DeepIM: Deep Iterative Matching for 6D Pose Estimation - Supplementary Material

Yi Li[1], Gu Wang[1], Xiangyang Ji[1], Yu Xiang[2], and Dieter Fox[2]

[1] Tsinghua University, BNRist
[2] University of Washington and NVIDIA Research
yili.matrix@gmail.com, wanggl6@mails.tsinghua.edu.cn,
xyji@tsinghua.edu.cn, {yux, dieterf}@nvidia.com

1 Details about Zooming In

Given the reference mask $\mathbf{m}_{\text{rend}}$ and observed mask $\mathbf{m}_{\text{obs}}$, the cropping patch is computed as Eq. 1

$$\begin{aligned}
y_{\text{dist}} &= \max(|u_{\text{obs}} - y_c|, |u_{\text{rend}} - y_c|, \\
&\qquad |d_{\text{obs}} - y_c|, |d_{\text{rend}} - y_c|) \\
x_{\text{dist}} &= \max(|l_{\text{obs}} - x_c|, |l_{\text{rend}} - y_c|, \\
&\qquad |r_{\text{obs}} - y_c|, |r_{\text{rend}} - y_c|) \\
\text{width} &= \max(x_{\text{dist}}, y_{\text{dist}} \cdot r) \cdot 2\lambda \\
\text{height} &= \max(x_{\text{dist}}/r, y_{\text{dist}}) \cdot 2\lambda
\end{aligned} \tag{1}$$

where u_*, d_*, l_*, r_* denotes the upper, lower, left, right bound of foreground mask of observed or rendered images, x_c, y_c represent the 2D projection of the center of the object in $\mathbf{x}_{\text{rend}}$, r represent the aspect ratio of the origin image (width/height), λ denotes the expand ratio, which is fixed to 1.4 in the experiment. Then this patch is bilinear sampled to the size of the original image, which is 480×640 in this paper. By doing so, not only do the object is zoomed in without being distorted, but also provide the network with the information about where the center of the object lies.

2 Details about Training Data

2.1 Reference Images during Training

The reference image $\mathbf{x}_{\text{rend}}$ and mask $\mathbf{m}_{\text{rend}}$ are randomly generated during training without using prior knowledge of the initial poses in the test set. Specifically, given a ground truth pose $\hat{\mathbf{p}}$, we add noises to $\hat{\mathbf{p}}$ to generate the initial poses. For rotation, we independently add a Gaussian noise $\mathcal{N}(0, 15)$ to each of the three Euler angles of the rotation. If the angular distance between the new pose and the ground truth pose is more than 45°, we discard the new pose and generate another one in order to make sure the initial pose for refinement is within 45°

of the ground truth pose during training. For translation, considering the fact that RGB-based pose estimation methods usually have larger variance on depth estimation, the following Gaussian noises are added to the three components of the translation: $\Delta x \sim \mathcal{N}(0, 0.01), \Delta y \sim \mathcal{N}(0, 0.01), \Delta z \sim \mathcal{N}(0, 0.05)$, where the variances are 1 cm, 1 cm and 5 cm, respectively.

2.2 Synthetic Training Data

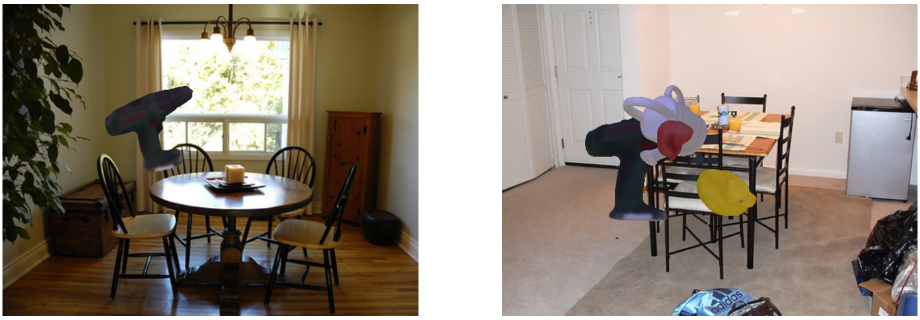
(a) Synthetic Data for LINEMOD (b) Synthetic Data for Occlusion

Fig. 1: Synthetic Data for LINEMOD or Occlusion. 1a shows the synthetic training data used when training on LINEMOD dataset, only one object is presented in the image so there is no occlusion. 1b shows the synthetic training data used when training on OCCLUISON dataset, multiple objects are presented in one image so one object may be occluded by other objects.

