Supplementary Material of GPV-Pose: Category-level Object Pose Estimation via Geometry-guided Point-wise Voting

Yan Di[1*], Ruida Zhang[2*], Zhiqiang Lou[2], Fabian Manhardt[3],
Xiangyang Ji[2], Nassir Navab[1] and Federico Tombari[1,3]
[1]Technical University of Munich, [2]Tsinghua University, [3]Google
{*zhangrd21@mails. lzq20@mails. xyji@}tsinghua.edu.cn, shangbuhuan13@gmail.com
fabianmanhardt@google.com, tombari@in.tum.de

1. Details of Different Loss Terms

1.1. Basic Loss Terms

We follow FS-Net [2] for the basic loss terms of translation, rotation and size.

For rotation, we decompose the rotation matrix R_{gt} as plane normals of the bounding box $R_{gt} = \left[r_x^{gt}, r_y^{gt}, r_z^{gt}\right]$. We predict the first two normals, denoted as r_x, r_y. The loss is defined as,

$$\mathcal{L}_{rot}^{Basic} = \left\|r_x^{gt} - r_x\right\|_1 + \left\|r_y^{gt} - r_y\right\|_1 \tag{1}$$

Note that we normalize the predicted plane normals before calculating the loss.

As for the translation t, GPV-Pose predicts the residual translation t_*, which is the difference between translation t and the mean M_p of input point cloud.

$$t = t_* + M_p \tag{2}$$

Then we have the translation loss as,

$$\mathcal{L}_{trans}^{Basic} = \left\|t_{gt} - t\right\|_1 \tag{3}$$

where t_{gt} is the ground truth translation.

For size, we predict the residual size s_*, which indicates the difference between the real size s and the pre-computed mean size C_m. C_m is the mean object size of all instances within a certain category in the training dataset. The predicted size is,

$$s = s_* + C_m \tag{4}$$

The size loss is defined as,

$$\mathcal{L}_{size}^{Basic} = \left\|s_{gt} - s\right\|_1 \tag{5}$$

where s_{gt} is the ground truth size. Finally, we have,

$$\begin{aligned}\mathcal{L}^{Basic} =& \lambda_{rot}\mathcal{L}_{rot}^{Basic} + \lambda_{trans}\mathcal{L}_{trans}^{Basic} + \lambda_{sym}\mathcal{L}_{sym}^{Basic} \\ & \lambda_{size}\mathcal{L}_{size}^{Basic} + \lambda_{conf}\mathcal{L}_{conf}^{Basic}\end{aligned} \tag{6}$$

[*]Authors with equal contributions.

where $\mathcal{L}_{sym}^{Basic}$ denotes the loss term for symmetry reconstruction, which will be introduced in Sec. 4, and $\mathcal{L}_{conf}^{Basic} = \mathcal{L}_{r_c}^{Basic} + \mathcal{L}_{p_c}^{Basic}$ denotes the loss for confidence, whose definition is illustrated in Eq. 1 and Eq. 6 in the main text. $\{\lambda_{rot}, \lambda_{trans}, \lambda_{size}, \lambda_{size}, \lambda_{conf}\} = \{1.0, 1.0, 1.0, 1.0, 1.0\}$ are weighting parameters. Note that we use $\mathcal{L}_1$ loss to replace the $\mathcal{L}_2$ loss and cosine distance loss used in FS-Net [2] for faster convergence.

1.2. Bounding Box Based Loss Terms

We construct consistency terms between the predicted pose and voted bounding box to boost performance. Following descriptions in the main text, firstly we recover the plane parameters N_i, D_i for each of the six faces of the bounding box $i \in \mathcal{B}$, $\mathcal{B} = \{y+, y-, x+, x-, z+, z-\}$, by the weighted least square method. The bounding box based loss term $\mathcal{L}_{(R,t,s)}^{BB}$ includes the rotation term $\mathcal{L}_{(R)}^{BB}$, translation term $\mathcal{L}_{(t)}^{BB}$ and size term $\mathcal{L}_{(s)}^{BB}$. The loss is calculated for each of the six voted faces of the bounding box respectively and summed up as the final loss term. For rotation, the consistency term is,

$$\mathcal{L}_{(R)}^{BB} = \sum_{i \in \mathcal{B}} \left\|r_i' - N_i\right\|_1 \tag{7}$$

where r_i' is the calibrated predicted rotation vectors for bounding box face $i \in \mathcal{B}$ (see Sec. 3),

