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.

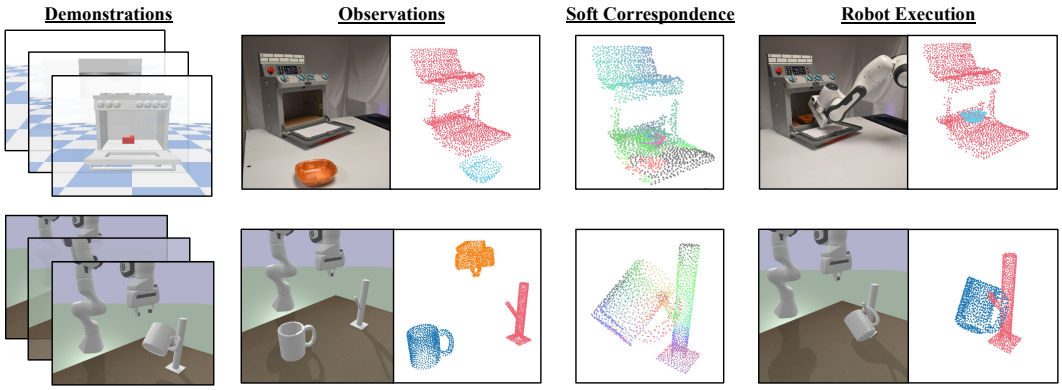


Figure 1: TAX-Pose in action. **Top:** PartNet-Mobility Placement Task. **Bottom:** Mug Hanging Task. The model is trained using demonstrations of anchor and action objects in their ground-truth cross-pose for the task. The model first observes the initial configuration of the objects, estimates correspondence 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. More recently, CLIPort [8] and Socratic Models [9] improve LfD agents' versatility by adding a multi-modal (language) component to the policy, making the decision-making process also conditioned on language instructions. In this project, we focus primary on the geometric relationships between objects, and use the demonstrations to estimate a "goal pose", which can be used by a motion planner to manipulate the objects into the desired poses.

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 [10, 11, 12, 13, 14, 15]. Recent work [16, 17, 18, 19] proposed to use 3D semantic

keypoints as an alternative form of object representation for manipulation. While keypoint-based methods can generalize within an object class, they require a significant amount of hand annotated data or access to simulated version of the task to estimate the keypoint locations. In contrast, our method is able to learn from just 10 demonstrations.

Dense descriptors [20, 21] and descriptor fields [22] achieve generalization across classes by predicting a dense embedding over the full image/point cloud space. To convert these task agnostic descriptors to a task specific pose, a subset of the descriptors are matched to demonstration objects. In Dense Object Nets (DON), a single point of reference is specified by a user [20]; in the Dense Object Nets baseline from NDF [22], this is extended to a set of user-selected points. In contrast, our method learns from just a set of demonstrations without needing extra user annotations. Further, we show that our method significantly outperforms the Dense Object Nets baseline [20].

Neural Descriptor Fields "assumes a static environment that remains fixed between demonstration-time and test-time" [22]. NDF needs this assumption because it uses a "known canonical configuration" for the reference object in the environment (e.g. the mug rack). In contrast, our method reasons about the "cross-pose" which describes a task-specific relationship between a pair of objects and thus does not need to assume a static environment. Our method learns from a small number of demonstrations to reason about the important task-specific relationships between object parts that enables our method to generalize to novel 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 [23] and its variants [24, 25, 26, 27, 28, 29] have been used to compute the optimal rigid alignment between two point clouds. Deep Closest Point (DCP) [30] 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. From a technical standpoint, our work is the first to introduce the notion of cross-pose, which we define in Section 3 and analyze theoretically in Supplement Section 1. Extending the framework from DCP, we learn a residual to the soft correspondences, allowing for points to match outside the convex hull of each object. This component is necessary when matching between objects of drastically different morphologies. Also, DCP does not use a weighted SVD; in contrast, we learn importance weights for different regions of the point cloud, allowing the system to focus on certain regions of each object that are important for the given task, which we integrate into a weighted differentiable SVD. Our ablation experiments show that each of these components contribute to our final performance.

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]. Relative placement tasks can either admit a single unambiguous solution (i.e. placing an asymmetric object in a specific pose, such as inserting an asymmetric peg into a hole), or a set of solutions (e.g. when there are object symmetries, or when the goal is semantically defined, such as "place the object somewhere on top of the table").

In this work, we make the simplifying assumption that, for a given pair of objects $\mathcal{A}$ and $\mathcal{B}$, there is a single, unambiguous 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

[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.

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.

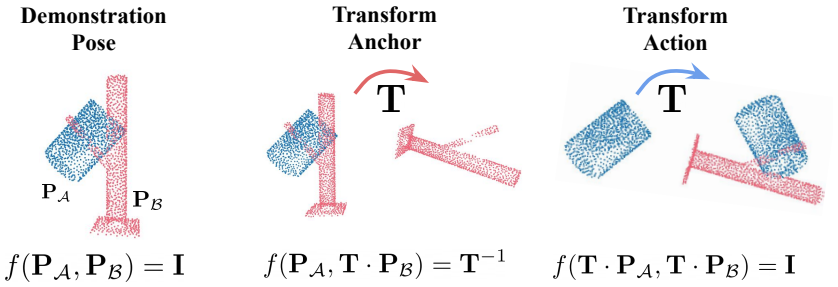


Figure 2: Visualization of the properties of cross-pose. If we transform both the action (mug) and the anchor (rack) objects by the same transform, then the relative pose between these objects is unchanged (the mug is still "on" the rack) so the cross-pose is unchanged.

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 $\mathbf{T}_\alpha \cdot \mathbf{T}_{\mathcal{AB}} \cdot \mathbf{T}_\beta^{-1} = \mathbf{I}$. 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

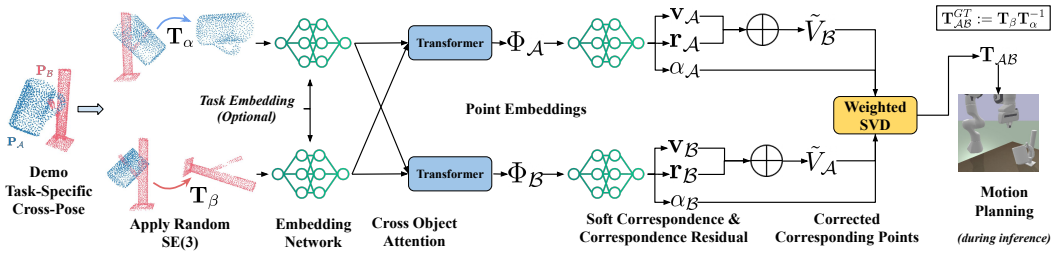


Figure 3: TAX-Pose Training Overview: Our method takes as input two point clouds given a specific task and outputs the cross-pose between them for the task. TAX-Pose first learns point clouds features using two DGCNN networks and two transformers. Then the learned features will be input to two point 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. This framing allows our cross-pose to adjust to novel positions of both the anchor and action objects, removing the restrictions to static anchor objects found in previous methods [22]. 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 virtual 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 virtual correspondence.

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) [30]. 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 corresponding 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 soft correspondence prediction problem to predicting $w_i^{\mathcal{A}\to\mathcal{B}} \in \mathbb{R}^{N_\mathcal{B}}$ and $w_i^{\mathcal{B}\to\mathcal{A}} \in \mathbb{R}^{N_\mathcal{A}}$ 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: $\Phi_\mathcal{A} = g_\mathcal{A}(\mathbf{P}_\mathcal{A}), \;\; \Phi_\mathcal{B} = g_\mathcal{B}(\mathbf{P}_\mathcal{B})$ 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-object 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:

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

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 corresponding point, allowing us to displace each virtual corresponding 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 correspondence*:

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

Least-Squares Cross-Pose Optimization with Weighted SVD: We now have two sets of points and their associated corrected virtual correspondence: $\left(\mathbf{P}_{\mathcal{A}}, \tilde{\mathbf{V}}_{\mathcal{B}}\right)$ and $\left(\mathbf{P}_{\mathcal{B}}, \tilde{\mathbf{V}}_{\mathcal{A}}\right)$, where $\tilde{\mathbf{V}}_{\mathcal{B}} = \left[\tilde{v}_1^{\mathcal{A}\to\mathcal{B}} \dots \tilde{v}_{N_{\mathcal{A}}}^{\mathcal{A}\to\mathcal{B}}\right]^\top$ 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 [31], 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 $g_{\mathcal{R}}\left(\phi_i^{\mathcal{A}}\right)$, $g_{\mathcal{R}}\left(\phi_i^{\mathcal{B}}\right)$, 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 [12, 32]: 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, $\hat{\mathbf{P}}_{\mathcal{A}} = \mathbf{T}_{\alpha}\mathbf{P}_{\mathcal{A}}$, and $\hat{\mathbf{P}}_{\mathcal{B}} = \mathbf{T}_{\beta}\mathbf{P}_{\mathcal{B}}$. This would give us a ground truth transform of $\mathbf{T}_{\mathcal{AB}}^{GT} = \mathbf{T}_{\beta}\mathbf{T}_{\alpha}^{-1}$; 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 $\tilde{V}_{\mathcal{B}}$:

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

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:

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

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 [30], which consists of DGCNN [33] and a Transformer [34]. We also optionally incorporate a contextual embedding vector into each DGCNN module - identical to the contextual encoding proposed in the original DGCNN paper - which can be used to provide an embedding of the specific placement relationship that is desired in a scene (e.g. selecting a "top" vs. "left" placement position) and thus enable goal conditioned placement. We refer to this variant as **TAX-Pose GC** (goal-conditioned). We briefly experimented with Vector Neurons [35] 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 [36] 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 position. We select a set of household furniture objects from the PartNet-Mobility dataset [37] as the anchor objects, and a set of small rigid objects released with the Ravens simulation environment [38] 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 two variants of our model, one **goal-conditioned** variant (**TAX-Pose GC**), and one **task-specific** variant (**TAX-Pose**): the only difference being that the TAX-Pose GC variant receives an encoding of the desired semantic goal position ('top', 'left', ...) for the task. The goal-conditioned variant is trained across all semantic goal positions, whereas the task-specific variant is trained separately on each semantic goal category (for a total of 4 models). Importantly, **both** variants are trained across **all PartNet-Mobility object categories**. 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 [39, 40] 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.

