A Network Configurations in Sec. 5.1

In Sec. 5.1, we conduct experiments on LineMOD dataset [3] to confirm the advantages of our SS-Conv in terms of accuracy and efficiency, compared to other convolutions. The experiments are performed on a plain architecture, as shown in Fig. 1; we use a global average pooling layer to aggregate the backbone features, and regress the rotation $\boldsymbol{r}$ and the translation $\boldsymbol{t}$ by two separate MLPs. Illustrations of **Plain12**, **Plain24** and **ResNet50** are shown in Fig. 2 respectively, where network specifics are also given. For Plain12 and Plain24, the channels of the two MLPs in Fig. 1 are both $(256, 128, 6)$ and $(256, 128, 3)$, respectively, while for ResNet50, they are set as $(1024, 128, 6)$ and $(1024, 128, 3)$. Note that for the rotation branch, a 6D representation of rotation is generated and then transformed into a 3×3 rotation matrix $\boldsymbol{r}$ following [11].

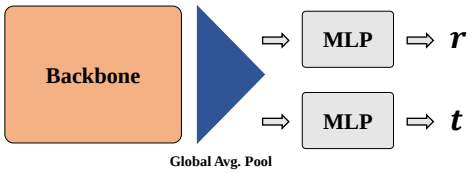

Figure 1: An illustration of the architecture used in Sec. 5.1 for instance-level 6D pose estimation.

B Experimental Details on Focused Tasks

B.1 Instance-level 6D Object Pose Estimation

Network Specifics We add a sparse average pooling on top of Plain24 (*c.f.* Fig. 2) to form the backbone for rotation-equivariant feature learning, which consists of a total of 4 pooling layers. The kernel size, stride, padding of the additional average pooling are $3 \times 3 \times 3$, 2, and 1, respectively. We use the backbone features $\{(\boldsymbol{H}_{n_i}, \boldsymbol{F}_{n_i})\}_{i=1}^{4}$ outputted by the 4 pooling layers for interpolation of point-wise features of the observed points in the first stage, and transform them into $\{(\boldsymbol{H}^{'}_{n_i}, \boldsymbol{F}^{'}_{n_i})\}_{i=1}^{4}$ with four Feature-Steering modules for the second-stage predictions. The Feature-Steering module is introduced in Sec. 4.1.1, and we supplement a more concrete illustration in Fig. 3, facilitating a better understanding. In each Feature-Steering module, we use two SS-Convs to enhance the feature representations; the kernel sizes, strides, paddings of both SS-Convs are set as $3 \times 3 \times 3$, 1, 1, respectively, and their output channels are kept consistent with the input ones. For both two stages, the output channels of the MLPs for regression of rotations are set as $(256, 128, 6)$, while those for translations are $(256, 128, 3)$.

Training Objective Given a point cloud $\mathcal{Q} = \{\mathbf{q}_i\}_{i=1}^{M}$ sampled from the CAD model of an asymmetric object with M points, we train the network in an end-to-end fashion by optimizing the following problem:

$$\min \mathcal{L} = \frac{1}{M} \sum_{i=1}^{M} \left\| (\boldsymbol{r}_1 \boldsymbol{q}_i + \boldsymbol{t}_1) - (\hat{\boldsymbol{r}} \boldsymbol{q}_i + \hat{\boldsymbol{t}}) \right\| + \lambda \left\| (\boldsymbol{r}_1 (\boldsymbol{r}_2 \boldsymbol{q}_i + \boldsymbol{t}_2) + \boldsymbol{t}_1) - (\hat{\boldsymbol{r}} \boldsymbol{q}_i + \hat{\boldsymbol{t}}) \right\|, \quad (1)$$

where $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{t}})$, $(\boldsymbol{r}_1, \boldsymbol{t}_1)$, and $(\boldsymbol{r}_2, \boldsymbol{t}_2)$ are the poses of ground truth, the first stage, and the second stage, respectively. For a symmetric object, we consider using Chamfer Distance to compute the distance between two point clouds in the optimization following [8]:

$$\min \mathcal{L} = \frac{1}{M} \sum_{i=1}^{M} \min_j \left\| (\boldsymbol{r}_1 \boldsymbol{q}_i + \boldsymbol{t}_1) - (\hat{\boldsymbol{r}} \boldsymbol{q}_j + \hat{\boldsymbol{t}}) \right\| + \min_j \lambda \left\| (\boldsymbol{r}_1 (\boldsymbol{r}_2 \boldsymbol{q}_i + \boldsymbol{t}_2) + \boldsymbol{t}_1) - (\hat{\boldsymbol{r}} \boldsymbol{q}_j + \hat{\boldsymbol{t}}) \right\|. \quad (2)$$

The balanced parameter λ is empirically set as 1.

