

Appendix for

T²SQNet: A Recognition Model for Manipulating Partially Observed Transparent Tableware Objects

Anonymous Author(s)
Affiliation
Address
email

1	Contents	
2	A Related Works	2
3	A.1 NeRF-based Transparent Objects Recognition for Manipulation	2
4	A.2 Learning-based Transparent Objects Recognition for Manipulation	2
5	B Implementation Details for Our Methods	4
6	B.1 Details for TablewareNet Objects	4
7	B.2 Details for Tableware Dataset	7
8	B.3 Examples of Class Supplementation Using Deformable Superquadrics	7
9	B.4 Details for Model Architecture and Training Process	7
10	B.4.1 Bounding Box Prediction Model Architecture	7
11	B.4.2 Bounding Box Prediction Model Training	8
12	B.4.3 3D Voxel Representation of Smoothed Visual Hull	8
13	B.4.4 Tableware Parameter Prediction Model Architecture	8
14	B.4.5 Tableware Parameter Prediction Model Training	8
15	B.5 Details for Geometry-Aware Object Manipulation	10
16	C Experimental Details	11
17	C.1 Additional Details for Recognition Experiments	11
18	C.2 Additional Details for Sequential Declutter Experiment	11
19	C.3 Additional Details for Target Retrieval Experiment	12
20	D Additional Experimental Results	13
21	D.1 Additional Results for Recognition Experiments	13
22	D.2 Additional Results for Sequential Declutter Experiments	15
23	D.3 Additional Results for Target Retrieval Experiments	17
24	D.4 Comparison of T ² SQNet with an End-to-End Method	18
25	D.5 Supplementary Videos for Pushing Manipulation Experiments	18

26 A Related Works

27 A.1 NeRF-based Transparent Objects Recognition for Manipulation

28 NeRF [1] and its variants use differentiable rendering to find optimal 3D vision models (e.g., 3D
29 color fields, occupancy fields) from multiple 2D RGB images. Due to NeRF’s ability to learn 3D
30 structures without using depth images, several studies [2, 3] have attempted to use NeRF for grasping
31 transparent objects. Dex-NeRF [2] has first proposed transparent object recognition and grasping
32 using NeRF, introducing a depth rendering technique suitable for grasping differing from traditional
33 NeRF approaches. This work utilizes the rendered depth images to grasp objects using a pre-trained
34 Dex-Net [4]. Building on this, Evo-NeRF [3] has enhanced performance by improving the time
35 efficiency of the training process and adding a geometry regularizer term. Recently, NFL [5] has
36 employed pre-trained mask estimators and 2D normal field estimators to more accurately capture
37 3D geometry.

38 The primary limitation of these NeRF-based methods is their dependency on having access to all-
39 around, hemispheric views. When only partial views are available, these methods experience catas-
40 trophic failures. Furthermore, while NeRFs effectively capture the overall geometry of a scene, they
41 do not provide details about specific object instances. This absence of instance-specific information
42 poses significant challenges for tasks requiring target-driven manipulation.

43 A.2 Learning-based Transparent Objects Recognition for Manipulation

44 In this section, we summarize learning-based methods developed for transparent object recognition
45 and manipulation. One of the dominant approaches attempts to refine a corrupted depth image using
46 information from an RGB image, and then apply existing depth image-based grasp pose generation
47 algorithms [6]. For instance, Cleargrasp [7] has proposed a method that takes an RGB-D image of
48 a scene containing transparent objects, uses the RGB portion to estimate surface normals, detect
49 boundaries, and segment objects with a neural network, and then globally optimizes these outputs
50 to restore the damaged depth image. DepthGrasp [8] has developed a GAN-based generator to
51 directly produce a completed depth image from raw RGB-D input. Similarly, TransCG [9] has
52 provided a dataset of raw and refined depth images of real-world transparent objects and train a
53 U-Net structure neural network to predict refined depth images from RGB images and incomplete
54 depth images. Moreover, SwinDRNet [10] has proposed depth image completion using a two-stream
55 Swin Transformer [11], introducing domain randomization to tackle sim-to-real domain shift issues.

56 However, the grasp pose generation methods based on depth images have fundamental limitations.
57 First, full 6-DoF grasping is not possible, resulting in reduced diversity of grasp poses. Second,
58 there is often a lack of information on the 3D geometry of objects, making it difficult to generate
59 collision-free grasping trajectories. Third, most cases do not involve instance-wise representation,
60 which complicates target-driven manipulation.

61 GraspNeRF [12] predicts TSDF values from multiple RGB images and trains a model to predict
62 grasp poses using VGN [13]. This method overcomes some of the disadvantages mentioned above.
63 Since it directly generates grasp poses, it can also be trained to create 6-DoF grasp poses. Addition-
64 ally, because it outputs TSDF values, it has information on 3D geometry, enabling the generation of
65 collision-free trajectories. However, since the current version does not provide instance information,
66 target-aware manipulation is challenging. One of the most significant differences in our research is
67 that we use extended deformable superquadrics for 3D scene representation, which, compared to
68 TSDF, is more memory-efficient and allows for much faster collision checks and grasp sampling.

69 Lastly, we would like to emphasize that all the learning-based methods mentioned above face a
70 sim-to-real issue. Because RGB images in simulation differ from real images, models trained on
71 simulated RGB images commonly experience a significant performance drop when applied to real
72 images. It would be ideal to use real-world data, but collecting it is quite challenging. Although
73 various methods have been introduced in previous studies [10, 12] to enhance robustness to the
74 sim-to-real gap, none have been particularly effective. In our research, we developed a model that

75 uses mask images as input, which we found to be significantly more robust to the sim-to-real gap
76 compared to models that directly take RGB images as input.

77 B Implementation Details for Our Methods

78 B.1 Details for TablewareNet Objects

79 In this section, we describe the constraints on the superquadric parameters and poses for each table-
80 ware object mentioned in Section 2.2. – wine glasses, bottles, beer bottles, bowls, dishes, handleless
81 cups, and mugs – and explain how these constraints are represented using tableware parameters.

82 **Notations.** Following the notation used in Section 2.2., each tableware object is composed of n
83 superquadrics $\{S_i | i = 1, \dots, n\}$, with their function type, pose, size parameter, and shape parameter
84 denoted as f_i, T_i, \mathbf{a}_i , and \mathbf{e}_i , respectively. Additionally, their deformation parameters are denoted as
85 $\mathbf{d}_i = (k_i, b_i, \alpha_i, s_i)$. If there is no deformation, $\mathbf{d}_i = (0, 0, 0, 0)$; we denote the absence of bending
86 as $b_i = 0$ for convenience, as bending diminishes when b approaches zero. In actual implementation,
87 deformation D_b is not computed when $b_i = 0$.

