Supplemental Materials for "SAR-Net: Shape Alignment and Recovery Network for Category-level 6D Object Pose and Size Estimation"

Haitao Lin Zichang Liu Chilam Cheang Yanwei Fu† Guodong Guo Xiangyang Xue†

A. Social Impact

This study helps the tasks of augmented reality, scene understanding, and robotic manipulation. Our method grants the generalization ability of robots to grasp novel but category-known objects in open-environment tasks such as home care services. The efficacy of our proposed framework should facilitate the deployment of such an algorithm on various real-world scenarios of robotic tasks, and contribute to the development of robotic research communities.

B. Network Architecture of SAR-Net

Our network uses the PointNet-like [11, 17] architecture. We use two types of general encoders as shown in Tab. 1 and Tab. 2, where V is the vertex number of input point cloud with 3 channels, and "conv1d" is the 1-dimensional convolutional filter. "IN" indicates the Instance Normalization. **Encoder1**$(\cdot)$ and **Encoder2**$(\cdot)$ mean choosing the output of **Index** $(\cdot)$ from their unit. Also, we define the general decoder as in Table 3, **Decoder1**$(\cdot)$ takes C channels of tensors as inputs. The architecture of three main components are shown in Tab. 4, Tab. 5 and Tab. 6, respectively. Notably, $cate$ indicates the number of total categoriess.

Table 1. The network architecture of Encoder1 unit.

Index	Input	Operation	Output Shape
(1)	Input	Input	$3 \times V$
(2)	(1)	conv1d(3→64),IN,ReLU	$64 \times V$
(3)	(2)	conv1d(64→64),IN,ReLU	$64 \times V$
(4)	(3)	conv1d(64→128),IN,ReLU	$128 \times V$
(5)	(4)	conv1d(128→256),IN,ReLU	$256 \times V$
(6)	(5)	conv1d(256→512),IN,ReLU	$512 \times V$
(7)	(4)	Maxpool	128×1
(8)	(5)	Maxpool	256×1
(9)	(6)	Maxpool	512×1
(10)	(7,8,9)	Concatenate	896×1

†indicates corresponding author.

Table 2. The network architecture of Encoder2 unit.

Index	Input	Operation	Output Shape
(1)	Input	Input	$3 \times V$
(2)	(1)	conv1d(3→64),ReLU	$64 \times V$
(3)	(2)	conv1d(64→64),ReLU	$64 \times V$
(4)	(3)	conv1d(64→64),ReLU	$64 \times V$
(5)	(4)	conv1d(64→128),ReLU	$128 \times V$
(6)	(5)	conv1d(128→1024),ReLU	$1024 \times V$
(7)	(6)	AveragePool	1024×1

Table 3. The network architecture of Decoder1 unit.

Index	Input	Operation	Output Shape
(1)	Input	Input	$C \times V$
(2)	(1)	conv1d(C→512), ReLU	$512 \times V$
(3)	(2)	conv1d(512→256), ReLU	$256 \times V$
(4)	(3)	conv1d(256→128), ReLU	$128 \times V$
(5)	(4)	conv1d(128→64), ReLU	$64 \times V$
(6)	(5)	conv1d(64→3× cate)	$3 \times cate \times V$
(7)	(6)	output branch selection	$3 \times V$

Table 4. The encoder-decoder architecture of Shape Alignment component $\mathbf{E}_{SA}$.

Index	Input	Operation	Output Size
(1)	Input	Observed Points $\mathcal{P}$	$3 \times N_o$
(2)	Input	Template Points $\mathcal{K}_c$	$3 \times N_k$
(3)	(1)	Encoder1(10)	896×1
(4)	(2)	Encoder2(4)	$64 \times N_k$
(5)	(2)	Encoder2(7)	1024×1
(6)	(2,3,4,5)	Concatenate	$1987 \times N_k$
(7)	(6)	Decoder1(1987)	$3 \times N_k$

Table 5. The encoder-decoder architecture of Symmetric Correspondence component $\mathbf{E}_{SC}$.

Index	Input	Operation	Output Size
(1)	Input	Observed Points $\mathcal{P}$	$3 \times N_o$
(2)	Input	Template Points $\mathcal{K}_c$	$3 \times N_k$
(3)	(1)	Encoder1(10)	896×1
(4)	(2)	Encoder2(7)	1024×1
(5)	(1,3,4)	Concatenate	$1923 \times N_o$
(6)	(5)	Decoder1(1923)	$3 \times N_o$

Table 6. The encoder-decoder architecture of Object Center and Size component $\mathbf{E}_{OCS}$.

Index	Input	Operation	Output Size
(1)	Input	Observed Points $\mathcal{P}$	$3 \times N_o$
(2)	Input	Symmetric Points $\tilde{\mathcal{P}}'$	$3 \times N_o$
(3)	Input	Template Points $\mathcal{K}_c$	$3 \times N_k$
(4)	(1,2)	Concatenate & Centralize	$3 \times 2N_o$
(5)	(4)	Encoder1(10)	896×1
(6)	(3)	Encoder2(7)	1024×1
(7)	(4,5,6)	Concatenate	$1923 \times 2N_o$
(8)	(7)	Decoder1(1923)	$3 \times 2N_o$
(9)	(5)	Linear(896→512), ReLU	512×1
(10)	(9)	Linear(512→256), ReLU	256×1
(11)	(10)	Linear(256→64), ReLU	64×1
(12)	(11)	Linear(64→3× cate)	$3 \times cate \times 1$
(13)	(12)	output branch selection	3×1

C. Train Details for 3D-GCN

Training data generation. To purify the point cloud back-projected from the masked depth, we utilize the 3D segmentation network 3D-GCN [14] to filter out the outliers belonging to the background. The synthetic training data NOCS CAMERA provides mixed real and synthetic data by rendering virtual objects on real backgrounds. Given ground-truth object masks, it is easy to process such a dataset to obtain the training data. For object detection or segmentation, the bounding box or mask is generally used for representing the 2D predicted results. To make the 3D segmentation network compatible with both widely used input of box and mask, we process the data to simulate these two kinds of 2D predicted results as in Fig. 1. In particular,

(1) For mask results, they are represented by polygons, which are limited to handling the objects with holes like the mug in Fig. 1. Thus, the points back-projected from such the masked depth should contain points from background. We generate the training data by filling the hole of the ground-truth mask and then dilating the mask by 0 to 5 pixels to simulate the imperfect segmentation.

