RNNPose: Recurrent 6-DoF Object Pose Refinement with Robust Correspondence Field Estimation and Pose Optimization — Supplementary Material

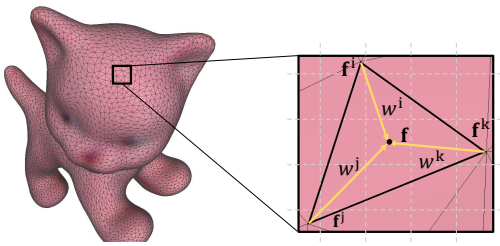

Figure S-1. Demonstration of the feature interpolation during 3D feature rendering.

1. Implementation Details

1.1. 3D Feature Rendering

To better encode the geometric information and improve the efficiency, we applied 3D networks in the 3D context feature encoder as well as in the 3D-2D hybrid network (for 3D descriptor learning). To adapt to the 2D features from image modality, we adopt a differentiable rendering technique to render these learned high-dimensional features/descriptors to 2D maps. Fig. S-1 demonstrates the feature rasterization procedure. We first find the triangle vertex indices in the 3D mesh for each spatial location of the 2D feature map using a modern rasterization pipeline. Then, the feature vector $\mathbf{f}$ being rendered to a 2D location is interpolated from the encoded 3D features of the indices found above based on the normalized weights w^i, w^j, w^k defined in the barycentric coordinate system, *i.e.*, $\mathbf{f} = w^i\mathbf{f}^i + w^j\mathbf{f}^j + w^k\mathbf{f}^k$ with $w^i + w^j + w^k = 1$.

1.2. Details About the Mechanism of GRU

We concatenate the local correlation volume, the rectified correspondence field and the previously encoded context feature map $\mathbf{F}_{ctx}$ as a data volume, denoted as $\mathbf{x}_t$, and input it to a GRU for correspondence field estimation. We adopt a GRU architecture same as [6], which controls the hidden state update in the following manner:

$$
\begin{aligned}
z_t &= sigmoid(K_{3\times3}([h_{t-1}, x_t]; W_z)) \\
r_t &= sigmoid(K_{3\times3}([h_{t-1}, x_t]; W_r)) \\
\bar{h}_t &= \tanh(K_{3\times3}([r_t \odot h_{t-1}, x_t]; W_h)) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \bar{h}_t
\end{aligned} \tag{1}
$$

where z_t is the update gate vector, r_t is the reset gate vector and h_t is the updated hidden state. Here, $K_{3\times3}(\cdot; W)$ denotes a 3×3 convolution kernel and $\odot$ denotes the Hadamard product. The correspondence field $\hat{\mathbf{C}}_t$ is then estimated from the hidden state h_t with two convolution layers.

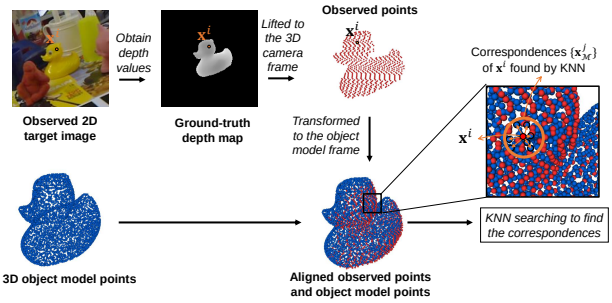

Figure S-2. Details about how to identify the correspondences in the 3D model for an observed 2D image point $\mathbf{x}^i$ for 3D-2D descriptor learning. The identified correspondences are used in the descriptor loss L_d to construct the positive set $\{\ \}_+$ for contrastive leaning. Note that, the ground-truth depth map is rendered according to the ground-truth object pose, and this correspondence identification process is conducted **only during training**.

1.3. Construction of the Positive Set and Negative Set for Descriptor Loss

To learn distinctive 3D and 2D descriptors for the similarity score modeling of Eq. (3) , we have proposed a descriptor loss denoted as L_d (Eq. (8)). To measure the first loss term $\mathcal{L}_{cir}(\boldsymbol{d}_{\mathcal{I}}^i, \{\boldsymbol{d}_{\mathcal{M}}^j\}_+, \{\boldsymbol{d}_{\mathcal{M}}^k\}_-)$ in L_d, for each foreground point $\mathbf{x}^i$ (with descriptor $\boldsymbol{d}_{\mathcal{I}}^i$) in the target image, we need to find a set of its correspondences $\{\mathbf{x}_{\mathcal{M}}^j\}$ (with associated descriptors $\{\boldsymbol{d}_{\mathcal{M}}^j\}_+$) in the model.