For translation and size consistency, we first locate the object center, which in fact is the predicted translation t. Then we calculate the distance between the object center and each bounding box plane, denoted as μ_i, $i \in \mathcal{B}$,

$$\mu_i = \left\|t^T N_i - D_i\right\|_1 \tag{8}$$

Finally, since the distance from the object center to two arbitrary opposite faces (*e.g.* face $y+$, $y-$) should be the same and equal to half of the size along the corresponding axis, we define the translation and size consistency terms as fol-

lows,

$$\mathcal{L}_{(t)}^{BB} = \sum_{i \in \{x,y,z\}} \|\mu_{i+} - \mu_{i-}\|_1 \tag{9}$$

$$\mathcal{L}_{(s)}^{BB} = \sum_{i \in \mathcal{B}} \|\mu_i - s_i/2\|_1 \tag{10}$$

2. Details of Bounding Box Voting

In order to recover the object bounding box via point-wise voting, we aggregate the normal direction, distance and confidence n_p, d_p, c_p of each point $p = [p_x, p_y, p_z]^T$ in the point cloud P_o. We first locate the corresponding point of p on the bounding box $p' = [p'_x, p'_y, p'_z]^T$, using $p' = p + d_p n_p$, and retrieve the predicted point cloud on the bounding box face P_b. Then we calculate the plane parameter by means of the weighted least square algorithm. The plane parameter is described as $A = [a_x, a_y, a_d]$, and the plane equation is $a_x p'_x + a_y p'_y + a_d = p'_z$. Then we define the weighting matrix $W = diag([c_p | p \in P_o])$, where $diag(*)$ denotes the diagonal matrix with $*$ as diagonal elements. And, we define the coefficient matrix C as,

$$C = \begin{bmatrix} p'^1_x, p'^1_y, 1 \\ \vdots \\ p'^i_x, p'^i_y, 1 \\ \vdots \\ p'^n_x, p'^n_y, 1 \end{bmatrix} \tag{11}$$

where for each point in P_b, its first and second coordinates are stacked to construct C. n is the number of points in P_b. Similarly, we define the fitting goal as b, consisting of the z-coordinate of each point in P_b.

$$b = \begin{bmatrix} p'^1_z \\ \vdots \\ p'^i_z \\ \vdots \\ p'^n_z \end{bmatrix} \tag{12}$$

Finally, we construct the optimization goal as $f(A) = ||W(CA - b)||_2^2$. Utilizing the least squared algorithm, the plane parameter can be recovered in a closed form as,

$$A = (C^T W^T W C)^{-1} C^T W^T W b \tag{13}$$

3. Details of Confidence-aware Rotation Recovery

In the main text, in Fig. 4(b), we briefly introduce how to calibrate the predicted plane normals r_y, r_x to be the perpendicular normals r'_y, r'_x, given their confidence c_y, c_x. we minimize the following cost function for calibration,

$$\begin{aligned} \theta_1^*, \theta_2^* = arg \ \ min \ \ c_y \theta_1^2 + c_x \theta_2^2 \\ s.t. \ \ \theta_1 + \theta_2 + \pi/2 = \theta, \end{aligned} \tag{14}$$

where θ denotes the angle between r_x and r_y, From Eq. 14 we then obtain

$$\begin{cases} \theta_1^* = \frac{c_x}{c_x + c_y}(\theta - \frac{\pi}{2}) \\ \theta_2^* = \frac{c_y}{c_x + c_y}(\theta - \frac{\pi}{2}). \end{cases} \tag{15}$$

To obtain the final plane normals r'_y and r'_x with θ_1^* and θ_2^*, we first calculate the rotation axis as $r'_z = r_x \times r_y = [k_x, k_y, k_z]$. By applying the Rodrigue's rotation formula, we have following equations,

$$\begin{aligned} R_{\Delta y} = cos\theta_1^* I + (1 - cos\theta_1^*) {r'_z}^T r'_z \\ + sin\theta_1^* \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \end{aligned} \tag{16}$$

$$\begin{aligned} R_{\Delta x} = cos\theta_2^* I + (1 - cos\theta_2^*) {r'_z}^T r'_z \\ + sin\theta_2^* \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \end{aligned} \tag{17}$$

$$r'_y = R_{\Delta y} r_y \tag{18}$$

$$r'_x = R_{\Delta x} r_x \tag{19}$$

The rotation matrix is finally recovered as

$$R' = \begin{bmatrix} r'_x & r'_y & r'_z \end{bmatrix} \tag{20}$$

4. Symmetry Prior and Symmetric Reconstruction

We exploit two types of symmetry: **Reflection Symmetry** and **Rotational Symmetry** [4]. More specifically, we subdivide the symmetry into 4 sub-classes for convenience : rotational symmetry around y axis (S_y), reflection symmetry w.r.t. the xy-plane (S_{xy}), reflection symmetry w.r.t. the xz-plane (S_{xz}) and reflection symmetry w.r.t. the yz-plane (S_{yz})). The detailed symmetry prior for each category in NOCS [8] and LineMod [3] dataset is listed in Tab. 1 and Tab. 2 respectively.

