TAX-Pose: Task-Specific Cross-Pose Estimation for Robot Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: How do we imbue robots with the ability to efficiently manipulate unseen objects and transfer relevant skills based on demonstrations? End-to-end learning methods often fail to generalize to novel objects or unseen configurations. Instead, we conjecture that the task-specific pose relationship between relevant parts of interacting objects is a generalizable notion of a manipulation task that can transfer to new objects in the same category; examples include the relationship between the pose of a lasagna relative to an oven or the pose of a mug relative to a mug rack. We call this task-specific pose relationship "cross-pose" and provide a mathematical definition of this concept. We propose a vision-based system that learns to estimate the cross-pose between two objects for a given manipulation task. The estimated cross-pose is then used to guide a downstream motion planner to manipulate the objects into the desired pose relationship (placing the lasagna into the oven or the mug onto the mug rack). We train a cross-pose estimator in simulation and we demonstrate the capability of our system to generalize to unseen objects in both simulation and the real world, deploying our policy on a Franka Emika with no finetuning. Results show that our system achieves state-of-the art performance in both simulated and real-world experiments. Supplementary information can be found at this anonymized website.

Keywords: Learning from Demonstration, Manipulation, 3D Learning

1 Introduction

Many manipulation tasks require that a robot is able to move an object to a location relative to another object. For example, a cooking robot may need to place a lasagna in an oven, place a pot on a stove, place a plate in a microwave, place a mug onto a mug rack, or place a cup onto a shelf. Understanding and placing objects in task-specific locations is a key skill for robots operating in human environments. Further, the robot should be able to generalize to novel objects within the training categories, such as placing new lasagnas into the oven or new mugs onto the mug rack. A common approach in robot learning is to train a policy "end-to-end," mapping from pixel observations to low-level robot actions. However, end-to-end trained policies cannot easily reason about complex pose relationships such as the ones described above, and they have difficulty generalizing to novel objects.

In contrast, we propose achieving these tasks by learning to reason about an object's three-dimensional geometry. For the type of tasks defined above, the robot needs to reason about the relationship between key parts on one object with respect to key parts on another object. For example, to place a mug on a mug rack, the robot must reason about the relationship between the pose of the mug handle and the pose of the mug rack; if the mug rack changes its pose, then the pose of the mug must change accordingly in order to still be placed on the rack (see Figure 2). We name this task-specific notion of the pose relationship between a pair of objects as "cross-pose" and we formally define it mathematically. Further, we propose a vision system that can efficiently estimate the cross-pose from a small number of demonstrations of a given task, generalizing to novel objects within the training categories (see Figure 1). To achieve the manipulation task, we input the

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

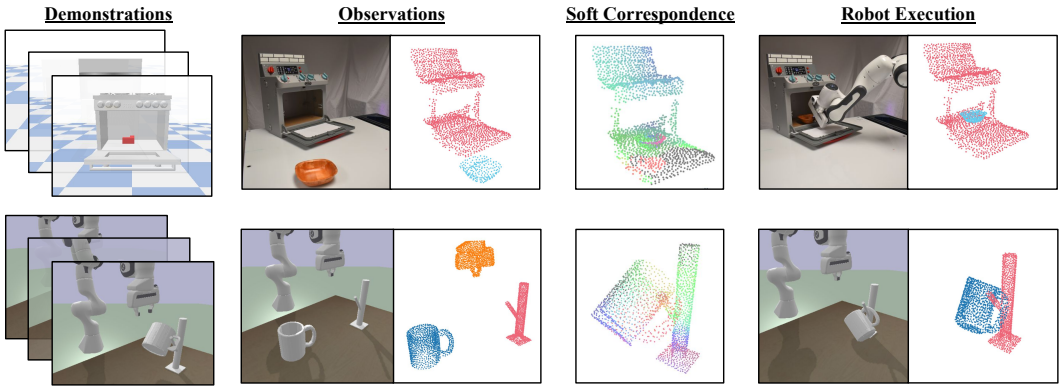

Figure 1: TAX-Pose in action. **Top:** PartNet-Mobility Placement Task. **Bottom:** Mug Hanging Task. The model is trained using demonstrations of anchor and action objects in their ground-truth cross-pose for the task. The model first observes the initial configuration of the objects, estimates correspondence between the pair of objects, and then calculates the desired pose. The desired pose is then used to guide robot motion planning.

estimated cross-pose into a motion planning algorithm which will manipulate the objects into the desired pose relationship (e.g. placing the mug onto the rack, placing the lasagna into the oven, etc).

In this paper, we present TAX-Pose (TAsk-specific Cross-Pose), a deep 3D vision-based robotics method that learns to predict a task-specific pose relationship between a pair of objects based on a set of demonstrations. We use this prediction to plan a trajectory that actuates the objects to achieve the desired relative pose. Our cross-pose estimation system is translation equivariant and can generalize from a small number of demonstrations to new objects in unseen poses.