As illustrated in Fig. S-2, to this end, we first lift the 2D image point $\mathbf{x}^i$ to the 3D camera frame according to its associated depth value and the camera intrinsics, and then the lifted points are transformed to the same frame of the object model for KNN searching. The KNN searching is based on a KD-tree for efficient training, and the K corresponding model descriptors $\{\boldsymbol{d}^j_{\mathcal{M}}\}_+$ are obtained according to the KNN searching result. The descriptors of the remaining non-corresponding model points constitute the negative set $\{\boldsymbol{d}^k_{\mathcal{M}}\}_-$ in the contrastive loss.

1.4. Derivation of Jacobians in LM

The objective function of the pose optimization is $E(\boldsymbol{\xi}) = \sum_{i=1}^{M}(\hat{\mathbf{C}}_t(\mathbf{x}^i) - \mathbf{C}(\mathbf{x}^i;\boldsymbol{\xi}))^T \mathbf{w}^i (\hat{\mathbf{C}}_t(\mathbf{x}^i) - \mathbf{C}(\mathbf{x}^i;\boldsymbol{\xi}))$ (Eq. (4)), where $\mathbf{w}^i = \begin{pmatrix} w^i & 0 \\ 0 & w^i \end{pmatrix}$. Here, the correspondence derived with residual pose parameter argument $\boldsymbol{\xi}$ is expressed as

$$\mathbf{C}(\mathbf{x}^i;\boldsymbol{\xi}) = \pi(\exp(\boldsymbol{\xi})\pi^{-1}(\mathbf{x}^i, z^i)), \tag{2}$$

where

$$\begin{aligned} \pi([X,Y,Z,W]^T) &= \begin{pmatrix} f_x \cdot X/Z + c_y \\ f_y \cdot Y/Z + c_y \end{pmatrix}, \\ \pi^{-1}([x,y]^T, z) &= \begin{pmatrix} (x - c_x)/f_x \\ (y - c_y)/f_y \\ z \\ 1 \end{pmatrix} \end{aligned} \tag{3}$$

We denote each residual term in the summation (Eq. (5)) as $\mathbf{r}^i = [r^i_x, r^i_y]^T = \hat{\mathbf{C}}_t(\mathbf{x}^i) - \mathbf{C}(\mathbf{x}^i;\boldsymbol{\xi})$ and its derivative w.r.t. pose argument $\boldsymbol{\xi}$ is derived as

$$\begin{aligned} \frac{\partial \mathbf{r}^i}{\partial \boldsymbol{\xi}^i} &= -\frac{\partial \mathbf{C}(\mathbf{x}^i;\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \\ &= -\frac{\partial \pi(\mathbf{X})}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \text{ (letting } \mathbf{X} = \pi(\exp(\boldsymbol{\xi})\pi^{-1}(\mathbf{x}^i, z^i))), \end{aligned} \tag{4}$$

where

$$\frac{\partial \pi(\mathbf{X})}{\partial \mathbf{X}} = \begin{pmatrix} f_x/Z & 0 & -f_x \cdot X/Z^2 & 0 \\ 0 & f_y/Z & -f_y \cdot Y/Z^2 & 0 \end{pmatrix}, \tag{5}$$

and

$$\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} = \begin{pmatrix} 1 & 0 & 0 & 0 & Z' & -Y' \\ 0 & 1 & 0 & -Z' & 0 & X' \\ 0 & 0 & 1 & Y' & -X' & 0 \end{pmatrix}. \tag{6}$$

Here, X, Y, Z are the components of $\mathbf{X}$, *i.e.*, $\mathbf{X} = [X,Y,Z]^T$, and X', Y', Z' denote the components of $\pi^{-1}(\mathbf{x}^i, z^i)$. For the back propagation, we follow [7] for numeric stability.

1.5. Network Architecture

The specific network architectures is provided in Fig. S-3. Note that, for the 3D descriptor net and the 3D context encoder, we adopt the layers and operations from KPConv [8].