88 **Wine Glass.** The wine glass template encompasses a wide range of wine glass and champagne glass
89 shapes. This template consists of three superquadrics: two superellipsoids S_1, S_2 for the foot and
90 stem, and a superparaboloid S_3 for the bowl. The tableware parameters and the constraints on the
91 superquadrics are given in Figure 1 and Table 1.

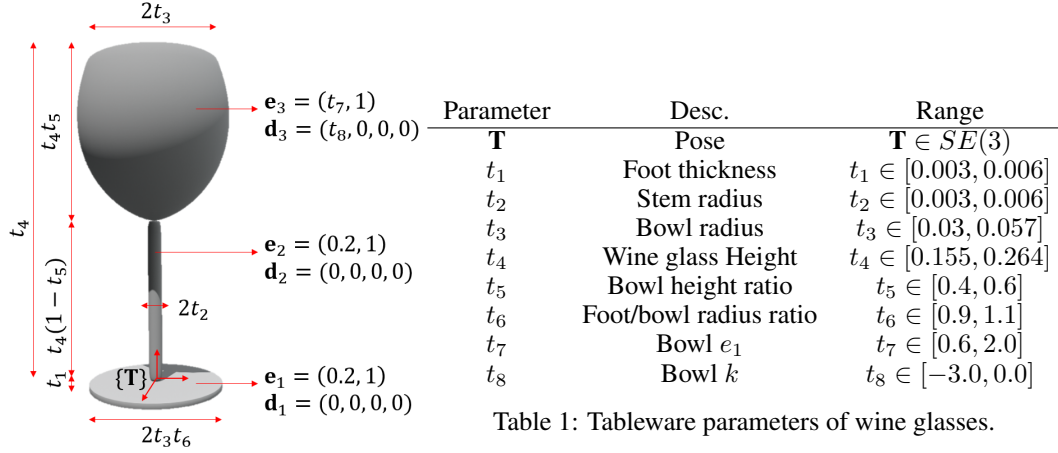


Table 1: Tableware parameters of wine glasses.

Figure 1: Description for the wine glass parameters.

92 **Bowl.** The bowl template consists of one superquadric: a superparaboloid S_1 for the bowl. The
93 tableware parameters are given in Figure 2 and Table 2.

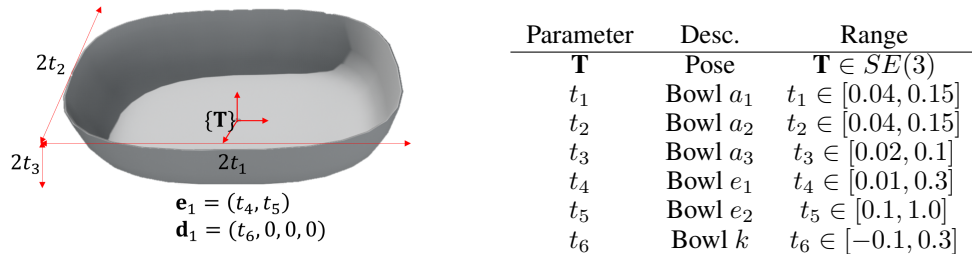
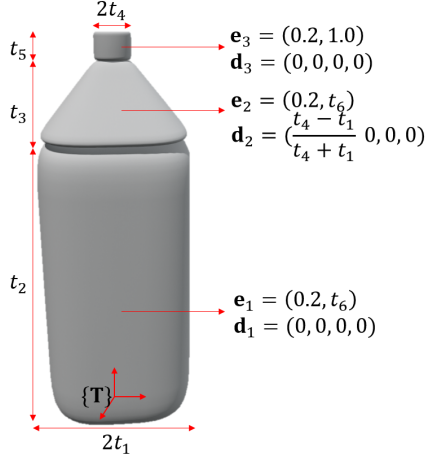


Table 2: Tableware parameters of bowls.

Figure 2: Description for the bowl parameters.

94 **Bottle.** The bottle template consists of three superquadrics: superellipsoids S_1, S_2, S_3 for the body,
 95 shoulder and finish. The tableware parameters and the constraints on the superquadrics are given in
 96 Figure 3 and Table 3.

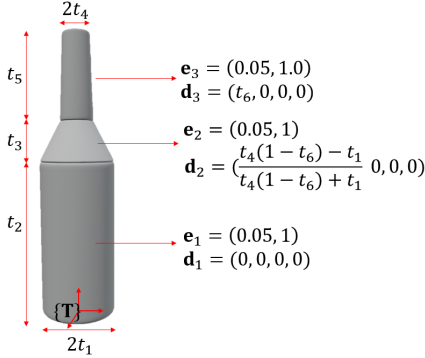


Parameter	Desc.	Range
\mathbf{T}	Pose	$\mathbf{T} \in SE(3)$
t_1	Body radius	$t_1 \in [0.03, 0.055]$
t_2	Body height	$t_2 \in [0.10, 0.23]$
t_3	Shoulder height	$t_3 \in [0.03, 0.05]$
t_4	Finish radius	$t_4 \in [0.008, 0.012]$
t_5	Finish height	$t_5 \in [0.01, 0.02]$
t_6	Overall e_2	$t_6 \in [0.2, 1.0]$

Table 3: Tableware parameters of bottles.

Figure 3: Description for the bottle parameters.

97 **Beer bottle.** The beer bottle template consists of three superquadrics: superellipsoids S_1, S_2, S_3 for
 98 the body, shoulder and neck. The tableware parameters and the constraints on the superquadrics are
 99 given in Figure 4 and Table 4.

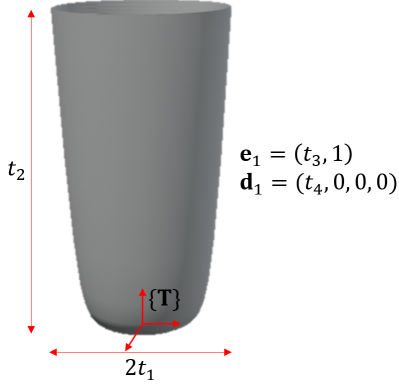


Parameter	Desc.	Range
\mathbf{T}	Pose	$\mathbf{T} \in SE(3)$
t_1	Body radius	$t_1 \in [0.025, 0.05]$
t_2	Body height	$t_2 \in [0.12, 0.19]$
t_3	Shoulder height	$t_3 \in [0.01, 0.07]$
t_4	Neck radius	$t_4 \in [0.014, 0.016]$
t_5	Neck height	$t_5 \in [0.07, 0.1]$
t_6	Neck k	$t_6 \in [-0.2, 0.0]$

Table 4: Tableware parameters of beer bottles.

