Appendix

A Visual Explanations of TAX-Pose

A.1 Illustration of Corrected Virtual Correspondence

The virtual corresponding points, $\mathbf{V}_{\mathcal{A}}$, $\mathbf{V}_{\mathcal{B}}$ given by Equation 3 in the main text, are constrained to be within the convex hull of each object. However, correspondences which are constrained to the convex hull 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, for each point-wise embedding $\boldsymbol{\phi}_i$, we further learn a *residual vector*, $\boldsymbol{\delta}_i^{\mathcal{A}} \in \boldsymbol{\Delta}_{\mathcal{A}}$ 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 3D residual vector:

$$\boldsymbol{\delta}_i^{\mathcal{A}} = g_{\mathcal{R}}\left(\boldsymbol{\phi}_i^{\mathcal{A}}\right) \in \mathbb{R}^3, \quad \boldsymbol{\delta}_i^{\mathcal{B}} = g_{\mathcal{R}}\left(\boldsymbol{\phi}_i^{\mathcal{B}}\right) \in \mathbb{R}^3$$

Applying these to the virtual points, we get a set of *corrected virtual correspondences*, $\tilde{\mathbf{v}}_i^{\mathcal{A}} \in \tilde{\mathbf{V}}_{\mathcal{A}}$ and $\tilde{\mathbf{v}}_i^{\mathcal{B}} \in \tilde{\mathbf{V}}_{\mathcal{B}}$, defined as

$$\tilde{\mathbf{v}}_i^{\mathcal{A}} = \mathbf{v}_i^{\mathcal{A}} + \boldsymbol{\delta}_i^{\mathcal{A}}, \quad \tilde{\mathbf{v}}_i^{\mathcal{B}} = \mathbf{v}_i^{\mathcal{B}} + \boldsymbol{\delta}_i^{\mathcal{B}} \tag{10}$$

These corrected virtual correspondences $\tilde{\mathbf{v}}_i^{\mathcal{A}}$ define the estimated goal location relative to object $\mathcal{B}$ for each point $\mathbf{p}_i \in \mathbf{P}_{\mathcal{A}}$ in object $\mathcal{A}$, and likewise for each point in object $\mathcal{B}$, as shown in Figure 6.

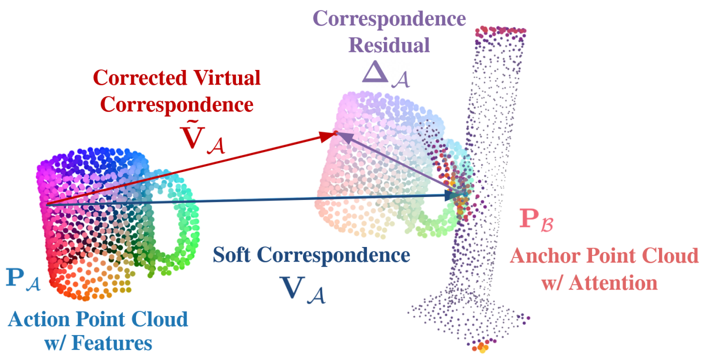

Correspondence Residual
$\boldsymbol{\Delta}_{\mathcal{A}}$
Corrected Virtual Correspondence
$\tilde{\mathbf{V}}_{\mathcal{A}}$
$\mathbf{P}_{\mathcal{B}}$
Anchor Point Cloud w/ Attention
Soft Correspondence
$\mathbf{V}_{\mathcal{A}}$
$\mathbf{P}_{\mathcal{A}}$
Action Point Cloud w/ Features

Figure 6: Computation of Corrected Virtual Correspondence. Given a pair of object point clouds $\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}$, a per-point *soft correspondence* $\mathbf{V}_{\mathcal{A}}$ is first computed. Next, to allow the predicted correspondence to lie beyond object's convex hull, these soft correspondences are adjusted with *correspondence residuals*, $\boldsymbol{\Delta}_{\mathcal{A}}$, which results in the *corrected virtual correspondence*, $\tilde{\mathbf{V}}_{\mathcal{A}}$. The coloring scheme and the point size on the rack represent the the value of the the attention weights, where the more red and larger the point, the higher the attention weights, the more gray and smaller the point the lower the attention weights.