Note that in the PartNet-Mobility Placement experiments, the pose of the anchor object poses are randomly varied. As such, we omit comparison to methods that assume a static anchor, such as Neural Descriptor Field (NDF) [22] and Dense Object Nets (DON) [20], as both methods assume that the anchor objects are in a consistent position. Comparison to these baseline methods is reserved for Section 5.2.

		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 [41]	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 [41]	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**
	TAX-Pose GC	7.74	0.17	**4.43**	**0.13**	3.27	0.13	4.16	**0.12**	**3.3**	**0.08**	**10.76**	0.26	7.36	0.30	6.28	0.14	22.37	0.21

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

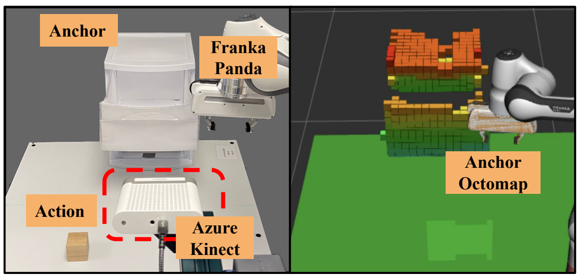


Figure 4: Real-world experiments illustration. **Left:** workspace setup for physical experiments. **Center:** Octomap visualization of the perceived anchor object.

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 [42] 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 [43] 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) [22] and Dense Object Nets (DON) [20]. Details of these methods can be found in [22].

	Grasp	Place	Overall	Grasp	Place	Overall
	Upright Pose			Arbitrary Pose		
DON [20]	0.91	0.50	0.45	0.35	0.45	0.17
NDF [22]	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 [20] and NDF [22]. See Table 3 for full simulation results.

Model	# Demos Used		
	1	5	10
DON [20]	0.32	0.36	0.45
NDF [22]	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] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[9] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. Florence. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv*, 2022.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] D. Turpin, L. Wang, S. Tsogkas, S. Dickinson, and A. Garg. Gift: Generalizable interaction-aware functional tool affordances without labels. *Robotics: Science and Systems (RSS)*, 2021.

[16] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. *International Symposium on Robotics Research (ISRR) 2019*, 2019.

[17] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese. Keto: Learning keypoint representations for tool manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7278–7285. IEEE, 2020.

[18] M. Vecerik, J.-B. Regli, O. Sushkov, D. Barker, R. Pevceviciute, T. Rothörl, R. Hadsell, L. Agapito, and J. Scholz. S3k: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency. In *Conference on Robot Learning*, pages 449–460. PMLR, 2021.

[19] L. Manuelli, Y. Li, P. Florence, and R. Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. In *Conference on Robot Learning*, pages 693–710. PMLR, 2021.

[20] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *Conference on Robot Learning*, pages 373–385. PMLR, 2018.

[21] Y. Liu, Z. Shen, Z. Lin, S. Peng, H. Bao, and X. Zhou. Gift: Learning transformation-invariant dense visual descriptors via group cnns. *Advances in Neural Information Processing Systems*, 32, 2019.

[22] 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)*, pages 6394–6400. IEEE, 2022.

[23] 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.

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

[25] 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.

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

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

[38] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv:2010.14406*, 2020.

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

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

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

[42] 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.

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

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

Anonymous Author(s)
Affiliation
Address
email

1 Limitations of Other Definitions of Cross-Pose

Relative placement tasks: In this paper, we are specifically interested in "relative placement tasks," which we define here. Loosely speaking, a relative placement task is a task such that only the relationship between objects $\mathcal{A}$ and $\mathcal{B}$ is important for task success. Specifically, suppose that $\mathbf{T}^*_{\mathcal{A}}$ and $\mathbf{T}^*_{\mathcal{B}}$ are poses for objects $\mathcal{A}$ and $\mathcal{B}$ respectively (in some reference frame) for which a desired task is complete (lasagna is in the oven; mug is on the rack, etc). Then for a relative placement task, if objects $\mathcal{A}$ and $\mathcal{B}$ are in poses $\mathbf{T} \cdot \mathbf{T}^*_{\mathcal{A}}$ and $\mathbf{T} \cdot \mathbf{T}^*_{\mathcal{B}}$ (respectively) for any transform $\mathbf{T}$, then the task will also be complete. In other words, if $\mathbf{T}^*_{\mathcal{B}}$ represents the pose of the oven and $\mathbf{T}^*_{\mathcal{A}}$ represents the pose of the lasagna in that oven (at task completion); then if we transform the lasagna pose by $\mathbf{T}$ and likewise transform the oven pose by $\mathbf{T}$, then the lasagna will still be located inside the oven.

Definition of Cross-Pose: In this section we will investigate the properties of different possible "cross-pose" formulations. Let us start by assuming that the "cross-pose" function for objects $\mathcal{A}$ and $\mathcal{B}$ takes the form $f(\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}) \in SE(3)$, where $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$ are the point clouds associated with objects $\mathcal{A}$ and $\mathcal{B}$, respectively. For convenience of the below analysis, we overload the function f to also receive as input the poses $\mathbf{T}_{\mathcal{A}}$, $\mathbf{T}_{\mathcal{B}}$ of objects $\mathcal{A}$ and $\mathcal{B}$ respectively (with respect to a global reference frame); in other words, we define cross-pose such that $f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}) := f(\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}})$.

We would like our definition of "cross-pose" to have the following properties:

1) Goal Consistency: Suppose that $\mathbf{T}^*_{\mathcal{A}}$ and $\mathbf{T}^*_{\mathcal{B}}$ are the poses of objects $\mathcal{A}$ and $\mathcal{B}$ in a desired relative configuration that achieves the relative placement task. Then if both objects are transformed by the same transform $\mathbf{T}$, their cross-pose should be unchanged. Specifically, we define "goal consistency" as the following property:

$$f(\mathbf{T}^*_{\mathcal{A}}, \mathbf{T}^*_{\mathcal{B}}) = f(\mathbf{T} \cdot \mathbf{T}^*_{\mathcal{A}}, \mathbf{T} \cdot \mathbf{T}^*_{\mathcal{B}})$$

for any transform $\mathbf{T} \in SE(3)$. This definition is consistent with the notion of success for a relative placement task defined above; if objects $\mathcal{A}$ and $\mathcal{B}$ are in a configuration such that the relative placement task is complete, then the cross-pose will be a constant value.

Let us define the cross-pose for which the task is complete as $f(\mathbf{T}^*_{\mathcal{A}}, \mathbf{T}^*_{\mathcal{B}}) = \mathbf{T}^*_{\mathcal{AB}} \in SE(3)$. In our paper, we chose $\mathbf{T}^*_{\mathcal{AB}} = \mathbf{I}$ where $\mathbf{I}$ is the identity. We explain below why the identity is a natural choice.

Note that we do not require that this property holds true if the objects are not in the desired relative configuration; thus, if objects $\mathcal{A}$ and $\mathcal{B}$ are in arbitrary poses $\mathbf{T}_{\mathcal{A}}$ and $\mathbf{T}_{\mathcal{B}}$ respectively (where $\mathbf{T}_{\mathcal{A}} \neq \mathbf{T}^*_{\mathcal{A}}$ and $\mathbf{T}_{\mathcal{B}} \neq \mathbf{T}^*_{\mathcal{B}}$), then we do not require that $f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}) = f(\mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}, \mathbf{T} \cdot \mathbf{T}_{\mathcal{B}})$.

2) Usability: The purpose of estimating the cross-pose is to determine how to transform the objects into a configuration such that the relative placement task is successful; in other words, suppose that objects $\mathcal{A}$ and $\mathcal{B}$ have a cross-pose of $f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}})$. Then we wish to use the "cross-pose" $f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}})$ and the desired "cross-pose" $f(\mathbf{T}^*_{\mathcal{A}}, \mathbf{T}^*_{\mathcal{B}}) = \mathbf{T}^*_{\mathcal{AB}}$ to compute a transform $\mathbf{T}_{\Delta}$ such that we can transform object $\mathcal{A}$ by $\mathbf{T}_{\Delta}$ to achieve the desired configuration for the relative placement task. In other words, we wish to compute $\mathbf{T}_{\Delta}$ such that $\mathbf{T}_{\Delta} \cdot \mathbf{T}_{\mathcal{A}}$ and $\mathbf{T}_{\mathcal{B}}$ are in a configuration that completes the task, i.e. objects $\mathcal{A}$ and $\mathcal{B}$ will be in poses $\mathbf{T} \cdot \mathbf{T}^*_{\mathcal{A}}$ and $\mathbf{T} \cdot \mathbf{T}^*_{\mathcal{B}}$ (respectively) for some transform $\mathbf{T}$; by the definition of the relative placement task above, this configuration is considered

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

a task success. As we will see below, some potential definitions of cross-pose allow us to compute $\mathbf{T}_\Delta$ more easily than others.