(2) For box results, it may usually come from a detection network like YOLOv3 [18]. The box regions contain background points and object points. We also randomly transform this region to simulate the imperfect detection results. Concretely, obtaining the ground-truth box region $\mathcal{B}_{gt} = (x_{left}, y_{top}, x_{right}, y_{bottom})$ of the target object, we transform this region into $(x_{left} + \delta_1, y_{top} + \delta_2, x_{right} + \delta_3, y_{bottom} + \delta_4)$, where (x_{left}, y_{top}) and (x_{right}, y_{bottom}) indicates the left-top and right-bottom coordinates of the corners of bounding box $\mathcal{B}_{gt}$, respectively; $\{\delta_n\}_{n=1}^{4}$ ranges from -5 to 15 pixels. At the data generation stage, the probability of using (1) and (2) is 0.5, separately.

Given the generated dataset, we train a single 3D-GCN model to segment points cloud from six categories. To make 3D-GCN robust to real scenarios, zero-mean Gaussian noise with standard deviation $\sigma = (\sigma_x, \sigma_y, \sigma_z)$ is also added. We use $\sigma_x = \sigma_y = 0.1mm$ and $\sigma_x = 1mm$ in our experiment.

Training details. We keep the same configurations for the segmentation task as in [14]. The 3D-GN is trained on 6 NVIDIA RTX2080Ti GPUs with a batch size of 48. We set the initial learning rate as 0.0012 and multiply it by 0.75 for every epoch. The network takes about 5 hours to converge. The 3D-GCN only has negligible 1.69M parameters.

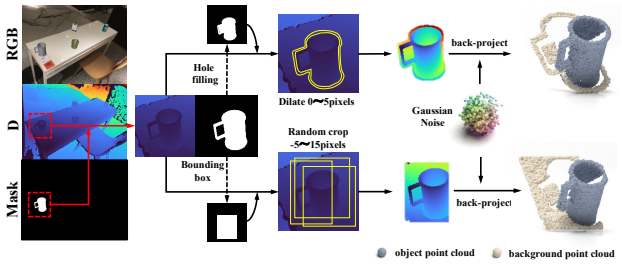

Figure 1. Training data generation. We use the NOCS CAMERA dataset to generate the data composed of object and background point clouds for training 3D-GCN.

D. Different 2D Input Representation

Our framework is applicable to the input of either a 2D bounding box or segmentation mask. Thus, we also report the results on the REAL275 by using the 2D detected bounding boxes as inputs, denoted as SAR-Net(box). As shown in the Tab. 7, our SAR-Net(box) still achieves comparable results, especially on the metrics for 6D pose recovery. Taking detected boxes as inputs saves inference times and achieves higher real-time performance. The baseline methods FS-Net takes the 2D detected box as input and achieves real-time performance. Compared to FS-Net, our SAR-Net(box) outperforms it by a large margin under $5°5cm$ metric.

E. Training Data Generation for SAR-Net

We prepare our small training data using models from ShapeNetCore [2]. ShapeNetCore consists of various realistic-looking synthetic models which are canonical, in terms of translation, orientation, and size. To generate the training set, we pick models as [21], covering 6 categories - `bottle`, `bowl`, `camera`, `can`, `laptop`, and `mug`. The instance models are located in the origin of the world coordinate and are surrounded by a camera for generating view-specific rendered depth images in Blender software [1]. To be specific, the parameters are as follows. We set the Field of View (FOV) of the camera as 60°. To cover varied viewpoints, we equally split the camera's azimuth into 60 regions and randomly sample the azimuth from each region.

Table 7. Results on REAL275 [21]: comparisons with other COPSE methods.(↑): higher better, (↓): lower better.

Dataset	Method	mAP (↑) IoU_{50}	IoU_{75}	$5°2cm$	$5°5cm$	$10°2cm$	$10°5cm$	Accuracy (↑) $5°5cm$	Parameters (↓) (M)
REAL275	NOCS [21]	78.0	30.1	7.2	10.0	13.8	25.2	18.2	-
	CASS [3]	77.7	-	-	23.5	-	58.0	-	47.2
	SPD [20]	77.3	53.2	19.3	21.4	43.2	54.1	30.4	18.3
	FS-Net [4]	**92.2**	63.5	-	28.2	-	60.8	-	41.2
	StablePose [19]	-	-	-	-	-	-	38.8	-
	DualPose [13]	79.8	62.2	29.3	35.9	50.0	66.8	50.1	67.9
	SAR-Net(small)	80.4	**63.7**	24.1	34.8	45.3	67.4	49.1	**6.3**
	SAR-Net(box)	76.1	55.4	28.0	35.4	48.4	61.9	49.6	**6.3**
	SAR-Net	79.3	62.4	**31.6**	**42.3**	**50.3**	**68.3**	**54.9**	**6.3**

Meanwhile, the camera position randomly ranges from 0.3 to 1.5 meters in the vertical direction and 1.0 to 1.2 meters in the horizontal direction, relative to the target object. We also consider the in-plane rotation ranging from $-40°$ to $40°$. Finally, we generate rendered depth images under 60 different camera poses for each instance. Given the known camera intrinsic parameters, the partial point clouds are then back-projected from depth images and sampled into 1024 points randomly.