The contributions of this paper include:

1. A precise definition of "cross-pose," which defines a task-specific pose relationship between two objects.
2. A novel method that estimates the cross-pose between two objects; this method is provably translation equivariant and can learn from a small number of demonstrations.
3. A robot system to manipulate objects into the desired cross-pose needed to achieve a given manipulation task.

We present simulated and real-world experiments to test the performance of our system in achieving a variety of cross-pose manipulation tasks, learning from a small number of demonstrations. We show the generalizabilty of our model though an object placement task, where the robot must accurately place objects in, on, or around novel objects. We then show the precision of the method, hanging unseen mugs onto a mug rack.

2 Related Work

Learning from Demonstration (LfD): LfD is a diverse field of study which focuses on enabling robots to learn skills from expert demonstrations. We refer the readers to previous survey papers [1, 2, 3] for a comprehensive review of LfD approaches. The commonly-used LfD technique is Behavior Cloning (BC) [4, 5], which involves imitating an expert agent given a set of demonstration trajectories by learning to predict the expert's action in a given state. This simple formulation has proven successful in a variety of tasks, including autonomous driving [6], robotic manipulation [7], and many more. In this project, we use the demonstrations to estimate a "goal pose" and then use motion planning to manipulate the objects into the desired pose.

Object Pose Estimation: Pose estimation is the task of detecting and inferring the 6DoF pose of an object, which includes its position and orientation, with respect to some previously defined object reference frame [8, 9, 10, 11, 12]. Although our task is related to that of object pose estimation, we do not aim to estimate the pose of a single object; rather, we define the notion of "cross-pose" which describes a task-specific relationship between a pair of objects.

Point Cloud Registration: Our method for estimating the cross-pose between two objects builds upon previous work in point cloud registration. The typical objective in point cloud registration is to find the optimal rigid alignment between two point clouds, to minimize the sum of squared distances between two sets of points. Traditionally, ICP [13] and its variants [14, 15, 16, 17, 18, 19] have been used to compute the optimal rigid alignment between two point clouds. Deep Closest Point (DCP) [20] avoids local minima common for ICP by seeking to approximate correspondence in a high-dimensional learned feature space. Our method builds upon the architecture of DCP for cross-pose estimation; however, in contrast to point cloud registration, in which the objective is to minimize the sum of squared distances between two sets of points, our objective is to estimate a task-specific pose relationship between two different objects.

3 Problem Statement

We first define the notion of cross-pose in the context of object placement tasks. Given two objects $\mathcal{A}$ and $\mathcal{B}$, we define the "relative placement" task of placing object $\mathcal{A}$ at a pose relative to object $\mathcal{B}$. For example, consider the task of placing a lasagna in an oven, placing a pot on a stove, or placing a mug on a rack, or placing a robot gripper on the rim of a mug. All of these tasks involve placing one object (which we call the "action" object $\mathcal{A}$) at a semantically meaningful location relative to another object (which we call the "anchor" object $\mathcal{B}$)[1].

In this work, we make the simplifying assumption that, for a given pair of objects $\mathcal{A}$ and $\mathcal{B}$, there is a single target relative pose needed to achieve a given task. Let $\mathbf{P}_\mathcal{A}$, $\mathbf{P}_\mathcal{B}$ be the point clouds of the objects, where $\mathbf{P}_k \in \mathbb{R}^{3\times N_k}$, and N_k denotes the number of (x, y, z) points in the point cloud of object k. We define the task-specific "cross-pose" between objects $\mathcal{A}$ and $\mathcal{B}$ via the function $f(\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B})$ which has the following properties: $f(\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B}) = \mathbf{I}$ (where $\mathbf{I}$ is the identity) when $\mathcal{A}$ and $\mathcal{B}$ are each in the target pose needed to complete the task (lasagna is in the oven; mug is on the rack, etc). The task-specific cross-pose function f has these further properties:

$$f(\mathbf{T}\cdot\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B}) = \mathbf{T}\cdot f(\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B}), \qquad f(\mathbf{P}_\mathcal{A}, \mathbf{T}\cdot\mathbf{P}_\mathcal{B}) = f(\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B})\cdot\mathbf{T}^{-1} \tag{1}$$

where $\mathbf{T} \in SE(3)$ is a pose transformation. We explore the limitations of other potential definitions of cross-pose in the appendix.

As a corollary, if $f(\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B}) = \mathbf{I}$, then $f(\mathbf{T}\cdot\mathbf{P}_\mathcal{A}, \mathbf{T}\cdot\mathbf{P}_\mathcal{B}) = \mathbf{I}$. In other words, the target cross-pose is invariant to the reference frame or a global pose transformation. Suppose that our task is to place a mug on a rack in a particular configuration. If we transform (rotate and translate) the mug by transformation $\mathbf{T}$ and we similarly transform the rack also by $\mathbf{T}$, then the cross-pose between the mug and the rack will be unchanged and the mug will still be "on" the rack, as seen in Figure 2.

