

GS-Pose: Generalizable Segmentation-based 6D Object Pose Estimation with 3D Gaussian Splatting

Supplementary Material

1. Supplement

1.1. Data Capture with ARKit

We follow OnePose and OnePose++ [1, 6] and leverage the off-the-shelf ARKit¹ to capture the RGB reference images using an iPhone. In our experiments, we take advantage of the ARKit-based OnePoseCap [6] application initially developed for OnePose [6]. In OnePoseCap, we first manually define a simple 3D bounding box around a stationary object, which serves as a minimal, 3D CAD model-free geometric proxy. When using this tool to capture the RGB reference sequence of an object, ARKit automatically tracks the frame-wise camera poses with respect to the predefined 3D box based on feature matching. These tracked poses are then transformed into the 3D bounding box coordinate system and used as the 6D object pose annotations. We kindly refer the reader to OnePose/OnePose++ [1, 6] for more details.

1.2. Preprocessing

In this part, we describe the image pre-processing step for creating the reference database. Given the reference data $\{\hat{I}_i^{ref}, R_i^{ref}, t_i^{ref}\}_{i=1}^{N_r}$ of the target object, where $\hat{I}_i^{ref} \in \mathbb{R}^{H \times W \times 3}$ denotes the i^{th} RGB image, 3D rotation matrix R_i^{ref} , and 3D translation vector t_i^{ref} , we preprocess these reference images to obtain normalized object-centric images $\{I_i^{ref}\}_{i=1}^{N_r}$ with a predefined resolution $S \times S$.

Specifically, we first compute a 2D square bounding box $\{C_i^{ref}, S_i^{ref}\}$ using the ground truth translation vector t_i^{ref} and the camera intrinsic K^{ref} , where $C_i^{ref} \in \mathbb{R}^2$ and $S_i^{ref} \in \mathbb{R}$ are the 2D center and scale of the bounding box, respectively. In particular, we generate the 2D box center C_i^{ref} by projecting the 3D translation t_i^{ref} to the image plane using K^{ref} and then compute the scale factor by $S_i^{ref} = d_{obj} \cdot f^{ref} / t_z^{ref}$, where d_{obj} is the diameter of the 3D object bounding box, f^{ref} is the camera focal length, and $t_z^{ref} \in \mathbb{R}$ denotes the z-axis component of the 3D translation vector t_i^{ref} . Subsequently, we crop the object region (I_i^{ref}) using the derived bounding box ($\{C_i^{ref}, S_i^{ref}\}$) and rescale it to the fixed size S . After preprocessing, the object in all normalized reference images is assumed to be located along the camera optical axis $[0, 0, t_z^{ref}]^T$ at the same distance $t_z^{ref} = d_{obj} \cdot f^{ref} / S$. We set $S = 224$ in all experiments.

¹ARKit. <https://developer.apple.com/augmented-reality>.

Algorithm 1 Iterative Pose Refinement within GS-Refiner

```

1: Input initial pose:  $P^{init}$ 
2: Input segmented object image:  $\bar{I}^{que}$ 
3: Initialize the maximum iteration steps:  $N_g = 400$ 
4: Initialize rotation quaternion parameters:  $q_0 = [1, 0, 0, 0]^T$ 
5: Initialize translation parameters:  $t_0 = [0, 0, 0]^T$ 
6: Initialize iteration counter:  $i = 0$ 
7: Initialize learning rate:  $r_0 = 0.005$ 
8: Form transformation matrix:  $P_0 = MatrixForm(q_0, t_0)$ 
9: while  $i < N_g$  do
10:   Transform object:  $\mathcal{G}_{P_i}^{obj} = RigidTransform(\mathcal{G}^{obj}, P_i)$ 
11:   Render object:  $I_{P_i}^{end} = GaussianRenderer(\mathcal{G}_{P_i}^{obj}, P^{init})$ 
12:   Compute loss:  $\mathcal{L}_{gs} = LossCriterion(I_{P_i}^{end}, I_{obj}^{que})$ 
13:   Compute gradients:  $\delta_q = \frac{\partial \mathcal{L}_{gs}}{\partial q_i}, \delta_t = \frac{\partial \mathcal{L}_{gs}}{\partial t_i}$ 
14:   Update LR:  $r_{i+1} = CosineAnnealingLRScheduler(r_i)$ 
15:   Update params:  $q_{i+1} = q_i + r_{i+1}\delta_q, t_{i+1} = t_i + r_{i+1}\delta_t$ 
16:   Transformation matrix:  $P_{i+1} = MatrixForm(q_{i+1}, t_{i+1})$ 
17:   Update iteration counter:  $i = i + 1$ 
18:   if  $\mathcal{L}_{gs}$  converges then
19:     break
20:   end if
21: end while
22: Update initial pose:  $P = P^{init} P_i$ 

```

1.3. Iterative Pose Refinement with GS-Refiner

We summarize the iterative pose refinement process in Algorithm 1.

