Supplementary Material for OnePose: One-Shot Object Pose Estimation without CAD Models

Jiaming Sun[1,2*] Zihao Wang[1*] Siyu Zhang[2*] Xingyi He[1] Hongcheng Zhao[3]
Guofeng Zhang[1] Xiaowei Zhou[1†]

[1]Zhejiang University [2]SenseTime Research [3]TUM

1. Implementation Details of the Pose Tracking Module

In this section, the implementation details of our online feature-based pose tracking module are explained. In general, our pose tracking module works similar to a SLAM system to reconstruct a 3D map which is optimized together with predicted poses on-the-fly.

1.1. Pose Initialization

The pose estimation module is executed every five frames, returning a set of 2D keypoints $\{\mathbf{p}_k^t\}$ as well as their corresponding 3D keypoints in the SfM map $\{\mathbf{P}_j\}$. $\mathcal{M}_{3D}^t(j)$ defines the mapping from the index of 2D keypoints index to 3D keypoints. The number of keypoints in each image n_k is omitted for simplicity. For the coming frame with index $t+1$, we use sparse optical flow [6] to track the 2D keypoints from the last frame:

$$\mathbf{p}^{t+1} = \text{opticalFlow}(\mathbf{I}_t, \mathbf{I}_{t+1}, \mathbf{p}^t)$$

where $\mathbf{I}_t$ and $\mathbf{I}_{t+1}$ denotes consecutive input images. Then the 2D-3D correspondences between the query image and the SfM map can be then propagated to the new frame as $\mathcal{M}_{3D}^{t+1}$, from which we can obtain the predicted pose with Perspective-n-Points (PnP) algorithm:

$$\xi_{\text{PnP}}^{t+1} = \text{PnP}(\mathbf{p}^{t+1}, \mathbf{P}_{\mathcal{M}_{3D}^{t+1}}).$$

To handle the potential failures of tracking by optical flow , we further implement a policy to reject ξ_{PnP}^{t+1} if they are too far from the last optimized frame:

$$\xi^{t+1} = \begin{cases} \xi_{\text{PnP}}^{t+1}, \text{if diff}(\xi_{\text{PnP}}^{t+1}, \hat{\xi}^t) < \delta_{\text{trans}} \\ \text{extrapolate}(\hat{\xi}^t, \hat{\xi}^{t-1}), \text{otherwise.} \end{cases}$$

where $\hat{\xi}_t$ denote the optimized pose for frame t and $\text{extrapolate}(\cdot)$ denotes the extrapolation operation given optimized poses of previous frames. We use 10 *cm* and 25° as the threshold δ_{trans} in our implementation.

1.2. Bundle Adjustment

After pose initialization, 2D features are extracted from current frame with SuperPoint [5] which are matched with 2D features of the last keyframe in the keyframe pools. Combined with the correspondence information of the matched keyframe, the 2D-3D correspondences between the current frame and the online-built 3D map can be established. With the predicted pose from pose initialization, we are able to apply bundle adjustment (BA) within a local window of keyframes to optimize for a more precise pose and refine the 3D map. The refined pose by BA is used as the output of the tracking module at each time step.

1.3. Keyframe Creation

We create a new keyframe every 3 frames instead of relying on the number of newly created map points in the counterpart [7], due the tremendous amount of detected keypoints in each frame. New map points are created with keyframes by triangulating 2D matches without 2D-3D correspondences within the frame.

2. 2D Object Detection via Feature Matching

When deploying OnePose to a real-world system, an off-the-shelf category-level 2D object detector like [3] can be used. However, this could defeat the category-agnostic nature of OnePose. We can instead use a feature-matching-based pipeline for 2D object detection, which locates the scanned object on the query image through 2D feature matching.

Specifically, we first sample n reference images from the reference images $\{\mathbf{I}_i\}$ and then perform 2D matching between the query image and the object Regions-of-Interest (RoIs) in each reference image. The object RoIs can be determined by projecting the annotated $\mathbf{B}$ to each reference view. We use SuperPoint [5] and SuperGlue [10] for the matching. The reference-query image pair with the most inliers is selected and used to estimate their affine transformation by the 2D matches. The 2D bounding box can be

#Samples	Large Objects			Medium Objects			Small Objects			Time (*ms*)
	1cm-1deg	3cm-3deg	5cm-5deg	1cm-1deg	3cm-3deg	5cm-5deg	1cm-1deg	3cm-3deg	5cm-5deg	
HLoc *(SIFT + NN)*										
2	0.214	0.418	0.449	0.292	0.489	0.519	0.177	0.357	0.398	47.79
5	0.311	0.559	0.597	0.346	0.571	0.610	0.223	0.429	0.487	116.27
10	0.359	0.622	0.668	0.384	0.608	0.649	0.228	0.443	0.507	218.82
HLoc *(SPP + NN)*										
2	0.257	0.486	0.517	0.350	0.538	0.561	0.229	0.455	0.524	54.61
5	0.333	0.618	0.661	0.425	0.654	0.693	0.304	0.576	0.651	136.98
10	0.388	0.682	0.730	0.464	0.707	0.747	0.325	0.609	0.705	231.89
HLoc *(SPP + SPG)*										
2	0.308	0.553	0.584	0.438	0.618	0.628	0.294	0.542	0.619	221.43
5	0.426	0.789	0.835	0.541	0.779	0.815	0.390	0.711	0.822	555.56
10	0.479	0.846	0.879	0.580	0.839	0.869	0.422	0.754	0.876	1094.25

Table 1. **Experiment on the effects of number of images sampled (i.e., retrieved) for HLoc.** We evaluate our visual localization baselines based on HLoc [9] with differnt number of images sampled.