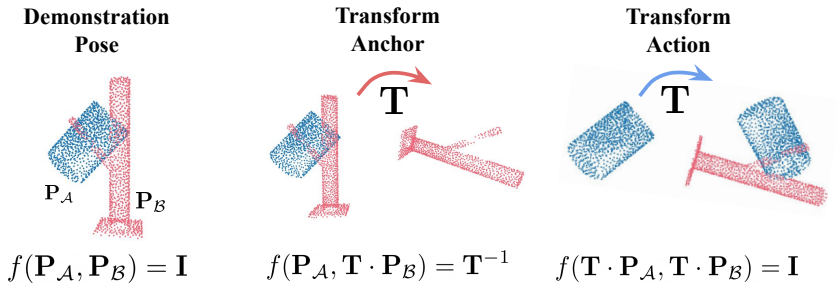

Figure 2: Visualization of the properties of cross-pose. If we transform both the action (mug) and the anchor (rack) objects by the same transform, then the relative pose between these objects is unchanged (the mug is still "on" the rack) so the cross-pose is unchanged.

We aim to learn a model, f_θ, that takes as input the two point clouds and predicts an SE(3) rigid transformation: $f_\theta(\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B}) = \mathbf{T}_{\mathcal{AB}}$, where $\mathbf{T}_{\mathcal{AB}}$ denotes the cross-pose between object $\mathcal{A}$ and object $\mathcal{B}$ as defined above.We can then transform object $\mathcal{A}$ by $\mathbf{T}_\alpha$ and $\mathcal{B}$ by $\mathbf{T}_\beta$ such that $\mathbf{T}_\alpha \cdot \mathbf{T}_{\mathcal{AB}} \cdot$

[1]Note that the definition of action and anchor is symmetric; either object can be treated as the action object and the other as the anchor.

$\mathbf{T}_\beta^{-1} = \mathbf{I}$. Given the properties of "cross-pose" described above, this will shift objects $\mathcal{A}$ and $\mathcal{B}$ into the desired target pose for the task. In practice, we typically transform only object $\mathcal{A}$ by $\mathbf{T}_\alpha = \mathbf{T}_{\mathcal{AB}}^{-1}$ without moving object $\mathcal{B}$, although in theory either (or both) objects can be moved.

4 Method

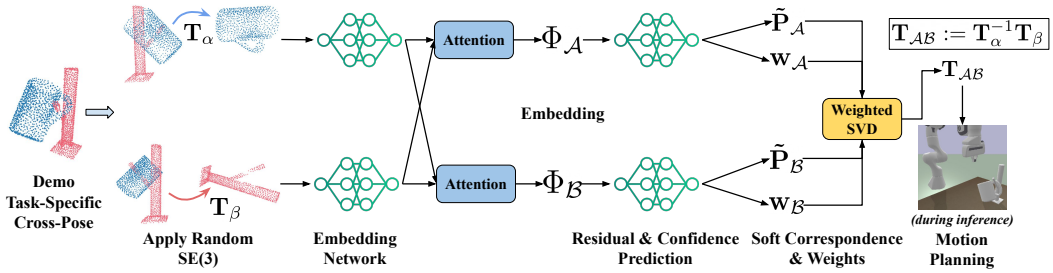

Figure 3: TAX-Pose Training Overview: Our method takes as input two point clouds given a specific task and outputs the cross-pose between them for the task. TAX-Pose first learns point clouds features using two DGCNN networks and two transformers. Then the learned features will be input to two point residual networks to predict per-point soft correspondence and weights between the two objects. Then the desired cross-pose can be inferred analytically using singular value decomposition.

Method Overview: We frame the task of cross-pose estimation as a correspondence-prediction task between a pair of point clouds, followed by an analytical least-squares optimization to find the optimal *cross-pose* for the predicted correspondences. At a high level, our method performs the following steps:

1. **Soft Correspondence Prediction**: For a pair of objects $\mathcal{A}, \mathcal{B}$, a neural network learns to predict a per-point embedding to establish a (soft) correspondence between $\mathcal{A}$ and $\mathcal{B}$.
2. **Adjustment via Correspondence Residuals**: To accommodate estimation tasks where $\mathcal{A}$ and $\mathcal{B}$ may not be in direct contact or overlap, we apply a pointwise residual vector to displace each of the predicted corresponding points. This allows points in $\mathcal{A}$ to correspond to points in free space near $\mathcal{B}$, for instance when a pot ($\mathcal{A}$) sits on top of a stove ($\mathcal{B}$).
3. **Find the optimal Cross-Pose Transform**: We use a standard SVD solution to the weighted Procrustes problem to find the optimal alignment given the corrected soft correspondences.

Because each step above is fully-differentiable, our method can learn arbitrary correspondences that solve arbitrary cross-pose estimation tasks. Our method is heavily inspired by Deep Closest Point (DCP) [20]. The key difference between our pose alignment model and DCP is that we are predicting the cross-pose between two *different* objects for a given task instead of registering two point clouds of an identical object. An overview of our method can be found in Figure 3. As we will discuss, this correspondence-based approach allows our method to be translation-equivariant: translating one object ($\mathcal{A}$ or $\mathcal{B}$) will lead to a translated cross-pose prediction.