1.4. Additional Results on LINEMOD

Following Gen6D [3], we additionally compare GS-Pose with baselines on a subset of objects in LINEMOD and report the results in Table 1. GS-Pose achieves 47.96% ADD(S)@0.1d without pose refinement and 90.86% after refinement, surpassing all baseline approaches by a significant margin. It is noteworthy that without using a subset of objects included in LINEMOD (using the same setup as ours) for training, Gen6D achieves 42.72% accuracy after pose refinement, falling behind even the initial results of GS-Pose (47.96%). As an additional experiment, we also leverage the feature volume-based pose refiner (Vol-Refiner) proposed in Gen6D [3] for pose refinement. Vol-Refiner improves the initial accuracy to 69.71%, lagging behind 90.86% achieved with GS-Refiner.

We show qualitative examples from LINEMOD in Fig. 1 and report the complete segmentation and detection results in Tab. 3 and the initial pose estimation results in Tab. 2.

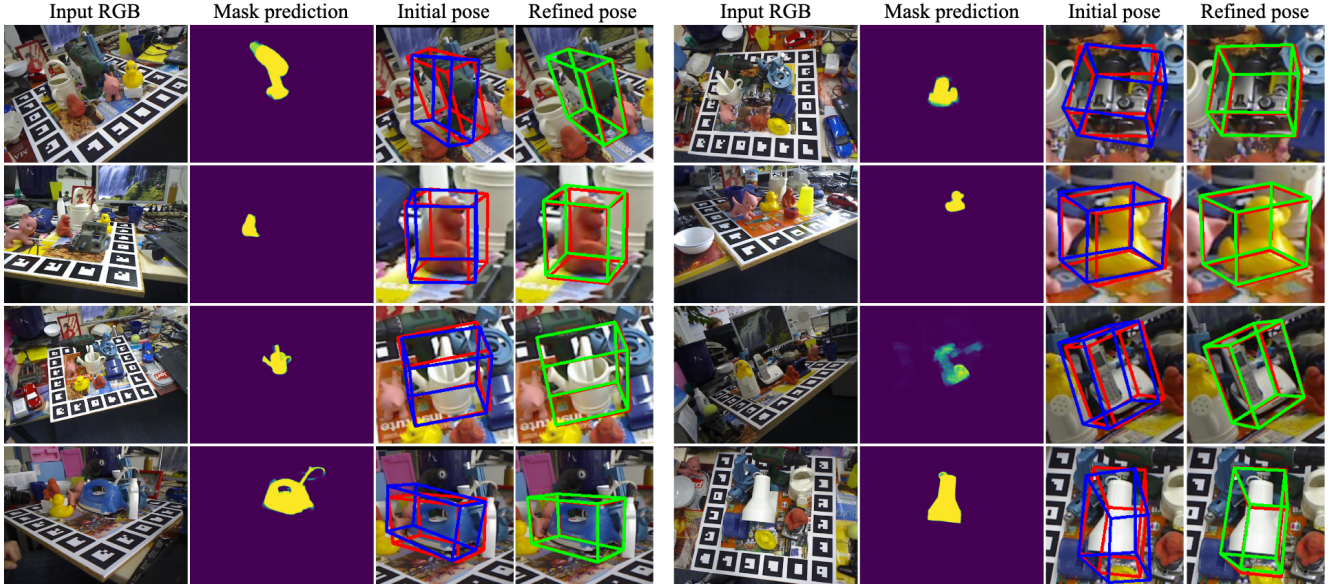


Figure 1. Qualitative evaluation on LINEMOD. We present the intermediate segmentation mask predictions (for localization) as well as the estimated 6D poses. Blue, green, and red boxes represent initial, refined, and ground truth poses, respectively.

Method	Pose Refinement	cat	duck	bvise	cam	driller	Avg.
Gen6D [3] [†]	✗	15.97	7.89	25.48	22.06	17.24	17.73
LocPoseNet [8] [†]	✗	-	-	-	-	-	27.27
OSOP [5]	✗	34.43	20.08	50.41	32.30	43.94	36.23
GS-Pose (ours)	✗	48.70	41.97	55.87	44.31	48.96	47.96
OSOP [5]	OSOP [5]	42.54	22.16	55.59	36.21	49.57	42.21
Gen6D [3]	Vol-Refiner [3]	40.92	16.24	62.11	45.59	48.76	42.72
Gen6D [3] [†]	Vol-Refiner [3]	60.68	40.47	77.03	66.67	67.39	62.45
LocPoseNet [8] [†]	Vol-Refiner [3]	-	-	-	-	-	68.58
Cas6D [4] [†]	Cas-Refiner [4]	60.58	51.27	86.72	70.10	84.84	70.72
GS-Pose (ours)	Vol-Refiner [3]	60.68	53.24	83.04	70.10	81.47	69.71
GS-Pose (ours)	GS-Refiner (ours)	88.82	74.74	99.61	95.98	95.14	90.86

Table 1. Additional quantitative results on the subset of objects in LINEMOD [2] regarding the ADD(S)@0.1d metric. [†] indicates that another heldout subset of objects in LINEMOD is included in the training data of the method. ”-” indicates unavailable results. We highlight the best in **Bold**.

