Generative Category-Level Shape and Pose Estimation with Semantic Primitives Supplementary Material

Guanglin Li[1,2] **Yifeng Li**[2] **Zhichao Ye**[1] **Qihang Zhang**[3]
Tao Kong[2] **Zhaopeng Cui**[1] **Guofeng Zhang**[1]
[1]State Key Lab of CAD&CG, Zhejiang University [2]ByteDance AI Lab
[3]Multimedia Laboratory, The Chinese University of Hong Kong

In this supplementary material, we describe more details of our method, including implementation details in Sec. A, mesh reconstruction in Sec. B, cases with shape ambiguity in Sec. C, more qualitative results in Sec. D and Sec. E. Besides, we also provide real world experiments in Sec. F.

A Implementation Details

A.1 Training Details

The object point cloud is back-projected from the depth image with the Mask-RCNN segmentation results provided by [1]. We randomly sample 1024 points from the object point cloud as our input. The network architecture and training protocol of our generative model g keep the same as [2]. We train a single generative model for each category with the CAD models provided by ShapeNetCore [3]. Our part segmentation network architecture follows the part segmentation version of 3D-GCN [4], and we set 256 primitives in our main experiments. The segmentation network is trained on CAMERA dataset (about 600K instances) with a batch size of 32 on a single NVIDIA Tesla V100 graphics card. Moreover, we set the initial learning rate as 0.0005 and multiply it by a factor 0.2 every 10 epochs. We use the ground truth pose provided by [5] and apply the pose on generated primitives to annotate the ground truth of part segmentation.

A.2 Training Data Augmentation

To obtain the object point cloud which is the input of our method from given images, we utilize Mask R-CNN [6] to segment the object masks from images. Because the results provided by Mask R-CNN are often imperfect, the object point cloud back-projected from the masked depth image would contain background points. To filter out these outlier points, we train our part segmentation network (Sec. 3.2) by making augmentation on the ground truth segmentation masks. As shown in Fig. A, we adopt the following two strategies [7] on the ground truth mask : (a) random 0 to 5 dilations on the 2D mask. (b) random 0 to 15-pixel crop on the mask's bounding box.

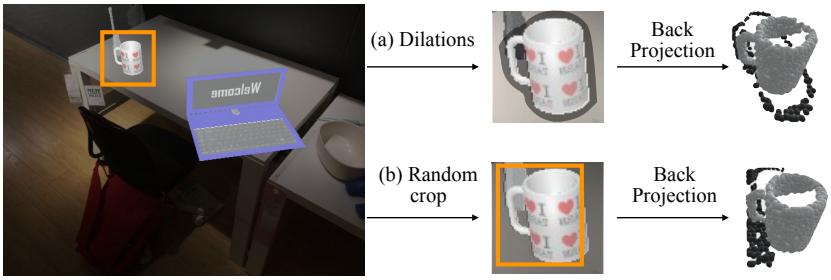

Figure A: Augmentation on synthetic training data. (a) Dilation on 2D mask. (b) Random crop on mask's bounding box. The black points in the object point cloud are noise points.

Work done during Guanglin Li's internship at ByteDance.

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

Table 1: Comparison of shape reconstruction accuracy of our reconstructed primitives and mesh in CD metric ($\times 10^{-3}$) on REAL275.

Shape	bottle	bowl	camera	can	laptop	mug	all
Semantic Primitives	2.05	1.55	10.1	1.63	2.12	2.93	3.45
Mesh	3.51	3.27	8.55	3.55	2.68	2.89	4.08

A.3 Training a DualSDF Model

In order to generate the semantic primitive representation for each shape, we pre-trained a generative model for each category following the network structure of DualSDF [2]. The generative model represents an object shape in two granularities. One coarse-level branch represents the coarse structure by a certain number of simple shape primitives, and the other fine-level branch represents the fine structure of the object by SDF.