Training and Testing Strategies We use ADAM to optimize the network, with an initial learning rate of 0.01; the learning rate is halved every $1,500$ iterations, until a total of $30,000$ ones. We firstly compute the geometric center $\boldsymbol{t}_0 = \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{p}_i$ of the observed point cloud $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^{M}$, and center the point cloud by subtracting each individual point $\boldsymbol{p}_i$ from $\boldsymbol{t}_0$. The input data is then voxelized into

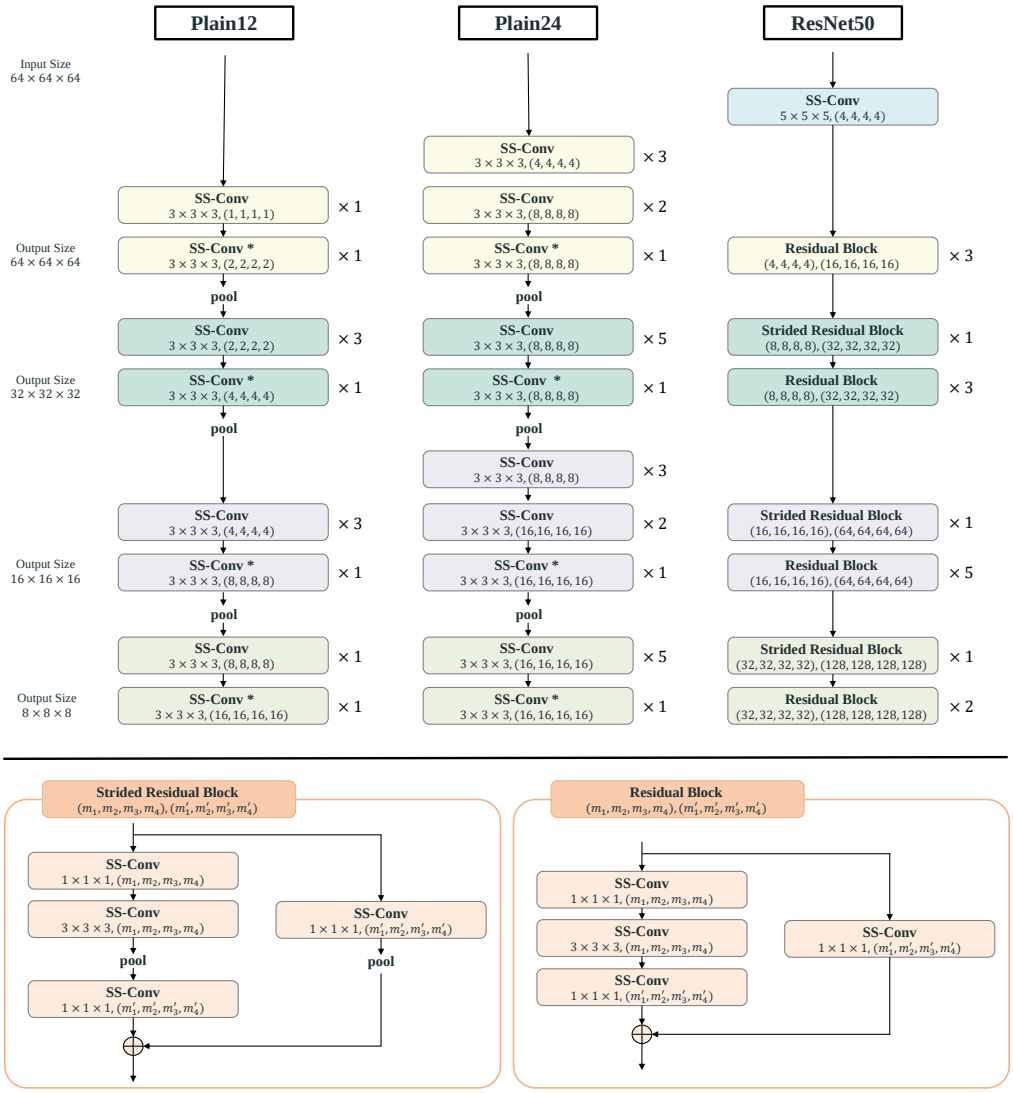

Figure 2: Illustrations of different backbones. "**SS-Conv**" denotes a submanifold sparse steerable convolution, while "**SS-Conv ***" denotes a general one. For each SS-Conv, the number of output channels is $\sum_{i=0}^{3} m_i(2i+1)$, defined by a tuple (m_0, m_1, m_2, m_3), where m_i denotes the number of irreducible features of order i; we only use irreducible features whose orders are smaller than 4 in the three backbones. The strides and paddings of $3 \times 3 \times 3$ SS-Convs are all set as 1, while those of the first $5 \times 5 \times 5$ SS-Conv in ResNet50 are 1 and 2, respectively. We use sparse average pooling to reduce the spatial size of features; the kernel sizes, strides, paddings of the pooling layers are set as $3 \times 3 \times 3$, 2 and 1, respectively.

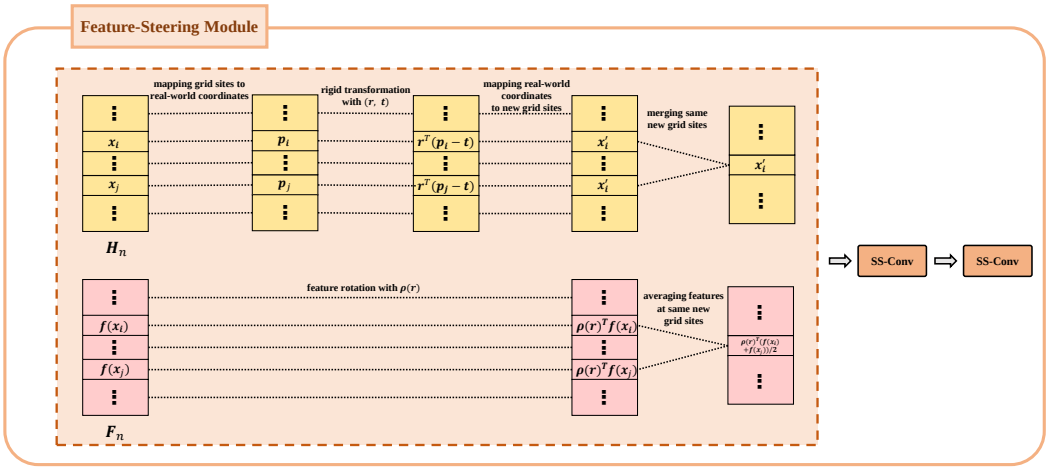

Figure 3: An illustration of Feature-Steering module introduced in Sec. 4.1.1.