Since the handle of a mug is oftentimes occluded, we separate *mug* into two sub-categories, with and without handle. Also, in order to apply symmetry prior for *mug* category, we follow SPD [7] and move the origin of object coordinate along x axis to the middle of the cylinder part.

category	S_y	S_{xy}	S_{xz}	S_{yz}	Type
bottle	1	1	0	1	T2
bowl	1	1	0	1	T2
camera	0	0	0	0	T1
can	1	1	1	1	T2
laptop	0	1	0	0	T3
mug w.h.	0	1	0	0	T3
mug wo.h.	1	0	0	0	T2

Table 1. Symmetry prior for each category in NOCS dataset [8]. w.h. and wo.h. denote with handle and without handle respectively.

category	S_y	S_{xy}	S_{xz}	S_{yz}	Type
ape	0	0	0	0	T1
benchvise	0	1	0	1	T3
bowl	1	1	0	1	T2
camera	0	0	0	0	T1
can	0	1	0	0	T3
cat	0	0	0	0	T1
cup	0	1	0	0	T3
driller	0	1	0	0	T3
duck	0	1	0	1	T3
eggbox	0	1	0	0	T3
glue	0	1	0	0	T3
holepuncher	0	1	0	0	T3
iron	0	1	0	0	T3
lamp	0	1	0	0	T3
phone	0	0	0	0	T1

Table 2. Symmetry prior for each category in LineMod dataset [3].

We first define the basic loss term for directly supervising the symmetry reconstruction depending on the reconstruction type. For type T1, *e.g. camera*, we directly reconstruct the input point cloud. For type T2, *e.g. bottle*, we reconstruct the input point cloud w.r.t. the symmetry axis. For type T3, *e.g. laptop*, we reconstruct the point cloud w.r.t. the corresponding reflection plane. Thus for a point p, its corresponding point $G_{sym}(p)$ under the canonical view w.r.t. different symmetry types is defined as follows,

$$G_{sym}(p) = \begin{cases} [p_x, p_y, p_z]^T & T1 \\ [-p_x, p_y, -p_z]^T & T2 \\ [p_x, p_y, -p_z]^T & T3 \end{cases} \tag{21}$$

Thereby, for any point in the observed point cloud, its corresponding symmetry point under the camera view is,

$$G(p) = R_{gt} G_{sym}(R_{gt}^T p - t_{gt}) + t_{gt} \tag{22}$$

Therefore, we define the symmetry reconstruction loss as,

$$\mathcal{L}_{sym,1}^{Basic} = \sum_{p \in P_o} \|p'_s - G(p)\|_1 \tag{23}$$

where p'_s denotes the predicted symmetry point of p.

We also adopt a consistency loss between the reconstructed points and the predicted pose to boost performance.

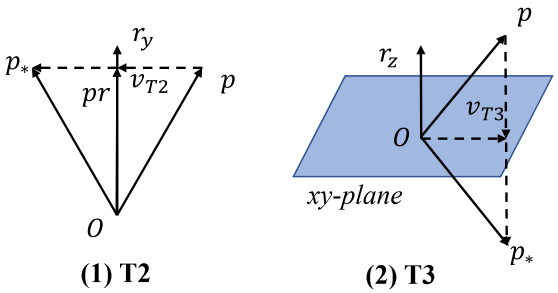

Figure 1. Illustration of the computation of the consistency loss in symmetry reconstruction.

We first use the predicted pose to transform the input points to the corresponding symmetry points and then compare the difference between the transformed points and predicted symmetry points. The procedure is described in Fig. 1. Then the function $\varepsilon(*)$ in Eq. 4 in the main text is defined according to the symmetry type as follows. For type T1, the category has no symmetric characteristics, so we directly use,

$$\varepsilon_{T1}(p) = p \tag{24}$$

For type T2, as shown in Fig. 1 (1), we need to calculate the symmetric point p_* of point p w.r.t. the y-axis. We first compute its projection p_{proj} on the y-axis,

$$p_{proj} = \langle p - t, r'_y \rangle r'_y = (p-t)^T r'_y r'_y \tag{25}$$

where r'_y denotes the predicted y-axis rotation vector and $\langle *, * \rangle$ denotes the inner product. Then we calculate v_{T2} that is perpendicular to the y axis as,