2. Experimental Details

We train all of our networks end-to-end on a Tesla V100 GPU with a batch size of 1 for 150k iterations, using the Adam [3] optimizer with an initial learning rate of 10^{-4} and gradually decrease it with a cosine annealing strategy (of a half cosine cycle). The weights of the correspondence loss, the model alignment loss L_{ma}, and the descriptor loss L_d are set to 0.5, 1 and 1 respectively. All our models are trained agnostic to the initial pose sources where disturbed ground-truth poses with Gaussian noise are taken as initial poses for training following [4]. The trained model is directly tested with initial poses provided by different object pose estimation methods without further fine-tuning. Specifically, the translational variance and rotational variance of the Gaussian noise are consistently set to $\sigma_t = 0.05$ cm and $\sigma_r = 15°$ respectively following [4] for all the experiments for fair comparison. Note that $\sigma_t = 0.05$ denotes the variance along the z direction, while for the x and y directions, we decrease the variances by $\frac{1}{5}\times$, considering the conventional methods usually have larger variances in depth estimation.

For the LINEMOE dataset, besides the real data, we add more synthetic data for training with the same configurations as in [4]. For the Occlusion LINEMOE, we further create occlusions during training by cropping and pasting the foreground patches randomly following [5]. For the YCB-Video dataset, we use the provided data from the BOP benchmark [1]. We also apply data augmentations including random lighting and color jittering as in [4] during training.

For efficiency, we crop the input target image to size 320×320 with a cropping window centered at the projected object center based on the initial pose. This cropped image is sent to the 2D net of the 3D-2D hybrid network for 2D descriptor learning. For the correspondence field estimation and pose refinement, the input image is further zoomed in and cropped to size 240×240 according to the rendered image (according to the initial pose) for efficiency. The zoom-in and cropping procedure are similar to [4] but they adopt larger zoom-in size (*i.e.*, 640×480). Larger cropping window should produce more robust performance against large initial pose errors, because if the initial pose estimation error is significant, the target could be outside of the cropping window, whose pose by no means could be recovered.

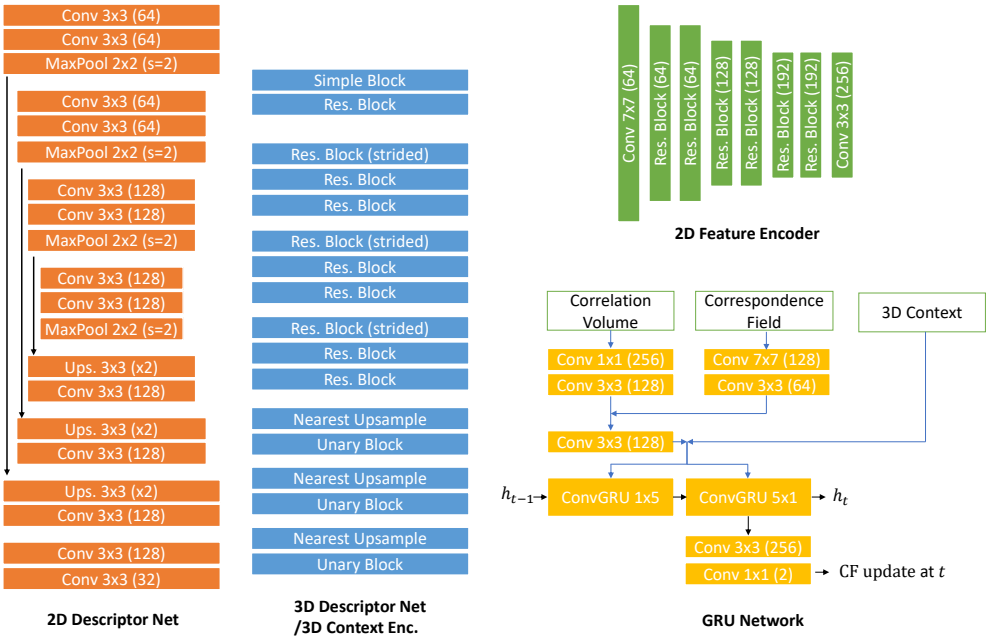

Figure S-3. Detailed architectures of the networks.

3. More Experimental Results

3.1. Learning Bandwidth Parameter σ of the Similarity score.

Fig. S-6 plots the variations of learnable bandwidth parameter σ of the similarity score $w^i = \exp\left(-\frac{|1-\mathbf{d}_{\mathcal{M}}^{i^T}\mathbf{d}_{\mathcal{I}}^i|}{\sigma}\right)$ (Eq. (3)) during training. It can be found that learnable parameter σ (initialized with 1) first decreases and stably stays low for all the sequences. A smaller σ value produces a similarity score function of a sharper shape. A sharper (strict) similarity measurement downweights the unreliable correspondences more effectively, because only the correspondence pairs with very similar descriptors could be recognized as reliable during pose optimization. Note that, we only supervise the final pose results estimated from the weighted correspondences (weighted by the similarity scores w^i), and this parameter is adjustment by the network automatically to pursue more reliable regions for more accurate pose estimations during training.