We now describe our Cross-Pose estimation algorithm in detail. To recap the problem statement, given objects $\mathcal{A}$ and $\mathcal{B}$ with point cloud observations $\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B}$ respectively, our objective is to estimate the task-specific cross-pose $\mathbf{T}_{\mathcal{AB}} = f(\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B}) \in SE(3)$. Note that the cross-pose between object $\mathcal{A}$ and $\mathcal{B}$ is defined with respect to a given task (e.g. putting a lasagna in the oven, putting a mug on the rack, etc).

4.1 Cross-Pose Estimation via Soft Correspondence Prediction

Soft Correspondence Prediction: The first step of the method is to compute two sets of correspondences between $\mathcal{A}$ and $\mathcal{B}$, one which maps from points in $\mathcal{A}$ to $\mathcal{B}$, and one which maps from points in $\mathcal{B}$ to $\mathcal{A}$. These need not be a bijection, and can be asymmetric. We desire for this correspondence to be differentiable, so following DCP we define a *soft correspondence*, which assigns for every point $p_i^\mathcal{A} \in \mathbf{P}_\mathcal{A}$ a corresponding *virtual corresponding point* $v_i^{\mathcal{A}\to\mathcal{B}}$, which is a convex combination of points in $\mathbf{P}_\mathcal{B}$, and vice versa. Formally:

$$v_i^{\mathcal{A}\to\mathcal{B}} = \mathbf{P}_{\mathcal{B}} w_i^{\mathcal{A}\to\mathcal{B}} \quad \text{s.t.} \quad \sum_{j=1}^{N_{\mathcal{B}}} w_{ij}^{\mathcal{A}\to\mathcal{B}} = 1 \qquad v_i^{\mathcal{B}\to\mathcal{A}} = \mathbf{P}_{\mathcal{A}} w_i^{\mathcal{B}\to\mathcal{A}} \quad \text{s.t.} \quad \sum_{j=1}^{N_{\mathcal{A}}} w_{ij}^{\mathcal{B}\to\mathcal{A}} = 1$$

with normalized weight vectors $w_i^{\mathcal{A}\to\mathcal{B}}$ and $w_i^{\mathcal{B}\to\mathcal{A}}$. Importantly, these virtual points are not constrained to the surfaces of $\mathcal{A}$ or $\mathcal{B}$; instead, they are constrained to the convex hulls of $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$, respectively. Thus, we can reduce the correspondence prediction problem to predicting $w_i^{\mathcal{A}\to\mathcal{B}}$ and $w_i^{\mathcal{B}\to\mathcal{A}}$ for each point in $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$, respectively.

To accomplish this, we first encode each point cloud $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$ into a latent space using a neural network encoder. This encoder head is comprised of two distinct encoders $g_{\mathcal{A}}$ and $g_{\mathcal{B}}$, each of which receives point cloud $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$, respectively, and outputs a dense, point-wise embedding for each object: $\Phi_{\mathcal{A}} = g_{\mathcal{A}}(\mathbf{P}_{\mathcal{A}}), \quad \Phi_{\mathcal{B}} = g_{\mathcal{B}}(\mathbf{P}_{\mathcal{B}})$ where $\phi_i^{\mathcal{A}} \in \Phi_{\mathcal{A}}$ is the d-dimensional embedding of the i-th point in object $\mathcal{A}$, and likewise for object $\mathcal{B}$ (see Figure 3). We zero-center each observation point cloud before passing it into its encoder, and we employ a cross-attention module between the two embedding spaces (architecture details can be found in the appendix).

Since the point-wise embeddings $\phi_i^{\mathcal{A}}$ and $\phi_i^{\mathcal{B}}$ have the same dimension d, we can select the inner product of the space as a similarity metric between two embeddings. For any point $p_i^{\mathcal{A}}$, we can extract the desired normalized weight vector $w_i^{\mathcal{B}\to\mathcal{A}}$ with the softmax function:

$$w_i^{\mathcal{A}\to\mathcal{B}} = \text{softmax}\left(\Phi_{\mathcal{B}}^\top \phi_i^{\mathcal{A}}\right), \quad w_i^{\mathcal{B}\to\mathcal{A}} = \text{softmax}\left(\Phi_{\mathcal{A}}^\top \phi_i^{\mathcal{B}}\right) \tag{3}$$

Adjustment via Correspondence Residuals: Correspondences which are constrained to the convex hull of objects are insufficient to express a large class of desired tasks. For instance, we might want a point on the handle of a teapot to correspond to some point above a stovetop, which lies outside the convex hull of the points on the stovetop. To allow for such placements, we further learn a *residual vector* that corrects each virtual point, allowing us to displace each virtual point to any arbitrary location that might be suitable for the task. Concretely, we use a point-wise neural network $g_{\mathcal{R}}$ which maps each embedding into a 3-D residual vector:

$$r_i^{\mathcal{A}\to\mathcal{B}} = g_{\mathcal{R}}\left(\phi_i^{\mathcal{A}}\right), \quad r_i^{\mathcal{B}\to\mathcal{A}} = g_{\mathcal{R}}\left(\phi_i^{\mathcal{B}}\right)$$