F. Effect of Different Template Shape

To learn the shape alignment of the observed point cloud $\mathcal{P}$, we randomly select three groups of category-level template shapes per category, which are sampled into sparse point cloud $\mathcal{K}_c$. Also, we use mean shapes provided by the SPD [20] as the template shapes. Notably, we choose a single fixed template shape for each category before training the network. During training and inference time, the category-level template point cloud is assigned according to the ground-truth and predicted category of the object, respectively. The network outputs the deformed template point cloud according to the given category-level template point cloud. Four groups of template shapes used in our experiments are shown in Fig. 7. As can be seen from Tab. 8, our models trained by using four different groups of template shapes show a slight performance difference among them. Such comparison results indicate that our proposed method, which learns the intra-class shape similarity, is robust to different template shape selections.

Table 8. Results on different template shape configurations. 'SAR-Net G1, G2, and G3' indicate our COPSE models are trained by using differently configured template shapes. 'SAR-Net MS' means using mean shape of each category. (↑): higher better.

Method	mAP (↑) IoU_{50}	IoU_{75}	$5°2cm$	$5°5cm$	$10°2cm$	$10°5cm$	Acc. (↑) $5°5cm$
SAR-Net G1	80.4	63.7	**24.1**	**34.8**	**45.3**	**67.4**	**49.1**
SAR-Net G2	81.5	62.9	22.8	32.6	44.4	66.6	48.1
SAR-Net G3	81.1	**63.9**	23.0	32.9	44.9	66.3	47.6
SAR-Net MS	**82.0**	62.6	23.8	33.0	45.1	67.1	48.9

G. Loss Function for Rotational Symmetry

For the objects with rotational symmetry, they should have the same appearance around the axis of symmetry. Nevertheless, the $\mathcal{L}_{def}$ encounters the ambiguities in dealing with instances with rotational symmetry, *e.g.*, *bowl*. Hence, given ground-truth deformed template points $\mathcal{K} = \{\boldsymbol{k}_i\}_{i=1}^{N_k}$ with N_k points, we adopt the strategy as [21] to generate a candidate ground-truth set. Finally, our SAR-Net reconstructs $\tilde{\mathcal{K}} = \{\tilde{\boldsymbol{k}}_i\}_{i=1}^{N_k}$ supervised by the loss as:

$$\mathcal{L}_{def} = \min_{S_j \in \mathcal{S}} \{ \frac{1}{N_k} \sum_{i=1}^{N_k} \left\| S_j \cdot \boldsymbol{k}_i - \tilde{\boldsymbol{k}}_i \right\|_1 \} \tag{1}$$

where $\mathcal{S} = \{S_j | j = 1, \cdots, M\}$ is a set of candidate symmetry transformations. Transformation S_j is applied to the ground-truth deformed template points $\mathcal{K}$, generating a candidate ground-truth set, *i.e.*, rotating ground-truth deformed template points along its symmetry axis by the angle of $j\frac{360°}{M}$. Thus, set $\mathcal{S}$ covers M discrete global rotational symmetries, and $M = 12$ in our experiment.

H. Rotation Representation Implementation

To compare the performance of different rotation representations, we incorporate these representations into our SAR-Net by replacing the shape alignment component (SA). Specially, for the SVD 9D [12], we use the released official code [1]. For the continuity 6D [23], we use the code on the website:[2]. For Vector [22], we use the implemented code released on the website[3] and the loss function defined in the paper. We firstly transform the representation of quaternion, SVD 9D, and continuity 6D into corresponding 3×3 orthogonal matrixes $\tilde{R}$, and calculate the geodesic distance between the predicted rotation matrix $\tilde{R}$ and ground-truth one R. The loss function is given as below:

$$\mathcal{L}_{rot} = arccos(\frac{tr(\tilde{R}R^T) - 1}{2}) \tag{2}$$

where $tr(\cdot)$ indicates the trace of a square matrix.

[1] https://github.com/google-research/google-research/tree/master/special_orthogonalization

[2] https://github.com/zawlin/6d_rot

[3] https://github.com/DC1991/FS_Net

I. Per-category Performance on NOCS

As in Fig. 2, we show the average precision (AP) curves per category versus different thresholds on 3D IoU, rotation error, and translation error on the CAMERA25 (top row) and REAL275 (bottom row) datasets. The mean average precision (mAP) curve depicted the mean value of average precision results for all 6 categories.

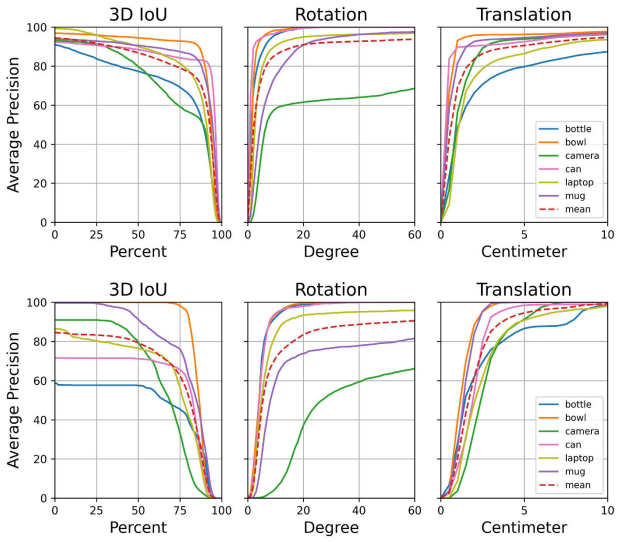

Figure 2. Average precision (AP) results versus different thresholds on 3D IoU, rotation error, and translation error on the CAMERA25 (top row) and REAL275 (bottom row) datasets.

