Supplementary Material: Gen6D: Generalizable Model-Free 6-DoF Object Pose Estimation from RGB Images

Yuan Liu[1], Yilin Wen[1], Sida Peng[2], Cheng Lin[3], Xiaoxiao Long[1], Taku Komura[1], and Wenping Wang[4]

[1] The University of Hong Kong
[2] Zhejiang University
[3] Tencent
[4] Texas A&M University

1 Overview

This supplementary material has the following contents.

1. Sec. 2 contains implementation details of the data normalization (Sec. 2.1), the object detector (Sec. 2.2), the viewpoint selector (Sec. 2.3), the pose refiner (Sec. 2.4) and the inference process.
2. Sec. 3 includes experiment details about the training set, the GenMOP dataset and the reference/query splits.
3. Sec. 4 provides more qualitative results on the LINEMOD [2] dataset, the MOPED [4] dataset and the GenMOP dataset.
4. Sec. 5 contains additional analysis on comparison with finetuned DeepIM [3], reference image number, unevenly distributed reference images, refinement iterations, ablations on training data, symmetric objects and imperfect view selection.

2 Implementation detail

2.1 Data normalization

Object size and center. On the GenMOP dataset, the object size and object center are roughly estimated from reconstructed points of COLMAP [7]. On the LINEMOD [2] dataset and the MOPED [4] dataset, we directly use the provided 3D model to compute the object size and the object center. Note we do not need an exact object size and an object center.

Normalization of object coordinate system. Given the object size d and the object center c, we normalize the object coordinate system by

$$x_{norm} = (x - c)/d \times 2, \tag{1}$$

where x_{norm} is the normalized coordinate and x is the original coordinate.

Normalization of reference images. Since the raw reference images may not look at the object, we normalize these reference images by warping the image with a homography and changing the intrinsic correspondingly, as shown in Fig. 1. The normalized reference images all look at the object and they enclose the projection of the unit sphere at the origin. Note we can do this normalization because we have already normalized the object coordinate and we know the poses of reference images.

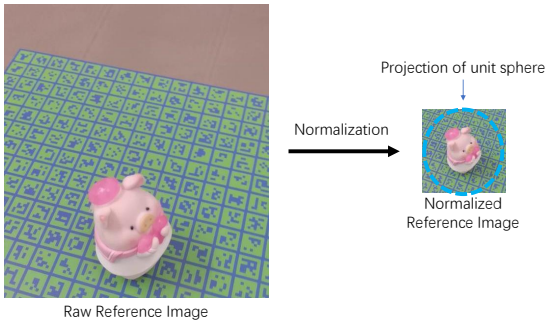

Fig. 1. Raw reference images are normalized to look at the object. The projections of the unit sphere are enclosed by the reference images.

2.2 Detector

Networks architecture. The detailed architecture is shown in Fig. 2. The input reference images are resized to 120×120. We actually use three levels of feature maps from the query image and the corresponding three levels of Conv kernels from reference images. We conduct convolutions on the feature map at each level with the corresponding conv kernel at the same level. $N_s = 5$ scales are used and we downsample the input query image with $\sqrt{2}$ as the factor.

Loss. Given the heat map H, we regard all values of the heat map as logits and use a binary classification loss to supervise the heat map prediction. Here, we first project the object center to the image using the ground-truth object pose. Then, a pixel on the heat map is assumed to be correct if it is within 1.5-pixel distance from the object center projection. Otherwise, it is incorrect.

$$\ell_{heat} = \sum_{p} -\mathbb{1}(\|p - c_{prj}\|_2 < 1.5) \log \sigma(H(p)) - (1 - \mathbb{1}(\|p - c_{prj}\|_2 < 1.5)) \log(1 - \sigma(H(p))), \tag{2}$$

where $\mathbb{1}$ is an indicator function, $c_{prj} \in \mathbb{R}^2$ is the 2D projection of the object center, p is a pixel on the heat map, σ is the Sigmoid function, $H(p)$ means the heat value on the pixel p. To supervise the scale map prediction, we apply a scale loss to minimize the L2 distance between the ground-truth scale and the

predicted scale in the log space. Note we only apply such scale loss on the pixels within 1.5-pixel distance away from the projected object center.

$$\ell_{scale} = \sum_{p} \mathbb{1}(\|p - c_{2d}\|_2 < 1.5) \| \log s_{gt} - S(p) \|_2^2, \tag{3}$$

where s_{gt} is the ground-truth scale, S is the scale map, $S(p)$ means the scale value at pixel p. s_{gt} can be computed from the distance l between the camera center and the object center by

$$s_{gt} = \frac{2f}{lS_r}, \tag{4}$$

where f is a virtual focal length by changing the principle point to the object center projection c_{prj}, 2 is the diameter of the unit sphere and $S_r = 120$ is the size of reference image.