Figure 4: Description for the beer bottle parameters.

100 **Handless Cup.** The handless cup template consists of one superquadric: a superparaboloid S_1 . The
 101 tableware parameters are given in Figure 5 and Table 5.

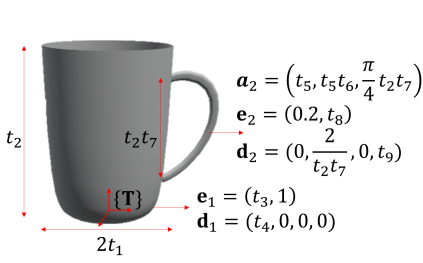


Parameter	Desc.	Range
\mathbf{T}	Pose	$\mathbf{T} \in SE(3)$
t_1	Cup radius	$t_1 \in [0.025, 0.05]$
t_2	Cup height	$t_2 \in [0.05, 0.22]$
t_3	Cup e_1	$t_3 \in [0.01, 0.3]$
t_4	Cup k	$t_4 \in [0.0, 0.3]$

Table 5: Tableware parameters of handless cups.

Figure 5: Description for the handless cup parameters.

102 **Mug.** The mug template consists of two superquadric: superparaboloids S_1 , S_2 for the cup and
 103 handle. The tableware parameters and the constraints on the superquadrics are given in Figure 6 and
 104 Table 6.

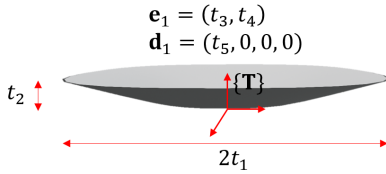


Parameter	Desc.	Range
\mathbf{T}	Pose	$\mathbf{T} \in SE(3)$
t_1	Cup radius	$t_1 \in [0.025, 0.05]$
t_2	Cup height	$t_2 \in [0.08, 0.12]$
t_3	Cup e_1	$t_3 \in [0.01, 0.3]$
t_4	Cup k	$t_4 \in [-0.2, 0.2]$
t_5	Handle a_1	$t_5 \in [0.002, 0.003]$
t_6	Handle a_2/a_1	$t_6 \in [1.0, 2.0]$
t_7	Handle length ratio	$t_7 \in [0.5, 0.7]$
t_8	Handle e_2	$t_8 \in [0.2, 1.0]$
t_9	Handle s	$t_9 \in [-0.5, -0.0001]$

Figure 6: Description for the mug parameters.

Table 6: Tableware parameters of mugs.

105 **Dish.** The dish template consists of one superquadric: a superparaboloid S_1 . The tableware param-
 106 eters are given in Figure 7 and Table 7.



Parameter	Desc.	Range
\mathbf{T}	Pose	$\mathbf{T} \in SE(3)$
t_1	Dish radius	$t_1 \in [0.08, 0.14]$
t_2	Dish height	$t_2 \in [0.015, 0.03]$
t_3	Dish e_1	$t_3 \in [0.01, 0.3]$
t_4	Dish e_2	$t_4 \in [0.5, 1.0]$
t_5	Dish k	$t_5 \in [0.3, 0.6]$

Figure 7: Description for the dish parameters.

Table 7: Tableware parameters of dishes.

107 B.2 Details for Tableware Dataset

108 Our dataset employs uniform random sampling for each object class, offering infinitely continuous
 109 variations in object shapes, unlike other datasets with a finite number of fixed shapes. We capture
 110 comprehensive data, including object poses, tableware parameters, class labels, bounding boxes,
 111 meshes, and TSDF values. For each scene, we provide mask images, depth images, and RGB
 112 images from seven different camera poses using the synthetic camera parameters of the RealSense
 113 D435. The dataset features two versions: one with objects on a shelf and another with objects on
 114 a table. Each includes 120,000 scenes, with 30,000 scenes each containing 1, 2, 3, or 4 objects.
 115 Among them, 12,000 scenes include RGB images; we do not capture RGB images for other scenes.
 116 Therefore, the dataset comprises 840,000 depth and mask images and 84,000 RGB images.

117 B.3 Examples of Class Supplementation Using Deformable Superquadrics

118 In this section, we present examples of additional objects that can be generated using de-
 119 formable superquadrics, which are not covered in our dataset. **Perfume Bottle.** Perfume bot-
 120 tles come in various shapes and are transpar-
 121 ent objects commonly seen in daily life. How-
 122 ever, they are not included in our dataset be-
 123 cause they are not tableware. The perfume bot-
 124 tle template consists of two superquadrics: su-
 125 perellipsoids S_1, S_2 for the body and cap. **Plas-
 126 tic Cup with Spherical Lid.** Plastic cups with
 127 spherical lids are easily seen in the most cof-
 128 fee shops. This template consists of two su-
 129 perquadrics: superellipsoids S_1, S_2 for the cup
 130 and lid. The example of perfume bottle and cup
 131 with spherical lid represented by deformable
 132 superquadrics can be found in Figure 8.

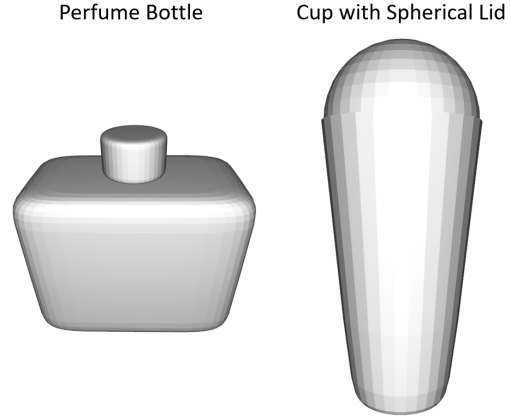


Figure 8: Perfume bottle and plastic cup with spherical lid represented by deformable superquadrics.

135 B.4 Details for Model Architecture and Training Process

136 B.4.1 Bounding Box Prediction Model Architecture

137 In this section, we describe the structure of the neural network used for bounding box prediction.
 138 We utilize the architecture of DETR3D [14], which takes fixed multi-view RGB images as input
 139 and outputs multiple 3D bounding boxes. DETR3D first obtains 2D features from the multi-view
 140 images using ResNet [15] and FPN [16]. It employs a Transformer architecture, where each layer
 141 decodes candidate positions for bounding box centers using a sub-network for object queries. These
 142 decoded positions are projected onto each image plane, and the corresponding image feature values
 143 are incorporated into the object queries using multi-head attention. The refined object queries are
 144 then used as input for the next layer. From each object query passing through the layers, the bound-
 145 ing box center, size, and class are predicted using an MLP structure for training. During inference,
 146 the bounding boxes and classes are predicted from the object queries of the final layer. For more
 147 details, refer to [14].