As we mentioned in Sec. 3.1, given a shape embedding z, the coarse-level branch could be expressed as $g_c(z) = \{\boldsymbol{\alpha}_i | i = 1, ..., N_c\}$, and the fine-level branch could be expressed as $g_f(z, \boldsymbol{x}) = SDF(\boldsymbol{x})$. A signed distance field (SDF) refers to the closest distance from a point $\boldsymbol{x}$ to the surface of the object model. $SDF(\boldsymbol{x}) < 0$ indicates $\boldsymbol{x}$ is inside the model and $SDF(\boldsymbol{x}) > 0$ indicates outside. The two auto-decoders g_f and g_c are supervised by a set of pairs of 3D points $\boldsymbol{x}$ and their corresponding ground truth signed distance values $s = SDF(\boldsymbol{x})$. The primitive set $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_i | i = 1, ..., N_c\}$ can be learned by minimizing the difference between predicted and ground truth signed distance values:

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \, L_{SDF}(d(\boldsymbol{x}, \boldsymbol{\alpha}), s), \tag{1}$$

where L_{SDF} is a truncated L1 loss and d indicates the distance from $\boldsymbol{x}$ to the nearest surface of $\boldsymbol{\alpha}$. We use sphere as primitive here, thus $\boldsymbol{\alpha}_i = (\boldsymbol{c}_i, r_i)$, where $\boldsymbol{c}_i$ is the sphere center and r_i is the sphere radius. d is expressed as:

$$d(\boldsymbol{x}, \boldsymbol{\alpha}) = \min_{1 \le i \le N_c} ||\boldsymbol{p} - \boldsymbol{c}_i||_2 - r_i, \tag{2}$$

In our implementation, we add a truncated value to the ground truth s to make these primitives evenly distributed on the object's surface. Specifically, the Eq. 1 can be rewritten as:

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \, L_{SDF}(d(\boldsymbol{x}, \boldsymbol{\alpha}), \overline{s}), \tag{3}$$

where

$$\overline{s} = \begin{cases} s, & s \ge -\frac{t}{2} \\ -s - t, & s < -\frac{t}{2} \end{cases}. \tag{4}$$

and $t = 0.02$ is a truncated value in our experiments.

The generated primitive examples of six categories are shown in Fig. B, and we can see that fewer primitives lack geometric details but will make it easier for our part segmentation network (Sec. 3.2) to learn. Fig. B also shows that 512 primitives are redundant to represent a shape, which drops all metrics on pose estimation. Sec. 4.3 shows the ablation study on the different number of primitives.

B By-Product: Object Mesh Reconstruction

Although the fine-level decoder g_f provides a fine-level structural representation of the objects, we do not exploit it in our main experiment (shape and pose estimation). To explore the potential of the shared latent space shared by g_c and g_f, we use our optimized shape embedding $\hat{z}$ to generate a mesh model for each object through the fine-level decoder g_f.

Specifically, for each object, we grid the canonical frame to 48^3 points and get the SDF value of each point by the optimized shape embedding $\hat{z}$ and the trained fine-level generative model g_f. After that, we utilize the Marching Cubes algorithm [8] to recover the object mesh. Table 1 shows a comparison of shape reconstruction of the primitives and mesh. With the mesh of each object, we can reconstruct the object-level scene by transforming these meshes to the world coordinate using our estimated pose. Fig. C shows qualitative results of our object mesh reconstruction. Without further optimization, the different reconstructed meshes keep consistent with their observations. For example, the reconstructed camera lens in the first scene (first column) is longer, while the one in the second scene (second column) is shorter.

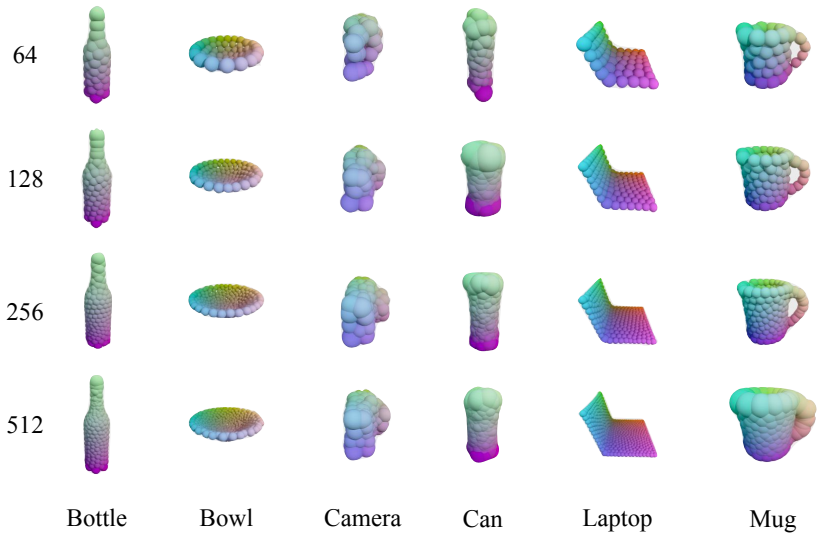

Figure B: Visualization of the generated primitive representations of six categories (*bottle, bowl, camera, can, laptop* and *mug*) with different numbers (64, 128, 256 and 512) of primitives. We can see that 64 primitives (top row) lack of geometric details and 512 primitives (bottom row) are redundant to represent a shape.

