A Method Details

We included the code for both our simulated and real-world experiments for reference. Please find it in the supplementary material under `iw_code`. Algorithms 1 and 2 describe our warp learning and inference.

Algorithm 1 Warp Learning

Input: Meshes of K example object instances $\{\text{obj}_1, \text{obj}_2, ..., \text{obj}_K\}$.
Output: Canonical point cloud, vertices and faces and a latent space of warps.
Parameters: Smoothness of CPD warping α and number of PCA components L.

$\text{PCD} = \left\langle \text{SampleS}(\text{obj}_i) \right\rangle_{i=1}^{K}$. ▷ Sample a small point cloud per object (Appendix A.1).
$C = \text{SelectCanonical}(\text{PCD})$. ▷ Select a canonical object with index C (Appendix A.2).
$\text{canon} = \text{Concat}(\text{obj}_C.\text{vertices}, \text{SampleL}(\text{obj}_C))$. ▷ Use both vertices and surface samples.
for $i \in \{1, 2, ..., K\}, i \neq C$ **do**
    $W_{C \to i} = \text{CPD}(\text{canon}, \text{PCD}_i, \alpha)$. ▷ Coherent Point Drift warping (Section 3).
end for
$D_W = \{\text{Flatten}(W_{C \to i})\}_{i=1, i \neq C}^{K}$. ▷ Dataset of displacements of canon.
$\text{PCA} = \text{FitPCA}(D_W, \text{n\_components} = L)$. ▷ Learn a latent space of canonical object warps.
return $\text{Canon}(\text{points} = \text{canon}, \text{vertices} = \text{obj}_C.\text{vertices}, \text{faces} = \text{obj}_C.\text{faces}), \text{PCA}$.

Algorithm 2 Warp Inference and Mesh Reconstruction

Input: Observed point cloud pcd, canonical object canon and latent space PCA.
Output: Predicted latent shape v and pose T.
Parameters: Number of random starts S, number of gradient descent steps T, learning rate η and object size regularization β.

$t_g = \frac{1}{|\text{pcd}|} \sum_{i=1}^{|\text{pcd}|} \text{pcd}_i$.
$\text{pcd} = \text{pcd} - t_g$. ▷ Center the point cloud.
for $i = 1$ **to** S **do**
    $R_{\text{init}} =$ Random initial 3D rotation matrix.
    Initialize $v = \begin{pmatrix} 0 & 0 & ... & 0 \end{pmatrix}, s = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, t_l = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}, \hat{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$.
    Initialize Adam [36] with parameters v, s, t_l, r and learning rate η.
    for $j = 1$ **to** T **do**
        $\delta = \text{Reshape}(Wv)$.
        $X = \text{canon.points} + \delta$. ▷ Warped canonical point cloud.
        $R = \text{GramSchmidt}(\hat{\text{R}})$.
        $X = (X \odot s) R_{\text{init}}^T R^T + t_l$. ▷ Scaled, rotated and translated point cloud.
        $\mathcal{L} = \frac{1}{|\text{pcd}|} \sum_k^{|\text{pcd}|} \min_l^{|X|} \|\text{pcd}_k - X_l\|_2^2$. ▷ One-sided Chamfer distance.
        $\mathcal{L} = \mathcal{L} + \beta \max_l^{|X|} \|X_l\|_2^2$. ▷ Object size regularization.
        Take a gradient descent step to minimize $\mathcal{L}$ using Adam.
    end for
end for
Find parameters $v^*, s^*, t_l^*, R_{\text{init}}^*, R^*$ with the lowest final loss across $i \in \{1, 2, ..., S\}$.
$X = \text{canon.points} + \text{Reshape}(Wv^*)$.
$X = (X \odot s^*)(R_{\text{init}}^*)^T (R^*)^T + t_l^* + t_g$. ▷ Complete point cloud in workspace coordinates.
$\text{vertices} = \langle X_1, X_2, ..., X_{|\text{canon.vertices}|} \rangle$. ▷ First $|\text{canon.vertices}|$ points of X are vertices.
return $\text{Mesh}(\text{vertices} = \text{vertices}, \text{faces} = \text{canon.faces})$. ▷ Warped mesh.