J. Per-Instance Performance on LINEMOD

For the evaluation of the LINEMOD dataset, we follow the testing protocol as [6, 7]. We report the quantitative results of each instance with other RGB(D) or depth-only methods, as shown in Tab. 9. Compare to synthetic-only approach CP(ICP) [7] and CAAE [6], our SAR-Net outperforms the CP(ICP) by a margin and achieves comparable performance with CAAE in terms of some instances like *camera*, *cat* and *lamp*, etc. Especially, our SAR-Net has a low performance for the *benchwise* instance under ADD metric. That is because the lack of distinctive details from the real depth makes the back-projected point cloud look like an object with rotational symmetry, which makes the network hard to distinguish the orientation of the object.

K. More Qualitative Results

NOCS-REAL275 dataset. To ensure a fair comparison with the prior work [13], we use the segmented results of the Mask-RCNN [8] provided by [13], to evaluate the pose and size accuracy. The qualitative comparison results are shown in Fig. 9.

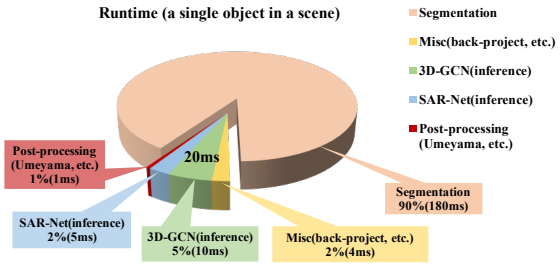

Figure 3. Runtime analysis of our framework.

Additional real-world scenarios. Besides, we also test our algorithm on 6 different real-world clutter scenarios by using novel 25 instances selected from the known 4 categories. Visualized results are shown in Fig. 6.

Intermediate results of our framework. We visualize the intermediate results from our framework tested on the REAL275 dataset in Fig. 10, including the estimated 6D pose & size, back-projected points filtered by 2D segmented mask, purified points processed by the 3D-GCN, deformed template point cloud from SA component, symmetric point cloud from SC component, and coarse object shape. Additionally, we visualize the intermediate results from our framework tested on real-world scenarios in Fig. 11.

Ground-truth symmetric point cloud. Figure 8 shows exemplars of ground-truth symmetric point cloud generated by chosen symmetric planes. We show the symmetric point cloud of the different categories under same camera view, and that of the same instance from different camera views.

L. Runtime Analysis

Given a scene image with a resolution of 640 $\times$ 480 and a single target object, our framework takes 180ms for segmentation, 4ms for processing the depth image, 10ms for 3D-GCN inference, 5ms for SAR-Net inference, and 1ms for post-processing, as shown in Fig. 3. For a more general scene with multiple objects, *i.e.*, five instances, the segmentation part takes nearly the same time as the case of a single object (180ms), and the pose and size estimation part only takes about 100ms.

M. Comparison to kPAM

kPAM [15] is an excellent algorithm which explicitly *detect* keypoints on the object as manipulation targets. The detected keypoints are used to perform tasks with semantic understanding of objects, *e.g.*, the task of hanging mugs on a rack by the mug handles. We show the difference between kPAM and SAR-Net as follows,

(1) **Training data.** kPAM demands intensive labor for manually semantic 3D keypoint annotations, but ours only use synthetic data. kPAM requires annotated 3D keypoints on the exact semantic position of the object, and ours only

Table 9. Results on LINEMOD [10]: comparisons with other instance-level methods. 'S' is synthetic data and 'R' is real data.

Training data	Methods	ape	benchwise	camera	can	cat	driller	duck	eggbox	glue	holepuncher	iron	lamp	phone	Mean
RGB(S+R)	PVNet [16]	43.6	99.9	86.9	95.5	79.3	96.4	52.6	99.2	95.7	82.0	98.9	99.3	92.4	86.3
RGBD(S+R)	FFB6D [9]	**98.4**	**100.0**	**99.9**	**99.8**	**99.9**	**100.0**	**98.4**	**100.0**	**100.0**	**99.8**	**99.9**	**99.9**	**99.7**	**99.7**
D(S)	CP(ICP) [7]	58.3	65.6	43.0	84.7	84.6	83.3	43.2	99.5	98.8	72.1	70.3	93.2	81.0‘	75.2
D(S)	CAAE [6]	74.5	86.6	65.6	90.2	90.7	97.3	50.0	99.7	93.5	57.9	85.0	82.1	94.4	82.1
D(S)	SAR-Net	64.5	4.0	68.3	83.6	91.4	84.0	66.0	99.4	**100.0**	74.0	70.5	94.5	84.0	75.7

demands the shape alignment of partial input points and category-level template shape.

(2) **Modality.** kPAM takes as input RGB-D images for 3D keypoint detection, and our SAR-Net uses the depth image for pose and size recovery.

(3) **Applicability of the task.** kPAM explicitly *detects* keypoints on the object as manipulation targets, and our SAR-Net *deforms* the category-level template point cloud for implicit 3D rotation representation. Thereby, kPAM is applicable to the task that demands a semantic understanding of objects in a category, while our SAR-Net focuses on the COPSE task.

N. Failure Cases Analysis

Lack of distinctive geometric details. The synthetic camera category is challenging, as some useful key geometry in NOCS-CAMERA25 is unavailable, *e.g.*, the geometry of some cameras represented by simple 3D boxes. Our SAR-Net only uses depth for geometry (not RGB), and lacked surface details confuse our network to distinguish front or back sides of the camera, as the red axis shown in Fig. 4, resulting in the angular error of $180°$. Thus lacked geometry details of cameras result in poor performance in rotation estimation, as the green curve shown in Fig. 2 (top row).

Imperfect segmentation. Imperfect segmentation results contains pixels of target object and background, as shown in Fig. 5. The back-projected points from depth images filtered by such an imperfect mask have outliers from pixels of background. The outliers will degrade the performance of the object center and size estimation, *i.e.*, oversize bounding box and offset of object center localization in Fig. 5, but the outliers less influence the 3D rotation estimation.

