RePOSE: Fast 6D Object Pose Refinement via Deep Texture Rendering Supplementary Material

Shun Iwase[1] Xingyu Liu[1] Rawal Khirodkar[1] Rio Yokota[2] Kris M. Kitani[1]
[1]Carnegie Mellon University [2]Tokyo Institute of Technology

1. Implementation Details

RePOSE uses the initial estimated pose of PVNet on the LineMOD and Occlusion LineMOD dataset, and of PoseCNN on the YCB-Video dataset. PoseCNN has VGG-16 [7] as the backbone network. However, since RePOSE reuses the deep feature from PoseCNN, its slow runtime of VGG-16 can be problematic in the case of tracking. Instead, we use ResNet-18 as the backbone network of PoseCNN. RePOSE makes use of the deep feature from ResNet-18, however, we still rely on the initial pose of the original PoseCNN with VGG-16. For tracking, the U-Net decoder of RePOSE also outputs the segmentation mask and use it as an additional signal to detect whether an object is lost or not. RePOSE learns its parameters separately per object on the LineMOD and Occlusion LineMOD dataset. In contrast to that, RePOSE learns the parameters for all the objects simultaneously on the YCB-Video dataset. We use `pvnet-rendering`[1] to generate synthetic images. However, we may be able to improve accuracy if images generated by photorealistic rendering are used for training. The neural textures, which are the outputs of the MLP that takes learnable parameters as input (ref. Fig.1 of the main paper), are trained along with the MLP and input learnable parameters.

2. Derivation of Camera Jacobian Matrix

Notation: $\mathbf{R} \in \mathrm{SO}(3)$ denotes a rotation matrix, $\mathbf{t} \in \mathbb{R}^3$ denotes a translation vector, $\mathbf{w} \in \mathfrak{so}(3)$ denotes a rotation vector, $\mathbf{P} = \begin{pmatrix}\mathbf{w} & \mathbf{t}\end{pmatrix} \in \mathbb{R}^6$ is a pose representation using a rotation and translation vector, f_x and f_y are focal lengths, p_x and p_y are the principal point offsets, $\mathbf{X}$ and $\mathbf{x}$ denote a homogeneous and inhomogeneous 2D image coordinate, subscript w, c denotes a coordinate is defined in the world, and camera coordinate system respectively, and $\mathbf{Id} \in \mathbb{R}^{3\times3}$ is an 3×3 identity matrix. $\mathbf{R}$ and $\mathbf{w}$ represents the same rotation.

[1]https://github.com/zju3dv/pvnet-rendering

Derivation: Jacobian matrix $\mathbf{J}$ of the objective function with respect to a pose $\mathbf{P}$ is required to perform deep feature-based pose optimization. We show the detailed derivation of a camera jacobian matrix $\frac{\partial \mathbf{x}}{\partial \mathbf{P}}$ at each pixel in Equation 9. The camera jacobian matrix can be decomposed more as follows;

$$\frac{\partial \mathbf{x}}{\partial \mathbf{P}} = \begin{pmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{w}} \frac{\partial \mathbf{x}}{\partial \mathbf{R}} \\ \frac{\partial \mathbf{x}}{\partial \mathbf{t}} \end{pmatrix} \in \mathbb{R}^{6\times2} \tag{1}$$

Using the derivative calculation method of a rotation matrix with respect to a rotation vector proposed in [2], the following equation is acquired.

$$\begin{aligned}\frac{\partial \mathbf{R}}{\partial \mathbf{w}} &= \begin{pmatrix} \frac{\partial \mathbf{R}}{\partial w_1} & \frac{\partial \mathbf{R}}{\partial w_2} & \frac{\partial \mathbf{R}}{\partial w_3} \end{pmatrix}^T \\ &= \begin{pmatrix} vec\left(\frac{w_1[\mathbf{w}]_\times + [\mathbf{w} \times (\mathbf{Id} - \mathbf{R})\mathbf{e}_1]_\times}{\|\mathbf{w}\|^2} \mathbf{R} \right) \\ vec\left(\frac{w_2[\mathbf{w}]_\times + [\mathbf{w} \times (\mathbf{Id} - \mathbf{R})\mathbf{e}_2]_\times}{\|\mathbf{w}\|^2} \mathbf{R} \right) \\ vec\left(\frac{w_3[\mathbf{w}]_\times + [\mathbf{w} \times (\mathbf{Id} - \mathbf{R})\mathbf{e}_3]_\times}{\|\mathbf{w}\|^2} \mathbf{R} \right) \end{pmatrix} \in \mathbb{R}^{3\times9}\end{aligned} \tag{2}$$

where *vec* is a vectorize operation, $[\mathbf{x}]_\times$ is a conversion from a 3-d vector to a skew-symmetric matrix, $\mathbf{e}_i$ is a ith 3-d basis vector. $\mathbf{R}$ is regarded as a 9-d vector for simplicity. A camera and image coordinate $\mathbf{x}_c$ and $\mathbf{x}$ can be calculated as follows;

$$\mathbf{x}_c = \mathbf{R}\mathbf{x}_w + \mathbf{t} \tag{3}$$

$$\mathbf{x} = \begin{pmatrix} \frac{f_x x_c}{z_c} + p_x \\ \frac{f_y y_c}{z_c} + p_y \end{pmatrix} \tag{4}$$