Real training images provided in existing datasets may be highly correlated or lack images in certain situations such as occlusions between objects. Therefore, generating synthetic training data is essential to enable the network to deal with different scenarios in testing. In generating synthetic training data for the LINEMOD dataset, considering the fact that the elevation variation is limited in this dataset, we calculate the elevation range of the objects in the provided training data. Then we rotate the object model with a randomly generated quaternion and repeat it until the elevation is within this range. The translation is randomly generated using the mean and the standard deviation computed from the training set. During training, the background of the synthetic image is replaced by a randomly chosen indoor image from the PASCAL VOC dataset as shown in Fig. 1.

For the Occlusion LINEMOD dataset, multiple objects are rendered into one image in order to introduce occlusions between objects. The number of objects ranges from 3 to 8 in these synthetic images. As in the LINEMOD dataset, the quaternion of each object is also randomly generated to ensure that the elevation range is within that of training data in the Occlusion LINEMOD dataset. The translations of the objects in the same image are drawn according to the distributions of the objects in the YCB-Video dataset [6] by adding a small Gaussian noise.

The real training images may also lack variations in light conditions exhibited in the real world or in the testing set. Therefore, we add a random light condition to each synthetic image in both the LINEMOD dataset and the Occlusion LINEMOD dataset.

3 Pose Initialization

Our framework takes an input image and an initial pose estimation of an object in the image as inputs, and then refine the initial pose iteratively. In our experiments, we have tested two pose initialization methods.

The first one is PoseCNN [6], a convolutional neural network designed for 6D object pose estimation. PoseCNN performs three tasks for 6D pose estimation, i.e., semantic labeling to classify image pixels into object classes, localizing the center of the object on the image to estimate the 3D translation of the object, and 3D rotation regression. In our experiments, we use the 6D poses from PoseCNN as initial poses for pose refinement.

To demonstrate the robustness of our framework on pose initialization, we have implemented a simple 6D pose estimation method for pose initialization, where we extend the Faster R-CNN framework designed for 2D object detection [3] to 6D pose estimation. As described in the main text, the bounding box of the object from Faster R-CNN is used to estimate the 3D translation of the object. We set the center of the bounding box as the center of the object to determine the translation along x axis and y axis. For the translation along z axis, or say, the distance of the object, we use the value which can maximize the overlap of the projection of the 3D object model with the bounding box. To estimate the 3D rotation of the object, we add a rotation regression branch after the last feature map of Faster R-CNN as in PoseCNN. In this way, we can obtain a 6D pose estimation for each detected object from Faster R-CNN.

In our experiments on the LINEMOD dataset, we have shown that, although the initial poses from Faster R-CNN are much worse than the poses from PoseCNN, our framework is still able to refine these poses using the *same* weights. The performance gap between using the two different pose initialization methods is quite small, which demonstrates the ability of our framework in using different methods for pose initialization.

4 Evaluation Metrics

$n°$, n cm Proposed in [4]. The 5°, 5cm metric considers an estimated pose to be correct if its rotation error is within 5° and the translation error is below 5cm. We also provided the results with 2°, 2cm and 10°, 10cm in Table 1 to give a comprehensive view about the performance.

For symmetric objects such as eggbox and glue in the LINEMOD dataset, we compute the rotation error and the translation error against all possible ground truth poses with respect to symmetry and accept the result when it matches one of these ground truth poses.

6D Pose Hintertoisser et al. [2] use the average distance (ADD) metric to compute the averaged distance between points transformed using the estimated pose and the ground truth pose as in Eq. 2:

$$\mathbf{ADD} = \frac{1}{m} \sum_{x \in \mathcal{M}} \|\text{3D\_proj}(\mathbf{x}, \mathbf{p}) - \text{3D\_proj}(\mathbf{x}, \hat{\mathbf{p}})\|, \tag{2}$$

where m is the number of points on the 3D object model, $\mathcal{M}$ is the set of all 3D points of this model, $\mathbf{p}$ is the estimated pose and $\hat{\mathbf{p}}$ is the ground truth pose. 3D_proj indicates transforming the point with the give SE(3) transformation (pose) $\mathbf{p}$. Following [1], we compute the distance between all pairs of points from the model and regard the maximum distance as the diameter d of this model. Then a pose estimation is considered to be correct if the computed average distance is within 10% of the model diameter. In addition to using $0.1d$ as the threshold, we also provided pose estimation accuracy using threshholds$0.02d$ and $0.05d$ in Table 1.