148 We adopt the DETR3D structure with a few modifications. Instead of using a classifier to predict the
 149 object class for each object query, we pre-assign classes to the object queries and use a confidence
 150 predictor to predict a value in $[0, 1]$ whether the query corresponds to a present object. For example,
 151 if there are 3500 object queries and 7 classes, the first 500 queries represent the first class, and if
 152 the confidence predictor assigns high confidence (we use a threshold value of 0.75) to one of these
 153 queries, it is predicted that a bounding box for an object of that class exists. Unlike the original
 154 DETR3D using RGB image inputs, we use binary mask images as inputs.

155 B.4.2 Bounding Box Prediction Model Training

156 **Matching and Loss Calculation.** As in the original DETR3D, we use bipartite matching to cal-
157 culate the loss. Since we have pre-assigned classes to each query, the matching occurs within each
158 class. We use binary cross-entropy for the confidence scores and L_1 loss for the bounding box co-
159 ordinates and size. The weights between the binary cross-entropy loss and the L_1 loss are set to 5:1.
160 After calculating the loss for the optimal matching within each class, we sum these losses across all
161 classes to obtain the final loss value.

162 **Training Process.** We train the network using 12,000 samples from the TablewareNet dataset;
163 although our model can utilize the entire 120,000 sample dataset, we use only the 12,000 samples
164 that include RGB images for comparison with other RGB-based baselines. We employ the Adam
165 optimizer with an initial learning rate of 0.00005. A cosine annealing scheduler is used, and the
166 training process spans 200 epochs.

167 **Cutout Augmentation.** For cutout augmentation, the number of holes per mask image is determined
168 by uniformly sampling an integer between 0 and 2. The size of each hole is determined by uniformly
169 sampling an integer between 50 and 100 for both the width and height; note that mask image size is
170 (240, 320). The position of each hole is also uniformly sampled within the image pixels.

171 B.4.3 3D Voxel Representation of Smoothed Visual Hull

172 We need to voxelize the predicted bounding boxes with fixed voxel size L_i , to create the raw 3D
173 space to be carved. Here, i represents the class index. Given that bounding boxes can vary in size,
174 voxelizing them directly would result in different resolution of voxels, disabling the inference of the
175 3D CNN and FCN architecture. To address this, we follow a specific procedure to standardize the
176 raw voxel representation.

177 First, we inspect the all size of the ground truth bounding boxes of class i and store the maximum size
178 ($W_{i_{\max}}, H_{i_{\max}}, D_{i_{\max}}$). Using this maximum size, we create a standardized 3D voxel space centered
179 on the predicted bbox center. This ensures that all voxel representations have a consistent resolution
180 ($W_{i_{\max}}/L_i, H_{i_{\max}}/L_i, D_{i_{\max}}/L_i$).

181 Next, we calculate the smoothed visual hull within this standardized voxel space. To further refine
182 the representation, we add a channel to the voxel grid, where voxels inside the predicted bounding
183 box are assigned a value of 1, and those outside are assigned a value of 0. The final voxel input
184 to the network thus takes the shape $(W_{i_{\max}}/L_i, H_{i_{\max}}/L_i, D_{i_{\max}}/L_i, 2)$, with the first channel repre-
185 senting the smoothed visual hull and the second channel indicating bounding box occupancy. This
186 standardized approach ensures consistent and accurate input for the 3D CNN, facilitating reliable
187 network inference.

188 B.4.4 Tableware Parameter Prediction Model Architecture

189 We employ a ResNet3D [17] + FCN architecture to predict the tableware parameters, including the
190 pose \mathbf{T} , using the smoothed visual hull voxel as input. Given that each tableware class has different
191 voxel resolutions and the number of parameters, we train separate predictors for each class. In other
192 words, there are as parameter predictors as the number of the classes, with each predictor dedicated
193 to estimating the parameters of a single class of tableware objects.

194 In addition, we assume that all objects are upright. Therefore, instead of predicting the entire $SE(3)$
195 matrix for the object pose \mathbf{T} , the model only predict the relative position $p \in \mathbb{R}^3$ to the bounding
196 box center and the rotation around the z -axis, $\theta \in [0, 2\pi)$.

197 B.4.5 Tableware Parameter Prediction Model Training

198 **Chamfer Distance as Loss function.** The tableware parameters output by the parameter predictor
199 could be directly trained via supervised learning using the L_1 or L_2 distance from the ground truth
200 tableware parameters. However, this approach may not yield a zero loss value even when the pre-

dicted object shape matches the ground truth shape due to symmetrical ambiguities. For instance, orientations of objects with circular symmetry such as wine glasses, beer bottles, handleless cups, and dishes, do not affect their shapes. In the case of a bottle, rotating the object by 90 degrees results in the same shape, thus the ground truth shape has four possible orientations. For bowls, swapping the width t_1 and length t_2 and rotating the orientation by 90 degrees along the z -axis results in the same shape. We want a loss function that yields a zero value when the predicted shape matches the ground truth shape, regardless of these symmetrical ambiguities.

To address this, we use the Chamfer distance between point clouds sampled from the predicted and ground truth object surfaces as the loss function. This requires a differentiable point cloud sampling method from deformable superquadric parameters, which we will discuss later. For a tableware object composed of multiple superquadrics S_1, \dots, S_n , we compute the Chamfer distance for each deformable superquadric part. Specifically, if $P_{i,\text{pred}}$ and $P_{i,\text{gt}}$ are the point clouds sampled from the predicted and ground truth i -th deformable superquadric $S_{i,\text{pred}}$ and $S_{i,\text{gt}}$, respectively, the Chamfer loss is defined as:

$$\sum_{i=1}^n \text{chamfer}(P_{i,\text{pred}}, P_{i,\text{gt}})$$

Differentiable Point Sampling on Deformable Superquadrics. For superquadrics without deformation, there exists an explicit parameterization as follows;

$$\mathbf{x}_{se} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_1 \cos^{e_1} \theta \cos^{e_2} \phi \\ a_2 \cos^{e_1} \theta \sin^{e_2} \phi \\ a_3 \sin^{e_1} \theta \end{bmatrix}, \mathbf{x}_{sp} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_1 h \cos^{e_2} \phi \\ a_2 h \sin^{e_2} \phi \\ a_3 (h^{2/e_1} - 1) \end{bmatrix},$$

where $-\pi/2 \leq \theta \leq \pi/2$, $-\pi \leq \phi \leq \pi$ and $0 \leq h \leq 1$, and $\cos^e \theta := \text{sgn}(\cos \theta) |\cos \theta|^e$ and $\sin^e \theta := \text{sgn}(\sin \theta) |\sin \theta|^e$.

