SE(3)-Equivariant Relational Rearrangement with Neural Descriptor Fields: Supplementary Material

In Section A1, we present details on data generation, model architecture, and training for NDFs. In Section A2 we detail the optimization method used to recover the pose of a local coordiante frame by minimizing descriptor distances (as in Equations (4), (5), and (6)). Section A3 describes the procedure for training the energy-based models used in relative transformation refinement. In Section A4, we describe more details about our experimental setup, Section A5 discusses more details on the evaluation tasks and robot execution pipelines, and Section A6 describes our alternating minimization method for aligning descriptors across a set of demonstrations. In section A7 we present an additional set of qualitative results showing how R-NDF can be used to handle collision avoidance and additional problem constraints and more complex rearrangement tasks, and in Section A8 we discuss applying R-NDF to relational rearrangement with more than two objects. Section A9 shows an example of the framework operating with partial point clouds and Section A10 contains additional visualizations of the tasks and objects used in the evaluation. Finally, in Section A11, we provide more thorough implementation details and an expansion on the limitations of the proposed approach.

A1 NDF Training

In this section, we present details on the data used for training NDFs, the neural network architectures we used in the NDF implementation, and model training.

A1.1 Training Data Generation

3D shape data for training NDFs. NDFs are trained to perform category-level 3D reconstruction from point cloud inputs. We supervise this training using ground truth 3D shape data obtained from synthetic 3D object models. The three tasks we consider include objects from five categories: mugs, bowls, bottles, racks, and containers. Our NDF training thus begins with obtaining a dataset of varying 3D models for a diverse set of object instances from each of these categories. We use ShapeNet [1] for the mugs, bowls, and bottles, and procedurally generate our own dataset of `.obj` files for the racks and containers. See Figure A1 for representative samples of the 3D models from each category.

NDFs based on regressing occupancy vs. signed-distance. Given an object dataset of 3D models, we generate a dataset of inputs and outputs for training the neural networks used in NDFs. While in [2] the underlying NDF decoder is trained to perform 3D reconstruction by representing an occupancy field [3], i.e., as an Occupancy Network (ONet) that predicts whether a point is inside or outside of a shape, we find performance improves by instead training the model to regress a signed-distance field (SDF) [4], i.e., as a DeepSDF that predicts the minimum distance from a point to the boundary of a shape and assigns negative/positive values for points inside/outside of the shape. Details on our pipeline for generating the data used to train the SDF decoder can be found in the next subsection. The following paragraphs discuss two reasons we hypothesize for the performance gap between occupancy field and signed-distance field training.

First, a signed-distance field (whose zero level set represents the boundary of a 3D shape) contains information about the underlying object geometry even at query points that are *far away* from the surface of the object, whereas the underlying occupancy field is flat everywhere except exactly at the crossing between the inside and outside of the shape. This feature of an SDF increases the likelihood of *unique* descriptors at different coordinates (e.g., $f(\mathbf{x}_i|\mathbf{P})$ and $f(\mathbf{x}_j|\mathbf{P})$). These factors appear

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

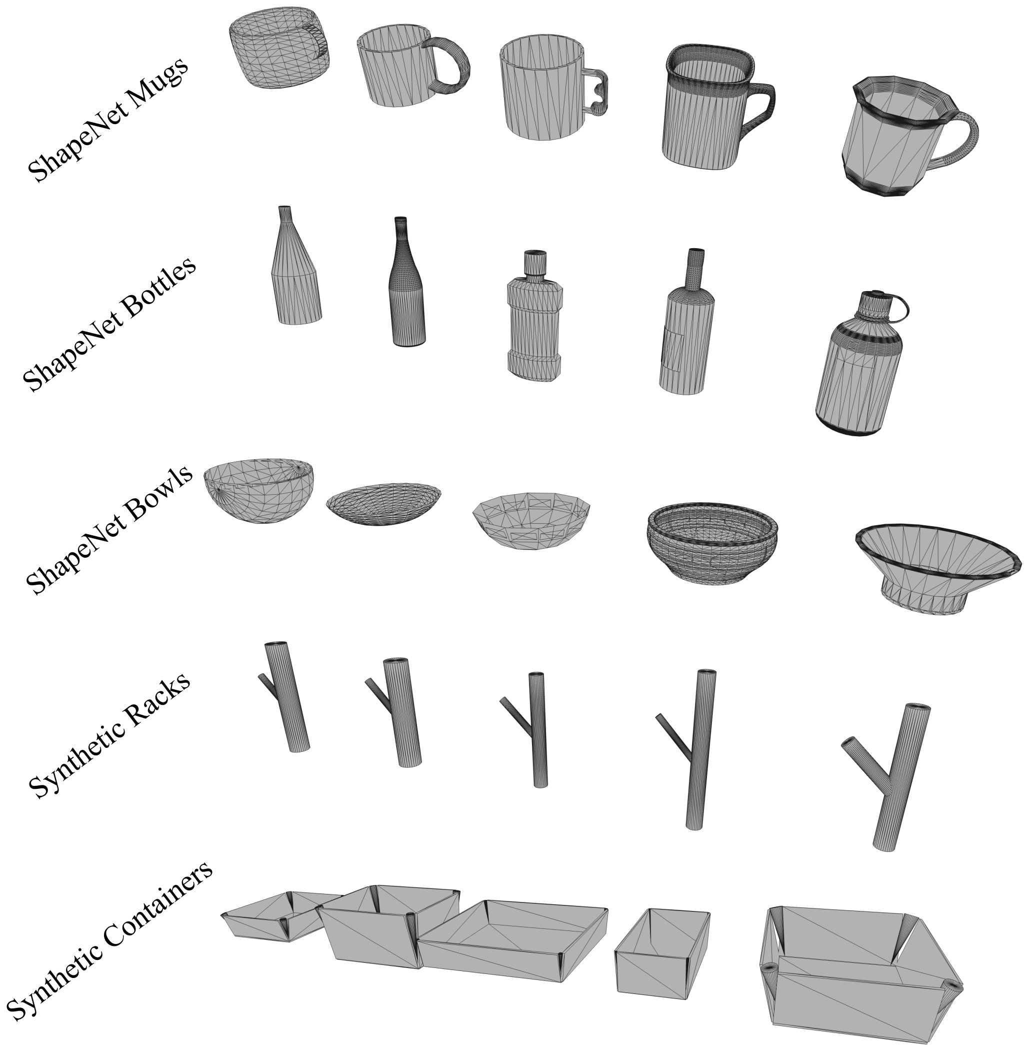


Figure A1: Example 3D models used to train NDFs and deploy NDFs on our rearrangement tasks. Mugs, bottles, and bowls are from ShapeNet [1] while we procedurally generated our own synthetic racks and box-shaped containers.

to shape the optimization landscape in Equations (4), (5), and (6) to enable smoother and more consistent convergence, along with less sensitivity to poor initialization.

Second, in addition to an (empirically observed) improvement in global optimization convergence, we also observe the SDF-based decoder leads to the optimization performing much better *near* the surface of the shape. The intuition is that an occupancy field has a sharp discontinuity at the boundary of the shape. When optimizing a query point set pose using Equation (4)-(6), the gradients become steep when the optimization reaches the region near the input point cloud. Some of the points in $\mathcal{X}$ end up inside the shape and get stuck. In contrast, although it may still have high frequency fluctuations near the shape boundary, an SDF varies much less rapidly near the surface, and we observe a corresponding reduction in local minima when running the NDF optimization using the SDF model.

Data generation. Based on the empirical observations discussed above, we convert each 3D model dataset into a corresponding dataset of input/output pairs for signed-distance function regression. For each shape, we normalize the object to a unit bounding box and use the 3D model to generate the distance from M query points to the boundary of the shape, where points inside and outside the shape are labeled with negative and positive sign, respectively. We use an open-source tool[*] for generating the ground truth signed distance. We computed signed-distance values for $M = 200,000$ query points per shape, where the query points are sampled inside a unit sphere and biased toward being near the surface of the shape (using about $M/2$ points near the shape boundary).

We also need a point cloud of each shape to provide as input during training. To generate these point clouds, we initialize the objects on a table in PyBullet [5] in random positions and orientations, and render depth images with the object segmented from the background using multiple simulated cameras. These depth maps are converted to 3D point clouds and fused into the world coordinate frame using known camera poses. To obtain a diverse set of point clouds, we randomize the number of cameras (1-4), camera viewing angles, distances between the cameras and objects, object scales, and object poses. Rendering point clouds in this way allows the model to see some of the occlusion patterns that occur when the objects are in different orientations and cannot be viewed from below the table.

A1.2 Architecture

Point cloud encoder. We follow the encoder/decoder architecture proposed in Vector Neurons [6] for rotation equivariant occupancy networks [3], and replace the output occupancy probability prediction with a signed distance regression.