Table 1: Robust evaluation of our method on LineMOD dataset [3] for instance-level 6D object pose estimation. The experiments are repeatedly conducted 5 times, and the evaluation metric is ADD(S).

Trial	1	2	3	4	5	MEAN	STD
ape	97.42	97.80	97.71	97.42	98.18	97.71	0.080
bench.	99.32	99.03	99.51	99.03	99.12	99.20	0.036
camera	99.50	99.80	99.60	99.50	99.70	99.62	0.013
can	99.60	99.31	99.70	99.21	99.60	99.48	0.036
cat	99.80	99.80	99.50	99.70	99.60	99.68	0.014
driller	99.60	99.50	99.00	99.30	99.50	99.38	0.045
duck	97.84	97.46	97.46	97.18	97.46	97.48	0.044
egg.	99.90	99.90	99.90	99.81	100.0	99.90	0.004
glue	99.60	99.61	99.80	99.51	99.42	99.59	0.016
hole.	99.40	99.23	99.71	99.14	99.04	99.31	0.055
iron	99.20	98.87	99.48	99.79	99.79	99.43	0.126
lamp	99.70	99.80	99.52	99.71	99.80	99.71	0.011
phone	99.23	99.13	98.84	99.13	99.23	99.11	0.020
MEAN	99.23	99.17	99.20	99.10	99.26	99.19	0.003

a cube with $64 \times 64 \times 64$ grids, covering a total area of $0.32 \times 0.32 \times 0.32\text{m}^3$. The training batch size is set as 64. For testing, we use the predicted masks provided in [10] to segment out the objects of interest for fair comparisons, and repeat the second stage twice for a refined pose; the predicted translation adding the geometric center $\boldsymbol{t}_0$ gives the final translation.

Error Bars The quantitative results of repeated experiments on LineMOD dataset are reported in Table 1. As shown in the table, the results of all 5 trials are of similar qualities across different object instances, showing the stability of our method. We report the results of the first trial in our paper.