For symmetric objects, we use the closet point distance in computing the average distance for 6D pose evaluation as in [2]:

$$\mathbf{ADD\text{-}S} = \frac{1}{m} \sum_{\mathbf{x}_1 \in \mathcal{M}} \min_{\mathbf{x}_2 \in \mathcal{M}} \|\text{3D\_proj}(\mathbf{x}_1, \mathbf{p}) - \text{3D\_proj}(\mathbf{x}_2, \hat{\mathbf{p}})\|. \tag{3}$$

2D Projection focuses on the matching of pose estimation on 2D images. This metric is considered to be important for applications such as augmented reality. We compute the error using Eq. 4 and accept a pose estimation when the 2D projection error is smaller than a predefined threshold:

$$\mathbf{Proj.\ 2D} = \frac{1}{m} \sum_{x \in \mathcal{M}} \|\text{2D\_proj}(\mathbf{x}, \mathbf{p}, \mathbf{K}) - \text{2D\_proj}(\mathbf{x}, \hat{\mathbf{p}}, \mathbf{K})\|, \tag{4}$$

where $\mathbf{K}$ denotes the intrinsic parameter matrix of the camera and 2D_proj indicates transforming a 3D point according to the SE(3) transformation and then projecting the transformed 3D point onto the image. In addition to using 5 pixels as the threshold, we also show our results with the thresholds 2 pixels and 10 pixels.

For symmetric objects such as eggbox and glue in the LINEMOD dataset, we compute the 2D projection error against all possible ground truth poses and accept the result when it matches one of these ground truth poses.

5 Detailed Results on LINEMOD

Table 1 shows our detailed results on all the 13 objects in the LINEMOD dataset. The network is trained and tested with 4 iteration. Initial poses are estimated by PoseCNN [6].

Table 1: Results of using more detailed thresholds on LINEMOD dataset

metric	(n°, n cm)			6D Pose			Projection 2D		
threshold	(2, 2)	(5, 5)	(10,10)	0.02d	0.05d	0.10d	2 px.	5 px.	10 px.
ape	37.71	90.38	98.00	14.29	48.57	76.95	92.19	98.38	99.62
benchvise	37.63	88.65	98.16	37.54	80.50	97.48	67.70	96.99	99.61
camera	56.08	95.78	99.22	30.88	74.02	93.53	86.27	98.92	99.71
can	57.97	92.81	99.02	41.44	84.25	96.46	98.62	99.70	99.80
cat	33.53	87.62	97.80	17.56	50.40	82.14	88.42	98.70	100.00
driller	49.36	92.86	99.11	35.68	79.19	94.95	64.22	96.13	99.41
duck	30.80	85.16	98.50	10.52	48.26	77.65	88.08	98.50	99.81
eggbox	32.11	63.85	94.46	34.74	77.84	97.09	53.43	96.15	99.62
glue	32.82	83.01	97.97	57.34	95.37	99.42	81.47	98.94	99.71
holepuncher	8.66	54.52	93.82	5.33	27.31	52.81	59.09	96.29	99.52
iron	47.50	92.65	99.28	47.91	86.31	98.26	67.42	97.24	99.90
lamp	47.50	90.88	98.37	45.30	86.76	97.50	59.98	94.24	99.04
phone	34.84	89.61	98.58	22.66	60.53	87.72	75.92	97.73	99.81
MEAN	38.96	85.21	97.87	30.86	69.16	88.61	75.60	97.53	99.66

6 Detailed Results on Occlusion LINEMOD

Table 2 shows our results on the Occlusion LINEMOD dataset. We can see that DeepIM can significantly improve the initial poses from PoseCNN. Notice that the diameter here is computed using the extents of the 3D model following the setting of [6] and other RGB-D based methods. Some qualitative results are shown in Figure 2.

7 Test on Unseen Objects

Notice that the transformation predicted from the network does not need to have prior knowledge about the model itself. In this experiment, we explore the ability of the network in refining poses of objects that has not been never seen during training. ModelNet [5] contains a large number of 3D models in different object categories. Here, we tested our network on three of them: *airplane, car and chair*, where we train the network on around 100 object models and test the trained network on another unseen 100 object models. Similar to the way that we generate synthetic data as described in Sec. 2.2, we generate 50 poses for each model as the target poses. We use uniform gray texture for each model and add a light source which has a fixed relative position to the object to reflect the norms of the object. The initial pose we use during training and testing is generated in the same way as we did in previous experiments as described in Sec. 2.1. The results are show in Table 3.