3.2. Qualitative Analysis

Fig. S-4 exhibits the estimated correspondence fields (before rectification), the estimated object poses, and the rectified correspondence fields according to the estimated poses, from different recurrent iterations of a single rendering cycle. To give a better view on the correspondence estimation quality, we also visualize the warped observed images, *i.e.*, obtained by warping the observed images to the rendered frame based on the estimated correspondence field. Generally, if the correspondence estimation is better, the warped image is more vivid and more similar to the rendered image in appearance. It can be found that the rectified correspondence field (conforming to rigid-transformation constraints) is less noisy and produces more vivid warped image compared with the non-rectified counterpart. It can also be found that the correspondence field estimations and the object pose estimations (visualized as red boxes) are gradually improved as the recurrent refinement going on.

Fig. S-5 includes more visualization results of the pose estimation in severely occluded cases with erroneous pose initializations. Our methods consistently exhibits strong robustness against these challenging cases.

3.3. Detailed Limitation Discussion

Although our method achieves robust performance against erroneous pose initializations and occlusions, our system is still limited to known objects with CAD models similar to many previous works [2, 4, 10]. We evaluate our model trained on the YCB-Video dataset on another dataset LINEMOD directly. The generalization ability of our model is still found limited currently and the average ADD-(S) score on LINEMOD is 71.49% as reported by the 2nd row of Table S-1. Though the accuracies are still improved compared with the provided initial poses (accuracy 63.26%), the performance is inferior to that of the dedicated model trained on LINEMOD (accuracy 97.37% reported in Table 1a in our paper). But we find that finetuning the pose