Remark. We discuss the case where inferring the symmetric correspondence does not help much on recovering the object shape. For example, if the observed point cloud is almost symmetric about the symmetric plane, *e.g.*, point cloud of mug's handle opposite or away from the camera, the predicted symmetric point cloud should be largely the same as the observed one. In this case, the concatenated point cloud is still quite partial. We show this particular case in our supplemental video, *e.g.*, visualized intermediate results of the mug category. However, inferring such a symmetric point cloud is still helpful as it highly depends on object pose. This could be considered as a prior symmetric plane constraint that guides the network to learn geometry details to help differentiate the object's current pose, boosting pose estimation performance.

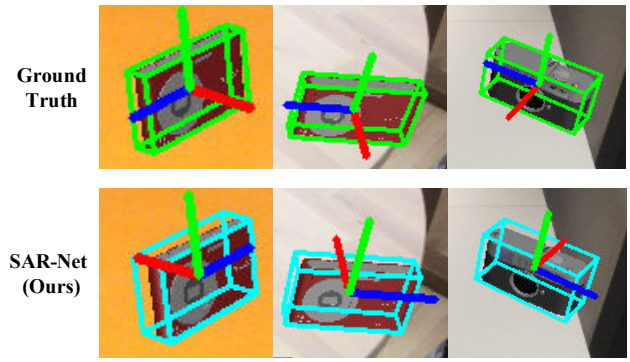

Figure 4. Exemplars of failure cases on the CAMERA25 dataset due to lacked distinctive geometry details of the camera category. We compare the visualized results of ground-truth (top row) and our SAR-Net (bottom row).

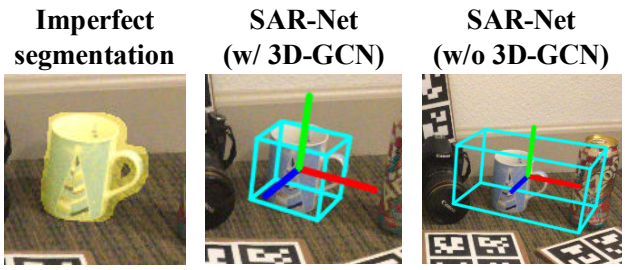

Figure 5. Failure case on REAL275 due to outliers given by imperfect segmentation (yellow mask). The 3D-GCN shows efficacy in filtering out outliers to enhance the performance of SAR-Net.

O. Robotic Experiment

In the robotic experiment, we use the top grasp for the grasping task and the side grasp for the object handover and pouring task. For the top grasp, the robot is programmed to grasp the handle of different mugs, sides of the bowls, and top of the bottles, according to the estimated 6D object pose and size. For the side grasp in the object handover task, we use the predicted pose and size to compute the gripper pose parallel to the narrower dimension of the bottle. For the side grasp in pouring task, the robot is commanded to grasp the mug body and then move it to the top of the moving bowl, according to the estimated pose of the bowl. We use the MoveIt! [5] to plan the feasible trajectory to grasp objects.

References

[1] Blender software. https://www.blender.org/. 2

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[3] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. Learning canonical shape space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11973–11982, 2020. 3

[4] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, Linlin Shen, and Ales Leonardis. Fs-net: Fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1581–1590, 2021. 3

[5] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012. 5

[6] Ge Gao, Mikko Lauri, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. Cloudaae: Learning 6d object pose regression with on-line data synthesis on point clouds. In *ICRA*, 2021. 4, 5

[7] Ge Gao, Mikko Lauri, Yulong Wang, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. 6d object pose regression via supervised learning on point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3643–3649. IEEE, 2020. 4, 5

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 4

[9] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11632–11641, 2020. 5

[10] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012. 5

[11] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7662–7670, 2020. 1

[12] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *arXiv preprint arXiv:2006.14616*, 2020. 3

[13] Jiehong Lin, Zewei Wei, Zhihao Li, Songcen Xu, Kui Jia, and Yuanqing Li. Dualposenet: Category-level 6d object pose and size estimation using dual pose network with refined learning of pose consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3560–3569, October 2021. 3, 4, 9

[14] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1800–1809, 2020. 2

[15] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2019. 4

[16] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019. 5

[17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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. 1

[18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[19] Yifei Shi, Junwen Huang, Xin Xu, Yifan Zhang, and Kai Xu. Stablepose: Learning 6d object poses from geometrically stable patches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15222–15231, 2021. 3

[20] Meng Tian, Marcelo H Ang Jr, and Gim Hee Lee. Shape prior deformation for categorical 6d object pose and size estimation. *arXiv preprint arXiv:2007.08454*, 2020. 3

[21] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 2, 3

[22] Jiaze Wang, Kai Chen, and Qi Dou. Category-level 6d object pose estimation via cascaded relation and recurrent reconstruction networks. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. 3

[23] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 3

Figure 6. More exemplar results from the real-world clutter environment. The colored coordinate axes indicate the predicted rotation of the object described in the camera frame, where the x, y, and z axes are colored as red, green, and blue, individually. We also visualize the predicted 6D object pose and size, represented by the tight-oriented bounding boxes.