$$v_{T2} = p_{proj} - (p - t) \tag{26}$$

The symmetric point can then be computed by,

$$p_* = p + 2v_{T2} \tag{27}$$

Thereby, we can define $\varepsilon_{T2}(p)$ as,

$$\varepsilon_{T2}(p) = p + 2v_{T2} \tag{28}$$

For type T3, as shown in Fig. 1 (2), we calculate p_* that is symmetric to p w.r.t. the xy-plane. For other reflection planes, the calculation is similar. We compute v_{T3} as,

$$v_{T3} = \langle t - p, r'_z \rangle r'_z \tag{29}$$

where r'_z is the predicted z-axis rotation vector. Then $\varepsilon_{T3}(p)$ is defined as,

$$\varepsilon_{T3}(p) = p + 2v_{T3} \tag{30}$$

To summarize, $\varepsilon(*)$ in Eq. 4 is defined as,

$$\varepsilon(p) = \begin{cases} \varepsilon_{T1}(p) & T1 \\ \varepsilon_{T2}(p) & T2 \\ \varepsilon_{T3}(p) & T3 \end{cases} \tag{31}$$

Finally, we define the symmetry-based consistency loss as follows,

$$\mathcal{L}_{sym,2}^{Basic} = \sum_{p \in P_o} \|p'_s - \varepsilon(p)\|_1 \tag{32}$$

where p'_s is the predicted symmetry point of p. The final symmetry reconstruction loss is $\mathcal{L}_{sym}^{Basic} = \mathcal{L}_{sym,1}^{Basic} + \lambda_s \mathcal{L}_{sym,2}^{Basic}$, where $\lambda_s = 1.0$ is the weighting parameter.

5. Limitations

Modality of Input Data. Our framework GPV-Pose only utilizes the point cloud sampled from the input depth image for pose estimation to boost the inference speed and get rid of the influences of intra-class color variations. However, this strategy has 2 shortcomings compared to RGB-D based methods. First, the translation is sometimes hard to accurately predict for a certain category, *e.g. laptop*. In GPV-Pose, the translation is recovered by adding the predicted residual translation and the mean of the input point cloud. However, if the object center is outside the object (*e.g. laptop*), the translation residual is hard to predict. Under $3D_{75}$, our result is 39.7, while SGPA [1] is about 59.0. Second, the input point cloud is obtained by randomly sampling on the cropped depth map, which introduces randomness into our method and undermines robustness. In a nutshell, our point cloud based pose estimator enables efficient inference, but adding RGB information may improve the accuracy.

Influence of Outliers. Due to the limits of depth sensor, the input depth around the boundary of the object is often of low quality. Meanwhile, the segmentation based on RGB image is also inaccurate on the object boundary. Thereby, the sampled point cloud contains a non-trivial number of outliers, which occupies more than one tenth of total points in some cases. However, our method doesn't use any point cloud segmentation method to remove outlier points as FS-Net [2], instead we only mitigate the influence of outliers with the geometric constraints in Eq. 9 for the sake of computation efficiency. Our geometric constraints perform robustly in most times, but outliers may still deteriorate our performance to some extent.

Dependency on 2D detection results. For fair comparison with previous methods [1, 5, 7], we employ an off-the-shelf object detector Mask-RCNN and train it solely on REAL275 dataset. This training strategy leads to some poor 2D detection results for ambiguous objects, *i.e. camera, thin bottles*. Since our method takes the point clouds back-projected from the cropped depth maps as input, poor detection results further limit our pose accuracy, especially under the IoU metric. This could be mitigated to some extent by utilizing specially designed detectors (similar to FS-Net [2]).

Potential Negative Societal Impact. Our method inherently has no significant negative societal impact. We will try our best to resolve the problems caused by our method.

6. Details of Network Design

Choice of backbone. We use a point cloud based approach as point clouds are generally better at describing 3D scenes than 2D depth maps. For example, neighboring pixels in 2D can be far away in 3D, which is hard to capture with CNNs as they are not strong at handling the resulting high frequencies. Hence, most recent SOTA methods (*e.g.* FS-Net [2], DualPoseNet [5], SGPA [1]) all operate on point clouds. 3DGC [6] is insensitive to shift and scale and capable to extract pose-sensitive feature, which is proved by FS-Net [2]. Thereby, we choose 3DGC as our backbone.