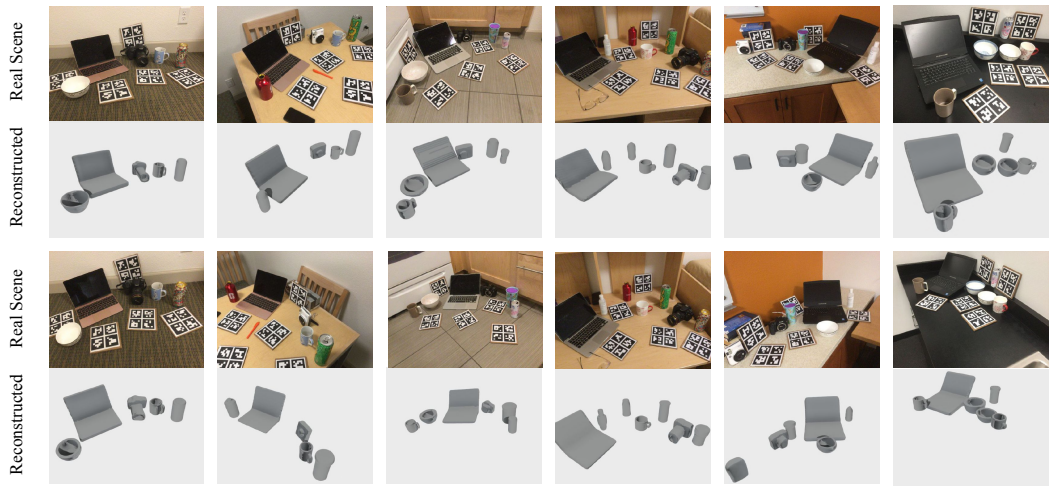

Figure C: Visualization of our mesh reconstruction experiment. The mesh of each object is recovered by the optimized shape embedding $\hat{z}$ and the fine-level decoder g_f. We transform the object model to the world coordinate by our estimated pose. The top row is the original scene and the bottom row is our object-level scene reconstruction.

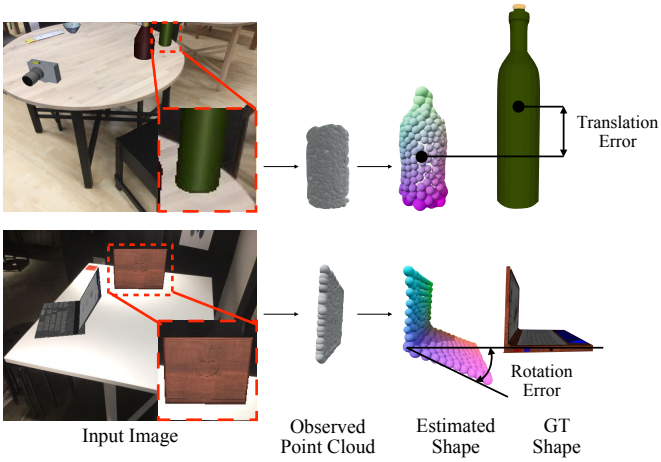

Figure D: Shape ambiguity cases. When an object is occluded causing shape ambiguity, although our estimated shape and GT shape are both consistent with the observation, this still causes relatively large translation errors (top row) and rotation errors (bottom row) in the evaluation.

C Shape Ambiguity of Partially Observed Point Cloud

Fig. D shows two cases of shape ambiguity when the objects are highly occluded, and these cases lead to some failure cases of our method on CAMERA [9] dataset. The top row shows a bottle with the top half occluded. Although our estimated shape is consistent with the partially observed point cloud, it still leads to translation errors when evaluating (about $10cm$ translation error in this case). The bottom row shows a laptop with its keyboard occluded. In this case, the screen of our estimated laptop is consistent with the observation, but there is still a rotation error when evaluating (about 20° rotation error in this case).