Method	YOLOv5	ape	bvise	cam	can	cat	driller	duck	ebox*	glue*	holep.	iron	lamp	phone	Avg.
GS-Pose _{init}		31.5	55.9	44.3	64.9	48.7	49.0	42.0	92.8	67.7	48.1	47.2	48.9	36.0	52.1
GS-Pose (ours)		59.6	99.6	96.0	97.6	88.9	95.1	74.9	99.3	92.2	86.8	98.2	96.7	80.7	89.7
GS-Pose _{init}	✓	39.3	58.8	45.1	64.3	53.6	50.7	38.8	93.7	74.4	52.1	55.9	56.3	37.8	55.5
GS-Pose (ours)	✓	71.0	99.8	98.2	97.7	86.7	96.2	77.2	99.6	98.4	87.4	99.2	98.9	85.0	92.0

Table 2. Results on LINEMOD [2] regarding the ADD(S)@0.1d metric. ✓ indicates using the detection results provided by YOLOv5 [7]. ”_{init}” indicates the initial pose estimation results of GS-Pose.

References

[1] Xingyi He, Jiaming Sun, Yuang Wang, Di Huang, Hujun Bao, and XiaoWei Zhou. Onepose++: Keypoint-free one-shot ob-

ject pose estimation without cad models. *Advances in Neural Information Processing Systems*, 35:35103–35115, 2022. 1

[2] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Ste-

Type	Metric	ape	bwise	cam	can	cat	driller	duck	ebox	glue	holep.	iron	lamp	phone	Avg.
BBox	mAP@50	62.0	100.0	97.2	100.0	96.4	98.9	98.8	97.2	88.7	93.6	96.8	93.8	85.1	93.0
	mAP@75	60.8	80.6	87.8	98.8	95.1	89.5	95.1	95.7	46.1	90.0	42.0	58.4	70.9	77.8
	mAP@[50:95]	56.2	71.8	79.0	87.8	84.4	80.6	86.9	81.4	53.4	74.1	52.3	56.4	61.8	71.2
Mask	mAP@50	66.6	100.0	98.5	100.0	96.9	99.0	99.0	98.6	89.0	95.6	100.0	94.9	91.6	94.6
	mAP@75	61.9	90.9	91.6	97.3	90.6	86.9	95.1	97.0	83.5	85.9	82.3	39.6	59.0	81.7
	mAP@[50:95]	53.0	66.5	75.5	70.9	69.1	73.1	80.1	79.4	61.8	67.1	62.3	47.0	53.0	71.2

Table 3. Complete segmentation and detection results on **LINEMOD** [2]. "BBox" represents the use of square 2D bounding boxes to evaluate the mask-induced detection results. "Mask" indicates 2D segmentation results.

- fan 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 *Asian conference on computer vision*, pages 548–562. Springer, 2012. 2, 3
- [3] Yuan Liu, Yilin Wen, Sida Peng, Cheng Lin, Xiaoxiao Long, Taku Komura, and Wenping Wang. Gen6d: Generalizable model-free 6-dof object pose estimation from rgb images. In *European Conference on Computer Vision*, pages 298–315. Springer, 2022. 1, 2
- [4] Panwang Pan, Zhiwen Fan, Brandon Y Feng, Peihao Wang, Chenxin Li, and Zhangyang Wang. Learning to estimate 6dof pose from limited data: A few-shot, generalizable approach using rgb images. *arXiv preprint arXiv:2306.07598*, 2023. 2
- [5] Ivan Shugurov, Fu Li, Benjamin Busam, and Slobodan Ilic. Osop: A multi-stage one shot object pose estimation framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6835–6844, 2022. 2
- [6] Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowei Zhou. Onepose: One-shot object pose estimation without cad models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6825–6834, 2022. 1
- [7] Ultralytics. Yolov5: Real-time object detection, 2023. 2
- [8] Chen Zhao, Yinlin Hu, and Mathieu Salzmann. Locposenet: Robust location prior for unseen object pose estimation. *arXiv preprint arXiv:2211.16290*, 2022. 2