The encoder $\mathcal{E}$ follows from the SO(3)-equivariant PointNet [7] model proposed in [6]. $\mathcal{E}$ takes $\mathbf{P} \in \mathbb{R}^{3\times N}$ as input and outputs a *matrix* latent representation $\mathbf{Z} \in \mathbb{R}^{3\times C}$ (i.e., to which applying a rotation $\mathbf{R} \in \text{SO}(3)$ is a meaningful operation). The composition of layer operations in $\mathcal{E}$ is rotation equivariant (see [6] for details on how this property is achieved). As a result, given a point cloud $\mathbf{P}$ and rotation $\mathbf{R}$, the following relationship holds by construction:

$$\mathcal{E}(\mathbf{RP}) = \mathbf{R}\mathcal{E}(\mathbf{P}) = \mathbf{RZ}. \tag{1}$$

Implicit function decoder. The decoder Φ consists of an MLP with residual connections that also contains Vector Neuron layers:

$$\Phi(\mathbf{x}, \mathcal{E}(\mathbf{P})) : \mathbb{R}^3 \times \mathbb{R}^{3\times C} \to \mathbb{R} \tag{2}$$

The decoder takes in a combination of features, computed using the point cloud embedding $\mathbf{Z}$ and a 3D query point $\mathbf{x}$, that is designed to make the output prediction rotation *invariant*, i.e., if $\mathbf{Z}$ and $\mathbf{x}$ have been rotated *together*, the distance prediction should not change. Specifically, following [6], the decoder predicts the signed distance $s \in \mathbb{R}$ from 3D coordinate $\mathbf{x}$ to the shape as:

$$s(\mathbf{x}, \mathbf{Z}) = \text{ResNet}(\left[\langle \mathbf{x}, \mathbf{Z}\rangle, ||\mathbf{x}||^2, \text{VN-In}(\mathbf{Z})\right]) \tag{3}$$

where VN-In($\mathbf{Z}$) is rotation *invariant*, obtained as $\text{VN-In}(\mathbf{Z}) := \mathbf{V}^T\mathbf{Z}$, where $\mathbf{V}$ is the output of another rotation equivariant function of $\mathbf{Z}$[†].

Building descriptors from MLP activations. Following prior work [2], we define an NDF as a function mapping an input coordinate and point cloud to the *vector of concatenated activations* of the decoder Φ:

$$f(\mathbf{x}|\mathbf{P}) = \bigoplus_{i=1}^{L} \Phi^i(\mathbf{x}, \mathcal{E}(\mathbf{P})) \tag{4}$$

where L denotes the total number of layers in Φ, Φ^i denotes the output activation of the ith layer, and $\bigoplus$ denotes the concatenation operator.

A1.3 Training Details

Here we discuss details on training the NDF models using the data and model architecture described in the sections above. Training samples consist of inputs (point cloud $\mathbf{P}_{train} \in \mathbb{R}^{N_{train}\times 3}$, with $N_{train} = 1000$) and a set of query points $\mathcal{X}_{train} \in \mathbb{R}^{N_{q,train}\times 3}$, with $N_{q,train} = 1500$) and ground truth outputs (distance between each point in $\mathcal{X}_{train}$ and the underlying shape of the object represented

[*] `https://github.com/marian42/shapegan` and `https://github.com/marian42/mesh_to_sdf`

[†] Following the explanation from [6], this is because the product of one rotation equivariant feature $\mathbf{Z} \in \mathbb{R}^{3\times C}$ with the transpose of another $\mathbf{V} \in \mathbb{R}^{3\times C'}$ is rotation invariant: $(\mathbf{RV})^T(\mathbf{RZ}) = \mathbf{V}^T\mathbf{R}^T\mathbf{RZ} = \mathbf{V}^T\mathbf{Z}$. See Sec. 3.5 of [6] for more details

by $\mathbf{P}_{train}$). Based on the corresponding object scale and pose used to generate the respective point cloud in simulation, we compute ground truth distances by transforming and scaling the distances relative to the normalized object shapes. We train $\mathcal{E}$ and Φ end-to-end to minimize the mean-squared error between the predicted and ground-truth distance values across the query points.

We generated a dataset of approximately 100,000 samples for each category and trained one NDF model per category using each respective dataset. We trained the models for 150 thousand iterations on a single NVIDIA 3090 GPU with a batch size of 16 and a learning rate of 1e-4, which takes about half a day. We train the models using the Adam [8] optimizer.

A2 NDF Coordinate Frame Localization via Descriptor Distance Minimization

This section describes how the optimization in Equation (4) is performed to recover a pose, relative to an unseen shape, that matches a demonstration pose which has been encoded into a target pose descriptor.

The inputs to the problem are the target pose descriptor $\hat{\mathcal{Z}}$, NDF f, point cloud $\mathbf{P}$, and query points $\mathcal{X}$ (in their canonical pose). We randomly initialize an SE(3) pose $\mathbf{T}^0$ as an axis-angle 3D rotation $\mathbf{R}^0_{\text{aa}} \in \mathbb{R}^3$ and a 3D translation $\mathbf{t}^0 \in \mathbb{R}^3$. We then perform N_{iter} iterations of gradient descent to update this pose.

On iteration j, we begin by constructing a pose $\mathbf{T}^j \in$ SE(3) by converting $\mathbf{R}^j_{\text{aa}}$ into rotation matrix $\mathbf{R}^j \in$ SO(3) using the SO(3) exponential map [9] and combining with the translation $\mathbf{t}^j$. We then obtain pose descriptor $\mathcal{Z}^j$ by transforming the query points $\mathcal{X}$ by $\mathbf{T}^j$ and applying Equation (3) using f. Finally, we compute the loss $\mathcal{L}$ using the L1 distance between $\hat{\mathcal{Z}}$ and $\mathcal{Z}^j$ and backpropagate gradients of this loss to update the rotation and translation:

$$\mathbf{R}^{j+1}_{\text{aa}} \leftarrow \mathbf{R}^j_{\text{aa}} - \lambda \nabla_{\mathbf{R}_{\text{aa}}} \mathcal{L} \tag{5}$$

$$\mathbf{t}^{j+1} \leftarrow \mathbf{t}^j - \lambda \nabla_{\mathbf{t}} \mathcal{L} \tag{6}$$

We use the Adam [8] optimizer with a learning rate λ of 1e-2 to run this procedure. The optimization is run for a fixed number $N_{iter} = 650$ iterations to optimize $\mathbf{T}$. We empirically observed this to be enough iterations to allow the solution to converge. Furthermore, since the loss landscape for this problem is non-convex, the solution is somewhat sensitive to the initialization. To help obtain some diversity in the solutions, we run the optimization multiple times in parallel from different initial values for the pose. We used a batch size of 10, which uses about 10GB of GPU memory when using $N_q = 500$ query points and a point cloud downsampled to $N = 1500$ points.

A3 EBM Training

In this section, we present the training details of utilizing an EBM to refine predictions of relative transformation between objects.

Training Data. To train EBMs to capture each relation, we utilize the 10 demonstrations provided for each task. To prevent overfitting, and to construct more diverse data to train EBM models, we heavily data augment point clouds in each demonstration. In particular, we skew point clouds, apply per-point Gaussian noise, and simulate different occlusion patterns on the demonstration point clouds.

Model Architecture. We utilize a three layer MLP (described in Table 3) to parameterize an EBM operating over the concatenation of descriptors at each point. We utilize a swish activation in the EBM to enable fully continuous gradients with respect to inputs.

Training Details. When training EBMs, we corrupt $\hat{\mathbf{T}}_B \hat{\mathbf{P}}_B$ by a transformation corresponding translation sampled from $[-0.05, 0.05]$ along each dimension and rotation perturbation of 15 degrees along yaw, pitch and roll. We utilize a gradient descent step size of 10 for translation and 20 for rotation during optimization in training, and run 8 steps of optimization. Optimized rotations are represented as Euler angles, as the perturbations of individual rotation components are small.

Each EBM is evaluated pointwise across 1000 points (with descriptors with respect to each object concatenated). EBMs are trained with a batch size of 16.

Computational Resources. To train each model, we utilize a single Volta 32GB machine for 6 hours and train models for 12000 iterations.

A4 Experimental Setup

This section describes the details of our experimental setup in simulation and the real world.

A4.1 Simulated Experimental Setup

We use PyBullet [5] and the AIRobot [10] library to setup the tasks in the simulation, provide demonstrations, and perform quantitative evaluation experiments. The simulation environment contains a Franka Panda arm with a Robotiq 2F140 gripper attached, a table, and a set of simulated RGB-D cameras. We obtain segmentation masks using the built-in segmentation abilities of the simulated cameras to separate object point clouds from the overall scene.

A4.2 Real World Experimental Setup

Our real world environment also contains a Franka Robot arm with a Robotiq 2F140 parallel jaw gripper. We also use four Realsense D415 RGB-D cameras, with extrinsics calibrated relative to the robot's base frame. We use a combination of point cloud cropping and Euclidean clustering to segment object point clouds from the scene, identify their class identify, and filter out noise. Specifically, we crop the overall scene to the known region above the table, and then crop $\mathbf{O}_A$ and $\mathbf{O}_B$ based on which side of the table each object is on (assumed to be known for this experiment, just for the purposes of obtaining the segmentation). Finally, we run DBSCAN [11] clustering to remove outliers and noise to obtain the final point clouds $\mathbf{P}_A$ and $\mathbf{P}_B$. To demonstrate R-NDF on objects in diverse initial orientations, we present some of the objects on a 3D printed stand with angled support surfaces. When using the stand, we remove it from the point cloud by estimating its pose using 3D registration and the known CAD model.