We utilize a uniform grid in θ , ϕ , and h coordinates to sample points on the superquadric surface using this explicit parameterization. This method is differentiable with respect to the superquadric parameters.

When dealing with deformations, we apply the transformations D_t , D_b , and D_s described in Section 2.1 to these sampled points. These transformations are also differentiable with respect to deformation parameters.

Training Process. During training, we observe empirically that adding L_1 loss for tableware parameters (excluding the object’s pose) and MSE loss for position as regularizers accelerate the training process. We do not use a regularizer for orientation due to symmetry issues previously discussed. The weights used between the position MSE loss, parameter L_1 loss, and Chamfer loss were set to 1 : 0.1 : 1. The model is trained with 12,000 data which include RGB images. The input smoothed visual hulls are generated using ground truth bounding boxes and mask images for each scene. Similar to the bounding box predictor training, we employed the Adam optimizer with an initial learning rate of 0.00005, using a cosine annealing scheduler. The training process spanned 200 epochs.

Data Augmentation with Bounding Box Perturbation. During inference, errors in the bounding box predictions may lead to low performance of the parameter predictor if it is trained solely on visual hulls generated from ground truth bounding boxes. To address this, we add noise to the bounding box’s position and size to create a perturbed visual hull during training. Specifically, as mentioned in B.4.3., while generating the voxel representation, we apply perturbations to the center and size of the bounding boxes when creating the second channel representing the bounding box occupancy. This data augmentation strategy ensures that the parameter predictor remains robust to prediction errors from the bounding box predictor, enabling T²SQNet to maintain high performance even when bounding box predictions are not perfectly accurate.

242 B.5 Details for Geometry-Aware Object Manipulation

243 **6-DoF Grasp Sampler.** After obtaining the deformable superquadric representation of the transpar-
244 ent objects, we sample feasible grasp poses from the obtained shapes. While it is possible to find the
245 antipodal point by utilizing the implicit function of the deformable superquadric and its closed-form
246 normal vector [18], we manually design a faster and more diverse grasp sampler in this paper. Since
247 each object consists of multiple deformable superquadric parts, we first generate grasp poses for
248 each part. Inspired by previous works [19], we manually generate top-down and side grasp poses
249 for the superellipsoids according to their shapes. For superparaboloids, we generate grasp poses that
250 grasp the edge. Example grasp poses can be found in Section 4. After generating grasp poses for
251 each part, we check whether the grasp poses avoid self-collision with the object. Grasp poses with
252 a distance between the antipodal points greater than 7.5 cm are removed from the candidates, as the
253 maximum gripper width of the Franka gripper is 8 cm.

254 **Object Rearrangement Method.** Using the graspability function – set to 1 if the object is graspable
255 and 0 otherwise – described in Section 5.2, we can generate robot actions that make the target object
256 graspable. We use sampling-based model predictive control (MPC) to maximize the graspability
257 function. From the recognized tableware objects, we sample 50 pick-and-place action sequences.
258 Since the dynamics of the pick-and-place actions are precisely known, we do not use an additional
259 learning-based model to predict the dynamics of the objects. Then, we find the optimal action that
260 best maximizes the graspability function.

C Experimental Details

C.1 Additional Details for Recognition Experiments

Baseline Implementations. NeRF and Dex-NeRF are trained using the seven partial views available in TablewareNet. For both methods, the view poses used for training and depth accuracy measurement are identical. Consequently, we use the model from the final epoch of training for evaluation. Depth rendering for NeRF and Dex-NeRF follows the approach described in [2], with a σ threshold of 15 for Dex-NeRF. To generate occupancy voxel grids, we render depth images from uniformly sampled view poses over a hemisphere using the trained NeRF and Dex-NeRF models. Subsequently, these depth images are converted to TSDFs, and regions with negative TSDF values are marked as occupied. DVGO and Mask DVGO are also trained using the seven partial views. For depth rendering, both methods employ the depth rendering technique used by NeRF in [2], rather than the Dex-NeRF method. Empirically, using the Dex-NeRF depth rendering approach decreases performance for DVGO-based methods. Predicted occupancy voxels are defined as those with an occupancy probability exceeding a threshold, which we set to 0.1. For GraspNeRF, we train the model with our 12,000 RGB-equipped TablewareNet dataset. We utilize rendering loss and TSDF loss, where ground truth TSDF values are used to derive the occupancy voxels directly from the predicted TSDF values. For our model, T²SQNet, we generate occupancy voxels using the implicit function representation of the superquadrics of the predicted tableware objects.

T²SPose Dataset. For the T²SPose dataset, we follow the same process as when generating the TablewareNet dataset, as described in Section 2.2. We first spawn random objects from the T²SPose object dataset [20] within PyBullet and then render RGB images of the scenes using Blender with transparent textures.

C.2 Additional Details for Sequential Declutter Experiment

We generate five scenarios for each number of objects, one (single) and four (cluttered), and for each environment type, including shelf and table, resulting in a total of 20 scenarios. The initial settings for all scenes, including the configuration and poses of objects, can be seen in Figure 9. When testing the three baselines, including Mask DVGO, GraspNeRF, and T²SQNet, we place the same objects in the same pose as consistently as possible for each scene.



Figure 9: Initial scene settings for real-world sequential declutter experiments.

289 C.3 Additional Details for Target Retrieval Experiment

290 We generate five scenarios with designated target objects in a shelf environment. The initial scene
291 settings, including the target tableware class name (e.g., wine glass), for all scenarios can be found
292 in Figure 10. For the target retrieval experiment, we place the target object so that it is not initially
293 graspable.



Figure 10: Initial scene settings for real-world target retrieval experiments.

294 D Additional Experimental Results

295 D.1 Additional Results for Recognition Experiments

296 Additional examples of transparent object recognition results are shown in Figure 11 and Figure 12.
 297 The trends of these additional results are similar to the representative example in Section 5.1. For
 298 the test sets of the Tableware dataset in Figure 11, T²SQNet succeeds in recognizing accurate 3D
 299 geometries of the transparent objects while also delivering instance information. GraspNeRF per-
 300 forms best among the baselines but predicts less accurate results than T²SQNet. Although T²SQNet
 301 has slightly lower performance on the Tableware dataset compared to TRansPose, it succeeds in
 302 predicting somewhat accurate instance-wise geometries, as shown in Figure 12.