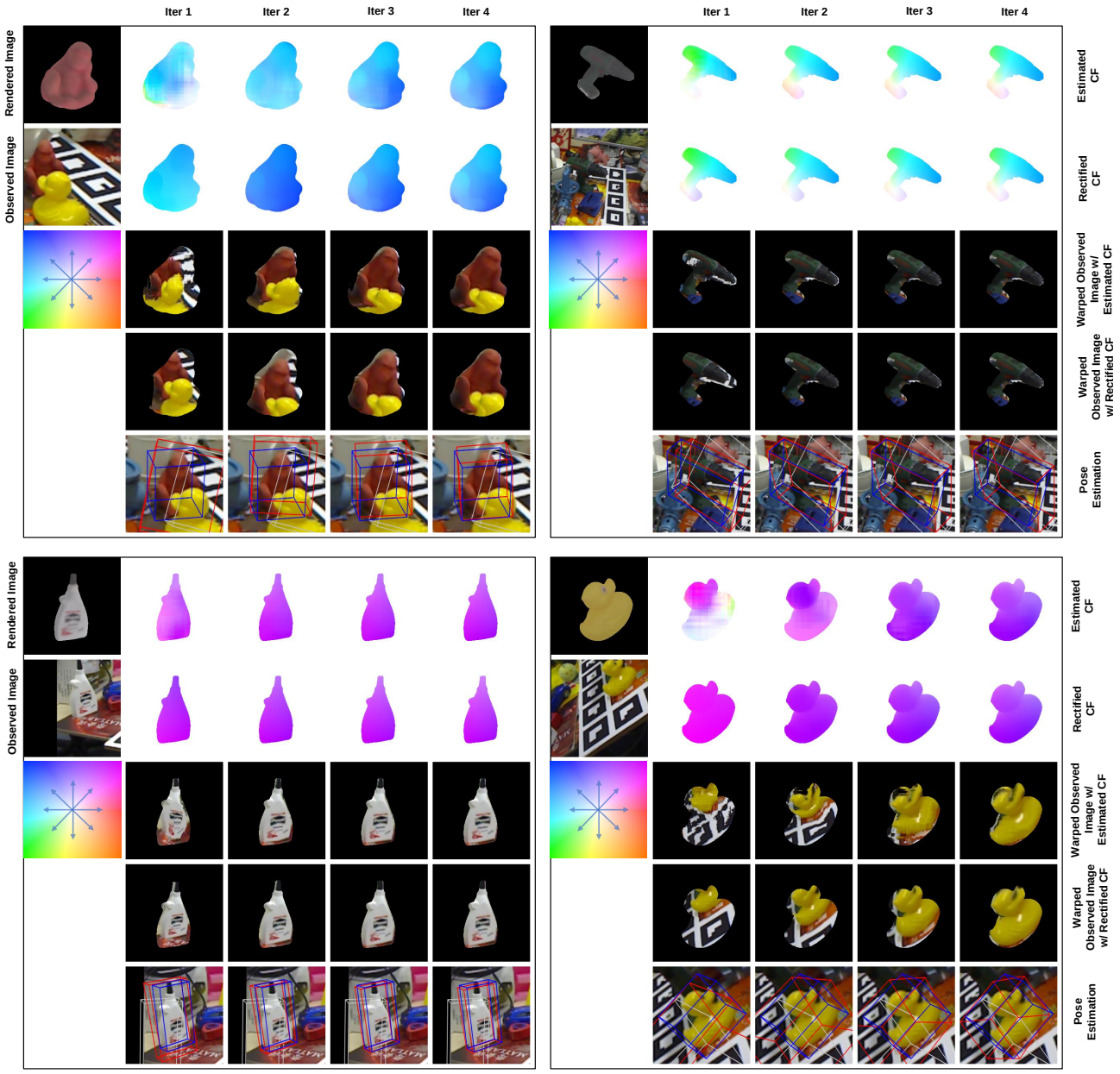

Figure S-4. Visualizations of the estimated correspondence fields (CFs), rectified CFs, warped observed image with the CFs and the pose estimation of each recurrent iteration (iter1-iter4) in the first rendering cycle. For pose visualization, the white boxes represent the input erroneous initial poses, the red boxes are estimated by our algorithm, while the ground-truth boxes are in blue.

Table S-1. Performance evaluation on unseen objects. ADD-(S) metric (%) is used for evaluation and the initial poses are provided by PoseCNN [9]. The 2nd row lists the performances on LINEMOD dataset when directly evaluating a model trained on a totally different dataset YCB-Video. The 3rd row shows the improved performances after only finetuning the pose refinement module.

	ape	benchvise	camera	can	cat	driller	duck	eggbox	glue	holepuncher	iron	lamp	phone	Avg.
Initial Pose	25.62	56.09	41.78	52.24	69.98	47.25	64.92	53.07	94.98	77.11	70.73	98.50	70.17	63.26
Before Pose Refinement Module Finetuning	33.24	70.90	80.59	87.40	68.96	84.34	45.92	94.74	86.68	24.93	96.83	83.69	71.20	71.49
After Pose Refinement Module Finetuning	80.38	99.90	99.02	99.21	97.80	99.90	88.08	99.91	99.61	98.00	99.90	99.90	99.06	96.98

refinement module, including the GRU and descriptor generation networks (with the image feature encoding and 3D context feature encoding modules frozen), on these unseen objects is able to improve the performance significantly. As