	1cm-1deg	3cm-3deg	5cm-5deg
Ground-truth box (same w/ the main paper)	0.497	0.775	0.841
Matching detector $n = 4$	0.442	0.726	0.791
Matching detector $n = 10$	0.471	0.762	0.832
Matching detector $n = 15$	0.473	0.770	0.839

Table 2. **Results of pose estimation with different object detectors.** The metrics are averaged across all objects in the evaluation set.

thus estimated by transforming the object RoI corners to the query image with the estimated affine transformation. With the estimated 2D bounding box, 2D-3D feature matching with GATs can be performed as described in the main paper to estimate the 6D poses of the object. Note that the 2D object detection will not slow down the entire pipeline since it is only necessary during the initialization. After the initialization, the 2D bounding box can be obtained by projecting the previously detected 3D bounding box to the current camera frame.

The quantitative results of pose estimation using a different number of reference images for object detection are shown in Tab. 2. In these results, feature-matching-based 2D object detection is used in every frame (instead of only the first frame of a video sequence) to better reflect the pose estimation accuracy.

3. Multi-Sequence Alignment in Dataset Preparation

In this section, we introduce the implementation details of the post-processing steps for dataset preparation. The main purposes of this step are to 1) ensure the 3D bounding box annotations are consistent across sequences and 2) eliminate the potential drifting in the camera poses tracked by ARKit.

The inconsistency of annotations mainly comes from the deviations in different bounding box annotations between multiple sequences of the same object. To make the bounding box annotations consistent, we first align the sequences of each object by converting the camera poses to the object frame with the annotated 3D bounding boxes. The inconsistency of the annotations is propagated to the camera poses during the conversion of coordinate frames and further introduces inaccuracies into the camera poses estimated by ARKit. We alleviate these problems by applying bundle adjustment to the camera frames. After the pose refinement, positions and orientations of the annotated 3D bounding boxes should be naturally aligned. To obtain a consistent definition for the dimensions, we simply compute the mean of dimensions in all annotations for each object.

We demonstrate the quality of our sequence alignment method by projecting points triangulated from one sequence to images from another sequence. As shown in Fig. 1, the reprojected points are highly overlapped with 2D keypoints extracted from the original image, indicating the accuracy of our sequence alignment method.

4. Experiment Details of Comparison with Baselines

In this section, some implementation details for the experiments of our baseline methods are further explained.

4.1. Implementation Details of the Evaluation of HLoc

As mentioned in the main paper, the original image retrieval module cannot generalize to the object-centric setting and we obtain retrieved images by sampling a fixed number of images with the same interval instead. To select a proper number for the images to be sampled, we further apply experiments on HLoc with different number of sampled images. As shown in Tab. 1, sampling 5 images can brought evident improvements of more than 10% compared to sampling 2 images, while sampling 5 more images only brings

Figure 1. **Alignment quality across sequences.** Triangulated point clouds from one sequence are projected to images from another sequence of the same object. The quality of alignment can be measured by the differences between projected 3D points (in blue) and the original 2D keypoints (in red) on the images.

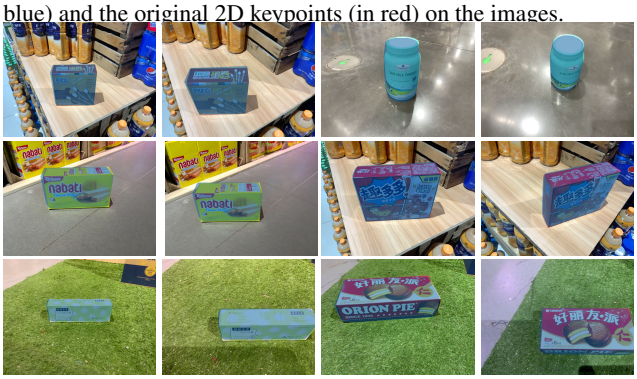

Figure 2. **Examples of generated masks for PVNet training.** The generated masks are overlayed on the original image in blue.

marginal improvement at an average 5% for most of metrics at a cost of doubled runtime. Thus, we choose to sample 5 images for our experiments in the main paper since it best balances the precision and the inference efficiency.

4.2. Implementation Details of the Evaluation of PVNet

For the experiments of evaluating PVNet [8], we directly use the original implementation and training configurations provided by the authors at [1]. To train PVNet, we additionally apply reconstruction and mask rendering on our dataset and manually pick objects with high reconstruction quality for experiments. Some results for the reconstructed meshes and generated masks for training are as shown in the Fig. 2.

4.3. Implementation Details of the Evaluation of Objectron

For the experiments of evaluating Objectron [4], we directly use the Python APIs provided in Mediapipe [2] for inference where only the models for *Shoe*, *Chair*, *Cup*, *Camera* are provided which greatly limited our choices for objects to use in the experiments. Objectron can only produce object poses up to a scale, and requires additional plane information for scale recovery which are provided in the original Objectron dataset but unavailable in our dataset. Therefore, we are unable to evaluate the 3D IoU metrics in the metric scale as the original paper.

References

[1] clean-pvnet. https://github.com/zju3dv/clean-pvnet. 3

[2] Mediapipe. https://google.github.io/mediapipe/solutions/objectron.html. 3

[3] Ultralytics/yolov5. https://github.com/ultralytics/yolov5, 2021. 1

[4] Adel Ahmadyan, Liangkai Zhang, Jianing Wei, Artsiom Ablavatski, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. *CVPR*, 2021. 3

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017. 1

[6] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981. 1

[7] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31:1147–1163, 2015. 1

[8] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019. 3

[9] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 2

[10] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *ICCV*, 2020. 1