A.2 Learned Importance Weights

A visualization of the learned importance weights, $\alpha_{\mathcal{A}}$ and $\alpha_{\mathcal{B}}$ for the mug and rack are visualized by both color scheme and point size in Figure 7

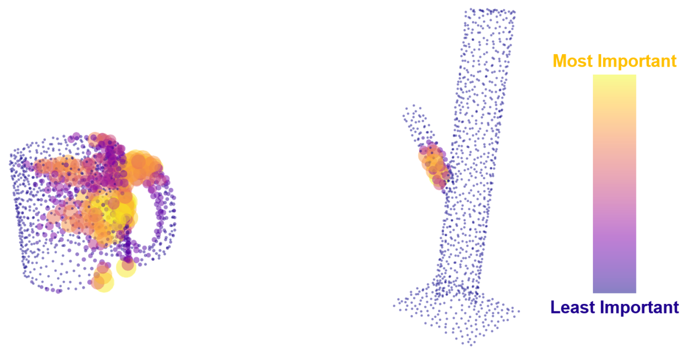


Figure 7: Learned Importance Weights for Weighted SVD on Mug and Rack. The coloring scheme and the point size on both objects represent the the value of the the learned importance weights, where the more yellow and larger the point, the higher the learned importance weights, the more purple and smaller the point the lower the learned importance weights.

B Proof of TAX-Pose 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

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

This mean is then subtracted from the clouds,

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

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, $\mathbf{v}_i^{k'}$ and the correspondence residuals, $\boldsymbol{\delta}_i^{k'}$. As we will see below, the "corrected virtual points" will be translationally equivariant by construction.

The virtual correspondence points, $\mathbf{v}_i^{k'}$, are computed using weights that are a function of only the translationally invariant query and key values from the cross-object attention transformer $g_{\mathcal{T}_\mathcal{K}}$, $\mathbf{Q}_\mathcal{K}$ and $\mathbf{K}_\mathcal{K}$, which are in turn functions of only the translationally invariant features, $\mathbf{\Phi}_k$:

$$\mathbf{w}_i^{\mathcal{A}'\to\mathcal{B}'} = \text{softmax}\left(\frac{\mathbf{K}_{\mathcal{B}'}\mathbf{q}_i^{\mathcal{A}'}}{\sqrt{d}}\right) = \text{softmax}\left(\frac{\mathbf{K}_{\mathcal{B}}\mathbf{q}_i^{\mathcal{A}}}{\sqrt{d}}\right) = \mathbf{w}_i^{\mathcal{A}\to\mathcal{B}}$$

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

$$\mathbf{v}_i^{\mathcal{A}'} = \mathbf{P}_{\mathcal{B}'}\mathbf{w}_i^{\mathcal{A}\to\mathcal{B}} = (\mathbf{P}_{\mathcal{B}} + \mathbf{t}_\beta)\mathbf{w}_i^{\mathcal{A}\to\mathcal{B}} = \sum_j \mathbf{p}_j^{\mathcal{B}} \cdot w_{i,j}^{\mathcal{A}\to\mathcal{B}} + \mathbf{t}_\beta \sum_j w_{i,j}^{\mathcal{A}\to\mathcal{B}} = \mathbf{P}_{\mathcal{B}}\mathbf{w}_i^{\mathcal{A}\to\mathcal{B}} + \mathbf{t}_\beta,$$

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

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

$$\boldsymbol{\delta}_i^{k'} = g_{\mathcal{R}_\mathcal{K}}(\boldsymbol{\phi}_i^{k'}) = g_{\mathcal{R}_\mathcal{K}}(\boldsymbol{\phi}_i^{k}) = \boldsymbol{\delta}_i^{k},$$

therefore they are also translationally invariant.

Since the virtual correspondence points are translationally equivariant, $\mathbf{v}_i^{\mathcal{A}'} = \mathbf{v}_i^{\mathcal{A}} + \mathbf{t}_\beta$ and the correspondence residuals are translationally invariant, $\boldsymbol{\delta}_i^{k'} = \boldsymbol{\delta}_i^{k}$, the final corrected virtual correspondence points, $\tilde{\mathbf{v}}_i^{\mathcal{A}'}$, are translationally equivariant, i.e. $\tilde{\mathbf{v}}_i^{\mathcal{A}'} = \mathbf{v}_i^{\mathcal{A}} + \boldsymbol{\delta}_i^{k} + \mathbf{t}_\beta$. This also holds for $\tilde{\mathbf{v}}_i^{\mathcal{B}'}$, 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 $\tilde{\mathbf{V}}_{\mathcal{B}} = \left[\tilde{\mathbf{v}}_1^{\mathcal{A}} \ldots \tilde{\mathbf{v}}_{N_{\mathcal{A}}}^{\mathcal{A}}\right]^\top$.