Applying these to the virtual points, we get our *corrected virtual points*:

$$\tilde{v}_i^{\mathcal{A}\to\mathcal{B}} = v_i^{\mathcal{A}\to\mathcal{B}} + r_i^{\mathcal{A}\to\mathcal{B}}, \quad \tilde{v}_i^{\mathcal{B}\to\mathcal{A}} = v_i^{\mathcal{B}\to\mathcal{A}} + r_i^{\mathcal{B}\to\mathcal{A}} \tag{4}$$

Least-Squares Cross-Pose Optimization with Weighted SVD: We now have two sets of points and their corresponding corrected virtual points: $\left(\mathbf{P}_{\mathcal{A}}, \tilde{\mathbf{V}}_{\mathcal{B}}\right)$ and $\left(\mathbf{P}_{\mathcal{B}}, \tilde{\mathbf{V}}_{\mathcal{A}}\right)$, where $\tilde{\mathbf{V}}_{\mathcal{B}} = \left[\tilde{v}_1^{\mathcal{A}\to\mathcal{B}} \ldots \tilde{v}_{N_{\mathcal{A}}}^{\mathcal{A}\to\mathcal{B}}\right]^\top$ and similarly for $\tilde{\mathbf{V}}_{\mathcal{A}}$. We would like to compute the cross-pose transformation $\mathbf{T}_{\mathcal{AB}}$ that minimizes the weighted distance between correspondences, where the weights signify the importance of specific correspondences and are predicted as an additional channel of the encoding neural networks. This is the well-known weighted Procrustes problem, for which there exists an analytical solution (see appendix for details). We use a differentiable SVD operation [21], which allows us to compute a rotation R and translation t that minimizes the weighted distance between correspondences (where $\mathbf{T}_{\mathcal{AB}} = [R, t]$).

4.2 Supervision

To train the encoders $g_{\mathcal{A}}(\mathbf{P}_{\mathcal{A}})$, $g_{\mathcal{B}}(\mathbf{P}_{\mathcal{B}})$ as well as the residual networks $g_{\mathcal{R}}\left(\phi_i^{\mathcal{A}}\right)$, $g_{\mathcal{R}}\left(\phi_i^{\mathcal{A}}\right)$, we use a set of losses defined below. We assume we have access to a set of demonstrations of the task, in which the action and anchor objects are in the target relative pose such that $\mathbf{T}_{\mathcal{AB}} = \mathbf{I}$.

Point Displacement Loss [10, 22]: Instead of directly supervising the rotation and translation (as is done in DCP), we supervise the predicted transformation using its effect on the points. For this loss, we take the point clouds of the objects in the demonstration configuration, and transform each cloud by a random transform, $\hat{\mathbf{P}}_{\mathcal{A}} = \mathbf{T}_{\mathcal{A}}\mathbf{P}_{\mathcal{A}}$, and $\hat{\mathbf{P}}_{\mathcal{B}} = \mathbf{T}_{\mathcal{B}}\mathbf{P}_{\mathcal{B}}$. This would give us a ground truth transform of $\mathbf{T}_{\mathcal{AB}}^{GT} = \mathbf{T}_{\mathcal{B}}\mathbf{T}_{\mathcal{A}}^{-1}$; the inverse of this transform would move object $\mathcal{B}$ to the correct position relative to object $\mathcal{A}$. Using this ground truth transform, we compute the MSE loss between the correctly transformed points and the points transformed using our prediction.

$$\mathcal{L}_{\text{disp}} = \left\|\mathbf{T}_{\mathcal{AB}}\mathbf{P}_{\mathcal{A}} - \mathbf{T}_{\mathcal{AB}}^{GT}\mathbf{P}_{\mathcal{A}}\right\|^2 + \left\|\mathbf{T}_{\mathcal{AB}}^{-1}\mathbf{P}_{\mathcal{B}} - \mathbf{T}_{\mathcal{AB}}^{GT-1}\mathbf{P}_{\mathcal{B}}\right\|^2 \tag{5}$$

Direct Correspondence Loss. While the Point Displacement Loss best describes errors seen at inference time, it can lead to correspondences that are inaccurate but whose errors average to the correct pose. To improve these errors we directly supervise the learned correspondences $\tilde{V}_{\mathcal{A}}$ and $\tilde{V}_{\mathcal{B}}$:

$$\mathcal{L}_{\text{corr}} = \left\| \tilde{\mathbf{V}}_{\mathcal{A}} - \mathbf{T}_{\mathcal{AB}}^{GT} \mathbf{P}_{\mathcal{A}} \right\|^2 + \left\| \tilde{\mathbf{V}}_{\mathcal{B}} - \mathbf{T}_{\mathcal{AB}}^{GT-1} \mathbf{P}_{\mathcal{B}} \right\|^2 . \tag{6}$$