A5 Evaluation Details

This section presents further details on the three tasks we used in our experiments and notes on the automatic detection methods used to obtain success rates over multiple simulated trials.

A5.1 Tasks and Evaluation Criteria

Task Descriptions. We consider three relational rearrangement tasks for evaluation: (1) Hanging a mug on the hook of a rack, (2) Stacking a bowl upright on top of a mug, and (3) Placing a bottle upright inside of a box-shaped container. For "Hanging a mug on a rack", we ensure the mug is oriented consistently relative to the single peg of the rack. This is achieved by providing demonstrations in which the handle always points left relative to a front-facing peg, and the opening of the mug is always points toward the top of the rack. Similarly for stacking bowls on mugs and putting bottles in containers, we provide demonstrations in which the "stacked" object is always upright, though in these tasks, the orientation about the radial axis of the bowls/bottles doesn't affect the relation result and is thus ignored. All objects are presented in an initial "upright" pose on the table for the demonstrations.

Evaluation Metrics and Success Criteria. To quantify performance, we report the success rate over 100 trials, where we use the ground truth simulator state to compute success. For a trial to be successful, objects $\mathbf{O}_A$ and $\mathbf{O}_B$ must be in contact, $\mathbf{O}_B$ must have the correct orientation relative to $\mathbf{O}_A$, and $\mathbf{O}_A$ and $\mathbf{O}_B$ must not interpenetrate.

A5.2 Providing Demonstrations

Here we re-describe the procedure for obtaining task demonstrations using a robot arm, a set of depth cameras, and a parallel jaw gripper, as outlined in Section 5.

When collecting a demonstration, initial object point clouds $\hat{\mathbf{P}}_A$ and $\hat{\mathbf{P}}_B$ of objects $\hat{\mathbf{O}}_A$ and $\hat{\mathbf{O}}_B$ are obtained by fusing a set of back projected depth images. The demonstrator moves the gripper to a pose $\hat{\mathbf{T}}_{\text{grasp}}$, grasps $\hat{\mathbf{O}}_B$, and finally moves the gripper to a pose $\hat{\mathbf{T}}_{\text{place}}$ that satisfies the desired relation between $\hat{\mathbf{O}}_A$ and $\hat{\mathbf{O}}_B$. $\hat{\mathbf{T}}_B$ is obtained as $\hat{\mathbf{T}}_{\text{place}}\hat{\mathbf{T}}_{\text{grasp}}^{-1}$. In *one* of the demonstrations, a 3D keypoint $\mathbf{x}_{AB}$ is labeled near the parts of the objects that interact with each other by moving the gripper to this region and recording its position.

Tuning the scale of the query point set $\mathcal{X}_A$. As mentioned in Section 4.1, $\mathcal{X}_A$ is obtained by sampling from a zero-mean Gaussian, with a variance that must be chosen by the user. This variance will impact the scale of the query point set, and can be thought of as a hyperparameter. The original NDF paper discusses the implications of different query point sizes (see Table III in [2]). We therefore did not repeat the ablations performed in [2] showing that the scale of the query point cloud must be tuned based on the rough scale of the shapes in the object set. We instead tuned the variance parameter and settled on values that led to good task performance. For real-world objects, the value we used typically falls between 0.015 and 0.025. The heuristic used in [2] to simplify this tuning procedure was to use a bounding box around the whole shape $\mathbf{O}_A$.

A5.3 Baselines

Pose Regression. The Pose Regression baseline method consists of an MLP trained to predict $\mathbf{T}_B$ using the concatenated embeddings of $\mathbf{P}_A$ and $\mathbf{P}_B$ obtained from a pretrained VNN encoder (the one used for NDFs). We supervise transform prediction $\hat{\mathbf{T}}_B$ using the Chamfer distance between a transformed pointcloud $\hat{\mathbf{T}}_B\mathbf{P}_B$ and a ground truth pointcloud $\mathbf{T}_B\mathbf{P}_B$. Such a loss captures the symmetry in $\mathbf{P}_B$. A transformation is represented using vector of six dimensions, where the first three dimensions correspond to translation and the subsequent dimensions correspond to an axis-angle parameterization of rotations The architecture for pose regression is provided in Table 4.

Patch Match. The Patch match baseline uses 3D registration to align the test-time point clouds $\mathbf{P}_A$ and $\mathbf{P}_B$ to the point clouds of the corresponding shapes used in one of the demonstrations, and then using the demonstrated pose $\hat{\mathbf{T}}_B$ together with these registration results to compute the resulting pose $\mathbf{T}_B$. Formally, for demonstration $\mathcal{D}_i$, registering source $\mathbf{P}_A$ to target $\hat{\mathbf{P}}_A$ produces SE(3) transformation $\mathbf{T}_{A,\text{reg}}$, and similarly for $\mathbf{P}_B$ and $\hat{\mathbf{P}}_B$ to obtain $\mathbf{T}_{B,\text{reg}}$. $\mathbf{T}_B$ is then obtained as $\mathbf{T}_{A,\text{reg}}^{-1}\hat{\mathbf{T}}_B$.

A5.4 Automatic success detection and common failure modes

This section discusses the methods we used for checking each success criteria in the simulator, along with some of the common failure modes for each method.

Correct Orienttion. We compare the angle between the radial axis of the bowls/bottles and the positve z-axis in the world to check if the final orientation is correct for the "bowl on mug" and "bottle in container" tasks. We count the objects as ending in a valid "upright" orientation if this angle difference is below 15 degrees once the physics has been turned on and the object has settled. This is important because a common failure mode for the "bottle in container" task is to localize the top of the bottle instead of the bottom and try to place it upside down on the container. This occurs for both our method and the baselines. For "bowl on mug", this failure mode is much less apparent for our method but still occurs quite often with the baselines.

We did not explicitly check if the orientation is correct for the "mug on rack" task, as this would require comparing the angle between the axis pointing along the cylindrical body of the mug and the axis pointing along the rack's peg to ensure the mug opening points in the right direction relative to the rack. Since the pegs on the racks are all slightly different, obtaining this ground truth peg-axis angle was too cumbersome to manually set up. We instead relied on the "$\mathbf{O}_A$ and $\mathbf{O}_B$ in contact" criteria to imply the correct relative configuration between the mug and the rack. This is an effective method because for many incorrect relative orientations, the mug misses the rack and falls when the physics are turned on, leading to a final configuration where the objects do not touch. However, there may still be solutions found that satisfy the "hanging" criteria by passing this check but not being in the intended orientation. We thus allow any final configurations satisfying "hanging" (implied by the mug and rack ending up "in contact" and "not interpenetrating"), where the mug opening points down instead of up, to be counted as a success. We observe this happens much more frequently for the "Patch Match" baseline than R-NDF, as "Patch Match" struggles in disambiguating between an upright and an upside down mug during 3D registration.

In-Contact vs. Not-Interpenetrating. Since we reset $\mathbf{O}_A$ and $\mathbf{O}_B$ in the simulator to their final configuration after predicting the relative transformation, the objects can end up in physically-/geometrically-infeasible poses that intersect, and we don't want to count these configurations as successful for any of the tasks. As described above, we use the "in-contact" criterion to implicitly determine if the "mug on rack" relation is satisfied based on the objects' relative orientation. Even for "bottle in container" and "bowl on mug", where we explicitly check to ensure $\mathbf{O}_B$ ends in an upright orientation, we might obtain a prediction that places $\mathbf{O}_B$ too low relative to $\mathbf{O}_A$ and causes an infeasible intersection. Therefore, we can obtain many false positives if we don't take care to ensure objects that interpenetrate are not counted as being *successfully* "in-contact". However, automatically disambiguating the "in contact" criteria with the "not interpenetrating" criteria, both of which are required for success, turns out to be slightly nontrivial. Here we discuss the method we used to check these criteria in a disentangled fashion.

We first check that $\mathbf{O}_A$ and $\mathbf{O}_B$ are in contact after transforming $\mathbf{O}_B$ by $\mathbf{T}_B$ ("in contact") and allowing the physics simulation to proceed for 2 seconds,. We then ensure the objects are not only in contact because they are interpenetratig by checking whether or not $\mathbf{O}_B$ can be easily *removed* from its final configuration under dynamic physical effects. To achieve this, we transform $\mathbf{O}_A$ and $\mathbf{O}_B$ to maintain their relative configuration, but make $\mathbf{O}_A$ turn upside down in the world frame. For each of our tasks, if the objects are not in interpenetration, $\mathbf{O}_B$ should fall away from $\mathbf{O}_A$, whereas we observe that they regularly get stuck together in a physically implausible way if they are in interpenetration. We thus check whether or not the objects are still in contact after turning them upside down and waiting while the physics simulation proceeds, and label the pair as "not interpenetrating" if they are not in contact after this delay. The most common failure mode for R-NDFs on the "bottle in container" and "bowl on mug" tasks is to predict transformations that nearly satisfy the relation but cause the objects to interpenetrate (e.g., the bottle/bowl is too low, and thus intersects with the container/mug).