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

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

where $\bar{\tilde{\mathbf{v}}}_{\mathcal{A}}$ and $\bar{\mathbf{p}}_{\mathcal{A}}$ 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 $\tilde{\mathbf{V}}^{\mathcal{A}'}$ is

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

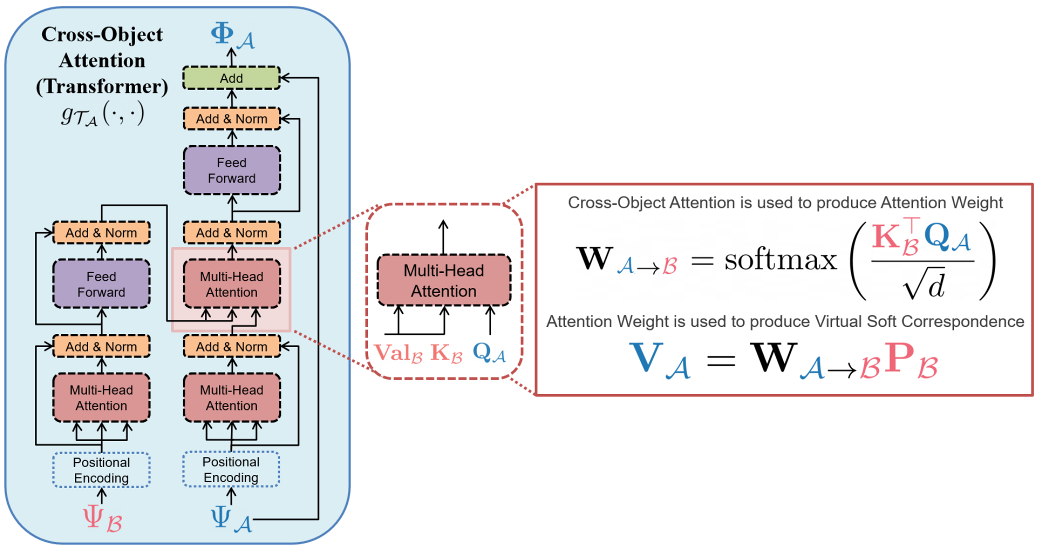
To simplify the analysis, if we assume that, for a given example, $\mathbf{R}_{\mathcal{A}\mathcal{B}} = \mathbf{I}$, then we get $\mathbf{t}_{\mathcal{A}'\mathcal{B}'} = \mathbf{t}_{\mathcal{A}\mathcal{B}} + \mathbf{t}_\beta - \mathbf{t}_\alpha$, demonstrating that the computed transformation is translation-equivariant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

C Description of Cross-Object Attention Weight Computation

To map our estimated features $\mathbf{\Psi}_{\mathcal{A}}$ and $\mathbf{\Psi}_{\mathcal{B}}$ obtained from object-specific embedding networks (DGCNN), $g_{\mathcal{A}}$ and $g_{\mathcal{B}}$ respectively, to a set of normalized weight vectors $\mathbf{W}_{\mathcal{A}\to\mathcal{B}}$ and $\mathbf{W}_{\mathcal{B}\to\mathcal{A}}$, we use the cross attention mechanism of our cross-object attention Transformer module [1]. Following Equations 5a and 5b from the paper, we can extract the desired normalized weight vector $\mathbf{w}_i^{\mathcal{A}\to\mathcal{B}}$ for the virtual corresponding point $\mathbf{v}_i^{\mathcal{A}}$ assigned to any point $\mathbf{p}_i^{\mathcal{A}} \in \mathbf{P}^{\mathcal{A}}$ using the intermediate attention embeddings of cross-object attention module as:

$$\mathbf{w}_i^{\mathcal{A}\to\mathcal{B}} = \mathrm{softmax}\left(\frac{\mathbf{K}_{\mathcal{B}}\mathbf{q}_i^{\mathcal{A}}}{\sqrt{d}}\right), \quad \mathbf{w}_i^{\mathcal{B}\to\mathcal{A}} = \mathrm{softmax}\left(\frac{\mathbf{K}_{\mathcal{A}}\mathbf{q}_i^{\mathcal{B}}}{\sqrt{d}}\right) \tag{11}$$

where $\mathbf{q}_i^{\mathcal{K}} \in \mathbf{Q}_{\mathcal{K}}$, and $\mathbf{Q}_{\mathcal{K}}, \mathbf{K}_{\mathcal{K}} \in \mathbb{R}^{\mathbf{N}_{\mathcal{K}} \times d}$ are the query and key (respectively) for object $\mathcal{K}$ associated with cross-object attention Transformer module $g_{\mathcal{T}_{\mathcal{K}}}$, as shown in Figure 8. These weights are then used to compute the virtual corresponding points $\mathbf{V}_{\mathcal{A}}$, $\mathbf{V}_{\mathcal{B}}$ using Equations 5a and 5b in the main paper.