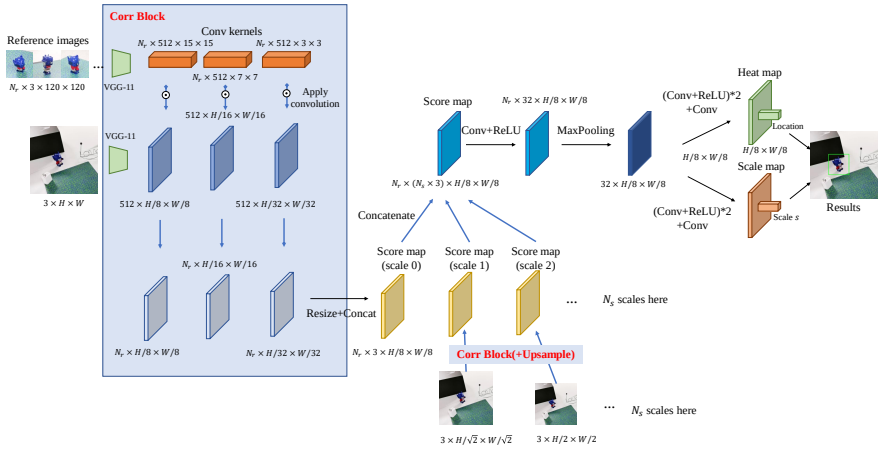

Fig. 2. Detailed pipeline of the detector. All "Conv" layers use 3×3 kernel size.

2.3 Selector

Network architecture. The detailed architecture of the selector is shown in Fig. 3. The input query image is cropped according to the detection results and resized to 128×128. All reference images are also cropped to the size 128×128. We also use three levels of feature maps to conduct the pixel-wise product. The viewpoint is represented by a 3-dimensional vector. The in-plane rotation is a scalar of the clock-wise rotation angle. The final similarity score for a reference image is also a scalar. We use $N_a = 5$ rotation angles in $[-\pi/2, \pi/2]$ to rotate every reference image.

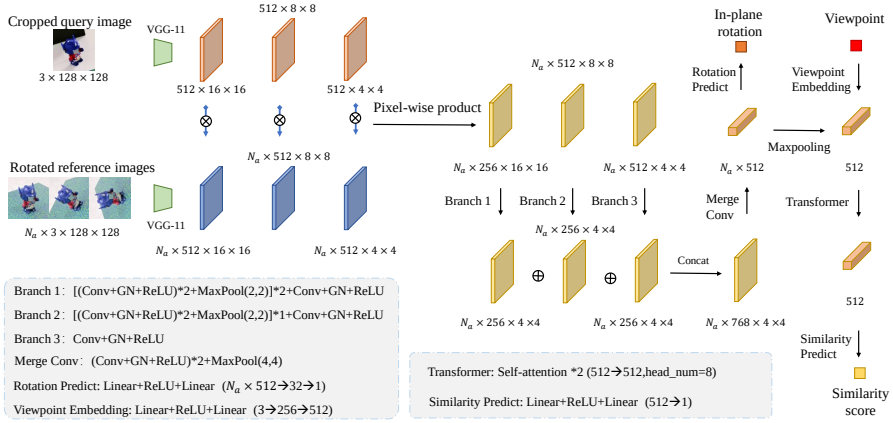

Fig. 3. Detailed pipeline of the selector. We only draw one reference image in the figure for clear visualization. "Conv" means a convolution layer with 3×3 kernels. "GN" means the global normalization that normalize the feature maps using the mean and variance computed from feature maps of all reference images. "Transformer" is applied among all feature vectors of reference images.