A5.5 Task Execution

This section describes additional details about the pipelines used for executing the inferred relations in simulation and the real world.

Simulated Execution Pipeline. The evaluation pipeline mirrors the demonstration setup. Objects from the 3D model dataset for the respective categories are loaded into the scene with randomly sampled position and orientation. In the "upright" pose case, a known upright orientation is obtained, and then adjusted with a random top-down yaw angle. In the "arbitrary" pose case, we sample a rotation matrix uniformly from SO(3), load the object with this orientation, and constrain the object in the world frame to be fixed in this orientation. We do not allow it to fall on the table under gravity, as this would bias the distribution of orientations covered to be those that are stable on a horizontal surface, whereas we want to evaluate the ability of each method to generalize over all of SO(3). In both cases, we randomly sample a position on/above the table that are in view for the simulated cameras. We also load the target pose descriptor $\hat{\mathcal{Z}}$, obtained by following the procedure described in Section 4, for use in inference.

After loading objects $\mathbf{O}_A$ and $\mathbf{O}_B$ and the target pose descriptor $\hat{\mathcal{Z}}$, we obtain segmented point clouds $\mathbf{P}_A$ and $\mathbf{P}_B$ and apply Equations (5), (6), and (8) to obtain a transformation $\mathbf{T}_B$. The output transformation is applied to $\mathbf{O}_B$ by transforming its initial pose $\mathbf{T}_{B,\text{start}}$ by $\mathbf{T}_B$ and resetting the state of the object in the simulation to the resulting pose $\mathbf{T}_{B,\text{final}} = \mathbf{T}_B \mathbf{T}_{B,\text{start}}$. Task success is then checked based on the criteria described in the section above.

Real World Execution Pipeline. Here, we repeat the description of how we execute the inferred transformation using a robot arm with additional details.

At test time, we are given point clouds $\mathbf{P}_A$ and $\mathbf{P}_B$ of new objects $\mathbf{O}_A$ and $\mathbf{O}_B$, each with potentially new shapes and poses, and Equations (5), (6), and (8) are applied in sequence to obtain $\mathbf{T}_B$. We first obtain an initial grasp pose $\mathbf{T}_{\text{grasp}}$. Our implementation uses NDF to generate these grasp poses, following the pipeline described in [2] and Section 3 (where $\mathbf{O}_A$ is the robot's gripper), but any generic grasp generation pipeline could be used instead. We then obtain a placing pose $\mathbf{T}_{\text{place}} = \mathbf{T}_B \mathbf{T}_{\text{grasp}}$, and plan a collision-free path between the grasp and place pose using MoveIt![‡]. The path is executed by following the joint trajectory in position control mode and opening/closing the fingers at the correct respective steps. The whole pipeline can be run multiple times in case the planner returns infeasibility, as the inference methods for both grasp and placement generation can produce somewhat different solutions depending on how the NDF optimization is initialized.

Pipeline for Executing Multiple Relations in Sequence. Figure 1 shows a multi-step rearrangement application of R-NDFs for the "bowl on mug" task, where a "mug upright on the table" relation is executed before the "bowl upright on the mug" relation. This section describes the setup for chaining these relations and executing them in sequence (see also Subsection A8 below for further discussion).

To specify the "mug upright on table" component of the task, we follow [2] and the steps described in Section 3 on "NDFs for Encoding Single Unknown Object Relations", where the table is the *known* object $\mathbf{O}_A$. Using this prior knowledge, we initialize a set of query points near a known placing region on the table, and use these points to obtain the target pose descriptor from a set of demonstrations (i.e., demos of placing mugs upright on the table near the query point set location). The "bowl upright on mug" part of the task is then encoded using the method described in Section 4.

During execution, both inference steps are run using the *initial* point clouds $\mathbf{P}_A$ and $\mathbf{P}_B$, i.e., we don't re-observe the mug after executing the upright placement on the table. Thus, we find $\mathbf{T}^1_B$ which transforms the mug relative to the table, and $\mathbf{T}^2_B$, which transforms the bowl relative to the mug in its *initial* configuration. Finally, we execute relative transformations $\mathbf{T}^1_{B,\text{exec}} = \mathbf{T}^1_B$ to the mug and $\mathbf{T}^2_{B,\text{exec}} = \mathbf{T}^1_B \mathbf{T}^2_B$ to the bowl, using the pick-and-place operation described in the section above.

A6 Alternating Minimization to Obtain an Average Pose Descriptor From Multiple Unaligned Demonstrations

This section describes details and further intuition behind our method for aligning descriptors across a set of demonstrations, as proposed in Section 4.

Assuming a specified interaction point $\mathbf{x}_{AB}$ in demonstration $\mathcal{D}_i$, we first construct $\hat{\mathbf{T}}^0_{\mathcal{X}_A,i}$. We then transform the canonical query points $\mathcal{X}_A$ by $\hat{\mathbf{T}}^0_{\mathcal{X}_A,i}$ to the region near the task-relevant features on the object and obtain $\hat{\mathcal{Z}}^0_{ref} = \hat{\mathcal{Z}}^0_i$ with Equation (3). Equation (5) is then used with $\hat{\mathcal{Z}}^0_{ref}$ to solve for a corresponding transformation $\hat{\mathbf{T}}_{\mathcal{X}_A,j}$ and a resulting pose descriptor $\hat{\mathcal{Z}}^0_j = F_A(\hat{\mathbf{T}}_{\mathcal{X}_A,j} | \hat{\mathbf{P}}_{A,j})$ for the remaining demonstrations $\{\mathcal{D}_j\}^K_{j=1,j\neq i}$. After running this for each demonstration, we compute an average pose descriptor $\hat{\mathcal{Z}}^1_{ref} = \frac{1}{K}\sum_{i=1}^K \hat{\mathcal{Z}}^0_j$. We then apply the same procedure to each demonstration (including the demo used for providing the initial reference pose) *again*, now using $\hat{\mathcal{Z}}^1_{ref}$ as the target descriptor. We repeat this process Q times, where Q is a hyperparameter (we used $Q = 3$ throughout the experiments). The result is a final target pose descriptor $\hat{\mathcal{Z}} = \hat{\mathcal{Z}}^Q_{ref} = \frac{1}{K}\sum_{i=1}^K \hat{\mathcal{Z}}^{Q-1}_j$.

Intuitively, this procedure starts with a target descriptor obtained from a single demonstration (whichever demonstration $\mathcal{D}_i$ corresponds to the one where $\mathbf{x}_{AB}$ was provided). As shown in the top row of Table 5a, some fraction of the poses obtained by matching a single demonstration correctly

[‡] https://moveit.ros.org/

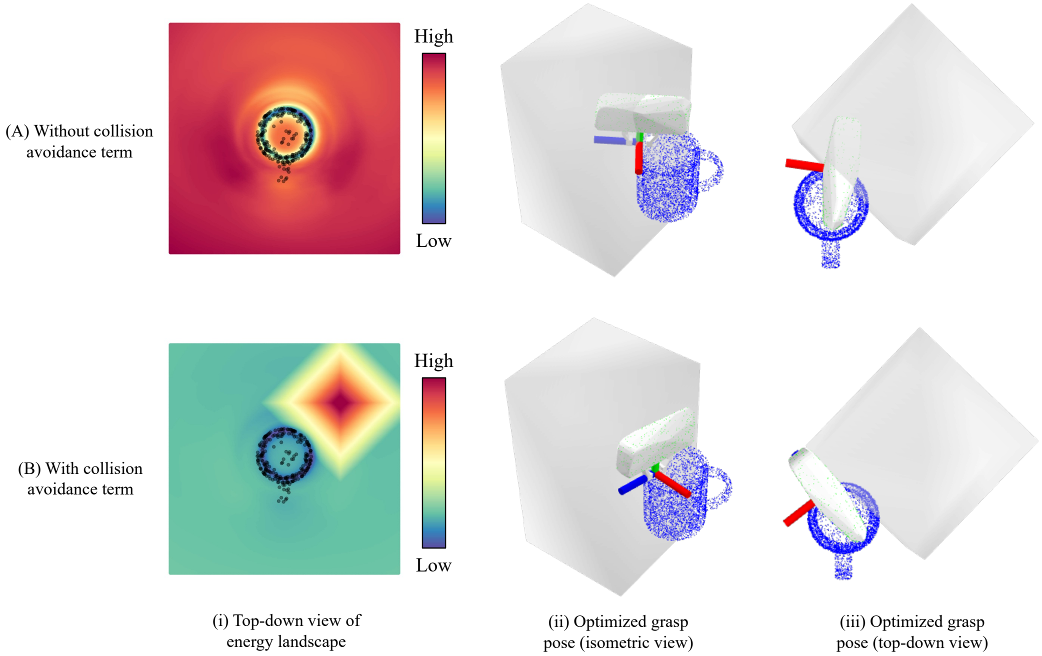


Figure A2: **Energy composition for collision avoidance and descriptor matching. (Top) NDF grasp pose inference without collision avoidance cost.** We recorded a grasp pose along the rim of a demo mug and recovered the corresponding pose on a new mug using NDF optimization. This procedure ignores collision avoidance with nearby obstacles, and thus finds a solution which would be infeasible due to interpenetration with the box-shaped obstacle next to the mug. **(Bottom) NDF grasp pose inference with collision avoidance cost.** We modify the NDF optimization with an additional cost term. This extra term penalizes query points that fall inside the obstacle. The resulting energy landscape takes both the collision avoidance and the descriptor matching into account. The optimizer finds a new solution that stays outside of the obstacle, while still achieving a grasp along the rim.