Cross-Object Attention (Transformer)
$g_{\mathcal{T}_\mathcal{A}}(\cdot,\cdot)$
$\Phi_\mathcal{A}$
Add
Add & Norm
Feed Forward
Multi-Head Attention
Positional Encoding
$\Psi_\mathcal{B}$
$\Psi_\mathcal{A}$
$\mathbf{Val}_\mathcal{B}$ $\mathbf{K}_\mathcal{B}$ $\mathbf{Q}_\mathcal{A}$
Cross-Object Attention is used to produce Attention Weight
$\mathbf{W}_{\mathcal{A}\to\mathcal{B}} = \mathrm{softmax}\left(\frac{\mathbf{K}_\mathcal{B}^\top \mathbf{Q}_\mathcal{A}}{\sqrt{d}}\right)$
Attention Weight is used to produce Virtual Soft Correspondence
$\mathbf{V}_\mathcal{A} = \mathbf{W}_{\mathcal{A}\to\mathcal{B}}\mathbf{P}_\mathcal{B}$

Figure 8: Cross-Object attention weight computation for virtual soft correspondence $\mathbf{V}_\mathcal{A}$ from object $\mathcal{A}$ to $\mathcal{B}$. $\mathbf{Q}_\mathcal{K}, \mathbf{K}_\mathcal{K}, \mathbf{Val}_\mathcal{K} \in \mathbb{R}^{\mathbf{N}_\mathcal{K}\times d}$ are the query, key and value (respectively) for object $\mathcal{K}$ associated with cross-object attention Transformer module $g_{\mathcal{T}_\mathcal{K}}$. The Transformer block is modified from Figure 2(b) in DCP [2].

C.1 Ablation

To explore the importance of this weight computation design choice described in Equation 11, we conducted an ablation experiment on this design choice against an alternative, arguably simpler method for cross-object attention weight computation that was used in prior work [2]. Since the point 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.

To compute the virtual corresponding point $\mathbf{v}_i^\mathcal{A}$ assigned to any point $\mathbf{p}_i^\mathcal{A} \in \mathbf{P}^\mathcal{A}$, we can extract the desired normalized weight vector $\mathbf{w}_i^{\mathcal{A}\to\mathcal{B}}$ with the softmax function:

$$\mathbf{w}_i^{\mathcal{A}\to\mathcal{B}} = \mathrm{softmax}\left(\mathbf{\Phi}_\mathcal{B}^\top \phi_i^\mathcal{A}\right), \quad \mathbf{w}_i^{\mathcal{B}\to\mathcal{A}} = \mathrm{softmax}\left(\mathbf{\Phi}_\mathcal{A}^\top \phi_i^\mathcal{B}\right) \tag{12}$$

This is the approach used in the prior work of Deep Closest Point (DCP) [2]. In the experiments below, we refer to this approach as *point embedding dot-product*.

We conducted an ablation experiment on the weight computation method used in TAX-Pose (Equation 11) against the simpler approach from DCP [2] (Equation 12), on the upright mug hanging task in simulation. The models are trained from 10 demonstrations and tested on 100 trials over the test mug set. As seen in Table 5, the TAX-Pose approach (Equation 11) outperforms *point embedding dot-product* (Equation 12) in all three evaluation categories on *grasp*, *place*, and *overall* in terms of test success rate.

Attention Weight Ablation	Grasp	Place	Overall
Point Embedding Dot-Product (Eqn. 12)	0.83	0.92	0.92
TAX-Pose (Ours) (Eqn. 11)	**0.99**	**0.97**	**0.96**

Table 5: Test success rate (↑) over 100 trials for mug hanging upright task, ablated on attention weight computation methods.

D Description of Weighted SVD

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

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

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

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

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

$$\mathbf{U\Sigma V}^{\top} = \text{svd}(\mathbf{A\Gamma B}^{\top}) \quad (13a) \qquad\qquad \mathbf{R}_{\mathcal{AB}} = \mathbf{U\Sigma}_*\mathbf{V}^{\top} \quad (13b)$$

where $\mathbf{\Sigma}_* = \text{diag}(\begin{bmatrix}1, 1, ...\text{det}(\mathbf{UV}^{\top})\end{bmatrix}$ and svd is a differentiable SVD operation [3].