Table 1: Comparison of the median of absolute angular and relative translation error on the LindMOD dataset [3]. We do not report the rotation error of symmetric objects (eggbox, and glue) because of its non-unique rotation representation.

Object	PVNet [6]		CNN w/ FW		Ours w/ FW		Ours	
	Rotation	Translation	Rotation	Translation	Rotation	Translation	Rotation	Translation
Ape	2.213°	0.119	1.849°	0.069	1.446°	0.055	1.197°	0.051
Benchvise	1.030°	0.022	0.857°	0.022	0.644°	0.014	0.757°	0.010
Cam	1.183°	0.045	0.896°	0.030	1.322°	0.073	0.713°	0.023
Can	0.958°	0.032	1.238°	0.033	0.995°	0.040	0.674°	0.017
Cat	1.260°	0.050	1.177°	0.041	0.941°	0.036	0.857°	0.035
Driller	1.008°	0.029	0.812°	0.023	1.057°	0.060	0.773°	0.014
Duck	1.701°	0.078	1.701°	0.055	1.531°	0.042	1.481°	0.053
Eggbox	-	0.056	-	0.074	-	0.048	-	0.025
Glue	-	0.050	-	0.049	-	0.040	-	0.036
Holepuncher	1.265°	0.052	1.548°	0.058	1.295°	0.060	1.009°	0.029
Iron	1.205°	0.027	1.223°	0.027	0.927°	0.020	0.911°	0.015
Lamp	1.050°	0.029	1.235°	0.042	1.054°	0.019	0.902°	0.020
Phone	1.208°	0.040	1.122°	0.032	0.925°	0.019	0.889°	0.025
Average	1.280°	0.048	1.242°	0.043	1.103°	0.040	0.924°	0.027

Table 2: Comparison of the median of absolute angular and relative translation error on the Occlusion LindMOD dataset [1]. We do not report the rotation error of symmetric objects (eggbox, and glue) because of its non-unique rotation representation.

Object	PVNet [6]		CNN w/ FW		Ours w/ FW		Ours	
	Rotation	Translation	Rotation	Translation	Rotation	Translation	Rotation	Translation
Ape	4.103°	0.210	4.222°	0.185	4.015°	0.171	3.871°	0.157
Can	2.722°	0.068	2.879°	0.069	2.793°	0.067	2.704°	0.043
Cat	6.012°	0.239	6.480°	0.363	6.552°	0.335	5.852°	0.274
Driller	2.774°	0.072	2.567°	0.131	2.762°	0.120	2.545°	0.056
Duck	6.923°	0.156	6.795°	0.115	6.537°	0.108	6.533°	0.102
Eggbox	-	0.325	-	0.295	-	0.284	-	0.318
Glue	-	0.275	-	0.246	-	0.235	-	0.266
Holepuncher	3.969°	0.121	3.917°	0.116	3.918°	0.126	3.629°	0.088
Average	4.417°	0.183	4.477°	0.190	4.430°	0.181	4.189°	0.163

Using Equations (3) and (4), the following derivatives can be obtained.

$$\frac{\partial \mathbf{x}}{\partial \mathbf{R}} = \begin{pmatrix} \frac{f_x x_w}{z_c} & 0 \\ \frac{f_x y_w}{z_c} & 0 \\ \frac{f_x z_w}{z_c} & 0 \\ 0 & \frac{f_y x_w}{z_c} \\ 0 & \frac{f_y y_w}{z_c} \\ 0 & \frac{f_y z_w}{z_c} \\ -\frac{f_x x_c x_w}{z_c^2} & -\frac{f_y y_c x_w}{z_c^2} \\ -\frac{f_x x_c y_w}{z_c^2} & -\frac{f_y y_c y_w}{z_c^2} \\ -\frac{f_x x_c z_w}{z_c^2} & -\frac{f_y y_c z_w}{z_c^2} \end{pmatrix} \in \mathbb{R}^{9\times 2} \qquad (5)$$

$$\frac{\partial \mathbf{x}}{\partial \mathbf{t}} = \begin{pmatrix} \frac{f_x}{z_c} & 0 \\ 0 & \frac{f_y}{z_c} \\ -\frac{f_x x_c}{z_c^2} & -\frac{f_y y_c}{z_c^2} \end{pmatrix} \in \mathbb{R}^{3\times 2} \qquad (6)$$