Direct pose regression. Many previous methods typically predict the NOCS coordinate for each observed point and rely on Umeyama algorithm + RANSAC methods for pose recovery. This strategy leads to inefficient inference process, thus we directly regress the pose for faster inference speed. In addition, using direct regression enabled our confidence-aware rotation representation, which demonstrates superior performance to competitors (*cf*. **Tab. 2 row: A1-A2** in the main text).

7. Choice of Hyperparameters

Loss weights. As several basic loss terms are borrowed from FS-Net, we simply adopt their weighting scheme. All other weights are chosen empirically such that the initial loss ranges are roughly the same to avoid any bias during training. We find that our obtained results are fairly resilient towards the choice of weights.

Other hyperparameters. First, $\{k_1, k_2\}$ in Eq. 1 control the number of points we use for calculating each bounding box face. Further, $\{k_n, k_s, k_p\}$ in Eq. 9-11 control the outlier filtering step. We select $\{k_n, k_s, k_p\}$ to preserve at least 90% points. Yet, similar to the loss weights, the performance of GPV-Pose is not sensitive to the hyperparameters as long as they are a sensible choice with respect to bounding box face and outlier filtering.

8. Details of Data Augmentation

To avoid overfitting, we follow FS-Net [2] and employ 4 kinds of augmentation on the point cloud: **random scaling, random uniform noise, random rotational and translational perturbations**, and **bounding box based adjustment**. Each kind of augmentation introduces small perturbations on the point cloud. We set the probabilities to conduct the first three kinds of augmentation to be 0.3, and 0.2 for the last one.

In **random scaling**, we re-scale the input point cloud along the x, y, z axis with the ratios e_x, e_y, e_z respec-

Method	$3D_{50}$	$3D_{75}$	$5°2cm$	$5°5cm$	$10°5cm$
NOCS [8]	83.9	69.5	32.3	40.9	64.6
SPD [7]	93.2	83.1	54.3	59.0	81.5
CR-Net [9]	**93.8**	88.0	72.0	76.4	87.7
SGPA [1]	93.2	88.1	70.7	74.5	88.4
DualPoseNet [5]	92.4	86.4	64.7	70.7	84.7
Ours	92.9	86.6	67.4	76.2	87.4
Ours(M)	93.4	**88.3**	**72.1**	**80.1**	**89.0**

Table 3. **Comparison with state-of-the-art methods on CAMERA25 dataset.**

category	$3D_{25}$	$3D_{50}$	$3D_{75}$	$5°2cm$	$5°5cm$	$10°5cm$	$10°10cm$
bottle	57.7	57.7	48.1	34.3	39.6	91.7	93.3
bowl	100	100	97.7	63.2	75.8	100	100
camera	90.9	86.0	46.4	0.4	0.6	10.4	10.4
can	71.4	71.4	64.0	46.9	56.5	97.5	97.5
laptop	85.4	84.6	39.7	36.0	73.2	90.0	95.0
mug	99.6	98.3	90.8	11.0	11.7	50.4	51.1
average	84.2	83.0	64.4	32.0	42.9	73.3	74.6

Table 4. **Per-category results of our method on REAL275 dataset.**

Category	$3D_{50}$	$3D_{75}$	$5°2cm$	$5°5cm$	$10°5cm$
bottle	93.7	87.8	75.5	95.2	97.2
bowl	96.8	96.5	95.7	96.9	99.6
camera	86.7	74.0	54.1	62.3	74.2
can	92.4	92.2	98.4	99.3	99.4
laptop	97.0	87.5	67.5	83.9	91.3
mug	93.8	91.9	42.2	42.7	72.0
average	93.4	88.3	72.1	80.1	89.0

Table 5. **Per-category results of Ours(M) on CAMERA25 dataset.**

tively. The ratios are generated from the uniform distribution $U(0.8, 1.2)$. For each point $p = [p_x, p_y, p_z]^T$ in the input point cloud, the point after random scaling is,

$$f_{rs}(p) = [p_x e_x, p_y e_y, p_z e_z]^T \tag{33}$$

To add **random uniform noise**, we add random noise sampled from the uniform distribution $U(-50mm, 50mm)$ onto the coordinates of each point.

In **random rotational and translational perturbations**, we apply random rotation R_{aug} and translation t_{aug} onto the ground truth rotation and translation R_{gt}, t_{gt}. For a point p, its corresponding point after augmentation is,

$$f_{rttp}(p) = R_{aug}(R_{gt}^T(p - t_{gt})) + t_{aug} \tag{34}$$

The random rotation R_{aug} is computed by sampling three Euler angles from the uniform distribution $U(-15°, 15°)$ respectively and then convert them into the rotation matrix. The random translation t_{aug} is sampled from the uniform distribution $U(-50mm, 50mm)$.