Loss. To compute the loss, we first introduce how we compute the ground-truth similarity between two viewpoints. In the object coordinate system, we denote the camera locations of all reference images as $\{u_i \in \mathbb{R}^3 | i = 1, ..., N_r\}$ where N_r is the number of reference images. Note that the object center is the origin so that $-u_i$ is the vector from the camera to the object center. In training, we know the query camera pose and compute the camera location v of the query image in the object coordinate system. Both u_i and v are normalized by $\tilde{u}_i = u_i/\|u_i\|_2$ and $\tilde{v} = v/\|v\|_2$. Then, the ground-truth viewpoint similarity between i-th reference image and the query image is their dot product $\tilde{u}_i \cdot \tilde{v}$.

We normalize the ground-truth viewpoint similarity to [0,1], which is denoted as $\tilde{s}_j$. Then, the similarity loss is

$$\ell_{sim} = \sum_j BCE(s_j, \tilde{s}_j) \tag{5}$$

where BCE means binary cross entropy loss and we force the predicted score to be consistent with the ground-truth viewpoint similarity.

To train the in-plane rotation angle prediction, we adopt a L2 loss,

$$\ell_{angle} = \|\alpha_j - \alpha_{gt}\|_2^2, \tag{6}$$

where α_j is the predicted in-plane angle on the ground-truth nearest reference image and α_{gt} is the ground-truth in-plane rotation between the query image and the reference image.

2.4 Refiner

Network architecture. The detailed architecture for the refiner is shown in Fig. 4. In the figure, we show the 2D CNN and the 3D CNN separately. In the 2D CNN, both reference images and query images are resized to 128×128 and the final feature map for unprojection is extracted from three levels of feature maps. In the 3D CNN, we first embed reference mean, reference variance and query features separately. Then, they are concatenated and processed by several 3D convolution layers. Final 3D pose residuals are regressed by linear layers from the flatten feature vector of the 3D volume. We use $N_n = 6$ neighboring reference images in refinement.

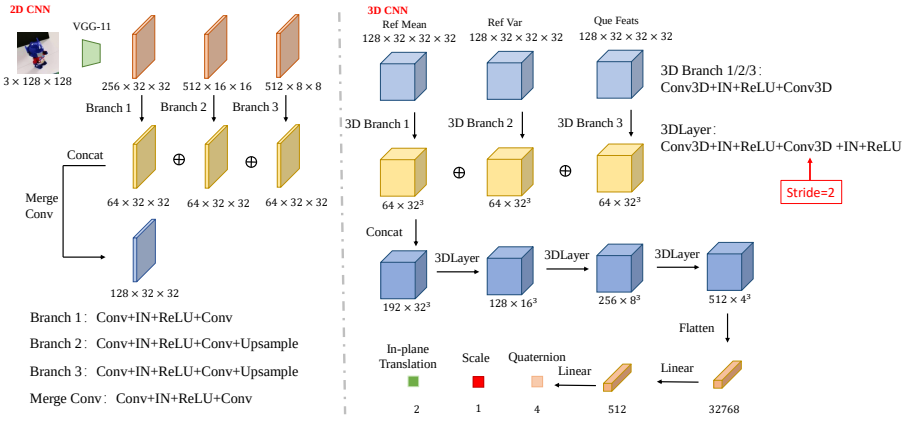

Fig. 4. Detailed architecture of the refiner. "IN" means instance normalization.

Loss. To train the refiner, we first sample 32^3 voxel points in the unit cube in the object coordinate system. Then, these points are transformed to the input camera coordinate system by the input pose. The loss is the distance between the sample points transformed by the ground-truth similarity transformation and the sample points transformed by the predicted similarity transformation.

$$\ell_{ref} = \sum_{k}^{32^3} \|s_{pr} R_{pr}(p_k + t'_{pr}) - s_{gt} R_{gt}(p_k + t'_{gt})\|_2, \tag{7}$$

where $p_k \in \mathbb{R}^3$ is the coordinate of a sample point in the input camera coordinate, s_{pr} and s_{gt} are predicted scale and ground-truth scale respectively, R_{pr} and R_{gt} are the predicted rotation and the ground-truth rotation respectively, t'_{pr} and t'_{gt} are the predicted 2D translation and the ground-with 2D translation respectively. We set the third element of t'_{pr} and t'_{gt} to 0 to form a 3D-translation. Note the predicted and ground-truth similarity transformations are used in transforming

the object (red circle in Fig. 5) with the input pose to the object with the ground-truth pose (green dotted circle in Fig. 5).