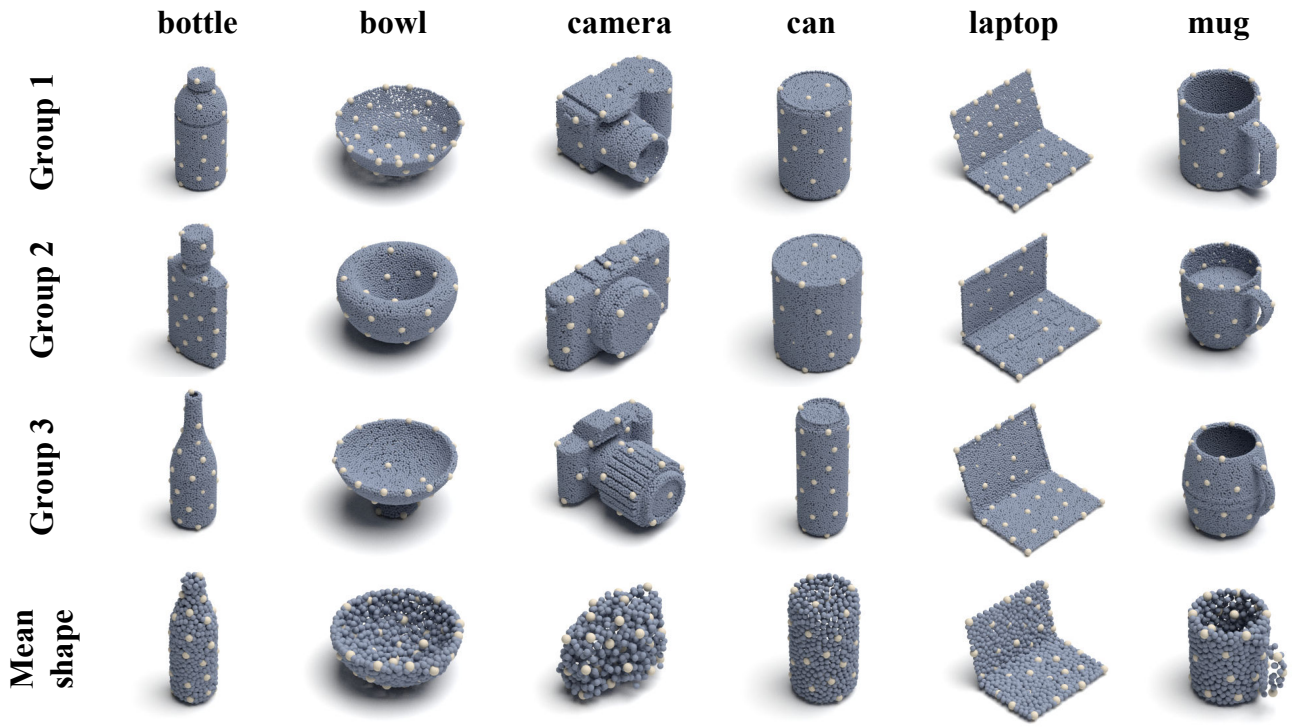

Figure 7. Four groups of category-level template point cloud (grey points) are used in our experiments, sampled into sparse point cloud (milky white points) by FPS algorithm. From left to right, we show in each row: *bottle*, *bowl*, *camera*, *can*, *laptop*, *mug*.

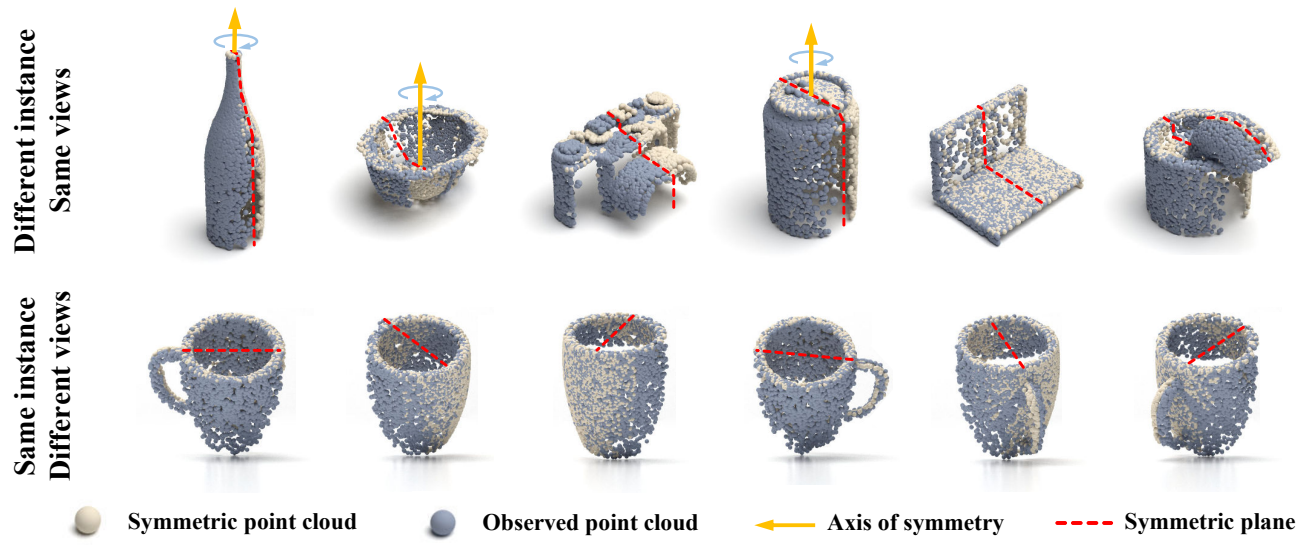

Figure 8. More exemplars of ground-truth symmetric point cloud generated by chosen symmetric planes. We show symmetric point clouds of different categories under the same camera view (top row) and that of the same instance from different camera views (bottom row).