A.1 Point Cloud Sampling

We use `trimesh`[1] to sample the surface of object meshes. The function `trimesh.sample.sample_surface_even` samples a specified number of points and then rejects points that are too close together. We sample 2k points for small point clouds (SampleS) and 10k point for large point clouds (SampleL).

A.2 Canonical Object Selection

Among the K example objects, we would like to find the one that is the easiest to warp to the other objects. For example, if we have ten examples of mugs, but only one mug has a square handle, we should not choose it as it might be difficult to warp it to conform to the round handles of the other nine mugs. We use Algorithm 3, which computes $K * K - 1$ warps and picks the object that warps to the other $K - 1$ objects with the lowest Chamfer distance. We also note an alternative and computationally cheaper algorithm from Thompson et al. [8], Algorithm 4. This algorithm simply finds the object that is the most similar to the other $K - 1$ objects without any warping.

Algorithm 3 Exhaustive Canonical Object Selection

Input: Point clouds of K training objects $\langle \mathrm{X}^{(1)}, \mathrm{X}^{(2)}, ..., \mathrm{X}^{(K)} \rangle$.
Output: Index of the canonical object.

for $i = 1$ **to** K **do**
  for $j = 1$ **to** $K, j \neq i$ **do**
    $W_{i \to j} = \mathrm{CPD}(\mathrm{X}^{(i)}, \mathrm{X}^{(j)})$ ▷ Warp point cloud i to point cloud j.
    $C_{i,j} = \frac{1}{|\mathrm{X}^{(j)}|} \sum_{k=1}^{|\mathrm{X}^{(j)}|} \min_{l=1}^{|\mathrm{X}^{(i)}|} \left\| \mathrm{X}_k^{(j)} - (\mathrm{X}^{(i)} + W_{i \to j})_l \right\|_2^2$
  end for
end for
for $i = 1$ **to** K **do**
  $C_i = \sum_{j=1, j \neq i}^{K} C_{i,j}$ ▷ Cumulative cost of point cloud i warps.
end for
return $\arg\min_{i=1}^{K} C_i$ ▷ Pick point cloud that is the easiest to warp.

Algorithm 4 Approximate Canonical Object Selection [8]

Input: Point clouds of K training objects $\langle \mathrm{X}^{(1)}, \mathrm{X}^{(2)}, ..., \mathrm{X}^{(K)} \rangle$.
Output: Index of the canonical object.

for $i = 1$ **to** K **do**
  for $j = 1$ **to** $K, j \neq i$ **do**
    $C_{i,j} = \frac{1}{|\mathrm{X}^{(j)}|} \sum_{k=1}^{|\mathrm{X}^{(j)}|} \min_{l=1}^{|\mathrm{X}^{(i)}|} \left\| \mathrm{X}_k^{(j)} - \mathrm{X}_l^{(i)} \right\|_2^2$
  end for
end for
for $i = 1$ **to** K **do**
  $C_i = \sum_{j=1, j \neq i}^{K} C_{i,j}$
end for
return $\arg\min_{i=1}^{K} C_i$

A.3 Gram-Schmidt Orthogonalization

We compute a rotation matrix from two 3D vectors using Algorithm 5 [38].

[1] https://github.com/mikedh/trimesh

Algorithm 5 Gram-Schmidt Orthogonalization

Input: 3D vectors u and v.
Output: Rotation matrix.