D More Results on NOCS-REAL275

Fig. E shows more qualitative results of our method compared with SGPA [10]. Compared with our method, the results of SGPA are relatively poor in some cases of laptops and cameras.

E More Visualization of Shape Optimization

Fig. F gives more visualization of the process of our shape optimization. As the optimization proceeds, the structure of the object is optimized to be consistent with the observation. For example, the lens of the camera is gradually stretched, the angle between the screen and keyboard of the laptop gradually increases, and the handle of the mug becomes rectangular. Throughout our framework, shape optimization accounts for the bulk of the time, taking about 0.005s per iteration.

F Real World Experiments

Applicability to unseen scenarios and settings is critical for robotic tasks. To verify the effectiveness of our model in real scenarios other than the dataset provided by NOCS [9], we test our method on several different real-world scenarios. These scenarios contain different kinds of household objects of different shapes, and the photos of these scenarios are taken from the view of a robotic arm. We use Detic [11] to pre-process the masks of each object and treat the cups with handles as mugs and those without handles as cans. Fig. G shows visualization results, and tight orientated bounding boxes mark out objects. The red, green and blue axes represent the x, y, z axes of the canonical coordinate.

Besides, we conduct robotic grasping experiments to demonstrate the application of our category-level pose estimation. Specifically, we first estimate pose of each object from the view of a robotic

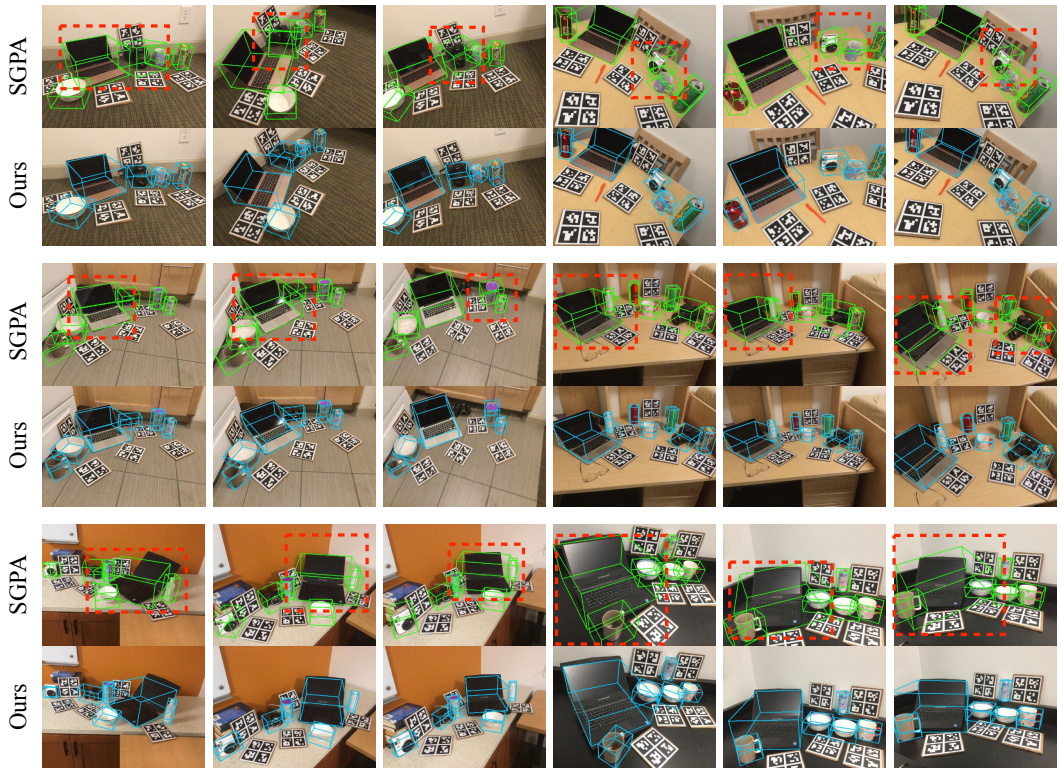

Figure E: More qualitative comparisons between our method and SGPA [10] on REAL275 dataset. The red dotted boxes indicate the results of SGPA with worse accuracy than ours.

arm. And then, we use the MoveIt![12] to plan a feasible path for the robot arm to grasp the top part of objects. Refer to our video for details.