The optimal translation can be computed as:

$$\mathbf{t}_{\mathcal{A}} = \bar{\tilde{\mathbf{v}}}_{\mathcal{B}} - \mathbf{R}_{\mathcal{AB}}\bar{\mathbf{p}}_{\mathcal{A}} \qquad \mathbf{t}_{\mathcal{B}} = \bar{\mathbf{p}}_{\mathcal{B}} - \mathbf{R}_{\mathcal{AB}}\bar{\tilde{\mathbf{v}}}_{\mathcal{A}} \qquad \mathbf{t} = \frac{N_{\mathcal{A}}}{N}\mathbf{t}_{\mathcal{A}} + \frac{N_{\mathcal{B}}}{N}\mathbf{t}_{\mathcal{B}} \quad (14a)$$

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:

$$\mathbf{T}_{\mathcal{AB}} = \begin{bmatrix}\mathbf{R}_{\mathcal{AB}} & \mathbf{t} \\ 0 & 1\end{bmatrix} \quad (15)$$

E Training Details

E.1 Supervision

To train the encoders $g_{\mathcal{A}}(\bar{\mathbf{P}}_{\mathcal{A}})$, $g_{\mathcal{B}}(\bar{\mathbf{P}}_{\mathcal{B}})$ as well as the residual networks $g_{\mathcal{R}_{\mathcal{A}}}\left(\phi_i^{\mathcal{A}}\right)$, $g_{\mathcal{R}_{\mathcal{B}}}\left(\phi_i^{\mathcal{B}}\right)$, 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 [4, 5]: 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.

$$\mathcal{L}_{\text{disp}} = \left\|\mathbf{T}_{\mathcal{AB}}\mathbf{P}_{\mathcal{A}} - \mathbf{T}_{\mathcal{AB}}^{GT}\mathbf{P}_{\mathcal{A}}\right\|^2 + \left\|\mathbf{T}_{\mathcal{AB}}^{-1}\mathbf{P}_{\mathcal{B}} - \mathbf{T}_{\mathcal{AB}}^{GT-1}\mathbf{P}_{\mathcal{B}}\right\|^2 \quad (16)$$

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

$$\mathcal{L}_{\text{corr}} = \left\|\tilde{\mathbf{V}}_{\mathcal{A}} - \mathbf{T}_{\mathcal{AB}}^{GT}\mathbf{P}_{\mathcal{A}}\right\|^2 + \left\|\tilde{\mathbf{V}}_{\mathcal{B}} - \mathbf{T}_{\mathcal{AB}}^{GT-1}\mathbf{P}_{\mathcal{B}}\right\|^2. \quad (17)$$

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

$$\mathcal{L}_{\text{cons}} = \left\| \tilde{\mathbf{V}}_{\mathcal{A}} - \mathbf{T}_{\mathcal{AB}} \mathbf{P}_{\mathcal{A}} \right\|^2 + \left\| \tilde{\mathbf{V}}_{\mathcal{B}} - \mathbf{T}_{\mathcal{AB}}^{-1} \mathbf{P}_{\mathcal{B}} \right\|^2 . \tag{18}$$

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 [2], which consists of DGCNN [6] and a Transformer [1].

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 [7] with a geometric distance weighting and random transformations, to learn $SE(3)$ invariant embeddings, see Appendix E.2 for details.

E.2 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 [8] 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 [9]; 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 [7] 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 $\mathbf{P}_{\mathcal{A}}$ 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 $\Psi_{\mathcal{A}} = g_{\mathcal{A}}(\mathbf{T}\mathbf{P}_{\mathcal{A}})$, where $\psi_i^{\mathcal{A}} \in \Psi_{\mathcal{A}}$ is the d-dimensional vector for the ith point $p_i^{\mathcal{A}} \in \mathbf{P}_{\mathcal{A}}$.