3. Additional Ablation Study

Ablation on effect of warping. We show the median of absolute angular and relative translation error on the LineMOD and Occlusion LineMOD datasets in Tables 1 and 2. The relative translation error is computed with respect to object diameter. The initial pose error on the LineMOD dataset [3] is relatively small and the methods using warping improve score properly. On the contrary, the initial pose error is large on the Occlusion LineMOD

Table 3: Ablation of the channel size on the LineMOD dataset

Channel Size	1	3	5	7	9
ADD(-S)	95.2	**96.1**	95.7	94.9	94.5

Table 4: Runtime comparison among recent methods. In CNN with feature warping (FW) the initial feature is obtained using CNN and then warp the feature based on an updated pose. We assume the number of iterations is 5 for RePOSE and FW denotes feature warping. The FPS is reported with refinement of 5 objects.

Method	Runtime
DeepIM [5] (1 iters)	45.8 ms
DeepIM [5] (4 iters)	166.8 ms
CosyPose [4] (1 iters)	38.3 ms
CosyPose [4] (2 iters)	77.1 ms
RePOSE w/ different feature extraction methods	
CNN w/ FW	19.6 ms
Ours w/ FW	13.5 ms
Ours	12.4 ms

dataset. In that case, feature warping becomes less effective. Being different from the methods using feature warping, our iterative deep feature rendering method can generate a feature with a complete shape. We believe this characteristics of feature rendering leads to successful reduction of the error of rotation and translation on both datasets.

Ablation on the number of channels in the neural textures We vary the number of channels in the neural textures. We report the results on the LineMOD dataset in Table 3. Note, the results are comparable, however, setting number of channels to 3 results in the best performance.

Runtime Ablation on Feature Warping. DeepIM [5] and CosyPose [4] run a CNN every iteration on a concatenated image of a zoomed input and rendered images to compare these two images and output a pose directly. According to the ablation study by [5], high-resolution zoomed-in is a key and it improves the ADD(-S) score by 23.4. However, as shown in Table 4, extracting image features from zoomed images multiple times leads to a slow runtime. Instead, RePOSE runs a CNN once for an input image with the original resolution. Additionally, an image representation of a rendered image can be extracted within 1ms because of deep texture rendering. This makes the runtime of RePOSE faster than prior methods while keeping a comparable accuracy to the prior methods. We also measure the runtime of RePOSE using different feature extraction methods for a rendered image. As shown in Table 4 in the supplemental and Table 4 and 5 in the main paper, RePOSE with deep texture rendering (Ours) achieves the fastest and highest accuracy among these three variants.

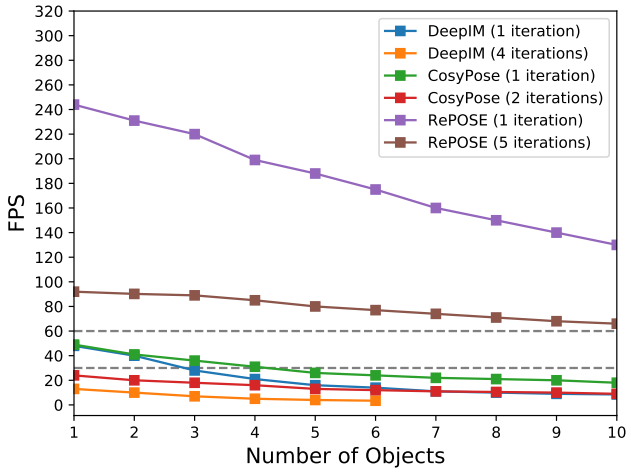

Figure 1: Trade-off between FPS and number of objects of recent methods.

Runtime Ablation on the number of objects We investigate the trade-off between FPS and number of objects. As shown in 1, RePOSE is the only method which runs at over than 60 FPS even with refinement of 10 objects. 6D pose refinement is always performed after initial pose estimation. Thus, it is crucial to make sure it runs at faster than real-time (30 FPS). DeepIM causes an out of memory error when the number of object is more than 6 with a NVIDIA RTX2080 which has 8GB memory.

References

[1] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, 2014. 2

[2] Guillermo Gallego and Anthony Yezzi. A compact formula for the derivative of a 3-d rotation in exponential coordinates. *Journal of Mathematical Imaging and Vision*, 2015. 1

[3] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*, 2012. 2

[4] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *ECCV*, 2020. 3

[5] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *ECCV*, 2018. 3

[6] Sida Peng, Xiaowei Liu, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019. 2

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1