Similarity approximation. We discuss how to convert the similarity transformation (red circle to dotted green circle in Fig. 5) to the rigid transformation (red circle to solid green circle in Fig. 5). Denoting the similarity transformation as (R, s, t), our target is to compute the rigid transformation (R, t'). Actually, we only need to convert (s, t) to t' because rotation R is the same in two transformations. Assuming the red circle center is (c_x, c_y, c_z) in the input camera coordinate, then the center of the green dotted circle is $(c_x + t_x, c_y + t_y, z)$ where t_x and t_y are the first and the second element of t respectively. The depth affects the scale, so the center of the green solid circle is $((c_x + t_x)/s, (c_y + t_y)/s, z/s)$. The final rigid translation is $t' = ((c_x + tx)/s - c_x, (c_y + t_y)/s - c_y, z/s - z)$.

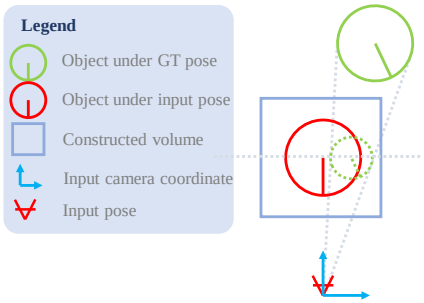

Fig. 5. Diagram to illustrate the transformation. Note all transformations are applied on coordinates in the input camera coordinate system.

2.5 Inference details

In inference, we apply pose refiner iteratively for 3 times. For the input data, not all reference images are used in the object detector and the viewpoint selector. Instead, we use farthest point sampling to sample 32 and 64 images for the detector and the selector respectively.

3 Experimental setting

Training set for Gen6D. The training objects on the LINEMOD [2] dataset are "ape", "can", "holepuncher", "iron" and "phone". There are ∼1200 images per object and we randomly use them as reference images and query images. The training objects on the GenMOP dataset are "cup", "knife", "love", "plugCN" and "Miffey", each of which contains ∼200 reference images and ∼200 query images. On every object from ShapeNet [1], we render 1024 images. On every object from Google Scanned Objects, we use 512 rendered images from IBRNet [9] for training. To train the 2D object detector, we additionally use the CO3D [6] for training.

Reference/query split on training sets. On the LINEMOD [2] dataset, the ShapeNet dataset and the Google Scanned objects dataset, we randomly select 128 images by farthest point sampling on camera locations as reference images while the other images are selected as query images. On the GenMOP

dataset, we use images from one video as reference images while images from the other video as query images.

Reference/query split on test sets. On the LINEMOD [2] dataset, we use the training set of previous instance-specific estimators [8,5] as reference images and the other images are selected as query images. On the GenMOP dataset, we also use images from one video sequence as reference images and images from the other video sequence as query images On the MOPED dataset [4], we use the provided reference video sequences as reference images and the other video sequences as query images. We list the number of reference images and query images in Table 1. Note that reference images are used in inference of Gen6D but not in training the Gen6D estimator while instance-specific estimators like PVNet [5] actually use these reference images to train their models.

Table 1. Numbers of reference images and test query images on different datasets.

	GenMOP				
	Chair	PlugEN	Piggy	Scissors	TFormer
Reference	212	199	227	202	206
Query	200	214	199	232	252
	LINEMOD [2]				
	cat	duck	bvise	cam	driller
Reference	177	189	183	182	179
Query	1002	1065	1032	1020	1009
	MOPED [4]				
	B.Drill	D.Dude	V.Mug	T.Plane	R.Aid
Reference	355	451	606	662	394
Query	215	297	91	250	58

GenMOP. On every object of the GenMOP dataset, we collect two 1-minute videos in different environments by a cell-phone. On each video, we sample 1 image per 10 frames, which results in ∼200 images for every video. On each video sequence, we apply COLMAP [7] to recover the extrinsics and intrinsics of all cameras. Then, in every sequence, we manually select two images, label 4 keypoints on the selected images, and use triangulation to compute the 3D points. For two difference sequences of the same object, we compute the transformation from the triangulated 3D points to align them.