Correspondence Consistency Loss. Furthermore, a consistency loss can be used. This loss penalizes correspondences that deviate from the final predicted transform. A benefit of this loss is that it can help the network learn to respect the rigidity of the object, while it is still learning to accurately place the object. Note, that this is similar to the Direct Correspondence Loss, but uses the predicted transform as opposed to the ground truth one. As such, this loss requires no ground truth:

$$\mathcal{L}_{\text{cons}} = \left\| \tilde{\mathbf{V}}_{\mathcal{A}} - \mathbf{T}_{\mathcal{AB}} \mathbf{P}_{\mathcal{A}} \right\|^2 + \left\| \tilde{\mathbf{V}}_{\mathcal{B}} - \mathbf{T}_{\mathcal{AB}}^{-1} \mathbf{P}_{\mathcal{B}} \right\|^2 . \tag{7}$$

Overall Training Procedure. We train with a combined loss $\mathcal{L}_{\text{net}} = \mathcal{L}_{\text{disp}} + \lambda_1 \mathcal{L}_{\text{corr}} + \lambda_2 \mathcal{L}_{\text{cons}}$, where λ_1 and λ_2 are hyperparameters. We use a similar network architecture as DCP [20], which consists of DGCNN [23] and a Transformer [24]. We briefly experimented with Vector Neurons [25] and found that this led to worse performance on this task. In order to quickly adapt to new tasks, we optionally pre-train the DGCNN embedding networks over a large set of individual objects using the InfoNCE loss [26] with a geometric distance weighting and random transformations, to learn $SE(3)$ invariant embeddings (see appendix for further details).

5 Experiments

To evaluate TAX-Pose, we conduct a wide range of simulated and real-world experiments on two classes of relative placement tasks: Object Placement and Mug Hanging. The Object Placement objective is to place an action object on a flat surface on or near an anchor object. The Mug Hanging task objective is to grasp and then hang unseen mugs on a rack.

5.1 PartNet-Mobility Placement

Task Description: The PartNet-Mobility Placement task is defined as placing a given action object relative to an anchor object based on a semantic goal positions. We select a set of household furniture objects from the PartNet-Mobility dataset [27] as the anchor objects, and a set of small rigid objects released with the Ravens simulation environment [28] as the action objects. For each anchor object, we define a set of semantic goal positions (i.e. 'top', 'left', 'right', 'in'), where action objects should be placed relative to each anchor. Each semantic goal position defines a unique task in our cross-pose prediction framework. Given a synthetic point cloud observation of both objects, the task is to predict a cross-pose that places the object at the specific semantic goal. We train a single model across anchor/action categories, one model per semantic task (for a total of 4 models). We train entirely on simulated data, and transfer directly to real world with no finetuning. Details can be found in the appendix. We report rotation ($\mathcal{E}_{\mathbf{R}}$) and translation ($\mathcal{E}_{\mathbf{t}}$) error between our predicted transform and the ground truth as geodesic rotational distance [29, 30] and $L2$ distance, respectively.

Baselines: We compare our method to the following baselines:

E2E Behavioral Cloning: Generate motion-planned trajectories using OMPL that take the action object from start to goal. These serve as "expert" trajectories for Behavioral Cloning (BC), where we train a neural network to output a policy that, at each time step, outputs an incremental 6-DOF transformation that imitates the expert trajectory.

E2E DAgger: Using the same BC dataset as above, we train a policy using DAgger [4].

Trajectory Flow: Using the same BC dataset with DAgger, we train a policy to predict a dense per-point 3D flow vector at each time step instead of a single incremental 6-DOF transformation. Given this dense per-point flow, we can extract a rigid transformation using SVD yielding the next pose.

Goal Flow: Instead of training a multi-step policy to reach the goal, train a network to output a single dense prediction which assigns a per-point 3D flow vector that points from each action object point directly to its corresponding goal location. We extract a rigid transformation from these flow vectors using SVD, yielding the goal pose.

		AVG.																	
		$\mathcal{E}_{\mathbf{R}}$	$\mathcal{E}_{\mathbf{t}}$	$\mathcal{E}_{\mathbf{R}}$	$\mathcal{E}_{\mathbf{t}}$	$\mathcal{E}_{\mathbf{R}}$	$\mathcal{E}_{\mathbf{t}}$	$\mathcal{E}_{\mathbf{R}}$	$\mathcal{E}_{\mathbf{t}}$	$\mathcal{E}_{\mathbf{R}}$	$\mathcal{E}_{\mathbf{t}}$	$\mathcal{E}_{\mathbf{R}}$	$\mathcal{E}_{\mathbf{t}}$	$\mathcal{E}_{\mathbf{R}}$	$\mathcal{E}_{\mathbf{t}}$	$\mathcal{E}_{\mathbf{R}}$	$\mathcal{E}_{\mathbf{t}}$	$\mathcal{E}_{\mathbf{R}}$	$\mathcal{E}_{\mathbf{t}}$
Baselines	E2E BC	42.26	0.73	37.82	0.82	37.15	0.65	44.84	0.68	30.69	1.06	40.38	0.69	45.09	0.76	45.00	0.79	45.65	0.64
	E2E DAgger [4]	37.96	0.69	34.15	0.76	36.61	0.66	40.91	0.65	24.87	0.97	35.95	0.70	40.34	0.74	32.86	0.79	39.45	0.53
Ablations	Traj. Flow [31]	35.95	0.67	31.24	0.82	39.21	0.72	34.35	0.66	28.48	0.75	37.14	0.59	29.49	0.70	39.60	0.76	39.69	0.48
	Goal Flow [31]	26.64	0.17	25.88	**0.15**	25.05	0.15	30.62	0.15	27.61	0.10	28.01	**0.18**	20.96	**0.24**	29.02	0.23	22.13	0.20
Ours	**TAX-Pose**	**6.64**	**0.16**	**6.85**	0.16	**2.05**	**0.10**	**3.87**	**0.12**	**4.04**	**0.08**	**12.71**	0.31	**6.87**	0.37	**5.89**	**0.13**	**14.93**	**0.18**