Option 1 (Ours): The definition of "cross-pose" used in our method is:

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

Using only this definition, we obtain the following properties:

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

Without any extra assumptions, goal consistency does not hold for this definition of cross-pose; if both objects are transformed by the same transform, the resulting transform can differ from the original cross-pose:

$$f(\mathbf{T} \cdot \mathbf{T}^*_\mathcal{A}, \mathbf{T} \cdot \mathbf{T}^*_\mathcal{B}) = \mathbf{T} \cdot \mathbf{T}^*_\mathcal{A} \cdot \mathbf{T}^{*-1}_\mathcal{B} \cdot \mathbf{T}^{-1} \neq \mathbf{T}^*_\mathcal{A} \cdot \mathbf{T}^{*-1}_\mathcal{B} = f(\mathbf{T}^*_\mathcal{A}, \mathbf{T}^*_\mathcal{B}).$$

In order to achieve the property of goal consistency, we can add the assumption that

$$f(\mathbf{T}^*_\mathcal{A}, \mathbf{T}^*_\mathcal{B}) = \mathbf{T}^*_{\mathcal{AB}} = \mathbf{I} \tag{2}$$

in the goal configuration. This implies that

$$f(\mathbf{T}^*_\mathcal{A}, \mathbf{T}^*_\mathcal{B}) = \mathbf{T}^*_\mathcal{A} \cdot \mathbf{T}^{*-1}_\mathcal{B} = \mathbf{I}.$$

Using this assumption, we then we obtain that

$$f(\mathbf{T} \cdot \mathbf{T}^*_\mathcal{A}, \mathbf{T} \cdot \mathbf{T}^*_\mathcal{B}) = \mathbf{I},$$

which satisfies the definition of goal consistency, since we then have $f(\mathbf{T}^*_\mathcal{A}, \mathbf{T}^*_\mathcal{B}) = f(\mathbf{T} \cdot \mathbf{T}^*_\mathcal{A}, \mathbf{T} \cdot \mathbf{T}^*_\mathcal{B}) = \mathbf{I}$ for any transform $\mathbf{T} \in SE(3)$.

Next we check the usability property: suppose that objects $\mathcal{A}$ and $\mathcal{B}$ have a desired "cross-pose" of $f(\mathbf{T}^*_\mathcal{A}, \mathbf{T}^*_\mathcal{B}) := \mathbf{T}^*_{\mathcal{AB}} = \mathbf{I}$. Let us assume that objects $\mathcal{A}$ and $\mathcal{B}$ have a current pose of $\mathbf{T}_\alpha \mathbf{T}^*_\mathcal{A}$ and $\mathbf{T}_\beta \mathbf{T}^*_\mathcal{B}$ respectively, for arbitrary transforms $\mathbf{T}_\alpha$ and $\mathbf{T}_\beta \in SE(3)$. Then the current "cross-pose" of objects $\mathcal{A}$ and $\mathcal{B}$ is:

$$f(\mathbf{T}_\alpha \mathbf{T}^*_\mathcal{A}, \mathbf{T}_\beta \mathbf{T}^*_\mathcal{B}) := \mathbf{T}_{\alpha\beta} = \mathbf{T}_\alpha \mathbf{T}^*_{\mathcal{AB}} \mathbf{T}_\beta^{-1} = \mathbf{T}_\alpha \cdot \mathbf{T}_\beta^{-1}. \tag{3}$$

Then we can compute a transform

$$\mathbf{T}_\Delta := \mathbf{T}^*_{\mathcal{AB}} \cdot \mathbf{T}^{-1}_{\alpha\beta} = \mathbf{T}_\beta \cdot \mathbf{T}^{-1}_\alpha;$$

such that if we transform object $\mathcal{A}$ by $\mathbf{T}_\Delta$ then object $\mathcal{A}$ will be in the pose

$$\mathbf{T}_\Delta \cdot \mathbf{T}_\alpha \cdot \mathbf{T}^*_\mathcal{A} = \mathbf{T}_\beta \cdot \mathbf{T}_\alpha^{-1} \cdot \mathbf{T}_\alpha \cdot \mathbf{T}^*_\mathcal{A} = \mathbf{T}_\beta \cdot \mathbf{T}^*_\mathcal{A}.$$

Since object $\mathcal{A}$ will be in the pose $\mathbf{T}_\beta \cdot \mathbf{T}^*_\mathcal{A}$ (after applying transformation $\mathbf{T}_\Delta$) and object $\mathcal{B}$ is already in the pose $\mathbf{T}_\beta \cdot \mathbf{T}^*_\mathcal{B}$, then the objects will now be in the desired relative configuration to complete the relative placement task. Note that, because of goal consistency, $\mathbf{T}^*_{\mathcal{AB}}$ is a constant, and above we have set it equal to the identity $\mathbf{I}$, so in this case $\mathbf{T}_\Delta = \mathbf{T}^{-1}_{\alpha\beta}$, which is the inverse of the cross-pose between objects $\mathcal{A}$ and $\mathcal{B}$ (see Equation 3). Thus we have shown the usability property: we can compute $\mathbf{T}_\Delta$ simply as the inverse of the cross-pose $\mathbf{T}^{-1}_{\alpha\beta}$; by transforming object $\mathcal{A}$ by $\mathbf{T}_\Delta$, we move the objects into the desired relative configuration, $\mathbf{T}_\beta \cdot \mathbf{T}^*_\mathcal{A}$ and $\mathbf{T}_\beta \cdot \mathbf{T}^*_\mathcal{B}$ that will complete the relative placement task.

Option 2: Alternatively, suppose we first disregard the need to satisfy the usability property. And instead we go with another definition that fully satisfies the goal consistency property without the need to add additional constraint, such as the one we have added for Option 1, where the demo cross pose is set to be the identity, as per Equation 2. To do this, we assume that

$$f(\mathbf{T}_\mathcal{A}, \mathbf{T}_\mathcal{B}) := \mathbf{T}_\mathcal{A}^{-1} \cdot \mathbf{T}_\mathcal{B} \tag{4}$$

In this case, transforming both objects by any transform $\mathbf{T}$ leaves the cross-pose unchanged:

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

so the goal consistency property holds.

This unfortunately comes at the cost of usability. First we notice that the definition of cross pose as defined in Equation 4 doesn't constitute a valid $SE(3)$ transformation, if we follow the same left multiplication convention for transformation composition, as is used in Option 1.

Next, we will show subsequently how this alternative option of cross pose definition violates the usability property. Suppose that objects $\mathcal{A}$ and $\mathcal{B}$ have a desired "cross-pose" of $f(\mathbf{T}^*_{\mathcal{A}}, \mathbf{T}^*_{\mathcal{B}}) := \mathbf{T}^{*-1}_{\mathcal{A}} \cdot \mathbf{T}^*_{\mathcal{B}} = \mathbf{T}^*_{\mathcal{AB}}$ and a current "cross-pose" of

$$f(\mathbf{T}_\alpha \cdot \mathbf{T}^*_{\mathcal{A}}, \mathbf{T}_\beta \cdot \mathbf{T}^*_{\mathcal{B}}) := \mathbf{T}_{\alpha\beta} = \mathbf{T}^{*-1}_{\mathcal{A}} \mathbf{T}^{-1}_\alpha \cdot \mathbf{T}_\beta \mathbf{T}^*_{\mathcal{B}},$$

for arbitrary transforms $\mathbf{T}_\alpha$ and $\mathbf{T}_\beta \in SE(3)$.To move object $\mathcal{A}$ into the desired relative configuration, with respect to the current pose of object $\mathcal{B}$, $\mathbf{T}_\beta \mathbf{T}^*_{\mathcal{B}}$, we need to transform it by $\mathbf{T}_\Delta := \mathbf{T}_\beta \cdot \mathbf{T}^{-1}_\alpha$, such that $\mathbf{T}_\Delta \cdot \mathbf{T}_\alpha \cdot \mathbf{T}^*_{\mathcal{A}} = \mathbf{T}_\beta \cdot \mathbf{T}^*_{\mathcal{A}}$. The only issue is that, in this case, the value for $\mathbf{T}_\Delta := \mathbf{T}_\beta \cdot \mathbf{T}^{-1}_\alpha$ cannot be easily computed from the current cross-pose $\mathbf{T}_{\alpha\beta} := \mathbf{T}^{*-1}_{\mathcal{A}} \cdot \mathbf{T}^{-1}_\alpha \cdot \mathbf{T}_\beta \cdot \mathbf{T}^*_{\mathcal{B}}$ and the desired cross-pose, $\mathbf{T}^*_{\mathcal{AB}} := \mathbf{T}^{*-1}_{\mathcal{A}} \cdot \mathbf{T}^*_{\mathcal{B}}$.