4 Qualitative Results

More qualitative results on the LINEMOD dataset, the MOPED [4] dataset and the GenMOP dataset are shown in Fig. 10, Fig. 11 and Fig. 12 respectively.

5 More analysis

5.1 Comparison with finetuned DeepIM

Based on the pretrained generalizable DeepIM [3] model in the experiments of the main paper, we further finetune it separately on every test object using the reference images of the object as the training set. The performance of finetuned DeepIM model is shown in Table 2, which shows that finetuning DeepIM brings significant improvements but still underperforms the Gen6D model which does not train on the object.

Table 2. Performance on the GenMOP dataset. "General" means generalizable or not. "DeepIM [3]-Ft" means we finetune DeepIM models on every object's reference images separately.

Metrics	Method	General	Object Name					avg.
			Chair	PlugEN	Piggy	Scissors	TFormer	
ADD-0.1d	DeepIM [3]	✓	12.50	6.54	29.15	18.10	31.35	19.53
	DeepIM [3]-Ft	✗	**65.50**	36.92	62.31	19.40	38.49	44.52
	Ours	✓	61.50	**19.63**	**75.38**	**32.76**	**62.70**	**50.39**
Prj-5	DeepIM	✓	4.50	52.34	18.50	61.64	73.81	42.16
	DeepIM [3]-Ft	✗	40.00	70.56	82.37	73.28	98.41	72.92
	Ours	✓	**55.00**	**72.90**	**92.96**	**93.53**	**98.81**	**82.64**

5.2 Fewer reference images

To show the performance of Gen6D estimator with less reference images, we reduce the reference images on the GenMOP dataset from 128 to 8 by farthest point sampling (FPS) as shown in Table 3. Sampling 128 images by FPS even slightly improves the performance because FPS makes the view distribute evenly. With 64 reference images, the Gen6D estimator still produces similar results as with all images. When only 16 or 8 reference images retain, the performance reduces reasonably. We further provide detailed Prj-5 of different objects on the GenMOP dataset in Table 4. This show that the suitable view numbers for different objects are different. 16 reference views on "piggy" still produce 91% Prj-5 while 16 views on "chair" will reduce Prj-5 to 34%.

5.3 Uneven reference image distribution

By default, reference images are preferred to be distributed evenly around the object as shown in Fig. 6 (a), which benefits the viewpoint selection and the refinement using neighboring views. When only a part of viewpoints are available like Fig. 6 (b), Gen6D is not able to accurately predict object poses on images captured from uncovered region. We show the performance of Gen6D using reference images of Fig. 6 (b) in Table 5.

Table 3. Performance of the Gen6D estimator on the GenMOP dataset with different numbers of reference images. Two metrics are averaged among all objects. "All" means using all images ($\sim$ 200) from the sequence as reference images.

Metrics	Reference image number					
	All	128	64	32	16	8
ADD-0.1d	50.39	51.30	49.68	39.45	33.56	27.55
Prj-5	82.64	83.28	82.30	81.03	73.51	37.33

Table 4. Prj-5 on GenMOP dataset of Gen6D with different reference image numbers.

# Ref. Img.	Object Name					avg.
	Chair	PlugEN	Piggy	Scissors	TFormer	
All	55.00	72.90	92.96	93.53	98.81	82.64
128	58.00	69.63	96.48	92.67	99.60	83.28
16	34.00	69.63	90.95	79.74	93.25	73.51
8	11.00	61.68	28.14	26.29	59.52	37.33

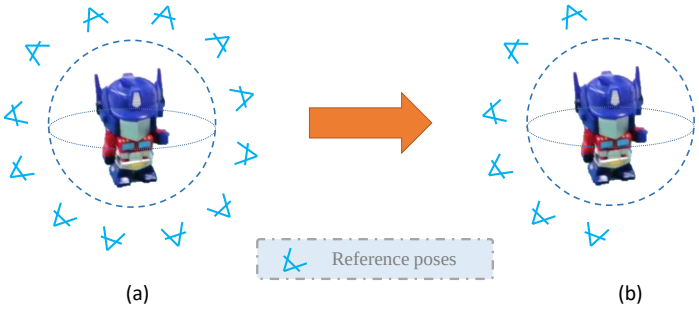

Fig. 6. (a) Reference images are homogeneously-distributed around the object. (b) Reference images only distribute in the Y- half space.