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

$$\mathcal{L}_{wc} := -\sum_i \log \left[\frac{\exp\left(\phi_i^\top \psi_i\right)}{\sum_j \exp\left(d_{ij}\left(\phi_i^\top \psi_j\right)\right)} \right] \tag{19}$$

$$d_{ij} := \begin{cases} \frac{1}{\mu} \tanh\left(\lambda \| p_i^{\mathcal{A}} - p_j^{\mathcal{A}} \|_2\right), & \text{if } i \neq j \\ 1, & \text{otherwise} \end{cases} \tag{20}$$

$$\mu := \max\left(\tanh\left(\lambda \| p_i^{\mathcal{A}} - p_j^{\mathcal{A}} \|_2\right)\right) \tag{21}$$

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

E.3 Architectural Variants

Goal-Conditioned TAX-Pose: To enable a single TAX-Pose model to scale to multiple related placement sub-tasks for a pair of action and anchor objects, we design a **goal-conditioned** variant (**TAX-Pose GC**), which receives a one-hot encoding of the desired semantic goal position (e.g. 'top', 'left', ...) for the task. This contextual encoding is incorporated into each DGCNN module in the same way as proposed in the original DGCNN paper. This encoding 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.

Vector Neurons: We briefly experimented with Vector Neurons [10] and found that this led to worse performance on this task.

F Additional Results

F.1 NDF Placement Tasks

F.1.1 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 6 for full ablation results along six ablated dimensions as detailed below. We also performed an ablation experiment on alternative cross-object attention weight computation, as explained in Appendix C and results can be found in Table 5. For consistency, all ablated models are trained for 15K 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

   $$\mathcal{L}_{\text{net}} = \mathcal{L}_{\text{disp}} + \lambda_1 \mathcal{L}_{\text{cons}} + \lambda_2 \mathcal{L}_{\text{corr}} \tag{22}$$

   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 6. 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 6: Mug Hanging Ablations Results

F.1.2 Effects of Pretraining on Mug Hanging Task

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 E.2, 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 7, 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 2 times longer (26K steps compared to 15K 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 for each object category is done in a task-agnostic 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**
No Pre-training for Embedding Network (trained for 15K steps)	**0.99**	0.72	0.71	15.39	**0.01**
TAX-Pose (Ours) (trained for 15K steps)	**0.99**	**0.97**	**0.96**	**4.33**	**0.01**

Table 7: 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).

F.1.3 Additional Simulation Experiments on Bowl and Bottle Placement Task

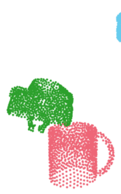
Object	**Algorithm**	**Grasp**	**Place**	**Overall**	**Grasp**	**Place**	**Overall**
		Upright Pose			**Arbitrary Pose**		
Mug	DON [11]	0.91	0.50	0.45	0.35	0.45	0.17
	NDF [12]	0.96	0.92	0.88	**0.78**	0.75	0.58
	TAX-Pose (Ours)	**0.99**	**0.97**	**0.96**	0.75	**0.84**	**0.63**
Bowl	DON [11]	0.50	0.35	0.11	0.08	0.20	0
	NDF [12]	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 [11]	0.79	0.24	0.24	0.05	0.02	0.01
	NDF [12]	**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 8: 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.

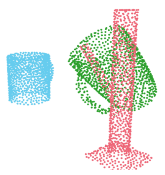
Additional results on *Grasp*, *Place* and *Overall* success rate in simulation for **Bowl** and **Bottle** are shown in Table 8. For bottle and bowl experiment, we follow the same experimentation setup as in [12], where the successful *grasp* is considered if a stable grasp of the object is obtained, 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.

F.1.4 Failure Cases

Some failure cases for TAX-Pose occur 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 9a. A common failure mode for the mug placement subtask is characterized by an erroneous transform prediction that results in the mug's handle completely missing the rack hanger, thus resulting in placement failure, as seen in Figure 9b.



(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 9: 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.

F.2 PartNet-Mobility Tasks

F.2.1 Expanded Results Tables

In the main text, we presented aggregated results of the performance of each method by averaging the quantitative metrics for each sub-task for each object ("In", "On", "Left", and "Right" in simulation and "In", "On" and "Left" in real-world), and then averaged across object classes to arrive at a single metric per method. Here, we present the per-class breakdown of performance. See Table 9 for simulated results, and Table 10 for real-world results.

		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 [13]	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 [14]	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 [14]	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	**4.94**	**0.16**	**6.18**	0.16	**1.75**	**0.10**	**2.94**	**0.10**	**3.02**	**0.06**	**10.15**	0.27	6.93	0.35	**3.76**	**0.11**	**4.76**	**0.11**

Table 9: 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.

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

We further provide results per-sub-task in simulation. For each category of anchor objects, sub-tasks may or may not all be well-defined. For example, the doors of safes might occlude the action object completely in a demonstration for "Left" and "Right" tasks due to the handedness of the door; and a table's height might be too tall for the camera to see the action object placed during the "Top" task. To avoid these ill-defined cases, we omit object-category / sub-task pairings which cannot be consistently defined from training and evaluation. We show visualizations of each defined task for each object category in Figure 10. Results for each sub-task can be found in Tables 11[4], 12, 13, and 14 respectively.