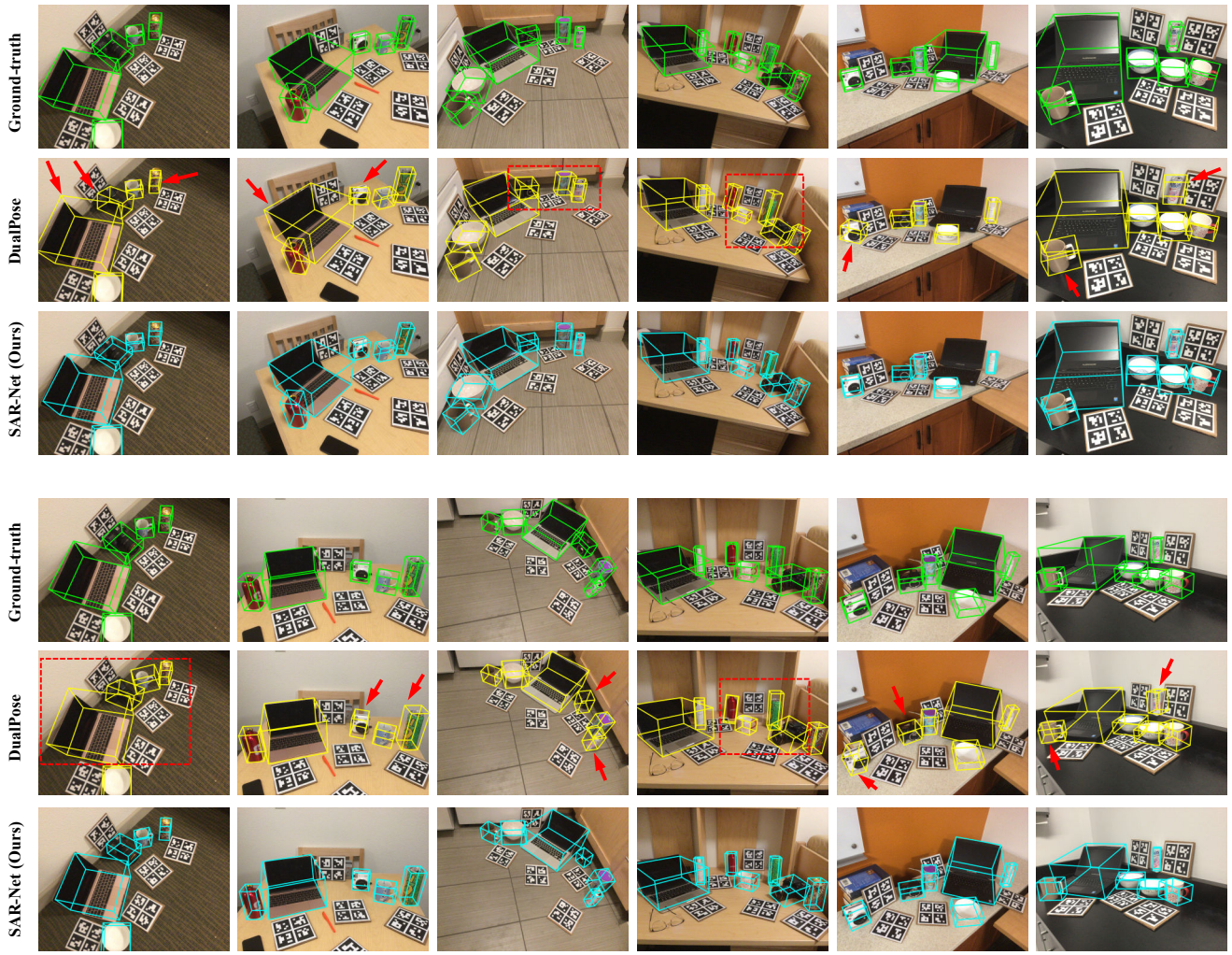

Figure 9. More qualitative comparisons between our SAR-Net and DualPose [13]. We visualize the ground-truth results with the green bounding boxes, our predicted results with the blue bounding boxes, and the competitor's with the yellow ones.