In **bounding box based adjustment**, we change the size of the top (y+) and bottom (y-) faces of bounding box with ratio e_{top}, e_{bottom} respectively. e_{top}, e_{bottom} are generated with the uniform distribution $U(0.8, 1.2)$. By adjusting the object bounding box, the points are also adjusted accordingly. Note that we only augment the bounding box for *bowl* and *mug*. Then for a point p, its corresponding point after augmentation is,

$$r(p) = (p_y/s_y + 0.5) * (e_{top} - e_{bottom}) + e_{bottom} \tag{35}$$

$$f_{bc}(p) = \{p_x * r(p), p_y, p_z * r(p)\} \tag{36}$$

where s_y denotes the ground truth size along y axis.

Moreover, during training, we add random noise to the ground truth mask before sampling the point cloud from the depth map in order to enhance robustness. In this manner, we ensure that a certain number of outliers are incorporated in the training stage and thus enables outlier removal capability during inference.

9. Overall Performance on CAMERA25.

In Tab. 3, we compare GPV-Pose with state-of-the-art competitors on the synthetic CAMERA25 [8] dataset for category-level tasks. Since DO-Net [4] and FS-Net [2] do not provide results on CAMERA25, we do not compare with them. Note that we provide two variants of our method. **Ours(M)** trains a separate model for each category,

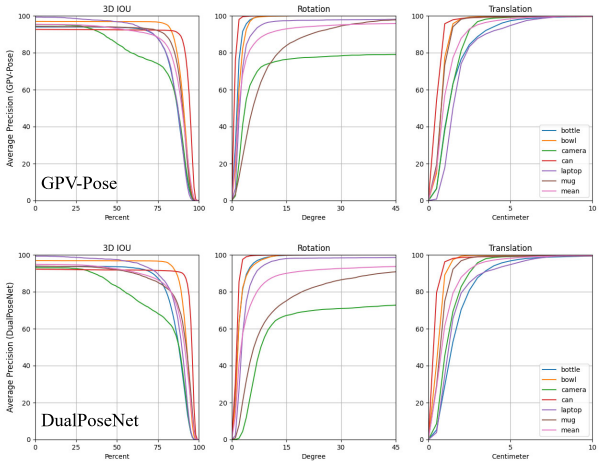

Figure 2. Per-category comparison of Ours(M) with DualPoseNet on CAMERA25

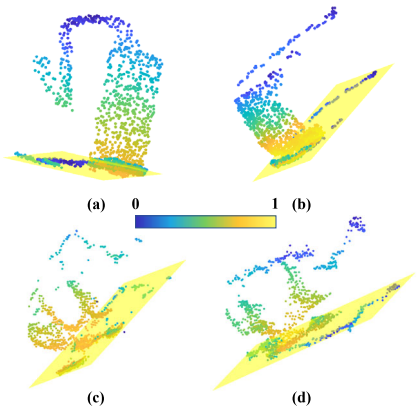

Figure 3. Confidence to support the bounding box plane y+ of each observed point and the recovered bounding box plane (yellow). Note that the confidence value is normalized beforehand for visualization. The outliers are preserved to show the robustness of our method.

while **Ours** trains a single model for all categories. We train the network for 300 epochs for **Ours** and 100 epochs for **Ours(M)**. Other hyperparameters keep the same as on REAL275.

From Tab. 3, it can be deduced that GPV-Pose achieves state-of-the-art performance w.r.t 4 out of total 5 metrics. Specifically, for the $5°5cm$ metric, we surpass CR-Net [9] with 80.1 compared to 76.4. This superior performance on the synthetic dataset proves the robustness of our method in various domains. In Fig. 2, we additionally present a detailed per-category comparison of our method with DualPoseNet [5]. As one can easily deduce, we outperform DualPoseNet by a large margin, especially when focusing on the rotation results for non-symmetrical objects such as *camera*. Moreover, Tab. 5 lists the detailed results of **Ours(M)** on each category on CAMERA25.

10. Supplementary Results on REAL275

In Tab. 4, we provide per-category results of GPV-Pose on REAL275. In Fig. 3, we additionally provide 4 examples with several outliers, and visualize the predicted confidence weights for the bounding box voting and the recovered bounding box plane. Our bounding box voting algorithm works robustly with outliers. In Fig. 4, 5, we demonstrate more qualitative results of our method in comparison with DualPoseNet [5]. It can be seen clearly that our method consistently outperforms DualPoseNet, especially in the estimation of rotation.