Alternative properties: The cross-pose function defined in option 1 has these properties:

$$f(\mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}) = \mathbf{T} \cdot f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}), \qquad f(\mathbf{T}_{\mathcal{A}}, \mathbf{T} \cdot \mathbf{T}_{\mathcal{B}}) = f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}) \cdot \mathbf{T}^{-1} \tag{5}$$

Now we ask whether we could instead define a cross-pose function that has the properties of

$$f(\mathbf{T} \cdot \mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}) = \mathbf{T} \cdot f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}), \qquad f(\mathbf{T}_{\mathcal{A}}, \mathbf{T} \cdot \mathbf{T}_{\mathcal{B}}) = \mathbf{T}^{-1} \cdot f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}). \tag{6}$$

As it turns out, we cannot. Suppose that objects $\mathcal{A}$ and $\mathcal{B}$ initially have poses of $\mathbf{T}_{\mathcal{A}}$ and $\mathbf{T}_{\mathcal{B}}$ respectively, with a cross-pose of $f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}})$. If you transform object $\mathcal{A}$ first by $\mathbf{T}_\alpha$ and then transform object $\mathcal{B}$ first by $\mathbf{T}_\beta$, then the final "cross-pose" is computed as follows:

$$f(\mathbf{T}_\alpha \cdot \mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}}) = \mathbf{T}_\alpha \cdot f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}})$$
$$f(\mathbf{T}_\alpha \cdot \mathbf{T}_{\mathcal{A}}, \mathbf{T}_\beta \cdot \mathbf{T}_{\mathcal{B}}) = \mathbf{T}^{-1}_\beta \cdot \mathbf{T}_\alpha \cdot f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}})$$

On the other hand, if you first transform $\mathcal{B}$ by $\mathbf{T}_\beta$ and then transform object $\mathcal{A}$ first by $\mathbf{T}_\alpha$, then the cross-pose would be computed as

$$f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_\beta \cdot \mathbf{T}_{\mathcal{B}}) = \mathbf{T}^{-1}_\beta \cdot f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}})$$
$$f(\mathbf{T}_\alpha \cdot \mathbf{T}_{\mathcal{A}}, \mathbf{T}_\beta \cdot \mathbf{T}_{\mathcal{B}}) = \mathbf{T}_\alpha \cdot \mathbf{T}^{-1}_\beta \cdot f(\mathbf{T}_{\mathcal{A}}, \mathbf{T}_{\mathcal{B}})$$

Note that we now would have two definitions of $f(\mathbf{T}_\alpha \cdot \mathbf{T}_{\mathcal{A}}, \mathbf{T}_\beta \cdot \mathbf{T}_{\mathcal{B}})$ which are not equivalent, since $\mathbf{T}^{-1}_\beta \cdot \mathbf{T}_\alpha \neq \mathbf{T}_\alpha \cdot \mathbf{T}^{-1}_\beta$. Thus, we cannot define a cross-pose function to have the properties of Equation 6 and instead we define our cross-pose function to have the properties of Equation 5.

2 Translational Equivariance

One benefit of our method is that it is translationally equivariant by construction. This mean that if the object point clouds, $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$, are translated by random translation $\mathbf{t}_\alpha$ and $\mathbf{t}_\beta$, respectively, i.e. $\mathbf{P}_{\mathcal{A}'} = \mathbf{P}_{\mathcal{A}} + \mathbf{t}_\alpha$ and $\mathbf{P}_{\mathcal{B}'} = \mathbf{P}_{\mathcal{B}} + \mathbf{t}_\beta$, then the resulting corrected virtual correspondences, $\tilde{\mathbf{V}}_{\mathcal{B}}$ and $\tilde{\mathbf{V}}_{\mathcal{A}}$, respectively, are transformed accordingly, i.e. $\tilde{\mathbf{V}}_{\mathcal{B}} + \mathbf{t}_\beta$ and $\tilde{\mathbf{V}}_{\mathcal{A}} + \mathbf{t}_\alpha$, respectively, as we will show below. This results in an estimated cross-pose transformation that is also equivariant to translation by construction. This is achieved because our learned features and correspondence residuals are invariant to translation, and our virtual correspondence points are equivariant to translation.

First, our point features are a function of centered point clouds. That is, given point clouds $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$, the mean of each point cloud is computed as

$$\bar{p}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{P}_k.$$

This mean is then subtracted from the clouds,

$$\bar{\mathbf{P}}_k = \mathbf{P}_k - \bar{p}_k,$$

which centers the cloud at the origin. The features are then computed on the centered point clouds:

$$\mathbf{\Phi}_k = g_k(\bar{\mathbf{P}}_k).$$

Since the point clouds are centered before features are computed, the features $\mathbf{\Phi}_k$ are invariant to an arbitrary translation $\mathbf{P}_{k'} = \mathbf{P}_k + \mathbf{t}_\kappa$.

These translationally invariant features are then used, along with the original point clouds, to compute "corrected virtual points" as a combination of virtual correspondence points, $v_i^{k'}$ and the correspondence residuals, $r_i^{k'}$. As we will see below, the "corrected virtual points" will be translationally equivariant by construction.

The virtual correspondence points, $v_i^{k'}$, are computed using weights that are a function of only the translationally invariant features, $\mathbf{\Phi}_k$:

$$w_i^{\mathcal{A}'\to\mathcal{B}'} = \text{softmax}\left(\Phi_{\mathcal{B}'}^\top \phi_i^{\mathcal{A}'}\right) = \text{softmax}\left(\Phi_{\mathcal{B}}^\top \phi_i^{\mathcal{A}}\right) = w_i^{\mathcal{A}\to\mathcal{B}};$$

thus the weights are also translationally invariant. These translationally invariant weights are applied to the translated cloud

$$v_i^{\mathcal{A}'\to\mathcal{B}'} = \mathbf{P}_{\mathcal{B}'} w_i^{\mathcal{A}\to\mathcal{B}} = (\mathbf{P}_{\mathcal{B}} + \mathbf{t}_\beta) w_i^{\mathcal{A}\to\mathcal{B}} = \sum_j p_{j,B} \cdot w_{i,j}^{\mathcal{A}\to\mathcal{B}} + \mathbf{t}_\beta \sum_j w_{i,j}^{\mathcal{A}\to\mathcal{B}} = \mathbf{P}_{\mathcal{B}} w_i^{\mathcal{A}\to\mathcal{B}} + \mathbf{t}_\beta,$$

since $\sum_{j=1}^{N_\mathcal{B}} w_{ij}^{\mathcal{A}\to\mathcal{B}} = 1$. Thus the virtual correspondence points $v_i^{\mathcal{A}'\to\mathcal{B}'}$ are equivalently translated. The same logic follows for the virtual correspondence points $v_i^{\mathcal{B}'\to\mathcal{A}'}$. This gives us a set of translationally equivaraint virtual correspondence points $v_i^{\mathcal{A}'\to\mathcal{B}'}$ and $v_i^{\mathcal{A}'\to\mathcal{B}'}$.

The correspondence residuals, $r_i^{k'}$, are a direct function of only the translationally invariant features $\mathbf{\Phi}_k$,

$$r_i^{k'} = g_\mathcal{R}(\phi_i^{k'}) = g_\mathcal{R}(\phi_i^k) = r_i^k,$$

therefore they are also translationally invariant.

Since the virtual correspondence points are translationally equivariant, $v_i^{\mathcal{A}'\to\mathcal{B}'} = v_i^{\mathcal{A}\to\mathcal{B}} + \mathbf{t}_\beta$ and the correspondence residuals are translationally invariant, $r_i^{k'} = r_i^k$, the final corrected virtual correspondence points, $\tilde{v}_i^{\mathcal{A}'\to\mathcal{B}'}$, are translationally equivariant, i.e. $\tilde{v}_i^{\mathcal{A}'\to\mathcal{B}'} = v_i^{\mathcal{A}\to\mathcal{B}} + r_i^k + \mathbf{t}_\beta$. This also holds for $\tilde{v}_i^{\mathcal{B}'\to\mathcal{A}'}$, giving us the final translationally equivariant correspondences between the translated object clouds as $\left(\mathbf{P}_\mathcal{A} + \mathbf{t}_\alpha, \tilde{\mathbf{V}}_\mathcal{B} + \mathbf{t}_\beta\right)$ and $\left(\mathbf{P}_\mathcal{B} + \mathbf{t}_\beta, \tilde{\mathbf{V}}_\mathcal{A} + \mathbf{t}_\alpha\right)$, where $\tilde{\mathbf{V}}_\mathcal{B} = \left[\tilde{v}_1^{\mathcal{A}\to\mathcal{B}} \dots \tilde{v}_{N_\mathcal{A}}^{\mathcal{A}\to\mathcal{B}}\right]^\top$.

As a result, the final computed transformation will be automatically adjusted accordingly. Given that we use weighted SVD to compute the optimal transform, $\mathbf{T}_{\mathcal{AB}}$, with rotational component $\mathbf{R}_{\mathcal{AB}}$ and translational component $\mathbf{t}_{\mathcal{AB}}$, the optimal rotation remains unchanged if the point cloud is translated, $\mathbf{R}_{\mathcal{A}'\mathcal{B}'} = \mathbf{R}_{\mathcal{AB}}$, since the rotation is computed as a function of the centered point clouds. The optimal translation is defined as

$$\mathbf{t}_{\mathcal{AB}} := \bar{\tilde{v}}_{\mathcal{A}\to\mathcal{B}} - \mathbf{R}_{\mathcal{AB}} \cdot \bar{p}_\mathcal{A},$$

where $\bar{\tilde{v}}_{\mathcal{A}\to\mathcal{B}}$ and $\bar{p}_\mathcal{A}$ are the means of the corrected virtual correspondence points, $\tilde{\mathbf{V}}_\mathcal{B}$, and the object cloud $\mathbf{P}_\mathcal{A}$, respectively, for object $\mathcal{A}$. Therefore, the optimal translation between the translated point cloud $\mathbf{P}_{\mathcal{A}'}$ and corrected virtual correspondence points $\tilde{\mathbf{V}}^{\mathcal{A}'\to\mathcal{B}'}$ is

$$\begin{aligned}
\mathbf{t}_{\mathcal{A}'\mathcal{B}'} &= \bar{\tilde{v}}_{\mathcal{A}'\to\mathcal{B}'} - \mathbf{R}_{\mathcal{AB}} \cdot \bar{p}_{\mathcal{A}'} \\
&= \bar{\tilde{v}}_{\mathcal{A}\to\mathcal{B}} + \mathbf{t}_\beta - \mathbf{R}_{\mathcal{AB}} \cdot (\bar{p}_\mathcal{A} + \mathbf{t}_\alpha) \\
&= \bar{\tilde{v}}_{\mathcal{A}\to\mathcal{B}} + \mathbf{t}_\beta - \mathbf{R}_{\mathcal{AB}} \cdot \bar{p}_\mathcal{A} - \mathbf{R}_{\mathcal{AB}} \cdot \mathbf{t}_\alpha \\
&= \mathbf{t}_{\mathcal{AB}} + \mathbf{t}_\beta - \mathbf{R}_{\mathcal{AB}} \cdot \mathbf{t}_\alpha
\end{aligned}$$

To simplify the analysis, if we assume that, for a given example, $\mathbf{R}_{\mathcal{AB}} = \mathbf{I}$, then we get $\mathbf{t}_{\mathcal{A}'\mathcal{B}'} = \mathbf{t}_{\mathcal{AB}} + \mathbf{t}_\beta - \mathbf{t}_\alpha$, demonstrating that the computed transformation is translation equivariant by construction.