$u' = u/\|u\|$
$v' = \frac{v-(u'\cdot v)u'}{\|v-(u'\cdot v)u'\|}$
$w' = u' \times v'$
return $\mathrm{Stack}(u', v', w')$

A.4 Shape and Pose Inference Details

The point clouds $Y \in \mathrm{R}^{n\times 3}$ starts in its canonical form with the latent shape v equal to zero. We set the initial scale s to one, translation t to zero and rotation $\hat{R}$ to identity,

$$v = \underbrace{\begin{pmatrix}0 & 0 & ... & 0\end{pmatrix}}_{d}, \quad s = \begin{pmatrix}1 & 1 & 1\end{pmatrix}, \quad t = \begin{pmatrix}0 & 0 & 0\end{pmatrix}, \quad \hat{R} = \begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{pmatrix}. \tag{9}$$

$\hat{R}$ is then transformed into $R \in \mathrm{SO}(3)$ using Algorithm 5. We minimize $\mathcal{L}$ with respect to v, s, t and $\hat{R}$ using the Adam optimizer [36] with learning rate 10^{-2} for 100 steps. We set $\beta = 10^{-2}$. We found the optimization process is prone to getting stuck in local minima; e.g., instead of aligning the handle of the decoded mug with the observed point cloud, the optimizer might change the shape of the decoded mug to hide its handle. Hence, we restart the process with many different random initial rotations and pick the solution with the lowest loss function. Further, we randomly subsample Y to 1k points at each gradient descent step – this allows us to run 12 random starting orientations at once on an NVIDIA RTX 2080Ti GPU.

A.5 Using Multiple Demonstrations

Our method transfers grasps and placements from a single demonstration, but in our simulated experiment, we have access to multiple demonstrations. We implement a simple heuristic for choosing the demonstration that fits our method the best: we make a prediction of the relational object placement from the initial state of each demonstration and select the demonstration where our prediction is closest to the demonstrated placement. The intuition is that we are choosing the demonstration where our method was able to warp the objects with the highest accuracy (leading to the best placement prediction). This is especially useful in filtering out demonstrations with strangely shaped objects.

B Experiment Details

B.1 Object re-arrangement on a physical robot

We use a UR5 robotic arm with a Robotiq gripper. We capture the point cloud using three RealSense D455 camera with extrinsics calibrated to the robot. For motion planning, we use MoveIt with ROS1. To segment the objects, we use DBSCAN to cluster the point clouds and simple heuristics (e.g. height, width) to detect the object class.

B.2 Grasp prediction in the wild

We use a single RealSense D435 RGB-D camera. Our goal is to be able to demonstrate any task in the real world without having to re-train our perception pipeline. Therefore, we chose an open-vocabulary object detection model Detic [42], which is able to detect object based on natural language descriptions. We used the following classes: "cup", "bowl", "mug", "bottle", "cardboard", "box", "Tripod", "Baseball bat", "Lamp", "Mug Rack", "Plate", "Toaster" and "Spoon". We use

(a) (b) (c)

Figure 8: Objects used for the real-world tasks: (a) mug on tree, (b) bowl (or plate) on mug and (c) bottle in box. We use a single pair of objects to generate demonstrations and test on novel objects.

the predicted bounding boxes from Detic to condition a Segment Anything model [43] to get accurate class-agnostic segmentation masks. Both Detic[2] and Segment Anything[3] come with several pre-trained models and we used the largest available. Finally, we select the pixels within each segmentation mask and use the depth information from our depth camera to create a per-object point cloud. We use DBSCAN to clouster the point cloud and filter out outlier points. Then, we perform mesh warping and interaction warping to predict object meshes and grasps.

Previously, we experimented with Mask R-CNN [44] and Mask2Former [45] trained on standard segmentation datasets, such as COCO [46] and ADE20k [47]. We found that these dataset lack the wide range of object classes we would see in a household environment and that the trained models struggle with out-of-distribution viewing angles, such as looking from a steep top-down angle. We also experimented with an open-vocabulary object detection model OWL-ViT [48] and found it to be sensitive to scene clutter and the viewing angle.

C Additional Results

Training and inference times: We measure the training and inference times of TAX-Pose, R-NDF and IW (Table 3). Both R-NDF and IW take tens of seconds to either perceive the environment or to predict an action. This is because both of these methods use gradient descent with many random restarts for inference. On the other hand, TAX-Pose performs inference in a fraction of second but requires around 16 hours of training for each task. Neither R-NDF nor IW require task-specific training. We do not include the time it takes to perform pre-training for each class of objects, which is required by all three methods, because we used checkpoints provided by the authors of TAX-Pose and R-NDF.