Table 5. Performance of Gen6D on the GenMOP dataset using homogeneously-distributed reference images ("Even") or only images whose camera centers are in the Y- space (Fig. 6(b)) ("Partial").

Metrics	Ref. Img.	Object Name					avg.
		Chair	PlugEN	Piggy	Scissors	TFormer	
ADD-0.1d	Even	61.50	19.63	75.38	32.76	62.70	50.39
	Partial	43.00	8.53	56.28	5.17	30.95	28.79
Prj-5	Even	55.00	72.90	92.96	93.53	98.81	82.64
	Partial	32.50	7.48	73.37	17.24	47.22	35.56

5.4 Refinement iterations

In Table 6, we show results on the GenMOP dataset with different refinement iterations. The results show that applying 1 refinement iteration already greatly improves performance from 17.90 to 38.59 on the ADD-0.1d metric. Further applying 2 refinement iteration will continuously improve the results while using 3 iterations does not.

Table 6. Results of our Gen6D estimator on the GenMOP dataset. "#Refine" means the number of refinement iterations used to produce the results.

Metrics	#Refine	Object Name					avg.
		Chair	PlugEN	Piggy	Scissors	TFormer	
ADD-0.1d	0	14.00	7.48	39.70	16.81	11.51	17.90
	1	50.50	9.81	55.28	24.57	52.78	38.59
	2	**62.00**	**29.91**	**80.90**	**37.07**	56.75	**53.32**
	3	61.50	19.63	75.38	32.76	**62.70**	50.39
Prj-5	0	11.50	40.65	33.17	34.05	64.29	36.73
	1	44.00	71.03	92.96	84.48	95.24	77.54
	2	51.50	**72.90**	**94.97**	**94.40**	**99.60**	**82.67**
	3	**55.00**	**72.90**	92.96	93.53	98.81	82.64

5.5 Ablations on training data

To show the effects of different training data, we show the performance of Gen6D using different training sets in Table 7. Training only on synthetic datasets suffers from the domain gap between the real data and synthetic data. Using real data for training greatly improves the results. LINEMOD [2] brings more obvious improvements than adding GenMOP. The main reason is that LINEMOD has more images ($\sim$1200) on every object while GenMOP only has $\sim$ 200 reference images and $\sim$ 200 query images for training.

5.6 Symmetric objects

Gen6D is able to predict poses for symmetric objects. We evaluate Gen6D on two unseen symmetric objects from LINEMOD [2], i.e. "glue" and "eggbox". Qualitative results are shown in Fig. 7 and quantitative results are shown in Table 8. The results show that our method is able to achieve reasonable performance on symmetric objects. The main reason is that Gen6D is based on matching query images with reference images. Though symmetry makes multiple feasible poses for a query image, the selector and refiner of Gen6D are able to find reference images near to one of such feasible poses.

5.7 Imperfect viewpoint selection

Table 7. Performance of Gen6D on the GenMOP and LINEMOD [2] datasets with different training sets. "Syn." means the ShapeNet [1] and Google Scanned Objects [9] datasets. "GMP" means 5 training objects from the GenMOP dataset. "LM" means 5 training objects from the LINEMOD [2] dataset. All test objects are not in the training set.

Metrics	Trainset	Object Name					avg.
		Chair	PlugEN	Piggy	Scissors	TFormer	
ADD-0.1d	Syn.+GMP+LM	**61.50**	**19.63**	**75.38**	32.76	**62.70**	**50.39**
	Syn.+GMP	39.00	11.21	71.86	**37.93**	39.68	39.94
	Syn.	30.00	21.96	61.81	24.57	30.16	33.70
Prj-5	Syn.+GMP+LM	**55.00**	72.90	**92.96**	**93.53**	**98.81**	**82.64**
	Syn.+GMP	34.50	**77.57**	92.46	91.38	89.68	77.18
	Syn.	10.05	**77.57**	68.84	74.57	98.41	65.98
Metrics	Trainset	cat	duck	bvise	cam	driller	avg.
ADD-0.1d	Syn.+GMP+LM	**60.68**	**40.47**	**77.03**	**66.67**	**67.39**	**62.45**
	Syn.+GMP	40.92	16.24	62.11	45.59	48.76	42.72
	Syn.	31.04	11.64	61.63	35.39	54.61	38.86