References

[1] Kai Chen and Qi 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. 4, 5

[2] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, Shen Linlin, 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 (CVPR)*, pages 1581–1590, June 2021. 1, 4, 5

[3] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient Response Maps for Real-Time Detection of Textureless Objects. *TPAMI*, 2012. 2, 3

[4] Haitao Lin, Zichang Liu, Chilam Cheang, Lingwei Zhang, Yanwei Fu, and Xiangyang Xue. Donet: Learning category-level 6d object pose and size estimation from depth observation. *arXiv preprint arXiv:2106.14193*, 2021. 2, 5

[5] 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. *arXiv preprint arXiv:2103.06526*, 2021. 4, 5, 6

[6] 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. 4

[7] Meng Tian, Marcelo H Ang, and Gim Hee Lee. Shape prior deformation for categorical 6d object pose and size estimation. In *European Conference on Computer Vision*, pages 530–546. Springer, 2020. 2, 4, 5

[8] 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/CVF Conference on*

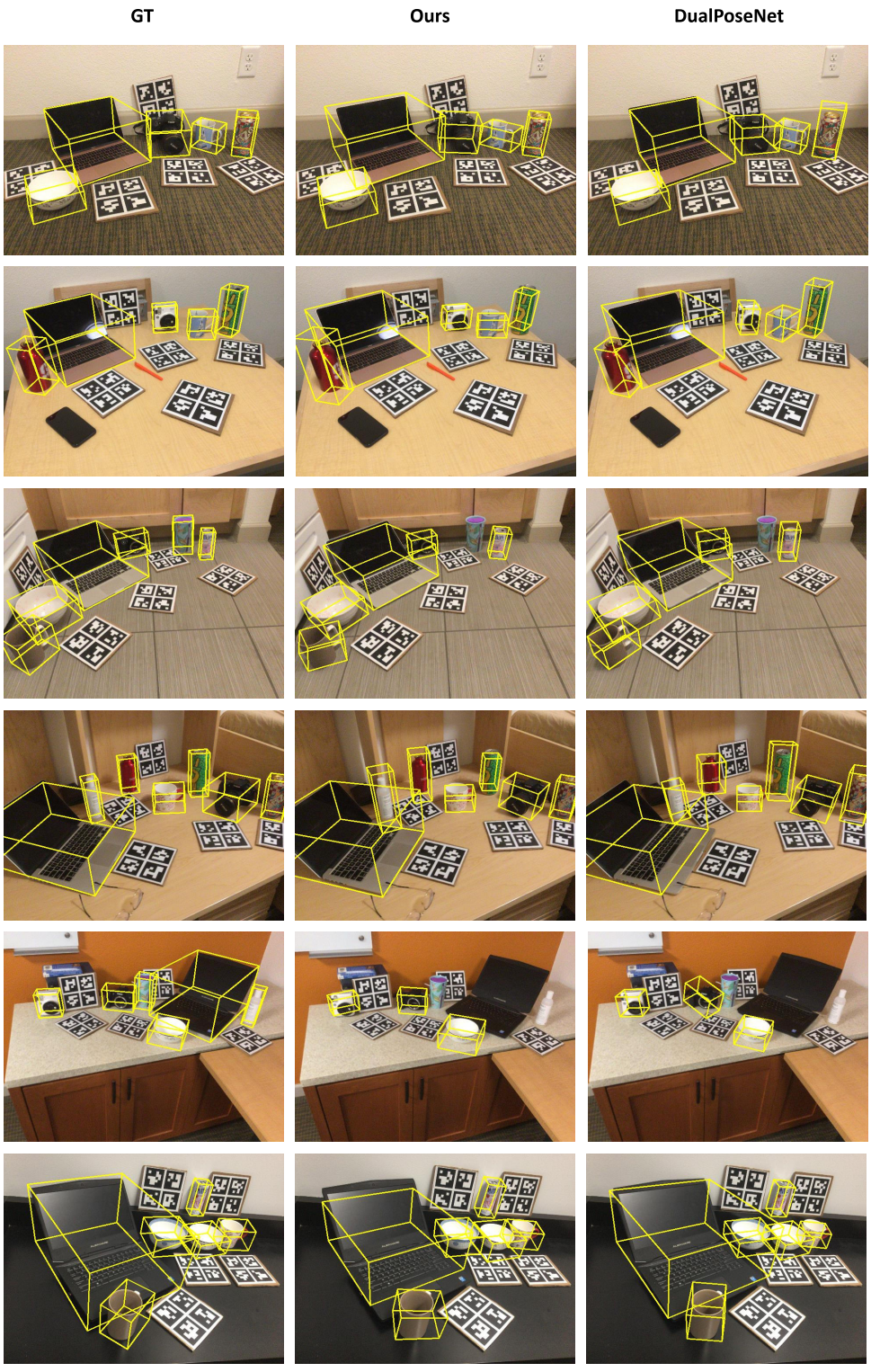

Figure 4. Qualitative results of our method and DualPoseNet on REAL275.