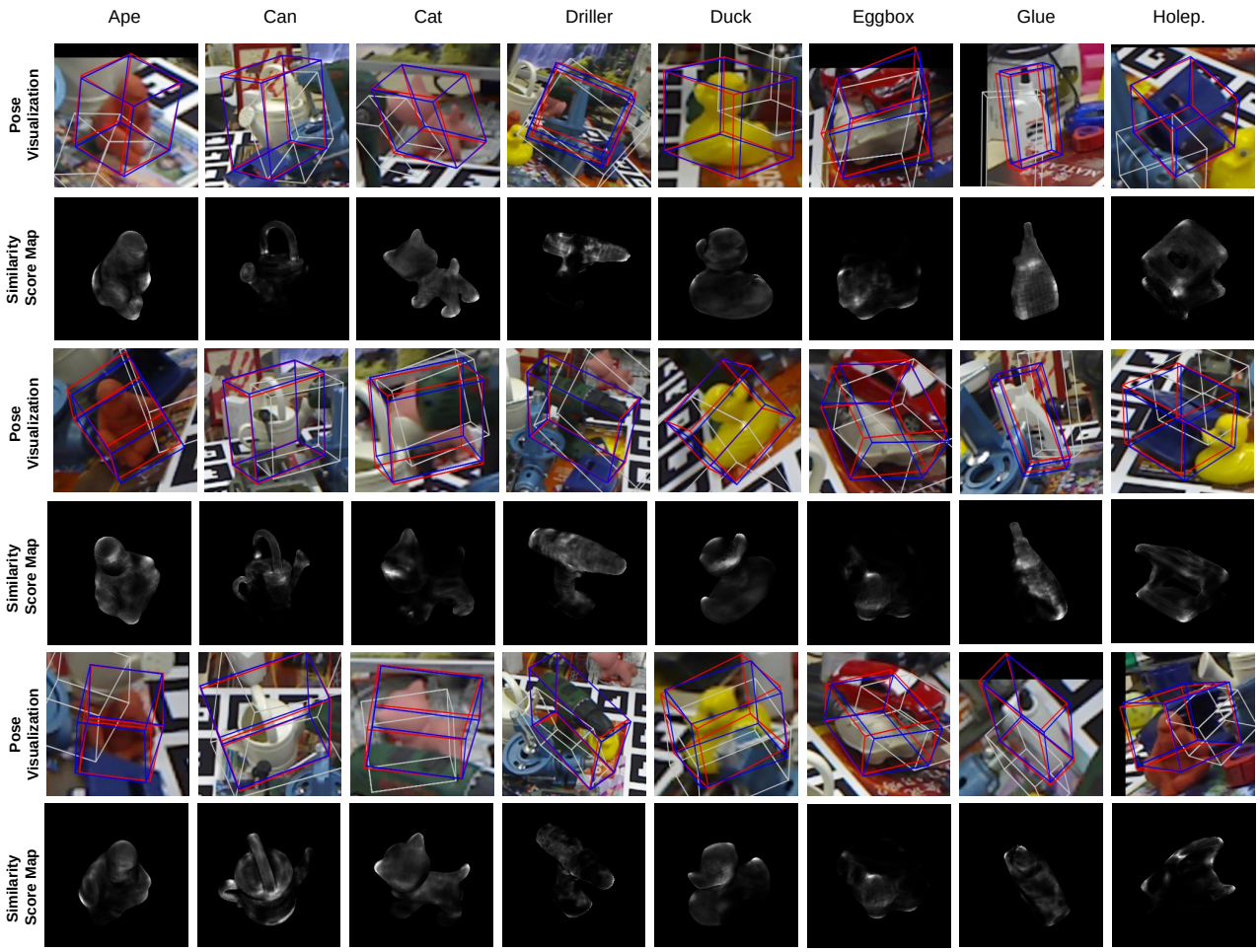

Figure S-5. More Visualizations of estimated poses and similarity score maps on Occlusion LINEMOE dataset. For pose visualization, the white boxes represent the erroneous initial poses, the red boxes are estimated by our algorithm and the ground-truth boxes are in blue. Here, the initial poses for pose refinement are originally from PVNet [5] but added with significant disturbances for robustness testing (adding Gaussian noise $\sigma_t = 0.09$ cm, $\sigma_r = 10^\circ$). For the similarity score map, brighter color represents a higher similarity.

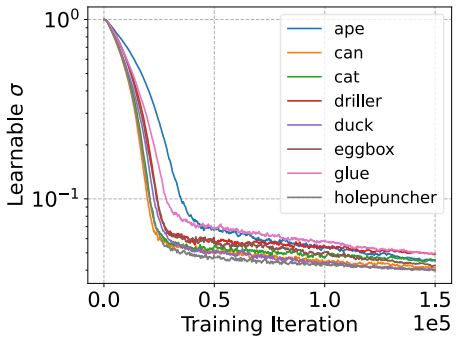

Figure S-6. The learning process of the parameter σ in the similarity score Eq. (3).

shown in Table S-1, the average accuracy improves from 71.49% to 96.97% which is comparable to training from the scratch (97.37%)

References

[1] Tomáš Hodaň, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiří Matas, and Carsten Rother. BOP: Benchmark for 6D object pose estimation. *European Conference on Computer Vision (ECCV)*, 2018. 2

[2] Shun Iwase, Xingyu Liu, Rawal Khirodkar, Rio Yokota, and Kris M Kitani. Repose: Real-time iterative rendering and refinement for 6d object pose estimation. *arXiv preprint arXiv:2104.00633*, 2021. 3

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[4] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vi-*

sion (ECCV), pages 683–698, 2018. 2, 3

[5] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019. 2, 5

[6] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *16th European Conference on Computer Vision, ECCV 2020*, pages 402–419. Springer Science and Business Media Deutschland GmbH, 2020. 1

[7] Zachary Teed and Jia Deng. Tangent space backpropagation for 3d transformation groups. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10338–10347, 2021. 2

[8] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019. 2

[9] 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. 4

[10] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1941–1950, 2019. 3