3 Weighted SVD

The objective function for computing the optimal rotation and translation given a set of correspondences, $\{p_i^k \rightarrow \tilde{v}_i^k\}_i^{N_k}$ and weights $\{\alpha_i^k\}_i^{N_k}$, is as follows:

$$\mathcal{J}(\mathbf{T}_{\mathcal{AB}}) = \sum_{i=1}^{N_\mathcal{A}} \alpha_i^\mathcal{A} ||\mathbf{T}_{\mathcal{AB}}\, p_i^\mathcal{A} - \tilde{v}_i^{\mathcal{A}\rightarrow\mathcal{B}}||_2^2 + \sum_{i=1}^{N_\mathcal{B}} \alpha_i^\mathcal{B} ||\mathbf{T}_{\mathcal{AB}}^{-1}\, p_i^\mathcal{B} - \tilde{v}_i^{\mathcal{B}\rightarrow\mathcal{A}}||_2^2$$

First we center (denoted with $*$) the point clouds and virtual points independently, and stack them into frame-specific matrices (along with weights) retaining their relative position and correspondence:

$$\mathbf{A} = \begin{bmatrix}\mathbf{P}_\mathcal{A}^* & \tilde{\mathbf{V}}_\mathcal{A}^*\end{bmatrix}^\top, \quad \mathbf{B} = \begin{bmatrix}\tilde{\mathbf{V}}_\mathcal{B}^* & \mathbf{P}_\mathcal{B}^*\end{bmatrix}^\top, \quad \mathbf{\Gamma} = \text{diag}\left([\alpha_\mathcal{A} \quad \alpha_\mathcal{B}]\right)$$

Then the minimizing rotation $\mathbf{R}_{\mathcal{AB}}$ is given by:

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \text{svd}(\mathbf{A}\mathbf{\Gamma}\mathbf{B}) \tag{7a}$$

$$\mathbf{R}_{\mathcal{AB}} = \mathbf{U}\mathbf{\Sigma}_*\mathbf{V}^\top \tag{7b}$$

where $\mathbf{\Sigma}_* = \text{diag}(\begin{bmatrix}1, 1, ...\det(\mathbf{U}\mathbf{V}^\top)\end{bmatrix}$ and svd is a differentiable SVD operation [1].

The optimal translation can be computed as:

$$\mathbf{t}_\mathcal{A} = \bar{\tilde{\mathbf{v}}}_\mathcal{B} - \mathbf{R}_{\mathcal{AB}}\bar{\mathbf{p}}_\mathcal{A} \qquad \mathbf{t}_\mathcal{B} = \bar{\mathbf{p}}_\mathcal{B} - \mathbf{R}_{\mathcal{AB}}\bar{\tilde{\mathbf{v}}}_\mathcal{A} \qquad \mathbf{t} = \frac{N_\mathcal{A}}{N}\mathbf{t}_\mathcal{A} + \frac{N_\mathcal{B}}{N}\mathbf{t}_\mathcal{B} \tag{8a}$$

with $N = N_\mathcal{A} + N_\mathcal{B}$. In the special translation-only case, the optimal translation and be computed by setting $\mathbf{R}_{\mathcal{AB}}$ to identity in above equations. The final transform can be assembled:

$$\mathbf{T}_{\mathcal{AB}} = \begin{bmatrix}\mathbf{R}_{\mathcal{AB}} & \mathbf{t} \\ 0 & 1\end{bmatrix} \tag{9}$$

4 Additional Experiments

4.1 Further Ablations on Mug Hanging Task

In order to examine the effects of different design choices in the training pipeline, we conduct ablation experiments with final task-success (*grasp*, *place*, *overall*) as evaluation metrics for Mug Hanging task with upright pose initialization for the following components of our method, see Table S1 for full ablation results. For consistency, all ablated models are trained to 15K batch steps.

1. **Loss.** In the full pipeline reported, we use a weighted sum of the three types of losses described in **Section 4.2** of the paper. Specifically, the loss used $\mathcal{L}_\text{net}$ is given by

$$\mathcal{L}_\text{net} = \mathcal{L}_\text{disp} + \lambda_1\mathcal{L}_\text{cons} + \lambda_2\mathcal{L}_\text{corr} \tag{10}$$

where we chose $\lambda_1 = 0.1$, $\lambda_2 = 1$ after hyperparameter search.

We ablate usage of all three types of losses, by reporting the final task performance in simulation for all experiments, specifically, we report task success on the following $\mathcal{L}_\text{net}$ variants.

(a) Remove the point displacement loss term, $\mathcal{L}_\text{disp}$, after which we are left with

$$\mathcal{L}'_\text{net} = (0.1)\mathcal{L}_\text{cons} + \mathcal{L}_\text{corr}$$

(b) Remove the direct correspondence loss term, $\mathcal{L}_\text{corr}$, after which we are left with

$$\mathcal{L}'_\text{net} = \mathcal{L}_\text{disp} + (0.1)\mathcal{L}_\text{cons}$$

(c) Remove the correspondence consistency loss term, $\mathcal{L}_\text{cons}$, after which we are left with

$$\mathcal{L}'_\text{net} = \mathcal{L}_\text{disp} + \mathcal{L}_\text{corr}$$

(d) From testing loss variants above, we found that the point displacement loss is a vital contributing factor for task success, where removing this loss term results in no overall task success, as shown in Table S1. However, in practice, we have found that adding the correspondence consistency loss and direct correspondence loss generally help to lower the rotational error of predicted placement pose compared to the ground truth of collected demos. To further investigate the effects of the combination of these two loss terms, we used a scaled weighted combination of $\mathcal{L}_{\text{cons}}$ and $\mathcal{L}_{\text{corr}}$, such that the former weight of the displacement loss term is transferred to consistency loss term, with the new $\lambda_1 = 1.1$, and with $\lambda_2 = 1$ stays unchanged. Note that this is different from variant (a) above, as now the consistency loss given a comparable weight with dense correspondence loss term, which intuitively makes sense as the consistency loss is a function of the predicted transform $\mathbf{T}_{\mathcal{AB}}$ to be used, while the dense correspondence loss is instead a function of the ground truth transform, $\mathbf{T}^{GT}_{\mathcal{AB}}$, which provides a less direct supervision on the predicted transforms. Thus we are left with

$$\mathcal{L}'_{\text{net}} = (1.1)\mathcal{L}_{\text{cons}} + \mathcal{L}_{\text{corr}}$$

2. **Usage of Correspondence Residuals.** After predicting a per-point soft correspondence between objects $\mathcal{A}$ and $\mathcal{B}$, we adjust the location of the predicted corresponding points by further predicting a point-wise correspondence residual vector to displace each of the predicted corresponding point. This allows the predicted corresponding point to get mapped to free space outside of the convex hulls of points in object $\mathcal{A}$ and $\mathcal{B}$. This is a desirable adjustment for mug hanging task, as the desirable cross-pose usually require points on the mug handle to be placed somewhere near but not in contact with the mug rack, which can be outside of the convex hull of rack points. We ablate correspondence residuals by directly using the soft correspondence prediction to find the cross-pose transform through weighted SVD, without any correspondence adjustment via correspondence residual.

3. **Weighted SVD vs Non-weighted SVD.** We leverage weighted SVD as described in **Section 4.1** of the paper as we leverage predicted per-point weight to signify the importance of specific correspondence. We ablate the use of weighted SVD, and we use an un-weighted SVD, where instead of using the predicted weights, each correspondence is assign equal weights of $\frac{1}{N}$, where N is the number of points in the point cloud $\mathbf{P}$ used.

4. **Pretraining.** In our full pipeline, we pretrain the point cloud embedding network such that the embedding network is $SE(3)$ invariant. Specifically, the mug-specific embedding network is pretrained on 200 ShapeNet mug objects, while the rack-specific and gripper specific embedding network is trained on the same rack and Franka gripper used at test time, respectively. We conduct ablation experiments where

   (a) We omit the pretraining phase of embedding network
   (b) We do not finetune the embedding network during downstream training with task-specific demonstrations.

   Note that in practice, we find that pretraining helps speed up the downstream training by about a factor of 3, while models with or without pretraining both reach a similar final performance in terms of task success after both models converge.