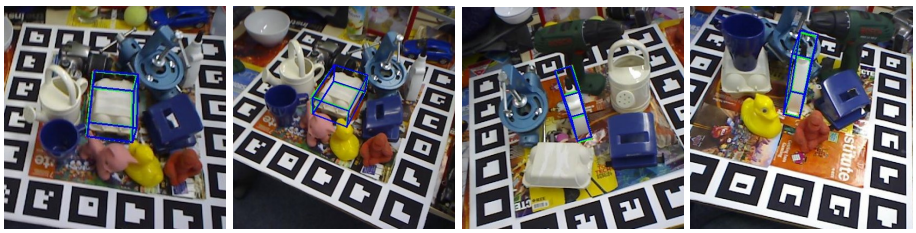

Fig. 7. Qualitative results on symmetric objects.

Table 8. Results on symmetric objects of the LINEMOD [2] dataset. For the ADD metric, we report "ADD-S-0.1d" which computes the nearest distance between the object points transformed by ground-truth pose and the estimated pose. Note PVNet [5] is trained on the specific test object with both synthetic and real images while our Gen6D is not trained on the test object.

Name	Prj-5		ADD-S-0.1d	
	PVNet [5]	Ours	PVNet [5]	Ours
Eggbox	99.34	97.84	99.15	98.40
Glue	98.45	96.24	95.66	87.16

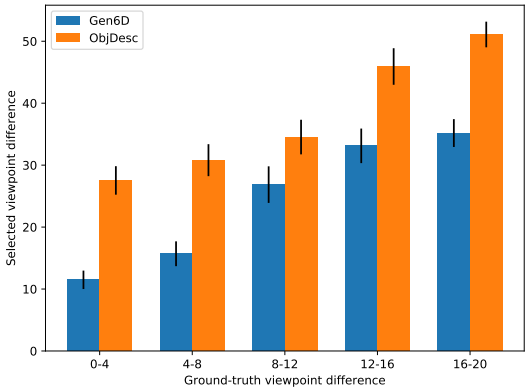

Fig. 8. Viewpoint difference between the selected reference image and the query image (y-axis); Viewpoint difference between the ground-truth reference image and the query image (x-axis).

As discussed in the Section 1 in the main paper, it would be challenging for selector to select the most similar reference image when there is no reference image with an exactly same viewpoint as the query image. To show this, we show Fig. 8, where the x-axis shows the viewpoint difference between the ground-truth nearest reference image and the query image while the y-axis shows the viewpoint difference between the selected reference image and the query image. The viewpoint difference is computed as the angle between the query viewpoint and the reference viewpoint by $\arccos \tilde{u} \cdot \tilde{v}$. With the increase of viewpoint difference between the ground-truth reference image and the query image, the viewpoint difference between the selected reference image and the query image is also increasing. However, the proposed selector is able to select more accurate reference image than the baseline ObjDesc [10]. Figure 9 also shows some examples.

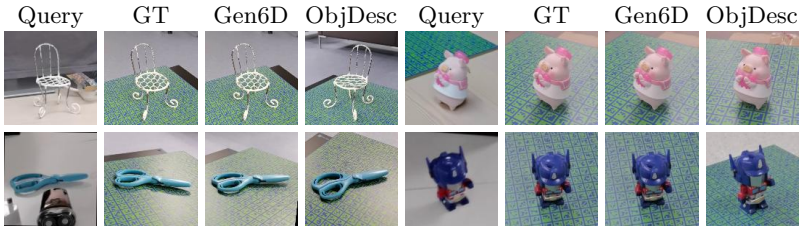

Fig. 9. The input query image, the ground-truth reference image with nearest viewpoint, the reference images selected by Gen6D and ObjDesc [10].

Fig. 10. Additional qualitative results on the LINEMOD [2] dataset. Ground-truth poses are drawn in green while predicted poses are drawn in blue.