match a target descriptor ($\sim$ 40% success). This means that *some* of the individual descriptors in $\{\hat{\mathcal{Z}}_j^k\}_{j=1,j\neq i}^K$ found on the kth iteration correctly align with the reference, and are somewhat near each other in descriptor space. The mean of this set $\hat{\mathcal{Z}}_{ref}^{k+1}$ thus provides a new descriptor that is both (on average) more similar to each of the descriptors than was the original reference, and still similar to the original descriptor that was obtained using the manually specified keypoint. By resetting the target using this mean, the next alignment round uses a target which is more sensitive to the part features that are shared among *some* of the demonstrations, and makes it more likely that a consistent pose will be found for *more* of the demonstrations than the previous iteration. The similarity among the resulting descriptors continues increasing accordingly. Overall, multiple rounds of this procedure leads to poses for each respective demonstration which are consistent with each other and map to descriptors with relatively high similarity.

By encouraging the individual descriptors in the set to converge to values that are similar to the mean among the whole set, this iterative procedure allows the final target descriptor to capture the shared parts of the demonstration objects in a consistent fashion.

A7 Modifying the R-NDF Optimization Objective for Collision Avoidance

This section shows how the optimization objective used for recovering coordinate frame poses can be modified to take collision avoidance into account.

Our framework models rearrangement tasks in the form of SE(3)-transformations of rigid bodies. The method described in Section 4 does not address other constraints like collision avoidance and robot kinematics, and our real-world task executions depend on a separate motion planning module, since our approach does not produce full paths/trajectories. However, because we perform optimization for pose localization, it is straightforward to incorporate extra cost terms that capture additional factors like collision avoidance.

In Figure A2, we highlight this ability with an example of grasping a mug while avoiding collisions. The top row optimizes the standard descriptor matching objective and doesn't consider nearby obstacles. In contrast, the bottom row shows the result when the optimizer minimizes a combined energy, composed of a descriptor matching cost and a collision avoidance cost. The grasp pose found in the top row collides with the nearby obstacle, since it only tries to match the target pose descriptor. On the other hand, in the bottom row, the energy landscape for the optimization takes the nearby obstacle into account, such that a different point along the rim of the mug achieves the new optimum. The new optimum still satisfies the desired grasp along the rim but also achieves the secondary constraint of keeping the gripper outside of the box-shaped obstacle.

This highlights the ability to incorporate additional task constraints together with the pose matching objective, using collision avoidance as one such example. The same principle can apply to other task constraints. For instance, recent work has shown that energy-based modeling is useful for taking robot kinematics into account by setting up joint space decision variables and using a differentiable forward kinematics module [12]. In this way, a user can add a similar additional cost term that penalizes solutions which cause self-collisions and violate joint limits. Other recent work has set up similar compositional energy optimization approaches for trajectory synthesis and motion generation [13]. In machine learning, energy-based models have been useful for compositional generative modeling and reasoning [14–17].

A8 Extending Beyond Two-Object Rearrangement

This section expands on how R-NDF can be used for rearrangement tasks involving more than two objects, via both sequential and compositional optimization.

Our approach models relational rearrangement via pair-wise relations, which facilitates a natural way to go beyond two-object rearrangement and handle tasks involving more objects. Here we discuss two approaches for achieving this with R-NDF. While we did not use the EBM for these examples, the same type of compositionality and sequential inference is also directly applicable to using the learned model, which also operates on pairs of object point clouds.

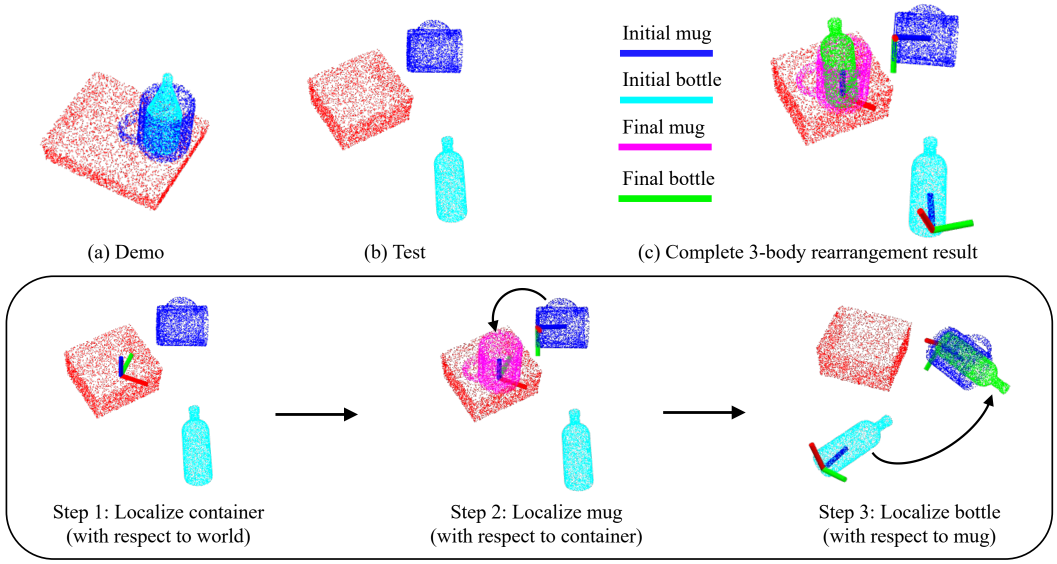
A8.1 Three-body rearrangement via sequential optimization

If a set of multi-object relations forms a tree structure, each coordinate frame can be found independently and then composed in sequence to satisfy the overall task. This example is highlighted in Figure 1 (where the table can be considered a third static object). For additional clarity, Figure A3 shows another example of this behavior. The task is to place the "mug upright in the container *and* the bottle in the mug". At test-time, the new container, mug, and bottle are all in different initial poses. First, the container is localized. Then, the mug is localized relative to the container, and finally, the bottle is localized relative to the mug. These relative transformations are composed to obtain the overall transformation applied to each object in execution.

A8.2 Three-body rearrangement via energy composition

Solving for task-relevant coordinate frame poses using optimization makes it straightforward to incorporate additional costs. We can apply this fact to the setting of three-object rearrangement, as shown in Figure A4. In particular, instead of matching descriptors with respect to one object, we can solve for a pose that attempts to match descriptors with respect to *two* objects *simultaneously*.

Here, the task is to place a mug upright both "next to the bowl" and "next to the bottle" (see Figure A4, top left). After localizing a coordinate frame on the bottom of a mug, we run the NDF



(a) Demo (b) Test (c) Complete 3-body rearrangement result

Step 1: Localize container (with respect to world)

Step 2: Localize mug (with respect to container)

Step 3: Localize bottle (with respect to mug)

(d) Sequential coordinate frame localization

Figure A3: **Three-object rearrangement by sequentially localizing multiple task-relevant coordinate frames**. **(a)** Demonstration of "mug in container" and "bottle in mug". **(b)** Initial configuration of test-time container, bottle, and mug. **(c)** Final configuration satisfying the desired set of relations among the three objects after inferring task-relevant coordinate frames on each of them. **(d)** Independent coordinate frame localization steps executed in sequence. First, the bottom of the container is found in the world frame. Then, the bottom of the sideways blue mug is found. Using the frame found in Step 1., the initial mug (dark blue) can be transformed to the final upright mug (pink). Finally, the bottom of the initially sideways bottle (teal) can be found, and transformed to the final bottle (green) inside of the mug. Compsing each of these steps together provides the final configuration shown in **(c)**.