5. **Usage of Transformer as Cross-object Attention Module.** In the full pipeline, we use transformer as the cross-object attention module, and we ablate this design choice by replacing the transformer architecture with a simple 3-layer MLP with ReLU activation and hidden dimension of 256, and found that this leads to worse place and grasp success.

6. **Dimension of Embedding.** In the full pipeline, the embedding is chosen to be of dimension 512. We conduct experiment on much lower dimension of 16, and found that with dimension =16, the place success is much lower, dropped from 0.97 to 0.59.

Ablation Experiment	Grasp	Place	Overall
No $\mathcal{L}_{\text{disp}}$	0.01	0	0
No $\mathcal{L}_{\text{corr}}$	0.89	0.91	0.84
No $\mathcal{L}_{\text{cons}}$	**0.99**	0.95	0.94
Scaled Combination: $1.1\mathcal{L}_{\text{cons}} + \mathcal{L}_{\text{corr}}$	0.10	0.01	0.01
No Adjustment via Correspondence Residuals	0.97	0.96	0.93
Unweighted SVD	0.92	0.94	0.88
No Finetuning for Embedding Network	0.98	0.93	0.91
No Pretraining for Embedding Network	**0.99**	0.72	0.71
3-Layer MLP In Place of Transformer	0.90	0.82	0.76
Embedding Network Feature Dim = 16	0.98	0.59	0.57
TAX-Pose (Ours)	**0.99**	**0.97**	**0.96**

Table S1: Mug Hanging Ablations Results

4.2 Effects of Pretraining

We explore the effects of pretraining on the final task performance, as well as training convergence speed. We have found that pretraining the point cloud embedding network as described in 5.1, is a helpful but not necessary component in our training pipeline. Specifically, we find that while utilizing pretraining reduces training time, allowing the model to reach similar task performance and train rotation/translation error with much fewer training steps, this component is not necessary if training time is not of concern. In fact, as see in Table S2, we find that for mug hanging tasks, by training the models from scratch without our pretraining, the models are able to reach similar level of task performance of 0.99 grasp, 0.92 for place and 0.92 for overall success rate. Furthermore, it is able to achieve similar level of train rotation error of 4.91° and translation error of $0.01m$, compared to the models with pretraining. However, without pre-trainig, the model needs to be trained for about 5 times longer (26K steps compared to 5K steps) to reach the similar level of performance. Thus we adopt our object-level pretraining in our overall pipeline to allow lower training time.

Another benefit of pretraining is that the pretraining is done in a multi-task way, so the network can be more quickly adapted to new tasks after the pretraining is performed. For example, we use the same pre-trained mug embeddings for both the gripper-mug cross-pose estimation for grasping as well as the mug-rack cross-pose estimation for mug hanging

Ablation Experiment	Grasp	Place	Overall	Train Rotation Error (°)	Train Translation Error (m)
No Pre-Training for Embedding Network (trained for 26K steps)	**0.99**	0.92	0.92	4.91	**0.01**
TAX-Pose (Ours) (trained for 5K steps)	**0.99**	**0.97**	**0.96**	**4.33**	**0.01**

Table S2: Ablation Experiments on the Effects of Pre-Training. We report the task success rate for upright mug hanging task over 100 trials each, as well as the grasping model's training rotational error (°) and translation error (m).

4.3 Additional Simulation Results on Bowl and Bottle

Object	Algorithm	Grasp	Place	Overall	Grasp	Place	Overall
		Upright Pose			Arbitrary Pose		
Mug	DON [2]	0.91	0.50	0.45	0.35	0.45	0.17
	NDF [3]	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**
Bowl	DON [2]	0.50	0.35	0.11	0.08	0.20	0
	NDF [3]	0.91	**1**	0.91	**0.79**	**0.97**	0.78
	TAX-Pose (Ours)	**0.99**	0.92	**0.92**	0.74	0.85	**0.85**
Bottle	DON [2]	0.79	0.24	0.24	0.05	0.02	0.01
	NDF [3]	**0.87**	**1**	**0.87**	**0.78**	**0.99**	**0.77**
	TAX-Pose (Ours)	0.55	0.99	0.55	0.61	0.55	0.52

Table S3: Unseen Object Instance Manipulation Task Success Rates in Simulation on Mug, *Bowl* and *Bottle* for Upright and Arbitrary Initial Pose. Each result is the success rate over 100 trials.

Additional results on *Grasp*, *Place* and *Overall* success rate in simulation for **Bowl** and **Bottle** are shown in Table S3. For bottle and bowl experiment, we follow the same experimentation setup as in [3], where the successful *grasp* is considered if a stable grasp of the object is obtained, and a successful *place* is considered when the bottle or bowl is stably placed upright on the elevated flat slab over the table without falling on the table. Reported task success results in are for both *Upright Pose* and *Arbitrary Pose* run over 100 trials each.

5 Additional Training Details

5.1 Pretraining

We utilize pretraining for the embedding network for the mug hanging task, and describe the details below.

We pretrain embedding network for each object category (mug, rack, gripper), such that the embedding network is $SE(3)$ *invariant* with respect to the point clouds of that specific object category. Specifically, the mug-specific embedding network is pretrained on 200 ShapeNet [4] mug instances, while the rack-specific and gripper-specific embedding network is trained on the same rack and Franka gripper used at test time, respectively. Note that before our pretraining, the network is randomly initialized with the Kaiming initialization scheme [5]; we don't adopt any third-party pretrained models.

For the network to be trained to be $SE(3)$ invariant, we pre-train with InfoNCE loss [6] with a geometric distance weighting and random $SE(3)$ transformations. Specifically, given a point cloud of an object instance, $\mathbf{P}_\mathcal{A}$, of a specific object category $\mathcal{A}$, and an embedding network $g_\mathcal{A}$, we define the point-wise embedding for as $\Phi_\mathcal{A} = g_\mathcal{A}(\mathbf{P}_\mathcal{A})$, where $\phi_i^\mathcal{A} \in \Phi_\mathcal{A}$ is a d-dimensional vector for each point $p_i^\mathcal{A} \in \mathbf{P}_\mathcal{A}$. Given a random $SE(3)$ transformation, $\mathbf{T}$, we define $\Psi_\mathcal{A} = g_\mathcal{A}(\mathbf{T}\mathbf{P}_\mathcal{A})$.

The weighted contrastive loss used for pretraining, $\mathcal{L}_{wc}$, is defined as

$$\mathcal{L}_{wc} := -\sum_{p_i^\mathcal{A} \in \mathbf{P}_\mathcal{A}} \log \left[\frac{\exp(d_{ij} \cdot (\phi_i^{\mathcal{A}\top} \cdot \psi_j^\mathcal{A}))}{\sum_{p_k^\mathcal{A} \in \mathbf{P}_\mathcal{A}} \exp(d_{ik} \cdot (\phi_i^{\mathcal{A}\top}, \psi_k^\mathcal{A}))} \right] \tag{11}$$

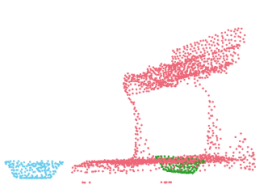
$$d_{ij} := \begin{cases} \mu \tanh(\lambda \|p_i^\mathcal{A} - p_j^\mathcal{A}\|_2), & \text{if } i \neq j \\ 1, & \text{otherwise} \end{cases} \tag{12}$$

$$\mu := \max(\tanh(\lambda \|p_i^\mathcal{A} - p_j^\mathcal{A}\|_2)) \tag{13}$$

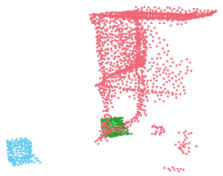
For this pretraining, we use $\lambda := 10$.

6 PartNet-Mobility Objects Placement Task Details

In this section, we describe the PartNet-Mobility Objects Placement experiments in details.



(a) Failure of "In" prediction. Predicted TAX-Pose violates the physical constraints by penetrating the oven base too much.



(b) Failure of "Left" prediction. Predicted TAX-Pose violates the physical constraints by being in collision with the leg of the drawer.

Figure S.1: An illustration of unsuccessful real-world TAX-Pose predictions. In both subfigures, red points represent the anchor object, blue points represent action object's starting pose, and green points represent action object's predicted pose.

6.1 Dataset Preparation

Simulation Setup. We leverage the PartNet-Mobility dataset [7] to find common household objects as the anchor object for TAX-Pose prediction. The selected subset of the dataset contains 8 categories of objects. We split the objects into 54 seen and 14 unseen instances. During training, for a specific task of each of the seen objects, we generate an action-anchor objects pair by randomly sampling transformations from $SE(3)$ as cross-poses. The action object is chosen from the Ravens simulator's rigid body objects dataset [8]. We define a subset of four tasks ("In", "On", "Left" and "Right") for each selected anchor object. Thus, there exists a ground-truth cross-pose (defined by human manually) associated with each defined specific task. We then use the ground-truth TAX-Poses to supervise each task's TAX-Pose prediction model. For each observation action-anchor objects pair, we sample 100 times using the aforementioned procedure for the training and testing datasets.

Real-World Setup. In real-world, we select a set of anchor objects: Drawer, Fridge, and Oven and a set of action objects: Block and Bowl. We test 3 ("In", "On", and "Left") TAX-Pose models in real-world without retraining or finetuning. The point here is to show the method capability of generalizing to unseen real-world objects.