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

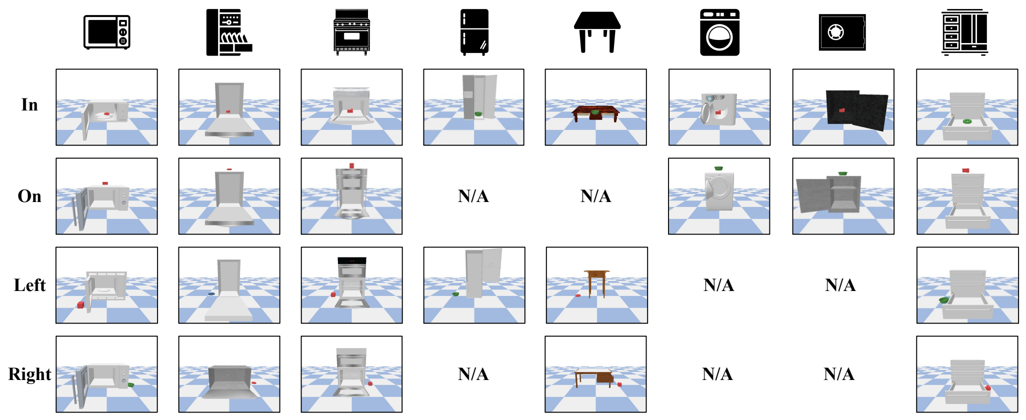


Figure 10: A visualization of all categories of anchor objects and associated semantic tasks, with action objects in ground-truth TAX-Poses used in simulation training.

		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 [13]	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 [14]	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 [14]	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 11: 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 [13]	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 [14]	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 [14]	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 12: 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 [13]	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 [14]	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 [14]	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 13: Goal Inference Rotational and Translational Error Results (↓) for the "**Left**" 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.											
		$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$	$\mathcal{E}_\mathbf{R}$	$\mathcal{E}_\mathbf{t}$
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 [13]	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 [14]	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 [14]	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 14: Goal Inference Rotational and Translational Error Results (↓) for the "**Right**" 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.

F.2.2 Goal-Conditioned Variant

We train our goal-conditioned variant **TAX-Pose GC** (Appendix E.3) to predict the correct cross-pose across sub-tasks, incorporating a one-hot encoding of each sub-task (i.e. 'top', 'in', 'left', 'right') so the model can infer the desired semantic goal location. Importantly, as with the task-specific model (**TAX-Pose**), the **TAX-Pose GC** model is trained across **all PartNet-Mobility object categories**. We report the performance of the variants in Table 9.

F.2.3 Failure Cases

Some failure cases for TAX-Pose occur when the predicted cross-pose does not respect the physical constraints in the scene. For example, as seen in Fig. 11. TAX-Pose would fail when the prediction violates the physical constraints of the objects, which in this case the action object collides with the anchor object. In the real world, this would yield the robot unable to plan a correct path.

(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 11: 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.

G Task Details

G.1 NDF Task Details

In this section, we describe the Mug Hanging task of the NDF Tasks and experiments in detail. The Mug Hanging task is consisted of two sub tasks: *grasp* and *place*. A *grasp* success is achieved when the mug is grasped stably by the gripper, while a *place* success is achieved when the mug is hung stably on the hanger of the rack. Overall mug hanging success is achieved when the predicted transforms enable both grasp and place success for the same trial. See Figure 12 for a detailed breakdown of the mug hanging task in stages.

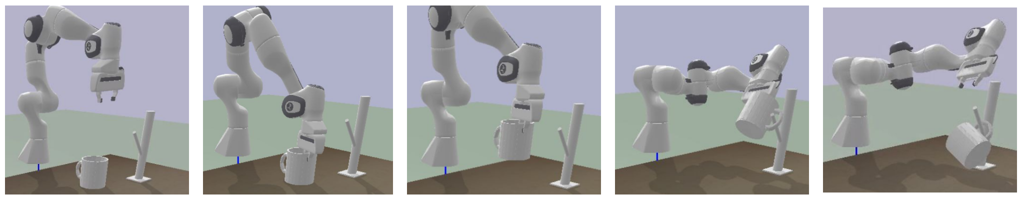


Figure 12: Visualization of Mug Hanging Task (Upright Pose). Mug hanging task is consisted of two stages, given a mug that is randomly initialized on the table, the model first predicts a $SE(3)$ transform from gripper end effector to the mug rim $\mathbf{T}_{g\to m}$, then grasp it by the rim. Next, 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.