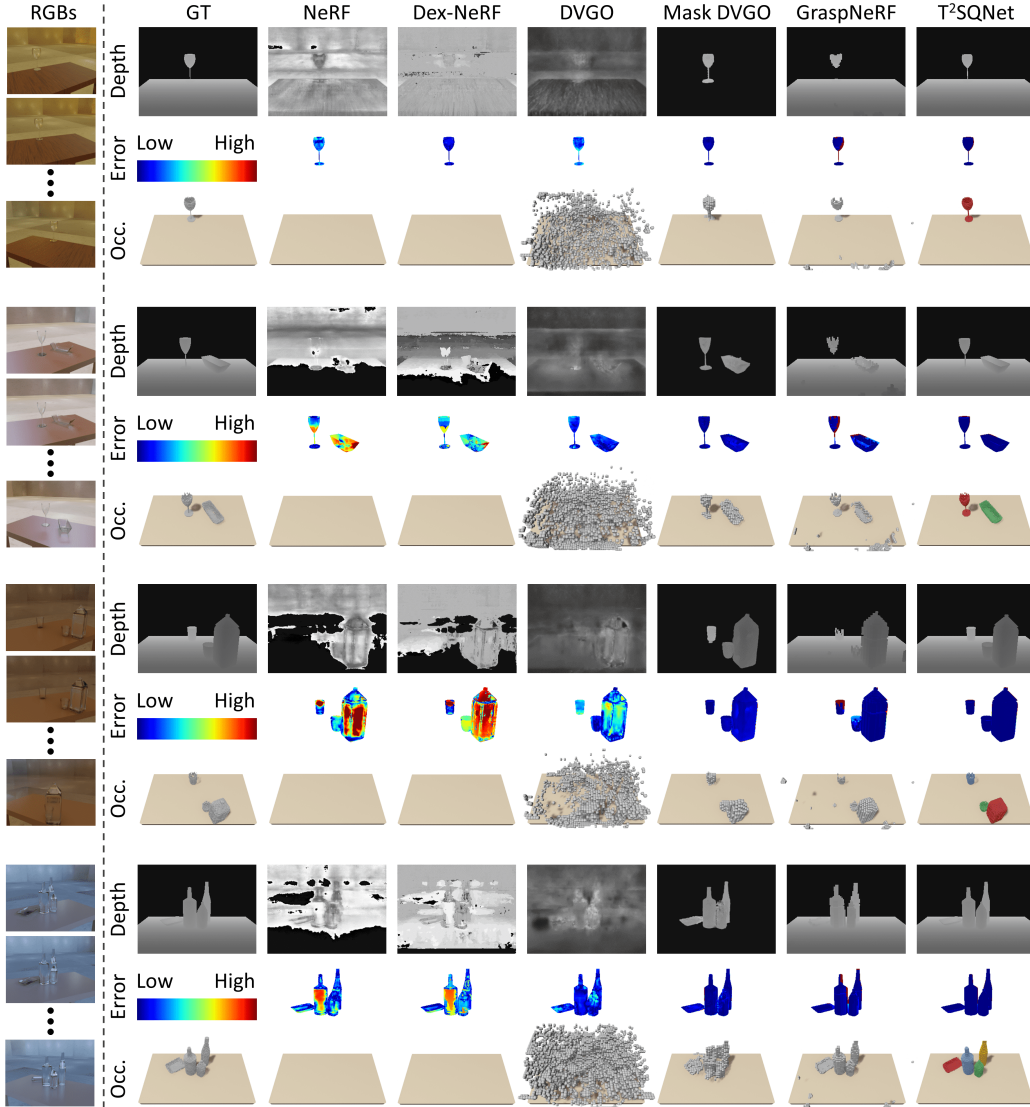


Figure 11: Recognition results from RGB images from test sets of Tableware dataset.