Table 1: Goal Inference Rotational and Translational Error Results (↓). Rotational errors ($\mathcal{E}_{\mathbf{R}}$) are in degrees (°) and translational errors ($\mathcal{E}_{\mathbf{t}}$) are in meters (m). The lower the better.

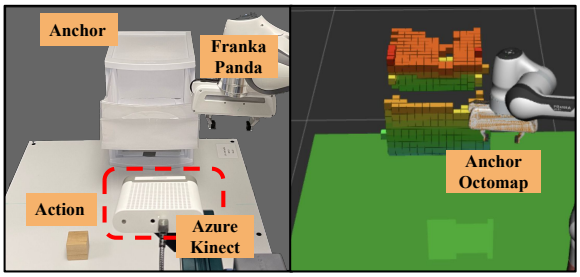

Figure 4: Real-world experiments illustration. **Left:** workspace setup for physical experiments. **Center:** Octomap visualization of the perceived anchor object.

Goal Flow	0.18	0.38	0.37
TAX-Pose	**0.95**	**0.95**	**0.85**

Table 2: Real-world goal placement success rate

Real-World Experiments: We design a set of of real world experiments to evaluate the performance of our cross-pose prediction model on real objects. We choose several real-world furniture objects similar to those found in the simulated training categories, annotate semantic goal locations for each, and choose several analogous action objects (bowl, block) to place at the goals. For each semantic goal, the task is to predict the appropriate cross-pose directly from point clouds recorded by a depth camera and have a Franka Emika Panda robot place the action object at the cross-pose (see Fig. 4 for the workspace). We compare TAX-Pose and the Goal Flow baseline for pose estimation and use OMPL motion planning on top of an Octomap [32] scene reconstruction to plan placement trajectories. As ground-truth cross-pose is hard to define in the real-world, we qualitatively define a "goal region" for each anchor object. Success is defined as the inferred pose of the action object landing inside the goal region of the anchor object. Details can be found in the appendix.

Results: In both our simulated experiments (Table 1) and our real-world experiments (Table 2), we find that TAX-Pose outperforms the baseline methods. In simulated experiments, while direct regression via goal-flow outperforms TAX-Pose in some rare cases of translation prediction, TAX-Pose performs substantially better than all other baselines in rotation prediction. Moreover, in real-world experiments, due to the provably translation-equivariant property of TAX-Pose, it generalizes to novel distributions of starting poses better than the Goal Flow regression baseline, successfully placing action objects into the goal regions.

5.2 Mug Hanging

Task Description. To successfully execute the manipulation task of hanging a mug on a rack by the mug's handle requires the successful inference of two sequential task-specific cross-poses: 1) predict a successful grasp pose; 2) predict the hanging pose of mug relative to the rack. In the first stage, it requires our model to reason about the cross-pose between the gripper and the mug, while the second stage requires prediction of the cross-pose between the mug and the rack.

Task Setup. To evaluate the performance of our method in simulation, we utilize Pybullet [33] and simulate a Franka Panda arm situated above a table with 4 depth cameras placed at each table corner. For training, the model is provided with 10 demonstrations in simulation, each on a different mug instances. At test time, we measure the task execution success on unseen mug instances, with randomly generated initial poses. To evaluate robustness to different initial poses, we evaluate on two sets of initial poses: 1) *Upright Pose*: where the mug is initialized to have an upright orientation, and placed randomly on the surface of the table; 2) *Arbitrary Pose*: where the mug is initialized to

have arbitrary orientation and position, irrespective of table surface is. We measure task success rates of, 1) *Grasping*, where success is achieved when the object is grasped stably; 2) *Placing*, where success is achieved when the mug is placed stably on the rack; 3) *Overall*, when the predicted transforms enable both grasp and place success. We compare our method to Neural Descriptor Field (NDF) [34] and Dense Object Nets (DON) [35]. Details of these methods can be found in [34].

	Grasp	**Place**	**Overall**	**Grasp**	**Place**	**Overall**
	Upright Pose			**Arbitrary Pose**		
DON [35]	0.91	0.50	0.45	0.35	0.45	0.17
NDF [34]	0.96	0.92	0.88	**0.78**	0.75	0.58
TAX-Pose (Ours)	**0.99**	**0.97**	**0.96**	0.75	**0.84**	**0.63**

Table 3: Mug on Rack Simulation Task Success Results

Results. We quantitatively evaluate our method in simulation on 100 trials on different initial configurations on unseen mug instances with randomly generated pose configurations for both **Upright** and **Arbitrary** poses and compare the performance of our method against DON [35] and NDF [34]. See Table 3 for full simulation results.