Visualization In Fig. 4, we visualize the qualitative results of our proposed method for instance-level 6D pose estimation on LineMOD dataset [3].

B.2 Category-level 6D Object Pose and Size Estimation

Network Architecture As introduced in Sec. 4.2, the network configurations of the category-level pose estimation are the same as those of the instance-level one, except for two adaptive modifications. Firstly, in each stage, we add two regression branches for 3D sizes $\boldsymbol{s} \in \mathbb{R}^3$ and a transformed point cloud $\mathcal{Q}$ in the canonical space of the observed $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^{M}$, as illustrated in Fig. 5, where the specifics of the MLPs for regression are also given; for the 3D size $\boldsymbol{s}$, we regress the object scale

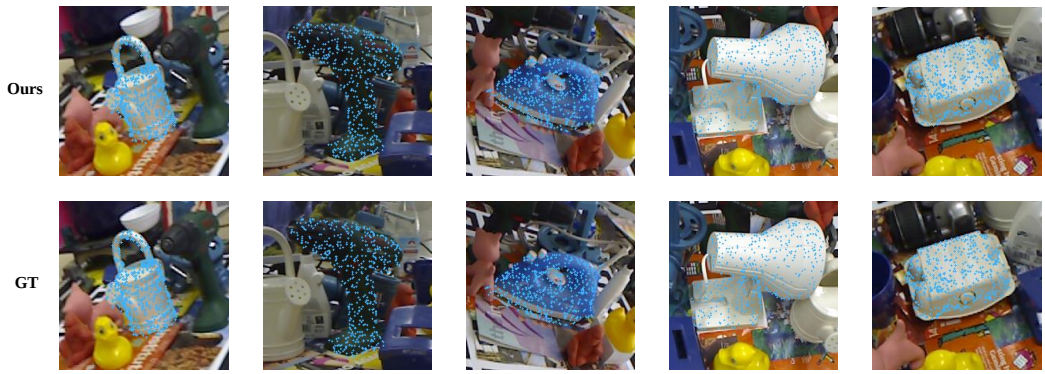

Figure 4: Qualitative results for instance-level 6D pose estimation on LineMOD dataset [3]. The sampled points (in blue) of object models are transformed by the predicted/ground truth poses and projected back to 2D images.

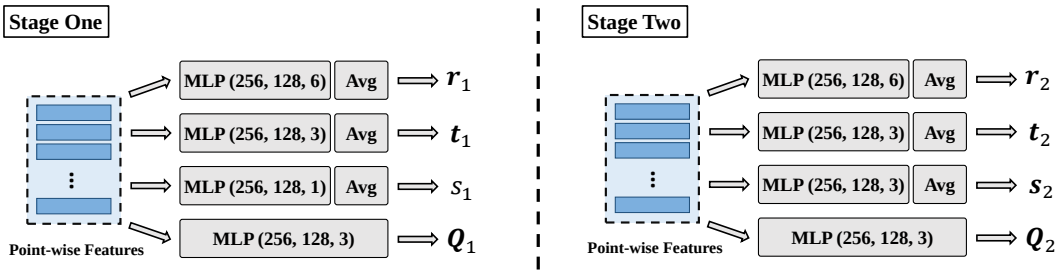

Figure 5: Illustrations of the regression branches in both two stages for category-level 6D object pose and size estimation.

$s = \|\boldsymbol{s}\|$ in the first stage, and regress the aspect ratio $\boldsymbol{s}/\|\boldsymbol{s}\|$ in the second one. Secondly, in each Feature-Steering module, the real world coordinates are also scaled with the predicted s by dividing it in the rigid transformation.

Training Objective Given the ground truth $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{s}})$, we optimize the following problem:

$$\mathcal{L} = (\|\boldsymbol{r}_1 - \hat{\boldsymbol{r}}\| + \left\|\boldsymbol{t}_1 - \hat{\boldsymbol{t}}\right\| + |s_1 - \|\hat{\boldsymbol{s}}\|\,| + \frac{1}{M}\sum_{i=1}^{M}\left\|\boldsymbol{q}_{1i} - \frac{1}{\|\hat{\boldsymbol{s}}\|}\hat{\boldsymbol{r}}^T(\boldsymbol{p}_i - \hat{\boldsymbol{t}})\right\|)$$
$$+ \lambda(\|\boldsymbol{r}_1\boldsymbol{r}_2 - \hat{\boldsymbol{r}}\| + \left\|s_1\boldsymbol{r}_1\boldsymbol{t}_2 + \boldsymbol{t}_1 - \hat{\boldsymbol{t}}\right\| + \|s_1\boldsymbol{s}_2 - \hat{\boldsymbol{s}}\| + \frac{1}{M}\sum_{i=1}^{M}\left\|\boldsymbol{q}_{2i} - \frac{1}{\|\hat{\boldsymbol{s}}\|}\hat{\boldsymbol{r}}^T(\boldsymbol{p}_i - \hat{\boldsymbol{t}})\right\|), \quad (3)$$