6.2 Metrics

Simulation Metrics. In simulation, with access to the object's ground-truth pose, we are able to quantitatively calculate translational and rotation error of the TAX-Pose prediction models. Thus, we report the following metrics on a held-out set of anchor objects in simulation:

Translational Error: The L2 distance between the inferred cross-pose translation ($\mathbf{t}_{\mathcal{AB}}^{\text{pred}}$) and the ground-truth pose translation ($\mathbf{t}_{\mathcal{AB}}^{\text{GT}}$).

$$\mathcal{E}_{\mathbf{t}} = ||\mathbf{t}_{\mathcal{AB}}^{\text{pred}} - \mathbf{t}_{\mathcal{AB}}^{\text{GT}}||_2$$

Rotational Error: The geodesic $SO(3)$ distance [9, 10] between the predicted cross-pose rotation ($\mathbf{R}_{\mathcal{AB}}^{\text{pred}}$) and the ground-truth rotation ($\mathbf{R}_{\mathcal{AB}}^{\text{GT}}$).

$$\mathcal{E}_{\mathbf{R}} = \frac{1}{2}\arccos\left(\frac{\text{tr}(\mathbf{R}_{\mathcal{AB}}^{\text{pred}\top}\mathbf{R}_{\mathcal{AB}}^{\text{GT}}) - 1}{2}\right)$$

Real-World Metrics. In real-world, due to the difficulty of defining ground-truth TAX-Pose, we instead manually, qualitatively define goal "regions" for each of the anchor-action pairs. The goal-region should have the following properties:

- The predicted TAX-Pose of the action object should appear visually correct. For example, if the specified task is "In", then the action object should be indeed contained within the anchor object after being transformed by predicted TAX-Pose.
- The predicted TAX-Pose of the action object should not violate physical constraints of the workspace and of the relation between the action and anchor objects. Specifically, the action object should not interfere with/collide with the anchor object after being transformed by the predicted TAX-Pose. See Fig. S.1 for an illustration of TAX-Pose predictions that fail to meet this criterion.

6.3 Motion Planning

In both simulated and real-world experiments, we use off-the-shelf motion-planning tools to find a path between the starting pose and goal pose of the action object.

Simulation. To actuate the action object from its starting pose $\mathbf{T}_0$ to its goal pose transformed by the predicted TAX-Pose $\hat{\mathbf{T}}_{\mathcal{AB}}\mathbf{T}_0$, we plan a path free of collision. Learning-based methods such as [11] deal with collision checking with point clouds by training a collision classifier. A more data-efficient method is by leveraging computer graphics techniques, transforming the point clouds into marching cubes [12], which can then be used to efficiently reconstruct meshes. Once the triangular meshes are reconstructed, we can deploy off-the-shelf collision checking methods such as FCL [13] to detect collisions in the planned path. Thus, in our case, we use position control to plan a trajectory of the action object $\mathcal{A}$ to move it from its starting pose to the predicted goal pose. We use OMPL [14] as the motion planning tool and the constraint function passed into the motion planner is from the output of FCL after converting the point clouds to meshes via marching cubes.

Real World. In real-world experiments, we need to resolve several practical issues to make TAX-Pose prediction model viable. First, we do not have access to a mask that labels action and anchor objects. Thus, we manually define a mask by using a threshold value of y-coordinate to automatically detect discontinuity in y-coordinates, representing the gap of spacing between action and anchor objects upon placement. Next, grasping action objects is a non-trivial task. Since, we are only using 2 action objects (a cube and a bowl), we manually define a grasping primitive for each action object. This is done by hand-picking an offset from the centroid of the action object before grasping, and an approach direction after the robot reaches the pre-grasp pose to make contacts with the object of interest. The offsets are chosen via kinesthetic teaching on the robot when the action object is under identity rotation (canonical pose). Finally, we need to make an estimation of the action's starting pose for motion planning. This is done by first statistically cleaning the point cloud [15] of the action object, and then calculating the centroid of the action object point cloud as the starting position. For starting rotation, we make sure the range of the rotation is not too large for the pre-defined grasping primitive to handle. Another implementation choice here is to use ICP [16] calculate a transformation between the current point cloud to a pre-scanned point cloud in canonical (identity) pose. We use the estimated starting pose to guide the pre-defined grasp primitive. Once a successful grasp is made, the robot end-effector is rigidly attached to the action object, and we can then use the same predicted TAX-Pose to calculate the end pose of the robot end effector, and thus feed the two poses into MoveIt! to get a full trajectory in joint space. Note here that the collision function in motion planning is comprised of two parts: workspace and anchor object. That is, we first reconstruct the workspace using boxes to avoid collision with the table top and camera mount, and we then reconstruct the anchor object in RViz using Octomap [17] using the cleaned anchor object point cloud. In this way, the robot is able to avoid collision with the anchor object as well.

6.4 Baselines Description

In simulation, we compare our method to a variety of baseline methods.

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). We then use a PointNet++ network to output a sparse policy that, at each time step, takes as input the point cloud observation of the action and anchor objects and outputs an incremental 6-DOF transformation that imitates the expert trajectory. The 6-DoF transformation is expressed using Euclidean xyz translation and rotation quaternion. The "prediction" is the final achieved pose of the action object at the terminal state.

E2E DAgger: Using the same BC dataset and the same PointNet++ architecture as above, we train a sparse policy that outputs the same transformation representation as in BC using DAgger [18]. The "prediction" is the final achieved pose of the action object at the terminal state.

Trajectory Flow: Using the same BC dataset with DAgger, we train a dense policy using PointNet++ 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 add the per-point flow to each point of the current time-step's point cloud, and we are able to extract a rigid transformation between the current

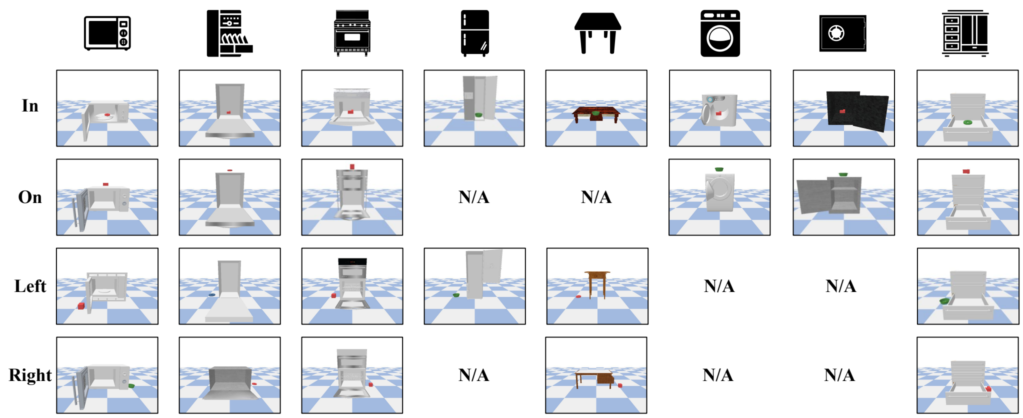


Figure S.2: A visualization of all categories of anchor objects and associated semantic tasks, with action objects in ground-truth TAX-Poses used in simulation training.

point cloud and the point cloud transformed by adding per-point flow vectors using SVD, yielding the next pose. The "prediction" is the final achieved pose of the action object at the terminal state.

Goal Flow: Instead of training a multi-step sparse/dense policy to reach the goal, train a PointNet++ network to output a single dense flow prediction which assigns a per-point 3D flow vector that points from each action object point from its starting pose directly to its corresponding goal location. Given this dense per-point flow, we add the per-point flow to each point of the start point cloud, and we are able to extract a rigid transformation between the start point cloud and the point cloud transformed by adding per-point flow vectors using SVD, yielding goal pose. We pass the start and goal pose into a motion planner (OMPL) and execute the planned trajectory. The "prediction" is thus given by the SVD output.

		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	42.37	0.69	40.49	0.80	50.79	0.59	48.02	0.61	30.69	1.09	36.59	0.81	48.48	0.42	41.42	0.84	42.49	0.37
	E2E DAgger [18]	36.06	0.67	38.57	0.68	43.99	0.63	42.34	0.57	24.87	0.96	30.87	0.90	42.96	0.46	29.79	0.83	35.08	0.33
Ablations	Traj. Flow [15]	34.48	0.65	35.39	0.85	43.42	0.63	35.51	0.60	28.26	0.80	27.67	0.68	25.91	0.44	43.59	0.82	36.05	0.36
	Goal Flow [15]	27.49	**0.21**	25.41	**0.08**	31.07	0.13	27.05	0.27	27.80	0.11	29.02	**0.38**	19.22	**0.36**	31.56	0.18	**28.81**	**0.19**
Ours	**TAX-Pose**	**11.74**	0.23	**5.81**	0.11	**1.82**	**0.08**	**5.92**	**0.11**	**3.67**	**0.07**	**19.54**	0.41	**7.96**	0.63	**5.96**	**0.12**	43.27	0.33

Table S4: Goal Inference Rotational and Translational Error Results (↓) for the "**In**" Goal. Rotational errors ($\mathcal{E}_\mathbf{R}$) are in degrees (°) and translational errors ($\mathcal{E}_\mathbf{t}$) are in meters (m). The lower the better.