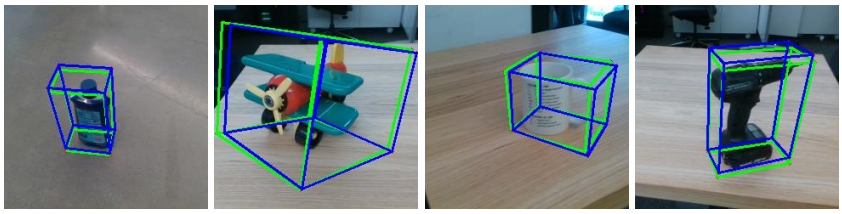

Fig. 11. Qualitative results on the MOPED [4] dataset. Ground-truth poses are drawn in green while predicted poses are drawn in blue.

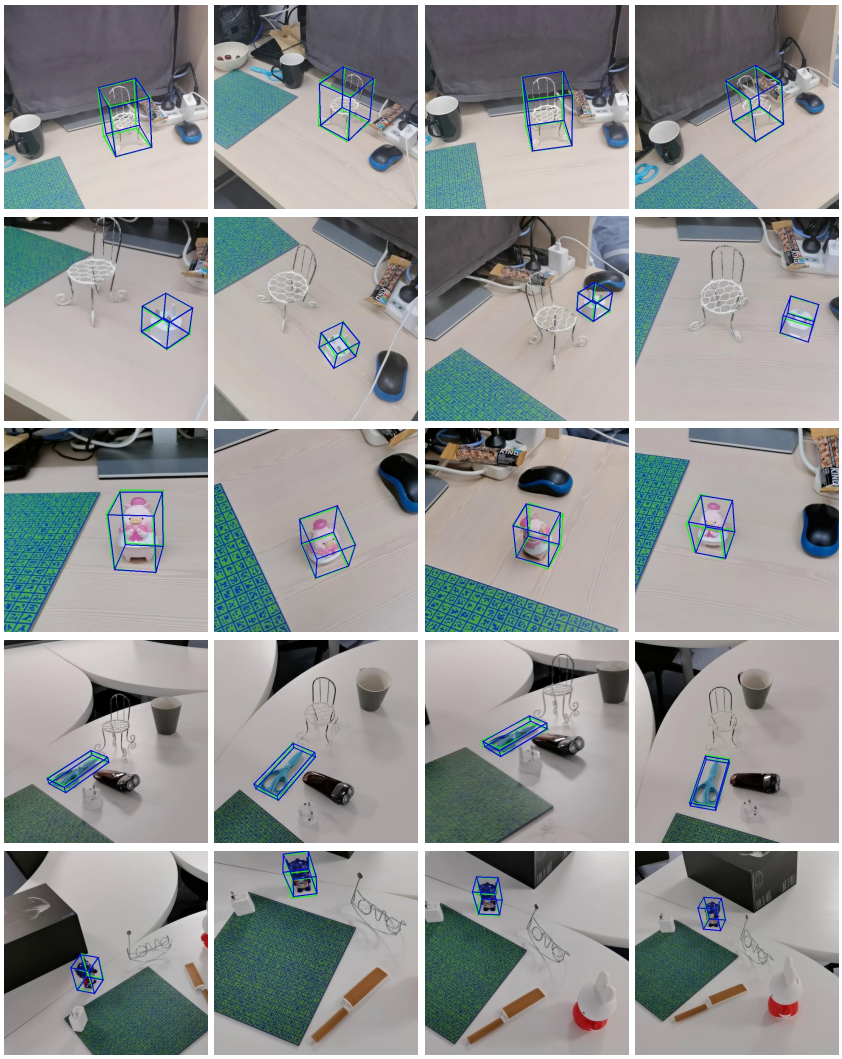

Fig. 12. More qualitative results of Gen6D on the GenMOP dataset.

References

1. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
2. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: ACCV (2012)
3. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: ECCV (2018)
4. Park, K., Mousavian, A., Xiang, Y., Fox, D.: Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In: CVPR (2020)
5. Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnet: Pixel-wise voting network for 6-dof pose estimation. In: CVPR (2019)
6. Reizenstein, J., Shapovalov, R., Henzler, P., Sbordone, L., Labatut, P., Novotny, D.: Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In: CVPR (2021)
7. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016)
8. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6d object pose prediction. In: CVPR (2018)
9. Wang, Q., Wang, Z., Genova, K., Srinivasan, P.P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: CVPR (2021)
10. Wohlhart, P., Lepetit, V.: Learning descriptors for object recognition and 3d pose estimation. In: CVPR (2015)