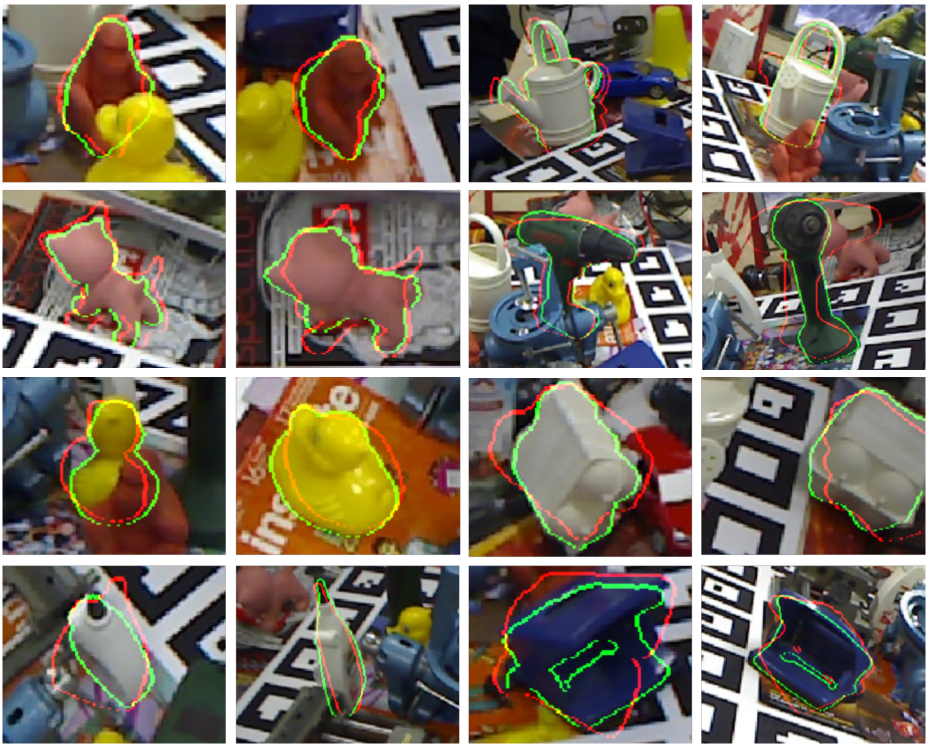

Fig. 2: Some pose refinement results on the Occlusion LINEMOD dataset. The red and green lines represent the edges of 3D model projected from the initial poses and our refined poses respectively.

8 Test on Unseen Categories

We also tested our framework on refining the poses of unseen object categories, where the training categories and the test categories are completely different. We train the network on 8 categories from ModelNet [5]: *airplane, bed, bench, car, chair, piano, sink, toilet* with 30 models in each category and 50 image pairs for each model. The network was trained with 4 iteration and 4 epoch. Then we tested the network on 7 other categories: *bathtub, booksehlf, guitar, range hood, sofa, wardrobe, and tv stand*. The results are shown in Table. 4. It shows that the network indeed learn some general features for pose refinement across different object categories.

Table 2: Results on the Occlusion LINEMOD dataset. The network is trained and tested with 4 iteration.

metric	(5°, 5cm)		6D Pose 0.1d		Projection 2D 5 px.	
method	Initial	Refined	Initial	Refined	Initial	Refined
ape	2.06	**51.75**	9.94	**59.18**	34.56	**69.02**
can	4.06	**35.82**	45.36	**63.52**	15.09	**56.14**
cat	0.28	**12.75**	0.83	**26.24**	10.37	**50.92**
driller	2.48	**45.24**	41.60	**55.58**	7.36	**52.94**
duck	1.84	**22.48**	19.51	**52.41**	31.76	**60.54**
eggbox	0.00	**17.81**	24.48	**62.95**	1.86	**49.18**
glue	0.90	**42.73**	46.18	**71.66**	13.79	**52.92**
holepuncher	1.74	**18.84**	27.02	**52.48**	23.06	**61.16**
MEAN	1.67	**30.93**	26.87	**55.50**	17.23	**56.60**

Table 3: Results on Unseen objects.

method	airplane	car	chair
5cm 5°	68.90	81.45	87.55
6D Pose	94.70	90.65	97.39
Proj. 2D	87.30	91.82	88.63

Table 4: Results on unseen categories. Those categories has never been seen by the network during training.

metric	(5°, 5cm)		6D Pose 0.1d		Projection 2D 5 px.	
method	Initial	Refined	Initial	Refined	Initial	Refined
bathtub	0.92	71.64	11.92	88.60	0.16	73.36
bookshelf	1.20	39.20	9.16	76.44	0.08	51.28
guitar	1.24	50.36	9.64	69.60	0.24	77.08
range hood	1.04	69.84	11.20	89.56	0.04	70.56
sofa	1.24	82.72	9.00	89.48	0.08	94.24
wardrobe	1.40	62.70	12.50	79.40	0.20	70.00
tv stand	1.20	73.56	8.80	92.12	0.16	76.56

References

1. Brachmann, E., Michel, F., Krull, A., Ying Yang, M., Gumhold, S., Rother, C.: Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3364–3372 (2016)
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: Asian Conference on Computer Vision (ACCV) (2012)
3. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS) (2015)
4. Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.: Scene coordinate regression forests for camera relocalization in RGB-D images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2930–2937 (2013)
5. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D shapenets: A deep representation for volumetric shapes. In: IEEE conference on Computer Vision and Pattern Recognition (CVPR). pp. 1912–1920 (2015)
6. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199 (2017)