where $\mathcal{Q}_1 = \{\boldsymbol{q}_{1i}\}_{i=1}^{M}$ and $\mathcal{Q}_2 = \{\boldsymbol{q}_{2i}\}_{i=1}^{M}$ are the predicted canonical point clouds in the first and second stages, respectively. We empirically set the balanced parameters as $\lambda = 1$.

Note that for symmetric objects along y-axis, we map $\hat{\boldsymbol{r}}$ to a canonical rotation $\hat{\boldsymbol{r}}\boldsymbol{\theta}$ following [6, 4], where

$$\boldsymbol{\theta} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \quad \text{s.t. } \theta = \arctan 2\,(\hat{r}_{13} - \hat{r}_{31}, \hat{r}_{11} + \hat{r}_{33})\,. \quad (4)$$

$\hat{r}_{11}$, $\hat{r}_{13}$, $\hat{r}_{31}$, and $\hat{r}_{33}$ are the elements of $\hat{\boldsymbol{r}}$.

Training and Testing Strategies We use ADAM to train the network with a total of $100,000$ iterations. The learning rate is initialized as 0.01 and halved every $10,000$ iterations. A MaskRCNN [2] is employed to segment out the objects of interest as the inputs, each of which is centered and

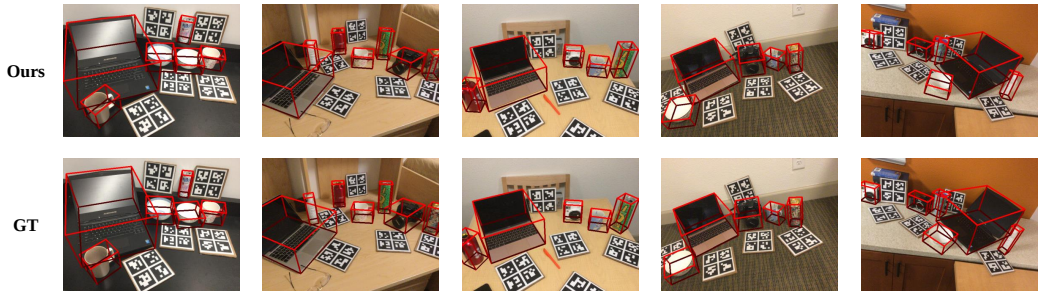

Figure 6: Qualitative results on REAL275 dataset [9] for category-level 6D pose and size estimation.

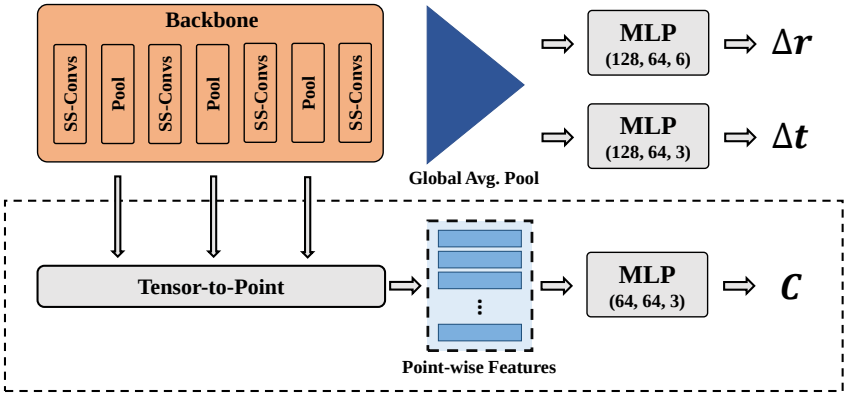

Figure 7: An illustration of the architecture for category-level 6D object pose tracking.

voxelized into $64 \times 64 \times 64$ grids; the volume of these grids is $0.96 \times 0.96 \times 0.96\text{m}^3$. The other training and testing settings are the same as those of instance-level task on LineMOD dataset.