		AVG.													
		$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$
Baselines	E2E BC	42.69	0.74	41.94	0.74	36.70	0.52	38.23	0.73	41.69	1.10	48.57	0.75	48.98	0.63
	E2E DAgger [18]	37.68	0.70	39.24	0.69	31.63	0.54	41.06	0.68	37.72	1.03	35.94	0.75	40.47	0.51
Ablations	Traj. Flow [15]	35.13	0.76	34.78	0.70	39.14	0.59	31.10	0.69	33.07	0.97	35.61	0.71	37.09	0.87
	Goal Flow [15]	22.10	0.20	27.82	0.26	20.43	**0.09**	34.66	0.10	22.71	0.12	26.48	0.27	0.48	0.32
Ours	**TAX-Pose**	**4.45**	**0.12**	**4.21**	**0.12**	**2.29**	0.10	**2.73**	**0.09**	**5.77**	**0.10**	**5.81**	**0.13**	**5.89**	**0.19**

Table S5: Goal Inference Rotational and Translational Error Results (↓) for the "**On**" Goal. Rotational errors ($\mathcal{E}_\mathbf{R}$) are in degrees (°) and translational errors ($\mathcal{E}_\mathbf{t}$) are in meters (m). The lower the better.

		AVG.													
		$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$
Baselines	E2E BC	44.87	0.74	30.95	0.89	36.86	0.72	56.86	0.52	34.35	1.03	31.69	0.77	46.86	0.78
	E2E DAgger [18]	41.32	0.68	31.40	0.84	38.49	0.73	47.64	0.51	36.47	0.99	27.72	0.73	39.83	0.51
Ablations	Traj. Flow [15]	38.85	0.58	31.87	1.07	39.48	0.44	39.48	0.44	28.71	0.69	41.06	0.73	40.70	0.31
	Goal Flow [15]	29.64	**0.10**	28.51	**0.10**	26.33	**0.08**	32.96	**0.07**	27.42	0.10	22.04	**0.09**	27.42	0.15
Ours	**TAX-Pose**	**6.02**	0.17	**12.73**	0.28	**1.59**	0.11	**2.91**	0.12	**4.41**	**0.08**	**12.12**	0.34	**6.38**	**0.12**

Table S6: Goal Inference Rotational and Translational Error Results (↓) for the "**Left**" Goal. Rotational errors ($\mathcal{E}_\mathbf{R}$) are in degrees ($^\circ$) and translational errors ($\mathcal{E}_\mathbf{t}$) are in meters (m). The lower the better.

		AVG.													
		$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$
Baselines	E2E BC	39.11	0.76	37.89	0.86	24.26	0.77	36.27	0.88	52.86	0.48	44.26	0.78		
	E2E DAgger [18]	36.80	0.73	27.40	0.84	32.31	0.74	32.61	0.82	49.27	0.46	42.40	0.78		
Ablations	Traj. Flow [15]	35.33	0.71	22.93	0.66	34.78	1.22	31.29	0.92	42.71	0.37	44.93	0.36		
	Goal Flow [15]	27.34	0.16	21.79	0.15	22.37	0.28	27.79	0.15	32.96	**0.07**	31.79	0.15		
Ours	**TAX-Pose**	**4.33**	**0.13**	**4.64**	**0.14**	**2.48**	**0.11**	**3.91**	**0.15**	**6.47**	0.17	**4.17**	**0.08**		

Table S7: Goal Inference Rotational and Translational Error Results (↓) for the "**Right**" Goal. Rotational errors ($\mathcal{E}_\mathbf{R}$) are in degrees ($^\circ$) and translational errors ($\mathcal{E}_\mathbf{t}$) are in meters (m). The lower the better.

6.5 Per-Task Results

In the main body of the paper, we have shown the meta-results of the performance of each method by averaging the quantitative metrics for each sub-task ("In", "On", "Left", and "Right" in simulation and "In", "On" and "Left" in real-world). Here we show each sub-task's results in Table S4[1], Table S5, Table S6, and Table S7 respectively.

As mentioned above, not all anchor objects have all 4 tasks due to practical reasons. For example, the doors of safes might occlude the action object completely and it is impossible to show the action object in the captured image under "Left" and "Right" tasks (due to handedness of the door); a table's height might be too tall for the camera to see the action object under the "Top" task. Under this circumstance, for sake of simplicity and consistency, we define a subset of the 4 goals for each object such that the anchor objects of the same category have consistent tasks definitions. We show a collection of visualizations of each task defined for each category in Fig.S.2.

Moreover, we also show per-task success rate for real-world experiments in Table S8.

	In			On			Left		
Goal Flow	0.00	0.10	0.30	0.05	N/A	0.20	0.50	0.65	0.60
TAX-Pose	**1.00**	**1.00**	**0.85**	**1.00**	N/A	**1.00**	**0.85**	**0.90**	**0.70**

Table S8: Combined per-task results for real-world goal placement success rate.

7 Mug Hanging Task Details

In this section, we describe the Mug Hanging task and experiments in details. The Mug Hanging task is consisted of two sub tasks: *grasp* and *place*. A success in grasp is achieved when the mug is

[1]Categories from left to right: microwave, dishwasher, oven, fridge, table, washing machine, safe, and drawer.

grasped stably by the gripper, while a success in place is achieved when the mug is hanged stably on the hanger of the rack. And the overall success mug hanging is considered when the predicted transforms enable both grasp and place success for the same trial. See Figure S.3 for a detailed breakdown of the mug hanging task in stages.

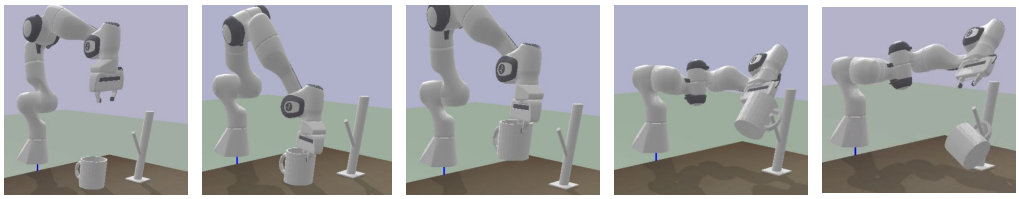


Figure S.3: Visualization of Mug Hanging Task (Upright Pose). Mug hanging task is consisted of two stages, given a mug that is randomly initialized on the table, the model first predicts a $SE(3)$ transform from gripper end effector to the mug rim $\mathbf{T}_{g\to m}$, then grasp it by the rim. Next, the model predicts another $SE(3)$ transform from the mug to the rack $\mathbf{T}_{m\to r}$ such that the mug handle gets hanged on the the mug rack.

7.1 Baseline Description

In simulation, we compare our method to the results described in [3].

- **Dense Object Nets (DON)** [2]: Using manually labeled semantic keypoints on the demonstration clouds, DON is used to compute sparse correspondences with the test objects. These correspondences are converted to a pose using SVD. A full description of usage of DON for the mug hanging task can be found in [3].
- **Neural Descriptor Field (NDF)** [3]: Using the learned descriptor field for the mug, the positions of a constellation of task specific query points are optimized to best match the demonstration using gradient descent.

7.2 Training Data

To be directly comparable with the baselines we compared to, we use the exact same sets of demonstration data used to train the network in NDF [3], where the data are generated via teleportation in PyBullet, collected on 10 mug instances with random pose initialization.

7.3 Training and Inference

Using the pretrained embedding network for mug and gripper, we train a grasping model for the grasping task to predict a transformation $\mathbf{T}_{g\to m}$ in gripper's frame from gripper to mug to complete the *grasp* stage of the task. Similarly, using the pretrained embedding network for rack and mug, we train a placement model for the placing task to predict a transformation $\mathbf{T}_{m\to r}$ in mug's frame from mug to rack to complete the *place* stage of the task. Both model are trained with the same combined loss $\mathcal{L}_{net}$ as described in the main paper. During inference, we simply use grasping model to predict the $\mathbf{T}_{g\to m}$ at test time, and placement model to predict $\mathbf{T}_{m\to r}$ at test time.

7.4 Motion Planning

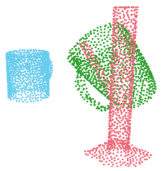
After the model predicts a transformation $\mathbf{T}_{g\to m}$ and $\mathbf{T}_{m\to r}$, using the known gripper's world frame pose, we calculate the desired gripper end effector pose at grasping and placement, and pass the end effector to IKFast to get the desired joint positions of Franka at grasping and placement. Next we pass the desired joint positions at gripper's initial pose, and desired grasping joint positions to OpenRAVE motion planning library to solve for trajectory from gripper's initial pose to grasp pose, and then grasp pose to placement pose for the gripper's end effector.

7.5 Failure Cases

Some failure cases for TAX-Pose happens when the predicted gripper misses the rim of the mug by a xy-plane translation error, thus resulting in failure of grasp, as seen in Figure S.4a. And common

failure mode for the mug placement subtask is charactereized by erroneous transform prediction that results in the mug's handle completely missing the rack hanger, thus resulting in placement failure, as seen in Figure S.4b.

(a) Failure of *grasp* prediction. Predicted TAX-Pose for the gripper misses the rim of mug.



(b) Failure of *place* prediction. Predicted TAX-Pose for mug results in the mug handle misses the rack hanger completely.

Figure S.4: An illustration of unsuccessful TAX-Pose predictions for mug hanging. In both subfigures, red points represent the anchor object, blue points represent action object's starting pose, and green points represent action object's predicted pose.

References

[1] 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.

[2] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *Conference on Robot Learning*, pages 373–385. PMLR, 2018.

[3] 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)*, pages 6394–6400. IEEE, 2022.

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

[5] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

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

[7] 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.

[8] 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.

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

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

[11] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox. Object rearrangement using learned implicit collision functions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6010–6017. IEEE, 2021.

[12] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[13] J. Pan, S. Chitta, and D. Manocha. Fcl: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866. IEEE, 2012.

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

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

[16] 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.

[17] 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.

[18] 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.