G.1.1 Baseline Description

In simulation, we compare our method to the results described in [12].

- **Dense Object Nets (DON)** [11]: 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 [12].
- **Neural Descriptor Field (NDF)** [12]: 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.

G.1.2 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 [12], where the data are generated via teleportation in PyBullet, collected on 10 mug instances with random pose initialization.

G.1.3 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 models 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.

G.1.4 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.

G.1.5 Real-World Experiments

We pre-train the DGCNN embedding network with rotation-equivariant loss on ShapeNet mugs' simulated point clouds in simulation. Using the pre-trained embedding, we then train the full TAX-Pose model with the 10 collected real-world point clouds.

G.2 PartNet-Mobility Object Placement Task Details

In this section, we describe the PartNet-Mobility Object Placement experiments in detail. We select a set of household furniture objects from the PartNet-Mobility dataset as the anchor objects, and a set of small rigid objects released with the Ravens simulation environment 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.

G.2.1 Dataset Preparation

Simulation Setup. We leverage the PartNet-Mobility dataset [15] 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 [16]. 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.

G.2.2 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 [17, 18] 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 Figure 11 for an illustration of TAX-Pose predictions that fail to meet this criterion.

G.2.3 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 [19] 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

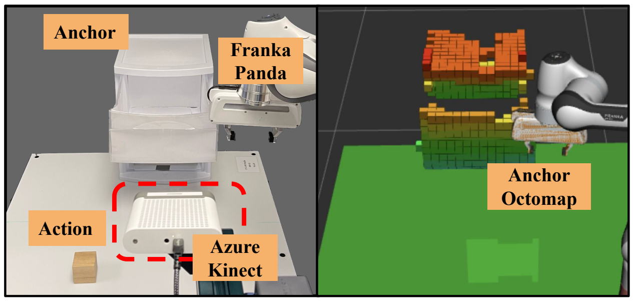


Figure 13: Real-world experiments illustration. **Left:** work-space setup for physical experiments. **Center:** Octomap visualization of the perceived anchor object.

marching cubes [20], 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 [21] 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 [22] 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 [14] 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 [23] 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 [24] using the cleaned anchor object point cloud. In this way, the robot is able to avoid collision with the anchor object as well. See Figure 13 for the workspace.

G.2.4 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). Our policy is represented as a PointNet++ network 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 final achieved pose of the action object at the terminal state is used for computing the evaluation metrics.
- **E2E DAgger**: Using the same BC dataset and the same PointNet++ architecture as above, we train a policy that outputs the same transformation representation as in BC using DAg-

ger [13]. The final achieved pose of the action object at the terminal state is used for computing the evaluation metrics.

- **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 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 point cloud and the point cloud transformed by adding per-point flow vectors using SVD, yielding the next pose. The final achieved pose of the action object at the terminal state is used for computing the evaluation metrics.
- **Goal Flow**: Instead of training a multi-step policy to reach the goal, we 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 final achieved pose of the action object at the terminal state is used for computing the evaluation metrics.

H Author Contributions

Chuer Pan designed and implemented the cross-correspondence TAX-Pose model, correspondence residuals, and the bi-directional weighted SVD method. She also designed, implemented, and conducted the simulation experiments of the NDF tasks as well as the study on varying the number of demos and the various ablation experiments. She also designed, implemented, and trained the models for mug hanging task in the real-world.

Brian Okorn proposed, designed, and implemented the cross-correspondence model, correspondence residuals, and the bi-directional weighted SVD method, as well as analyzed the invariances of the current framework.

Harry Zhang wrote early prototypes of TAX-Pose using residual flows without cross-attention, which later became a baseline of TAX-Pose. He also wrote the infrastructure of the PartNet-Mobility object placement task's simulated data collection and training. He also wrote all the baselines in the PartNet-Mobility placement task. He designed and conducted all real-world robot experiments for both tasks in the real-world.

Ben Eisner designed the PartNet-Mobility tasks, implemented the Pybullet physics environment required for their rendering and simulation, and conducted training and evaluation of TAX-Pose on the PartNet-Mobility dataset. He also designed and evaluated the goal-conditioned variant of TAX-Pose.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

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

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

[19] 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.

[20] 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.

[21] 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.

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

[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] 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.