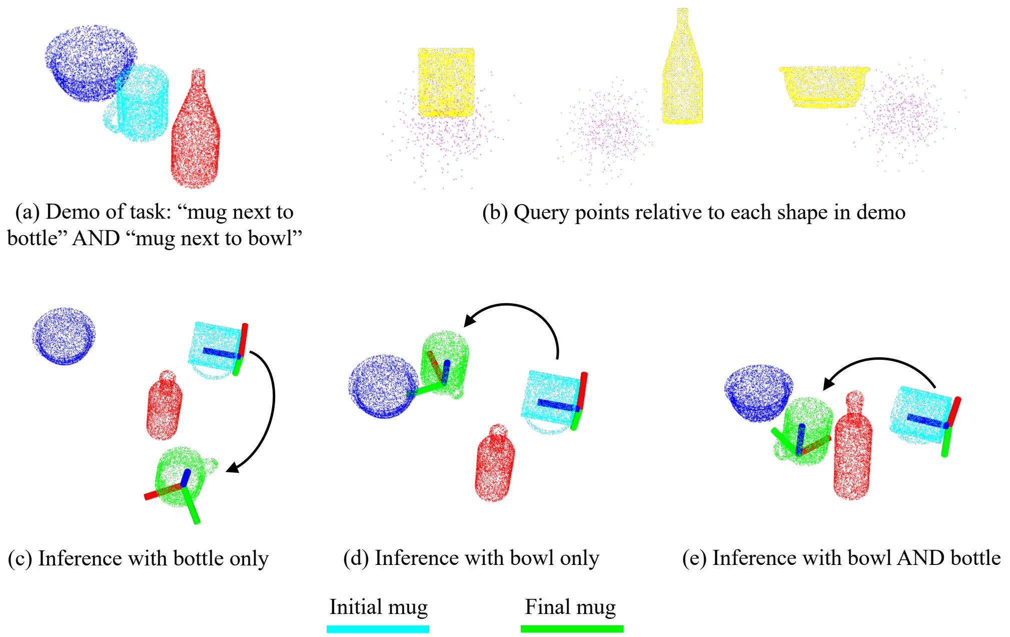
optimization of the query points to recover a coordinate frame that satisfies both of the "next to" relations simultaneously. This is achieved by computing the overall optimization loss as the sum of the two separate "bowl" and "bottle" descriptor matching losses. Note that due to the radial symmetry of the bowl and the bottle, if we only optimize the mug pose with respect to one object, the solution can end up anywhere along a circle surrounding the respective object. However, when we optimize with respect to both objects, the solver finds a unique solution that satisfies both "next to" objectives, placing the mug in a location that is unambiguously between the bowl and the bottle.

A9 Can R-NDF work with partial point clouds?

In this section, we provide evidence that R-NDF can work with partial point clouds obtained from a single viewpoint, and discuss the difficulties of handling the problem of rearrangement with heavy occlusions in its full generality.

Similar to [2], we obtained point clouds from multiple cameras at different viewing angles to ensure the point cloud was relatively complete. The purpose of this design choice was to focus the effort on expanding the rearrangement task capbilities, rather than handling scenes with large occlusions. However, we have observed the model's ability to deal with partial point clouds, so long as (1) they are included in the training distribution, and (2) the important part of the object for matching (i.e., the handle, or the peg, for the mug/rack task) is not entirely hidden from view. Figure A5 shows an example of successful coordinate frame localization on point clouds obtained from a single camera.

Dealing with the partial point cloud issue in its entirety requires addressing the possibility that the task-relevant part might be completely hidden from view. This difficult scenario greatly complicates matters. Techniques related to active perception, wherein the system can search for new viewpoints,


(a) Demo of task: “mug next to bottle” AND “mug next to bowl”

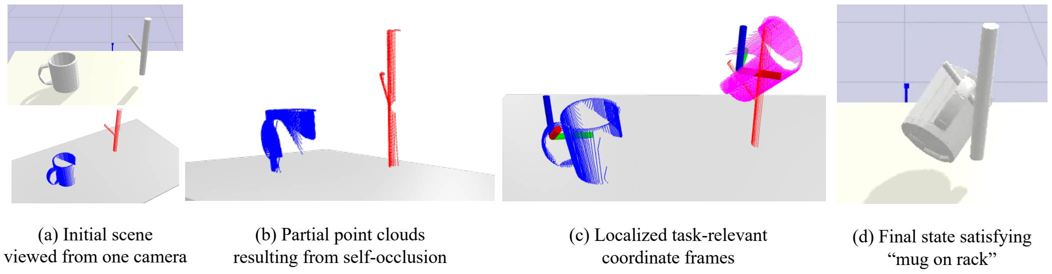
(b) Query points relative to each shape in demo

(c) Inference with bottle only

(d) Inference with bowl only

(e) Inference with bowl AND bottle

Figure A4: **Three-object rearrangement by composing multiple NDF descriptor distances**. **(a)** Demonstration showing “mug next to bottle” *and* ‘mug next to bowl”. **(b)** Using a single set of query points at the bottom of the demonstration mug, we obtain a set of descriptors for each shape, relative to the point cloud of each shape shown in the demonstrations. **(c)** At test-time, we first localize the world frame pose of the bottom of the mug. If we then optimize the final mug pose by matching the bottle NDF descriptors alone, a solution far away from the bowl is found. This is because the bottle has a radial symmetry which allows multiple solutions for descriptor matching. **(d)** Similarly, when finding the mug pose relative to the bowl on its own, the mug ends up at a position away from the bottle. **(e)** By optimizing the mug pose relative to both the bowl NDF and the bottle NDF *simultaneously*, the resulting solution is “next to” both the bowl *and* and the bottle.


(a) Initial scene viewed from one camera

(b) Partial point clouds resulting from self-occlusion

(c) Localized task-relevant coordinate frames

(d) Final state satisfying “mug on rack”

Figure A5: **NDFs can work with partial point clouds.** **(a)** Initial scene for “mug on rack” task viewed from only a single camera. **(b)** Alternate view of the resulting point clouds for each object, showing large missing regions of each point cloud. **(c)** Task-relevant coordinate frames found on each object using the partial point clouds. This shows that, as long as the occlusions aren’t too severe, NDF descriptor matching can still work for recovering corresponding coordinate frames that match a demonstration. **(d)** Final simulator state after applying the recovered relative transformation.

are perhaps the fundamentally correct way to deal with this problem. While this this was beyond the scope of our work, note that as these complementary methods improve, our proposed method will become applicable to more general occlusion settings.

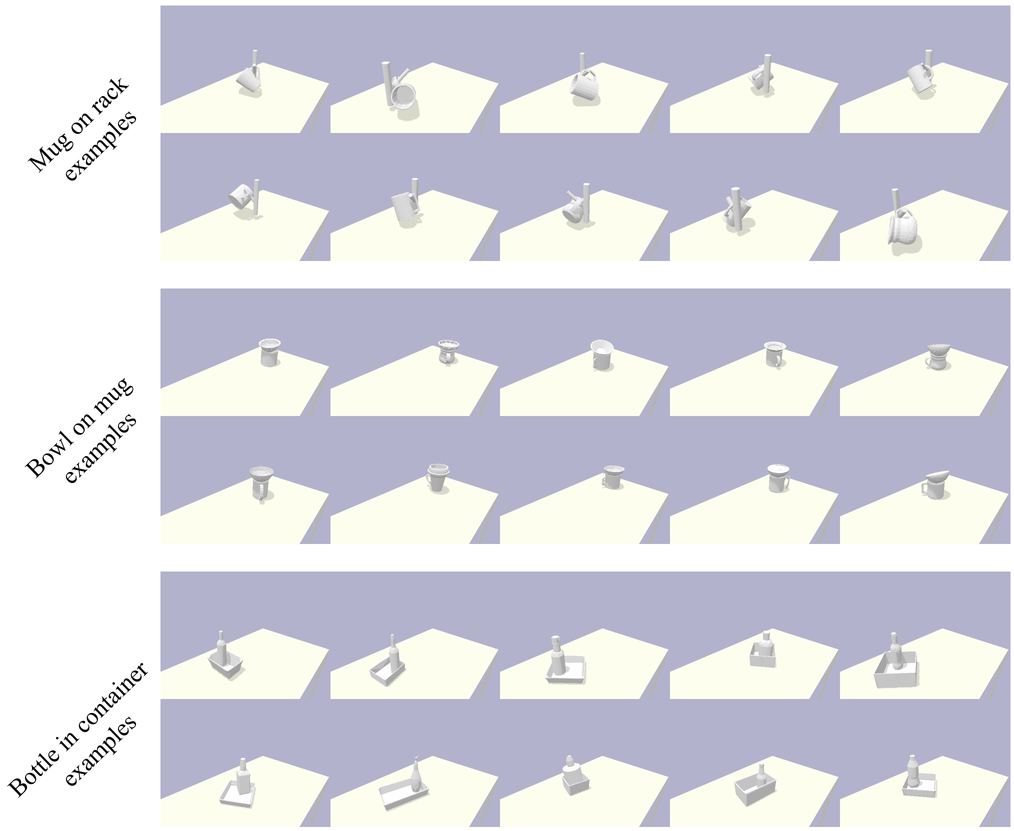


Figure A6: **Additional visualizations of unseen objects for evaluation tasks.** Snapshot of the simulator final state for each evaluation task. Simulator state is reached after transforming $\mathbf{O}_B$ into its relation-satisfying configuration using R-NDF. Each image shows a successful execution of the task.