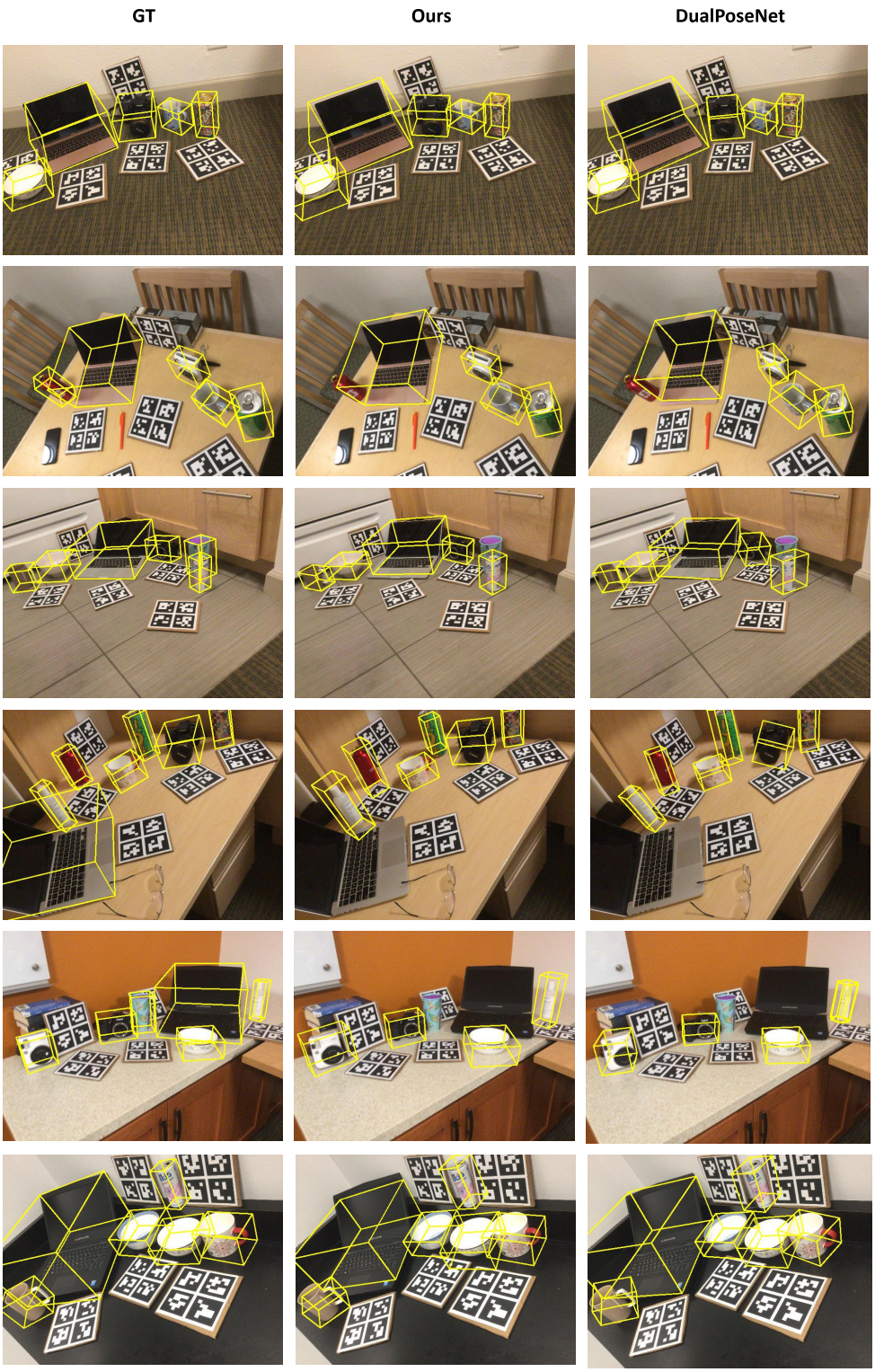

Figure 5. Qualitative results of our method and DualPoseNet on REAL275.

Computer Vision and Pattern Recognition, pages 2642–2651, 2019. 2, 3, 5

[9] Jiaze Wang, Kai Chen, and Qi Dou. Category-level 6d object pose estimation via cascaded relation and recurrent reconstruction networks. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. 5, 6