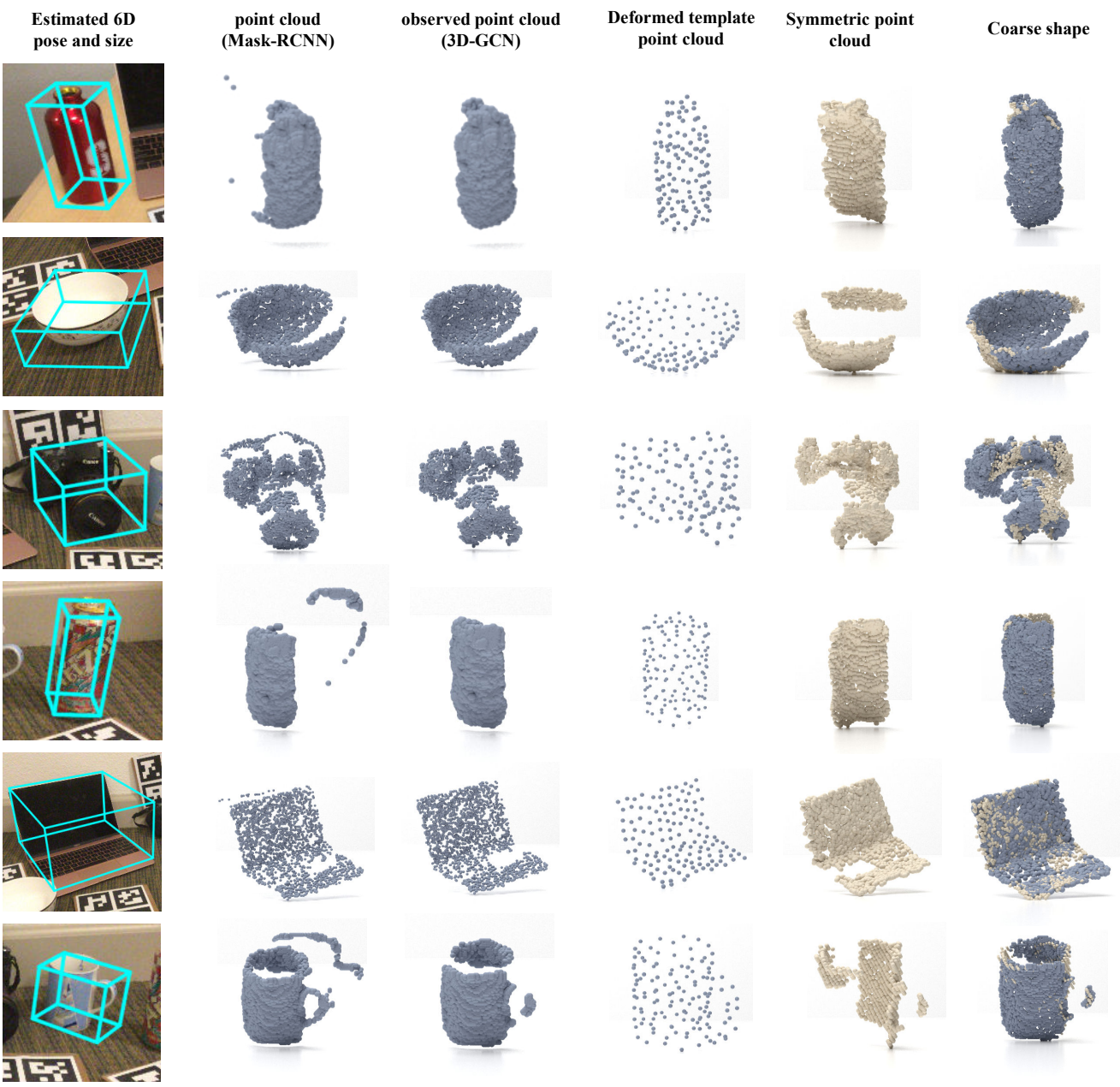

Figure 10. More visualized intermediate results from our framework. From left to right, we show in each row: the estimated 6D object pose & size, the back-projected point cloud filtered by 2D segmented mask, purified points processed by the 3D-GCN, deformed template point cloud, symmetric point cloud, and coarse object shape.

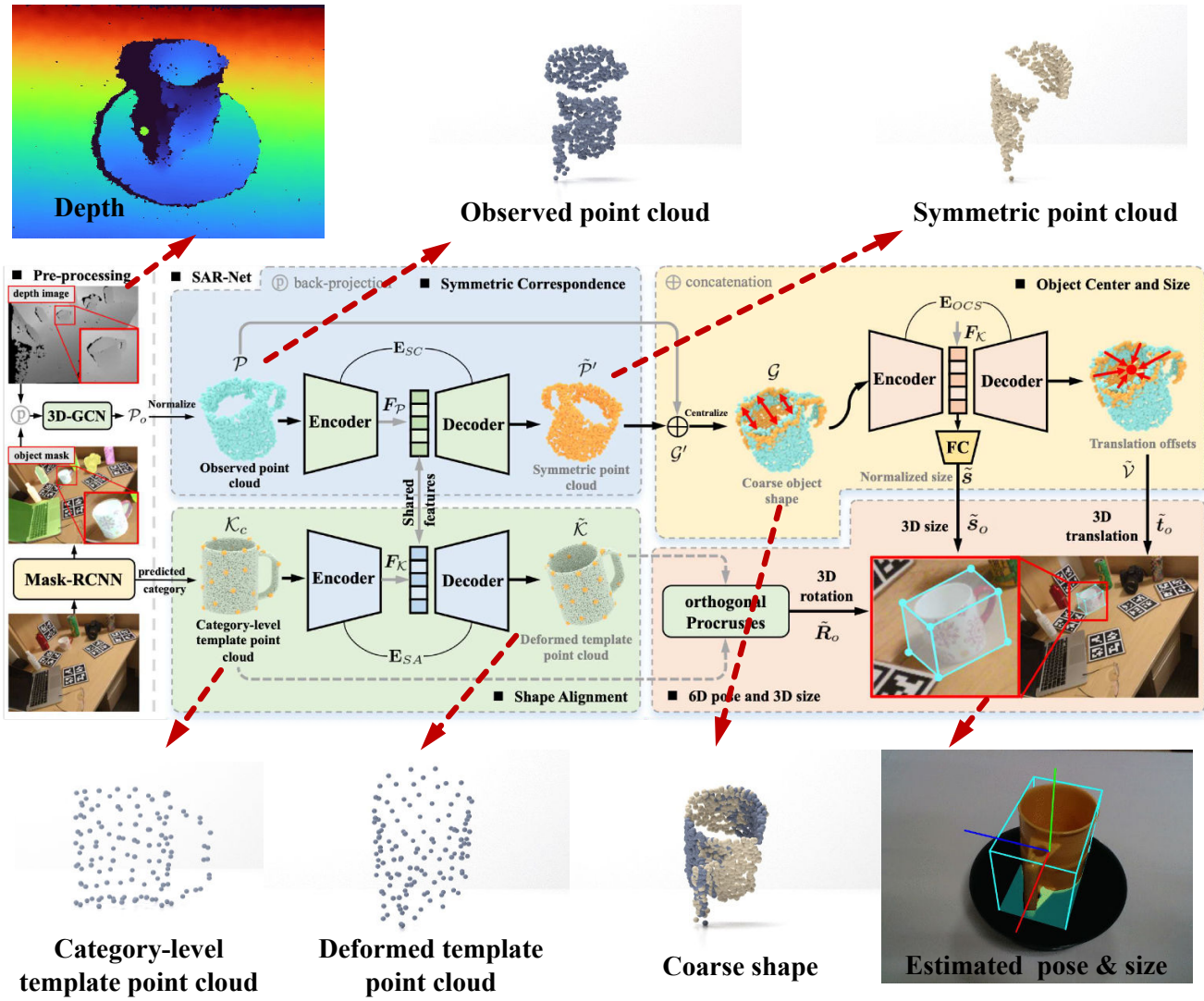

Figure 11. Visualized intermediate results from our framework tested on real-world scenario.