A10 Miscellaneous Visualizations

In this section, we show additional visualizations of the tasks used in our simulation experiments. Figure A6 shows more snapshots of the final simulator state for each of the tasks used in quantitative evaluation.

A11 Expanded Details on System Engineering, Implementation, and Limitations

This section provides more details on our assumptions, how the overall system is engineered, and the resulting implications and limitations. Section 8 discusses some of these considerations, and we provide a more thorough treatment here.

A11.1 Real World Systems Engineering and Implementation

Motion planning and collision avoidance in real-world experiments. This work models relational rearrangement in the form of relative SE(3) transformations. We focus on evaluating whether or not the inferred transformation achieves a desired relation between pairs of unseen objects. In simulation, we can directly evaluate this core capability by bypassing the physics and resetting the object state to its predicted goal configuration. However, in the real world, we must execute a feasible path to the goal with the object held by the robot's end-effector.

Naive trajectory generation (e.g., via linear interpolation in task or joint space) regularly fails to achieve this because the robot or the grasped object collides with part of the environment. For example, hanging the mug on the rack requires a particular path to be followed in the last few inches to avoid moving the rack (e.g., by the mug colliding with the peg). Solving this motion planning and collision avoidance issue in its full generality would require performing full collision-free planning of the arm with the grasped object, which remains a challenging task in itself when dealing with raw point clouds. Several recent works [18, 19] have dedicated specific effort to solving just this problem, highlighting that it remains an outstanding challenge without simple off-the-shelf solutions. As this was not the focus of our work, we used domain knowledge of the task and extra supervision to simplify path planning. Our approach was to create multiple Cartesian waypoints for the arm to reach along its path to the goal, each of which having a high likelihood of being collision-free (e.g., at a position high above the center of the table).

Additional "offset" waypoint pose in demonstrations. We also annotated an extra "offset waypoint" in the demonstrations, and used this extra waypoint annotation to solve for an offset relative to the inferred placement pose at test-time. This offset pose could be reached more easily without using fine-grained collision detection. Once at this offset pose, the final placement pose could be achieved by moving the end-effector in a straight line task-space path. For example, in the "mug on rack" task, this was a pose where the mug's handle was aligned with the rack's peg, but at a position just in front of the tip of the peg. We performed this offset pose annotation in the demonstrations by recording an additional end-effector pose before moving the gripper to the final placement pose ($\hat{\mathbf{T}}_{\text{place}}$). We then solved for the relative transformation between this offset pose and the placement pose. Finally, at test-time, we solved for the world-frame offset pose using this same relative transformation and the recovered placing pose.

A11.2 Limiting Assumptions, Resulting Implications, and Avenues for Future Work

A11.2.1

Modeling assumption: We model rearrangement tasks as SE(3)-transformation(s) of rigid bodies. R-NDF does not deal directly with other constraints like collision avoidance and kinematics, and depends on a separate motion planner for path planning. As discussed in the subsection above, full collision-free planning was not the focus of this work, and we therefore used heuristic solutions, domain knowledge, and extra real-world supervision for the purpose of simplifying path planning to avoid issues with additional constraints. We also describe in Section A7 how to add extra costs that take more problem constraints into account. Finally, note that the design choice of predicting relative transformations representing goal configurations allows our system to generalize much more easily and efficiently than it would have if we tried to learn full trajectory generators or closed-loop policies.

A11.2.2

Input/Task assumption: R-NDF performs category-level manipulation with known categories, and depends on the availability of an offline dataset of 3D shapes from each category. Assuming a known category does limit the ability to directly apply our method to brand new objects in unseen categories. However, this assumption is also what helps support generalization to new shape instances, as the model learns to associate the way different shapes are similar to each other. Several "category-level" manipulation systems and dense correspondence models have been proposed in the past [20–25], each with the reasoning that it's useful to enable programming a specific robot skill, or learning a category-specific concept, that works across all instances from a category. Obtaining a module that ignores global features of the object and focuses on local parts that may be shared across categories is an exciting direction for future work.

Depending on an offline set of 3D objects is another limitation. Many of the advancements in 3D vision and graphics have also used ground truth shapes for training [3, 4, 26], but it would be ideal if we could obtain a system with similar properties that does not depend on the availability of ground truth 3D shapes. We leave this for future work to address.

A11.2.3

Input assumption: R-NDFs use instance-segmented point clouds with known identity (i.e., the system knows which point cloud corresponds to $\mathbf{O}_A$ and $\mathbf{O}_B$). As the focus of this work is to relax the “static secondary/environmental object” assumption from the original NDF work, and expand the scope of achievable rearrangement tasks, we directly inherited the assumption that point clouds have been accurately segmented from the background. Note that significant progress has been made on instance/semantic segmentation [27–30] with other robotic systems deploying these as components to their overall pipeline [19, 31]. Based on this trend, it’s not unreasonable to assume access to an off-the-shelf module that provides segmented point clouds. However, it’s a fair concern that today, these systems are not robust enough to be entirely “plug and play” without substantial engineering effort. This is especially true in new environments that cause a distribution shift from the data they were trained on, which can negatively affect the downstream task performance.

Changing the underlying NDF formulation to reduce the dependence on performant off-the-shelf perception modules also has the potential to expand the system’s capabilities. An analogous progression in 6-DoF grasp generation has occured and shown meaningful improvements in overall system capability (i.e,. see [31] and [32] where [32] does not require segmentation). However, this improvement is roughly orthogonal to our proposed approach.

Recent works on using different neural network encoders for 3D data that use local features have shown promising results in not requiring accurate segmentation [33–35]. The implications of global vs. local encoding on generalization and robustness to diverse geometries have been discussed at length in various recent works on neural fields [26, 36–39]. In principle, transferring such approaches to our setting within the proposed R-NDF framework ought to provide similar benefits in the multi-object rearrangement tasks we consider.

A11.2.4

General limitation: We use NDF as the core component of the framework. If the descriptors learned in NDF pretraining don’t work well, then the downstream R-NDF framework also won’t work well. What if they fail on more difficult shapes? It’s a concern that if the underlying NDF models have not learned meaningful descriptors that encode correspondence in a correct/useful way, our approach will inherit these problems and suffer in performance. This could potentially occur on more difficult shape categories or with more difficult tests of generalization. For instance, we did not deploy our approach on shape categories with potentially varying topology (e.g., chairs) or on instances that are dramatically different from those seen in training (e.g., if we train on racks with a single peg, the performance will likely be reduced if we directly test on racks with four pegs).

Recent approaches in neural implicit modeling include components for learning deformation fields with explicit correction terms [25] and latent spaces with higher-dimension [24] to recover cross-instance correspondence for topologically varying shapes. These ideas could potentially be incorporated into future versions of the NDF to support improved performance more challenging objects. Investigating such changes to the underlying NDF setup were beyond the scope of this work, but it’s an important consideration for scaling the approach to enable more difficult object categories.

A11.2.5

General limitation: This work only shows results in empty scenes with minimal clutter, no distractor objects, and multiple cameras to help retrieve a relatively complete point cloud. We separated out the issue of handling point clouds with heavy occlusions from the goal of relaxing the “fixed placement object” assumption from [2]. However, as discussed in Section A9 and shown in Figure A5, the NDFs can work with partial point clouds under some specific settings. First, similar occlusion patterns should be included in the training distribution for 3D reconstruction pretraining. Second, the task-relevant part of the object that was used for obtaining the target pose descriptor should not be entirely out of view. Handling partial point clouds with full generality may require incorporating active perception that searches in camera pose space, and we decided to avoid complicating the system by factoring in this set of considerations.

New approaches to setting up the underlying NDF model (i.e., with point cloud encoders that operate on more local features) also have the potential to improve upon this issue. See the above paragraph regarding the quality of the underlying NDF models for further commentary on the potential implications of global vs. local feature encoders.

A11.2.6

Modeling assumption: We model relational rearrangement via pair-wise relations. Can this be extended to general N-body rearrangement tasks? R-NDF can be applied to rearrangement tasks with more than two objects. Depending on the nature of the multi-object task specification, there are different approaches available, including sequential localization and composing multiple NDF descriptor distance terms in the optimization objective. See Section A8 for further discussion and examples for handling rearrangement with more than two objects.

A11.2.7

Task limitation: We only show pick-and-place results. Can NDFs be used for tasks other than pick-and-place? NDFs can be used whenever it's useful to localize a coordinate frame near a task-specific local part of an object. Inferring SE(3) transformations for pick-and-place with rigid objects is one instantiation of this. However, there maybe other settings where this is useful. For example, in multi-finger manipulation, it may be useful to encode the pose of each fingertip relative to an object with a per-finger query point set and corresponding fingertip-pose descriptor.

Another point to consider is that other tasks of interest to the community may potentially be recast as a form of pick-and-place. For instance, tool-use consists of grasping an object, and then using some distal part of the grasped object to interact with the external environment. The geometric part of this "distal object part / environment" interaction could be solved in a way that is directly analogous to our "placing pose" inference. However, one fundamental limitation is that NDFs are a purely geometric representation. It would be interesting to see how to bring in dynamic/material/physical properties like mass, stiffness, friction, etc. and solve more dynamic tasks.