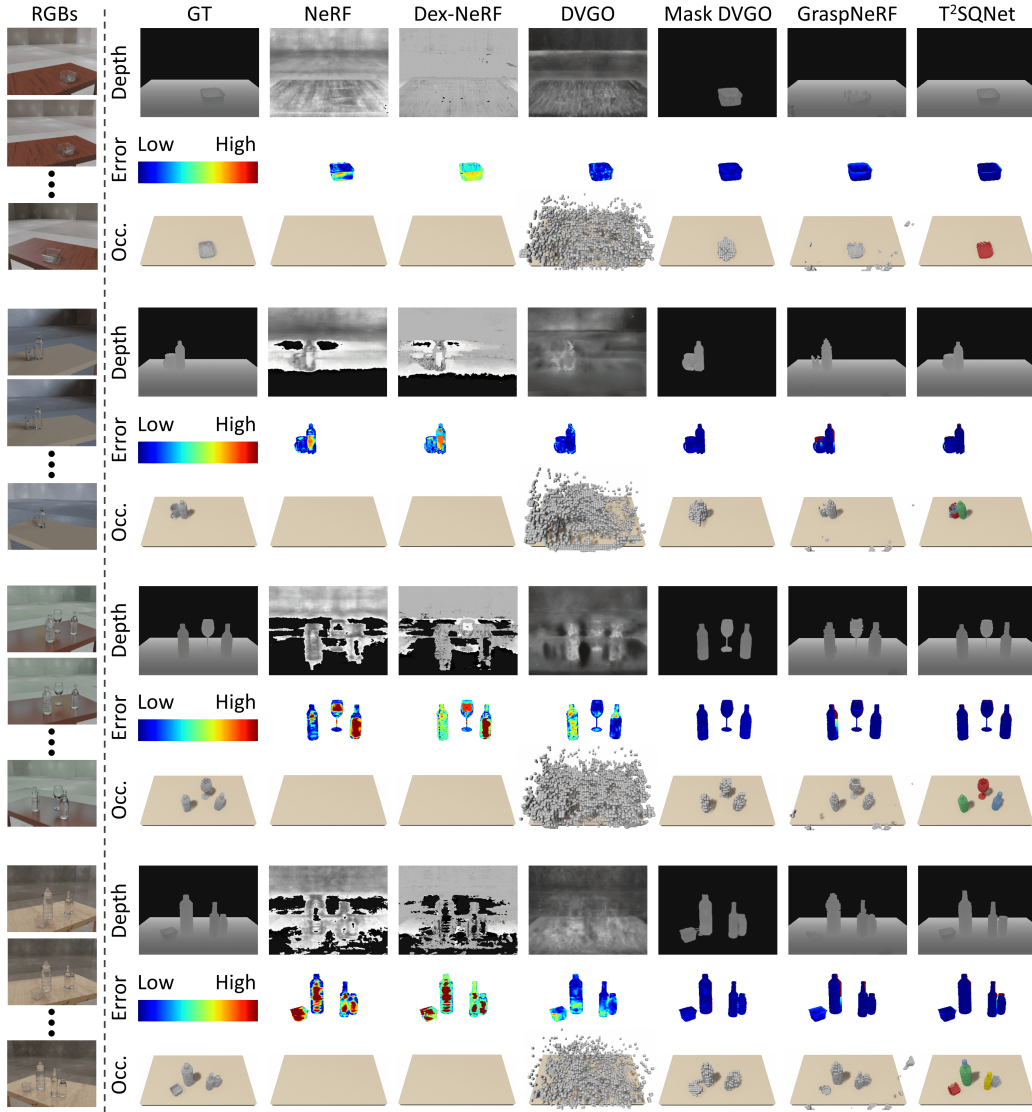


Figure 12: Recognition results from RGB images from TRansPose dataset.

303 D.2 Additional Results for Sequential Declutter Experiments

304 More examples of sequential decluttering with T²SQNet on shelves and tables are shown in Fig-
 305 ure 13 and Figure 14, respectively. Generally, our method succeeds in sequentially grasping objects
 306 without re-recognition, while avoiding collisions with other objects and the environment based on
 307 the accurately predicted geometries of the objects. However, there are several failure cases: (i) a
 308 slightly incorrect shape leads to an unstable grasp pose, as shown in the third example of Figure 13,
 309 (ii) some objects are not recognized by T²SQNet, as shown in the first example of Figure 14, and (iii)
 310 an inverse kinematics solution does not exist, as shown in the third example of Figure 14. The first
 311 and second failure cases can be resolved through a more accurate recognition model, as described in
 312 the future works section (Section 5.3). The third failure case can be addressed by designing a more
 313 diverse 6-DoF grasp sampler.



Figure 13: Examples of sequential declutter experiment results on shelves.

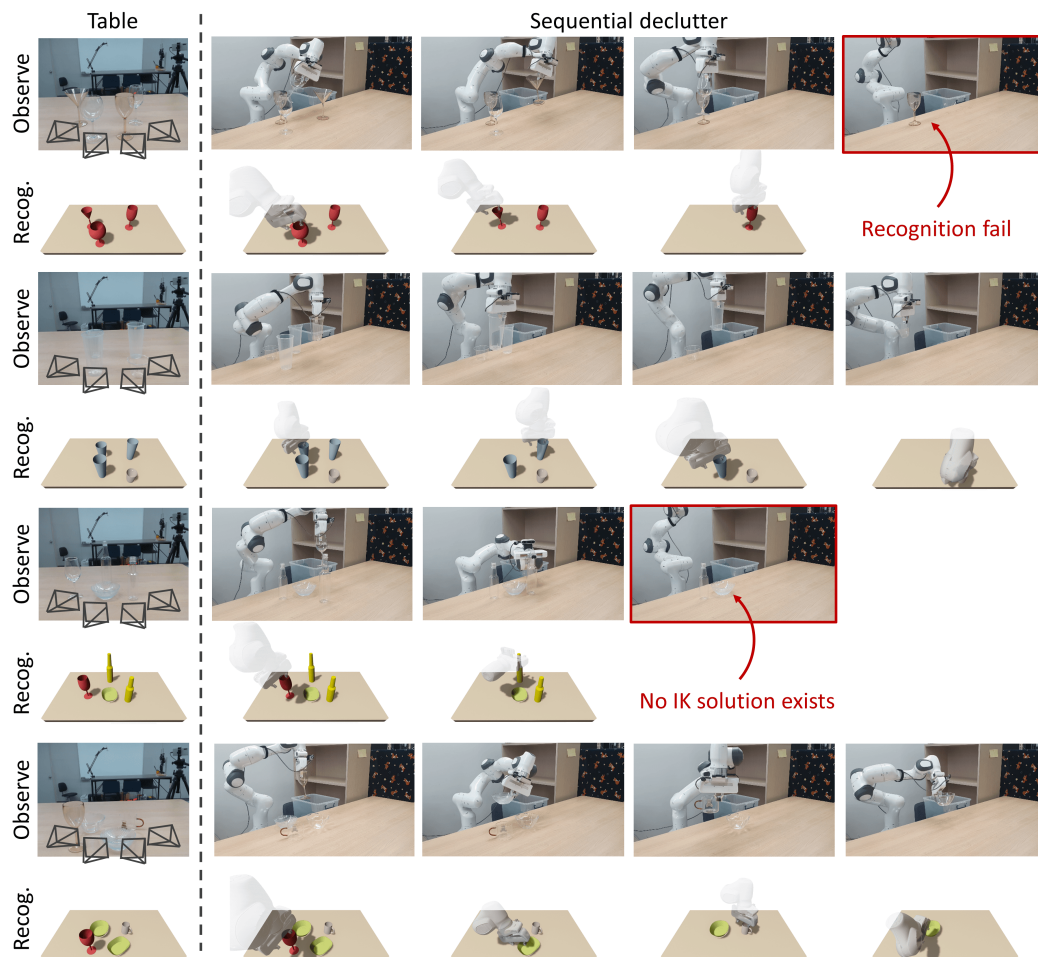


Figure 14: Examples of sequential declutter experiment results on tables.

314 D.3 Additional Results for Target Retrieval Experiments

315 Additional results for target retrieval with T²SQNet are shown in Figure 15. In the three examples
 316 above, T²SQNet-based method successfully rearranges surrounding objects and retrieves target ob-
 317 jects through appropriate pick-and-place actions. In the last example, T²SQNet fails to recognize
 318 one wine glass; consequently, the robot performs an action of directly retrieving the target object,
 319 the mug, and as a result, it grasps both the wine glass and the mug together.



Figure 15: Examples of target retrieval experiment results on shelves.