Visualization The qualitative results on REAL275 dataset [9] are visualized in Fig. 6.

B.3 Category-level 6D Object Pose Tracking

Problem Definition Given a sequence of RGB-D images $(\boldsymbol{I}_0, \boldsymbol{I}_1, ...)$ containing the object of interest, along with the initial 6D object pose $(\boldsymbol{r}_0, \boldsymbol{t}_0)$ in $\boldsymbol{I}_0$, the target of pose tracking is to estimate the small change $(\Delta\boldsymbol{r}_i, \Delta\boldsymbol{t}_i)$ of poses in every adjacent frames $(\boldsymbol{I}_{i-1}, \boldsymbol{I}_i)$, such that the object pose in $\boldsymbol{I}_i$ could be obtained based on that in $\boldsymbol{I}_{i-1}$: $(\boldsymbol{r}_i, \boldsymbol{t}_i) = (\boldsymbol{r}_{i-1}\Delta\boldsymbol{r}_i, \boldsymbol{r}_{i-1}\Delta\boldsymbol{t}_i + \boldsymbol{t}_{i-1})$.

Data Processing For an RGB-D image $\boldsymbol{I}_i$, we firstly convert the depth image into a point cloud, which is then transformed with $(\boldsymbol{r}_{i-1}, \boldsymbol{t}_{i-1})$ and scaled by dividing the object scale $\|\boldsymbol{s}\|$. The points within a bounding box, which is centered at the origin with the size of $1.5\boldsymbol{s}$, are cropped out as the input point cloud, denoted as $\mathcal{P} = \{\boldsymbol{p}_j\}_{j=1}^{M}$; each input point $\boldsymbol{p}_j$ is equipped with the point coordinate and RGB values. The input data is voxelized into $64 \times 64 \times 64$ grids in our experiments.

Network Architecture The network architecture is illustrated in Fig. 7. Specifically, we use Plain24 as backbone to extract SE(3)-equivariant features, which are aggregated by a global average pool and fed into two separate MLPs for regression of $\Delta\boldsymbol{r}$ and $\Delta\boldsymbol{t}$, respectively. Following [1], we also employ an auxiliary network for point-wise supervisions, visualized within a black dashed box in Fig. 7. In the auxiliary network, we interpolate the point-wise features of the observed $\mathcal{P}$ based on the output features of three pooling layers in Plain24, and use an MLP and a sigmoid function to obtain $\boldsymbol{C} = \{c_j\}_{j=1}^{M}$ for point-wise binary segmentation, where $\boldsymbol{c}_j$ denotes the foreground probability of $\boldsymbol{p}_j$. The auxiliary network could be detached after training.

Training Objective Given the ground truths of $(\Delta\hat{\boldsymbol{r}}, \Delta\hat{\boldsymbol{t}})$ and $\hat{\boldsymbol{C}} = \{\hat{c}_j\}_{j=1}^{M}$, we optimize the following problem:

$$\mathcal{L} = \lambda_1 \left\| \Delta\boldsymbol{r} - \Delta\hat{\boldsymbol{r}} \right\| + \lambda_2 \left\| \Delta\boldsymbol{t} - \Delta\hat{\boldsymbol{t}} \right\| + \lambda_3 \mathcal{L}_{seg}, \tag{5}$$

where we set the balanced parameters $\lambda_1, \lambda_2, \lambda_3$ as $10, 1, 2$, respectively. We use a focal loss [5] for the auxiliary network, which can be formulated as follows:

$$\mathcal{L}_{seg} = \frac{1}{M_{pos}} \sum_{j=1}^{M} -\alpha \hat{s}_j (1 - s_j)^{\gamma} log(s_j) - \alpha (1 - \hat{s}_j) s_j^{\gamma} log(1 - s_j), \tag{6}$$

where we empirically set $\alpha = 0.25$ and $\gamma = 2$.

Training and Testing Strategies Following [7], we train networks individually for different categories. The networks are optimized using ADAM for a total of $30,000$ iterations. The learning rate is initially set as 0.001 and halved after $20,000$ iterations, while the training batch size is set as 32. During test, the auxiliary network is detached for efficiency.

Visualization We show the qualitative results of our method in the video, named "SS-Conv-track.mp4".

References

[1] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020.

[2] 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.

[3] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *2011 international conference on computer vision*, pages 858–865. IEEE, 2011.

[4] 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.

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[6] 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.

[7] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke 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.

[8] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019.

[9] 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 Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[10] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[11] 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.