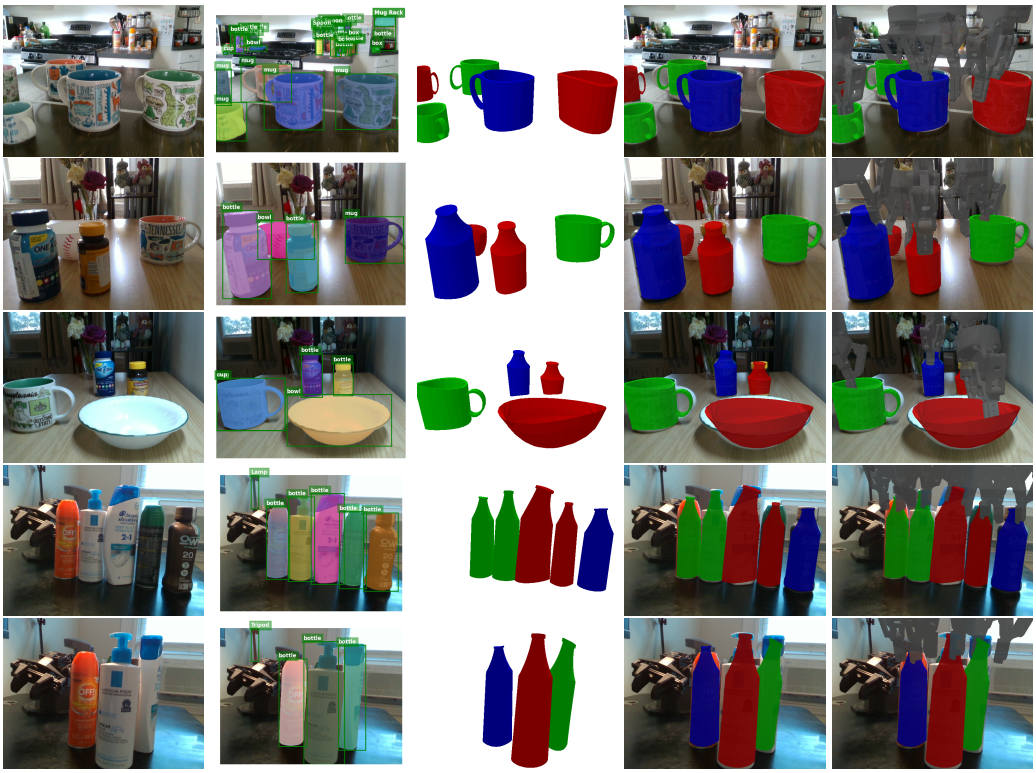
Additional real-world grasp predictions: We include additional examples of real-world object segmentation, mesh prediction and grasp prediction in Figure 9.

[2] https://github.com/facebookresearch/Detic

[3] https://github.com/facebookresearch/segment-anything

Method	Training	Perception	Grasp prediction	Placement prediction
TAX-Pose [2]	16.5 ± 1.3 h	-	0.02 ± 0.01 s	0.02 ± 0.01 s
R-NDF [17]	-	-	21.4 ± 0.5 s	42.5 ± 1.8 s
IW (Ours)	-	29.6 ± 0.2 s	0.01 ± 0.01 s	0.003 ± 0.004 s

Table 3: Approximate training and inference times for our method and baselines measured over five trials. R-NDF and IW do not have an explicit training phase, as they use demonstrations nonparametrically during inference. Only IW has a perception step that is separate from the action prediction step. We do not include the time it takes to capture a point cloud or to move the robot. Training and inference times were measured on a system with a single NVIDIA RTX 2080Ti GPU and an Intel i7-9700K CPU.



(a) (b) (c) (d) (e)

Figure 9: Additional examples, please see Figure 7.