A11.2.8

Implications of these assumptions and limitations: There are many moving parts to the overall R-NDF system and this creates multiple potential sources of error. Which considerations impact the performance the most? There can be multiple sources of error in NDF descriptor matching. A few common error sources, roughly in order of their impact on performance, include:

- Out-of-distribution/severe point cloud noise. This can come from depth sensor noise and bad segmentation. For example, some of the real-world racks we experimented with were quite thin, shiny, and reflective, causing the depth image to have holes. The background/table was also sometimes not cleanly removed from the point cloud, which left regions containing outliers that looked different from the data the PointNet encoder was trained on. A similar issue can occur if the cameras are not accurately calibrated relative to each other.
- Inaccuracies in the demonstrations. For example, in the real world, if the object moves while in the gripper, or the placement object is accidentally bumped, the final point clouds we record might be in a different location than the true real-world objects.
- NDFs that pick up on task-irrelevant features. For example, a common failure mode for hanging the mug on the rack is to rotate the mug to almost exactly the correct orientation, but use the wrong rotation about the cylindrical axis. In this case, the NDF matches the demonstrations via a the side and the opening, rather than the handle. Similarly, a somewhat common failure mode for "bottle in container" is to place the bottle upside down, which can occur when the top and the bottom both have a locally planar geometry.

What are the most important steps to help mitigate these errors?. To complement the discussion on multiple sources of error above, we list a set of key factors to pay particular attention to for reducing the likelihood of errors occurring:

- As discussed above, out-of-distribution point clouds are a common source of error that can cause failure in descriptor matching. The best remedy for this is to train the NDF on diverse point clouds that cover the distribution that is likely to be encountered at test-time. This can come from diverse random object scaling, point cloud masking, camera viewpoints, and other forms of 3D data augmentation.
- To complement the above point, making sure that the underlying implicit representation can fit the training data well for 3D reconstruction is important to ensure that the descriptors encode something meaningful. A useful debugging step when descriptor matching has errors is to examine the reconstruction and make sure it looks reasonable.
- We have found performance to be most consistent when using a handful ($\sim$10) of demonstrations, where the demonstrations themselves are somewhat diverse. They can be diverse in the shapes that are used and the pose of the objects in each demonstration. Using more demonstrations provides a better opportunity for the average target descriptor to accurately capture the task-relevant geometry, based on what is most saliently shared across the set of diverse demos. The issue of using a single demonstration (or a set of demonstrations which fail to disambiguate the task-relevant object parts) is discussed in Section 4.1 in the paper.
- With the current version of the framework, accurate segmentation and point cloud outlier removal are important. We notice a meaningful improvement when taking serious care to completely remove everything in the background/nearby environment, and leave just the object in the point cloud.
- It is important to run the NDF optimization from multiple diverse initializations. Because the optimization objective is non-convex, there exist many local minima. We have somewhat reduced the effects of this by moving toward a DeepSDF-based NDF (rather than the original OccNet-based NDF, see Section A1), but successfully obtaining a global optima is still subject to running pose optimization from multiple initial values.
- Real-world execution requires collision-free motion planning to work properly. The whole execution is prone to fail if the placement object is accidentally bumped by the grasped object during placing. Providing intermediate waypoints to reach during the path execution that are conservatively far away from any potential collisions increases the likelihood of success. We also assume the object doesn't move in the grasp while it's moving to the placing pose. More robustness could be achieved by tracking the motion of the object in the grasp to take any unanticipated motion into account and adjust the final placing pose.

A12 Model Architecture Diagrams

VN-LinearLeakyReLU 128
Linear 128
VN-ResnetBlock FC 128
VN-ResnetBlock FC 128
VN-ResnetBlock FC 128
VN-ResnetBlock FC 128
VN-ResnetBlock FC 128
Global Mean Pooling
VN-LinearLeakyReLU → 256

Table 1: Vector Neuron point cloud encoder architecture

ResnetBlock FC 128
ResnetBlock FC 128
ResnetBlock FC 128
ResnetBlock FC 128
ResnetBlock FC 128
Linear → 1

Table 2: Signed-distance function decoder architecture

Dense → 128
Dense → 128
Dense → 6

Table 3: Architecture of EBM capturing relations.

Dense → 256
Dense → 256
Dense → 6

Table 4: Architecture of pose regression baseline.

References

[1] 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. 1, 2

[2] 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. 1, 3, 6, 8, 11, 15

[3] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 1, 3, 14

[4] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*, 2019. 1, 14

[5] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016. 3, 5

[6] 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 *ICCV*, 2021. URL `https://arxiv.org/abs/2104.12229`. 3

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

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[9] J. Sola, J. Deray, and D. Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018. 4

[10] T. Chen, A. Simeonov, and P. Agrawal. AIRobot. https://github.com/Improbable-AI/airobot, 2019. 5

[11] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 5

[12] A. Ganapathi, P. Florence, J. Varley, K. Burns, K. Goldberg, and A. Zeng. Implicit kinematic policies: Unifying joint and cartesian action spaces in end-to-end robot learning. *arXiv*, 2022. 10

[13] J. Urain, P. Liu, A. Li, C. D'Eramo, and J. Peters. Composable Energy Policies for Reactive Motion Generation and Reinforcement Learning . In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi:10.15607/RSS.2021.XVII.052. 10

[14] Y. Du, S. Li, and I. Mordatch. Compositional visual generation with energy based models. In *Advances in Neural Information Processing Systems*, 2020. 10

[15] Y. Du, S. Li, Y. Sharma, B. J. Tenenbaum, and I. Mordatch. Unsupervised learning of compositional energy concepts. In *Advances in Neural Information Processing Systems*, 2021.

[16] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum. Compositional visual generation with composable diffusion models. *ECCV*, 2022.

[17] N. Liu, S. Li, Y. Du, J. Tenenbaum, and A. Torralba. Learning to compose visual relations. *Advances in Neural Information Processing Systems*, 34, 2021. 10

[18] Y. You, L. Shao, T. Migimatsu, and J. Bohg. Omnihang: Learning to hang arbitrary objects using contact point correspondences and neural collision estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5921–5927. IEEE, 2021. 14

[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. 14, 15

[20] W. Gao and R. Tedrake. kpam-sc: Generalizable manipulation planning using keypoint affordance and shape completion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6527–6533, 2021. doi:10.1109/ICRA48506.2021.9561428. 14

[21] W. Gao and R. Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *IEEE Robotics and Automation Letters*, 6(2):2962–2969, 2021.

[22] 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.

[23] L. Yen-Chen, P. Florence, J. T. Barron, T.-Y. Lin, A. Rodriguez, and P. Isola. NeRF-Supervision: Learning dense object descriptors from neural radiance fields. In *ICRA*, 2022.

[24] S. Duggal and D. Pathak. Topologically-aware deformation fields for single-view 3d reconstruction. *CVPR*, 2022. 15

[25] Y. Deng, J. Yang, and X. Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10286–10296, 2021. 14, 15

[26] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In *Proc. ECCV*, 2020. 14, 15

[27] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 15

[28] Y. Xiang, C. Xie, A. Mousavian, and D. Fox. Learning rgb-d feature embeddings for unseen object instance segmentation. In *Conference on Robot Learning*, pages 461–470. PMLR, 2021.

[29] S. Back, J. Lee, T. Kim, S. Noh, R. Kang, S. Bak, and K. Lee. Unseen object amodal instance segmentation via hierarchical occlusion modeling. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5085–5092. IEEE, 2022.

[30] C. Xie, Y. Xiang, A. Mousavian, and D. Fox. Unseen object instance segmentation for robotic environments. *IEEE Transactions on Robotics*, 37(5):1343–1359, 2021. 15

[31] A. Mousavian, C. Eppner, and D. Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages

2901–2910, 2019. 15

[32] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444. IEEE, 2021. 15

[33] H. Ryu, J.-H. Lee, H.-i. Lee, and J. Choi. Equivariant descriptor fields: Se (3)-equivariant energy-based models for end-to-end visual robotic manipulation learning. *arXiv preprint arXiv:2206.08321*, 2022. 15

[34] E. Chatzipantazis, S. Pertigkiozoglou, E. Dobriban, and K. Daniilidis. Se (3)-equivariant attention networks for shape reconstruction in function space. *arXiv preprint arXiv:2204.02394*, 2022.

[35] Y. Chen, B. Fernando, H. Bilen, M. Nießner, and E. Gavves. 3d equivariant graph implicit functions. *arXiv preprint arXiv:2203.17178*, 2022. 15

[36] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser. Local implicit grid representations for 3d scenes. In *Proc. CVPR*, pages 6001–6010, 2020. 15

[37] J. Chibane, T. Alldieck, and G. Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020.

[38] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020.

[39] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. ISSN 1467-8659. doi:10.1111/cgf.14505. 15