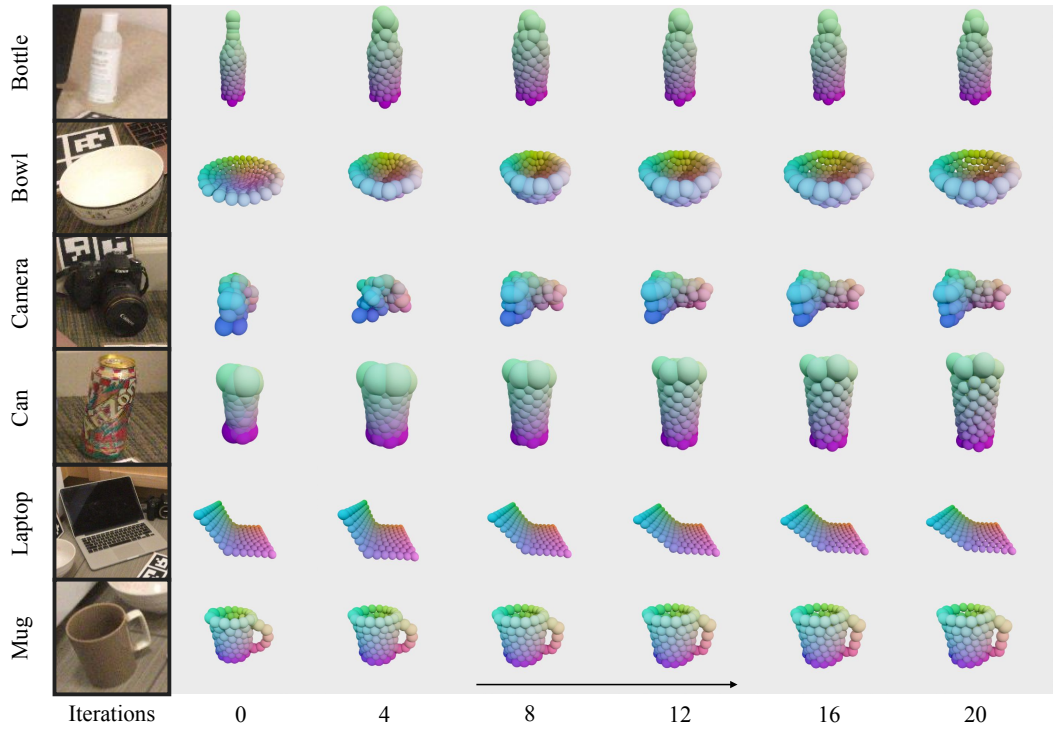

Figure F: More visualization of our shape optimization process. The objects in the figure are optimized by 20 iterations (from left to right). As the optimization proceeds, the shape of the objects changes significantly. For example, the lens of the camera is gradually stretched, the angle between the screen and keyboard of the laptop gradually increases, and the handle of the mug becomes rectangular.

Figure G: Results of real scenes. These scenarios contain different kinds of household objects of different shapes. The objects are marked out by tight orientated bounding boxes. The red, green and blue axes represent the x, y, and z axes of the canonical coordinate, respectively.

References

[1] M. Tian, M. H. Ang, and G. H. Lee. Shape prior deformation for categorical 6d object pose and size estimation. In *European Conference on Computer Vision*, pages 530–546. Springer, 2020.

[2] Z. Hao, H. Averbuch-Elor, N. Snavely, and S. Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7631–7641, 2020.

[3] 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.

[4] Z.-H. Lin, S.-Y. Huang, and Y.-C. F. 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.

[5] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066. IEEE, 2020.

[6] 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.

[7] H. Lin, Z. Liu, C. Cheang, Y. Fu, G. Guo, and X. Xue. SAR-Net: Shape alignment and recovery network for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2022.

[8] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Siggraph Computer Graphics*, 21(4):163–169, 1987.

[9] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[10] K. Chen and Q. Dou. Sgpa: Structure-guided prior adaptation for category-level 6d object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2773–2782, 2021.

[11] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra. Detecting twenty-thousand classes using image-level supervision. *arXiv preprint arXiv:2201.02605*, 2022.

[12] S. Chitta, I. Sucan, and S. Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012.