Model	**# Demos Used**		
	1	5	10
DON [35]	0.32	0.36	0.45
NDF [34]	0.46	0.70	0.88
TAX-Pose (Ours)	**0.77**	**0.90**	**0.96**

Table 4: # Demos vs. Overall Success

Ablation	**Grasp**	**Place**	**Overall**
No Res. Corresp.	0.97	0.96	0.93
Unweighted SVD	0.92	0.94	0.88
No Attention	0.90	0.82	0.76
TAX-Pose (Ours)	**0.99**	**0.97**	**0.96**

Table 5: Mug Hanging Ablations Results

Ablation Analysis. *Number of Demonstrations.* To study the effects of number of demonstrations used on the performance of our method, we report quantitative performance of our method alongside baseline methods trained on different numbers of demonstrations (10, 5, 1) for upright pose mug hanging task as seen in Table 4. Our method outperforms the baselines for all number of demonstrations; TAX-Pose can perform well even with just 5 demonstrations.

Cross-Pose Estimation Design Choices. We analyze the effects of the different design choices made in our Cross-Pose estimation algorithm for the upright pose mug hanging task. Specifically, we analyze the effects of 1) computing residual correspondence; 2) the use of *weighted* Procrustes over the non-weighted in computing cross-pose; 3) using a transformer as the cross-object attention as in Figure 3 method, as opposed to simpler model such as a 3-layer MLP. We report results in Table 5.

5.3 Limitations and Failure Cases

While our method is able to accurately predict the requisite transform to achieve a given task, it does require an accurate segmentation of the objects of importance. Additionally, while our method is tested with some occlusions, it performs better with a mostly complete cloud of the object being manipulated. This means that multiple views of that object must be captured. This can be done with multiple cameras, or by lifting the object and capturing multiple views. Additionally, as our method is a function of correspondences, symmetries could cause potential problems. This could be caused by interacting with symmetric objects or multimodality in the task to be completed, such as objects with multiple valid placement surfaces, or racks with multiple usable hangers. These problems could be alleviated using a consensus-based method for mapping from multimodal soft correspondences to a single transform. We leave this for future work.

6 Conclusion

In this paper, we show that dense soft correspondence can be used to learn task specific object relationships that generalize to novel object instances. Correspondence residuals allow our method to estimate correspondences to virtual points, outside of the objects convex hull, drastically increasing the number of tasks this method can complete. We further show that this "cross-pose" can be learned for a task, using a small number of demonstrations. Finally, we show that our method far outperforms the baselines on two challenging tasks in both real and simulated experiments.

References

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. Technical report, Springrer, 2008.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3 (6):233–242, 1999.

[4] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[5] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[6] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[7] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[8] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[9] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International journal of computer vision*, 66(3):231–259, 2006.

[10] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[11] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641, 2020.

[12] Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3003–3013, 2021.

[13] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[14] S. Bouaziz, A. Tagliasacchi, and M. Pauly. Sparse iterative closest point. In *Computer graphics forum*, volume 32, pages 113–123. Wiley Online Library, 2013.

[15] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001.

[16] A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[17] G. Agamennoni, S. Fontana, R. Y. Siegwart, and D. G. Sorrenti. Point clouds registration with probabilistic data association. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4092–4098. IEEE, 2016.

[18] T. Hinzmann, T. Stastny, G. Conte, P. Doherty, P. Rudol, M. Wzorek, E. Galceran, R. Siegwart, and I. Gilitschenski. Collaborative 3d reconstruction using heterogeneous uavs: System and experiments. In *International Symposium on Experimental Robotics*, pages 43–56. Springer, 2016.

[19] D. Hähnel and W. Burgard. Probabilistic matching for 3d scan registration. In *Proc. of the VDI-Conference Robotik*, volume 2002. Citeseer, 2002.

[20] Y. Wang and J. M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532, 2019.

[21] T. Papadopoulo and M. I. Lourakis. Estimating the jacobian of the singular value decomposition: Theory and applications. In *European Conference on Computer Vision*, pages 554–570. Springer, 2000.

[22] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[23] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[25] 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*, pages 12200–12209, 2021.

[26] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[27] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, and Others. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020.

[28] V. Zeng, T. E. Lee, J. Liang, and O. Kroemer. Visual identification of articulated object parts. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2443–2450. IEEE, 2020.

[29] D. Q. Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[30] R. Hartley, J. Trumpf, Y. Dai, and H. Li. Rotation averaging. *International journal of computer vision*, 103(3):267–305, 2013.

[31] B. Eisner*, H. Zhang*, and D. Held. Flowbot3d: Learning 3d articulation flow to manipulate articulated objects. In *Robotics: Science and Systems (RSS)*, 2022.

[32] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[33] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[34] 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. *arXiv preprint arXiv:2112.05124*, 2021.

[35] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.