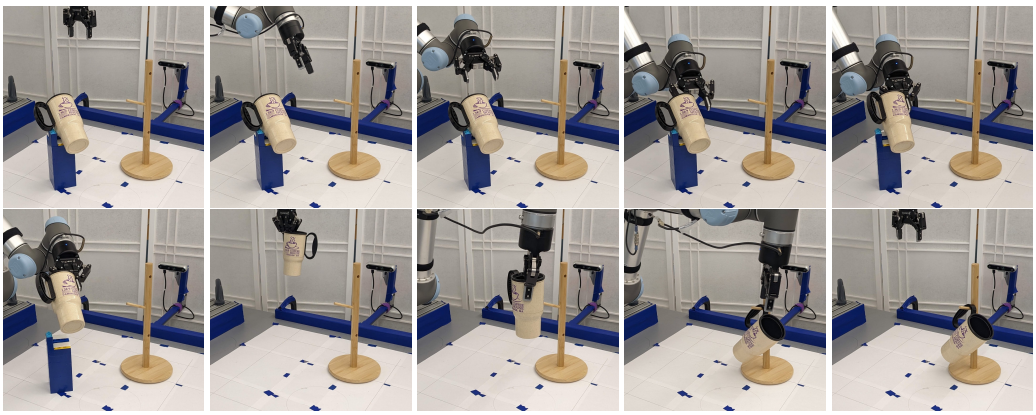


Figure 10: Example of mug on tree episode.

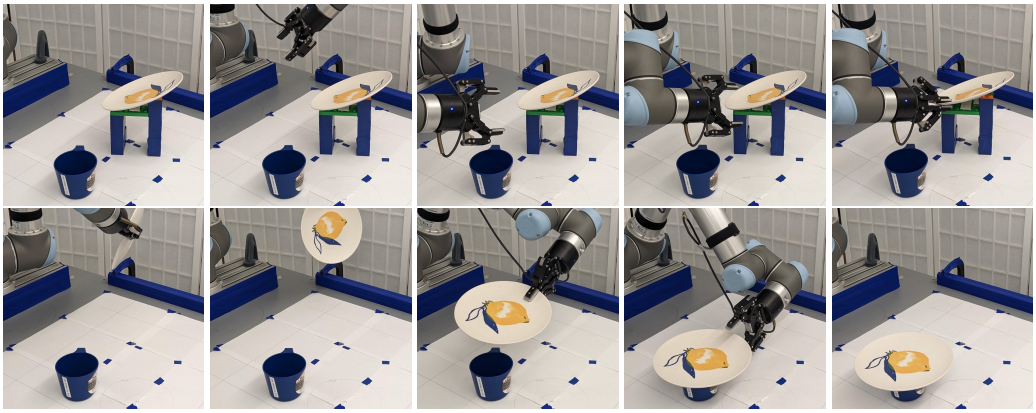


Figure 11: Example of bowl/plate on mug episode.

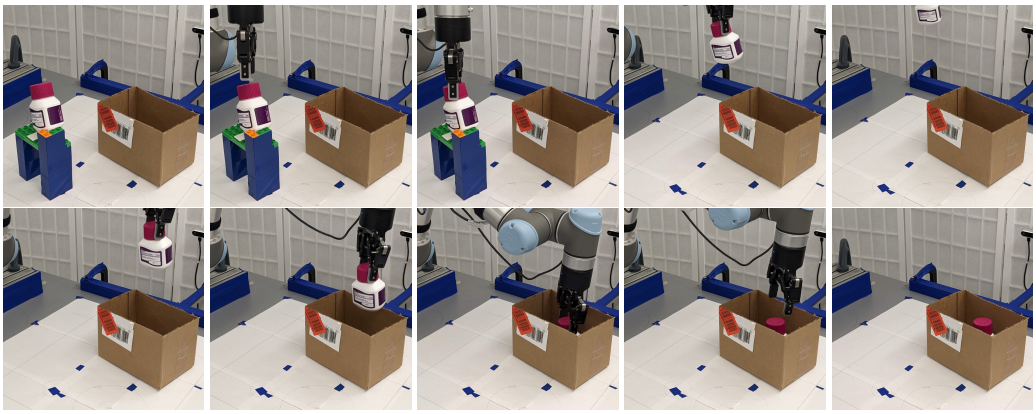


Figure 12: Example of bottle in box episode.