320 D.4 Comparison of T²SQNet with an End-to-End Method

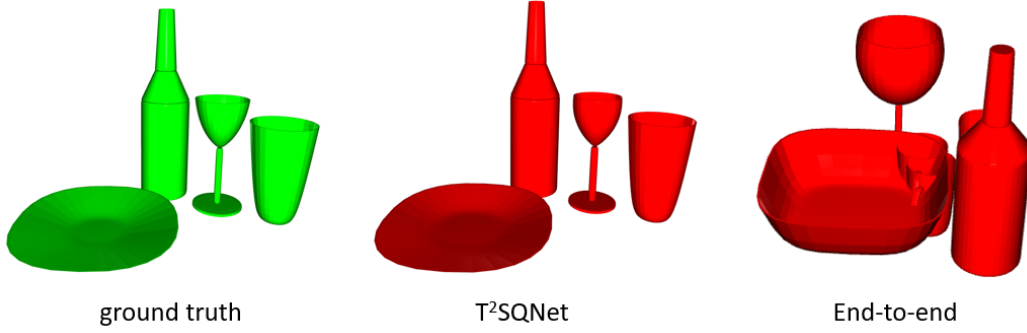


Figure 16: Visual comparison of T²SQNet output with an end-to-end method output on validation set

321 To validate the effectiveness of our T²SQNet framework, which combines several separate modules,
 322 we developed and trained a simple end-to-end model for comparison. The end-to-end model struc-
 323 ture is as follows: it utilizes our modified DETR3D structure (where queries are pre-assigned classes
 324 and a confidence estimation is incorporated) but replaces the bounding box predicting FCN with an
 325 FCN that predicts tableware parameters. Given that the dimension of tableware parameters varies by
 326 object class, separate FCNs are used for each class to predict the parameters. These class-specific
 327 FCNs take as input the queries assigned to the same class and output the corresponding tableware
 328 parameters. The learning loss for these parameters employs the same position regularizer, parameter
 329 regularizer, and chamfer loss as T²SQNet. In this end-to-end method, we set the weights for confi-
 330 dence loss, position regularizer, parameter regularizer, and chamfer loss to 1:1:1:0.1, respectively.

331 The training results are shown in Figure 16. We observed that training this end-to-end model is
 332 considerably challenging, and we hypothesize the following reasons:

333 **Chamfer Loss Scale.** As the centers of two objects diverge, the chamfer loss scale increases
 334 quadratically. During the initial training phases, failing to align the object’s position significantly
 335 inflates the chamfer loss scale relative to the confidence loss and position loss. Conversely, as ob-
 336 jects come closer, the chamfer loss scale becomes much smaller than the confidence loss and other
 337 loss terms. This large variation in loss scale during training leads to inconsistent bipartite matching
 338 results, hindering significant training progress.

339 **Positional Constraints.** In T²SQNet, the object center position is constrained to remain within the
 340 bounding box, providing a structured framework for the output. However, the end-to-end method
 341 bypasses the bounding box prediction process, resulting in the lack of such positional constraints.
 342 Consequently, the predicted object can be located anywhere within the workspace during the initial
 343 training stages, making it difficult to resolve the aforementioned issues.

344 In conclusion, the large scale variation of the chamfer loss during training makes it challenging
 345 to balance the losses, leading to unstable bipartite matching results and ultimately hindering the
 346 learning process. While the end-to-end method theoretically streamlines the process, the practical
 347 challenges in training and the inherent issues in loss scaling and positional constraints highlight the
 348 advantages of our modular T²SQNet framework.

349 D.5 Supplementary Videos for Pushing Manipulation Experiments

350 Videos for real-world sequential declutter and target retrieval can be found at the following link:

351 <https://www.youtube.com/watch?v=qUWOp6wUHb8>.

References

- [1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
- [2] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. *arXiv preprint arXiv:2110.14217*, 2021.
- [3] J. Kerr, L. Fu, H. Huang, Y. Avigal, M. Tancik, J. Ichnowski, A. Kanazawa, and K. Goldberg. Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects. In *6th annual conference on robot learning*, 2022.
- [4] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.
- [5] J. Lee, S. M. Kim, Y. Lee, and Y. M. Kim. Nfl: Normal field learning for 6-dof grasping of transparent objects. *IEEE Robotics and Automation Letters*, 2023.
- [6] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *The International Journal of Robotics Research*, 41(7): 690–705, 2022.
- [7] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song. Clear grasp: 3d shape estimation of transparent objects for manipulation. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 3634–3642. IEEE, 2020.
- [8] Y. Tang, J. Chen, Z. Yang, Z. Lin, Q. Li, and W. Liu. Depthgrasp: Depth completion of transparent objects using self-attentive adversarial network with spectral residual for grasping. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5710–5716. IEEE, 2021.
- [9] H. Fang, H.-S. Fang, S. Xu, and C. Lu. Transcg: A large-scale real-world dataset for transparent object depth completion and a grasping baseline. *IEEE Robotics and Automation Letters*, 7(3):7383–7390, 2022.
- [10] Q. Dai, J. Zhang, Q. Li, T. Wu, H. Dong, Z. Liu, P. Tan, and H. Wang. Domain randomization-enhanced depth simulation and restoration for perceiving and grasping specular and transparent objects. In *European Conference on Computer Vision*, pages 374–391. Springer, 2022.
- [11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.
- [12] Q. Dai, Y. Zhu, Y. Geng, C. Ruan, J. Zhang, and H. Wang. Graspnerf: Multiview-based 6-dof grasp detection for transparent and specular objects using generalizable nerf. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1757–1763. IEEE, 2023.
- [13] M. Breyer, J. J. Chung, L. Ott, S. Roland, and N. Juan. Volumetric grasping network: Real-time 6 dof grasp detection in clutter. In *Conference on Robot Learning*, 2020.
- [14] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, pages 180–191. PMLR, 2022.
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

- 396 [16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid
397 networks for object detection. In *Proceedings of the IEEE conference on computer vision and*
398 *pattern recognition*, pages 2117–2125, 2017.
- 399 [17] H. Kataoka, T. Wakamiya, K. Hara, and Y. Satoh. Would mega-scale datasets further enhance
400 spatiotemporal 3d cnns? *arXiv preprint arXiv:2004.04968*, 2020.
- 401 [18] S. Kim, T. Ahn, Y. Lee, J. Kim, M. Y. Wang, and F. C. Park. Dsqnet: A deformable model-
402 based supervised learning algorithm for grasping unknown occluded objects. *IEEE Transac-*
403 *tions on Automation Science and Engineering*, 2022.
- 404 [19] S. Kim, B. Lim, Y. Lee, and F. C. Park. Se (2)-equivariant pushing dynamics models for
405 tabletop object manipulations. In *Conference on Robot Learning*, pages 427–436. PMLR,
406 2023.
- 407 [20] J. Kim, M.-H. Jeon, S. Jung, W. Yang, M. Jung, J. Shin, and A. Kim. Transpose: Large-scale
408 multispectral dataset for transparent object. *The International Journal of Robotics Research*,
409 page 02783649231213117, 2023.