GS-Pose: Cascaded Framework for Generalizable Segmentation-based 6D Object Pose Estimation

Dingding Cai[1], Janne Heikkilä[2], and Esa Rahtu[1]

[1] Tampere University, Finland
[2] University of Oulu, Finland

Abstract. This paper introduces GS-Pose, an end-to-end framework for locating and estimating the 6D pose of objects. GS-Pose begins with a set of posed RGB images of a previously unseen object and builds three distinct representations stored in a database. At inference, GS-Pose operates sequentially by locating the object in the input image, estimating its initial 6D pose using a retrieval approach, and refining the pose with a render-and-compare method. The key insight is the application of the appropriate object representation at each stage of the process. In particular, for the refinement step, we utilize 3D Gaussian splatting, a novel differentiable rendering technique that offers high rendering speed and relatively low optimization time. Off-the-shelf toolchains and commodity hardware, such as mobile phones, can be used to capture new objects to be added to the database. Extensive evaluations on the LINEMOD and OnePose-LowTexture datasets demonstrate excellent performance, establishing the new state-of-the-art. Project page: https://dingdingcai.github.io/gs-pose

Keywords: 6D object pose estimation · Pose refinement

1 Introduction

The ability to obtain an object's 3D location and orientation based on RGB images is a long-standing and important problem in computer vision and robotics. This 6D pose information is vital in applications that interact with the physical world, such as robotic manipulation [8,9] and augmented reality [29,43]. Popular pose estimation approaches are based on training instance-specific models, and they often assume the availability of an external object detector for detecting the object from input RGB images. While some works have proposed approaches to circumvent this problem [25, 32, 42], they often rely on high-fidelity 3D CAD models of the object, which can be expensive and time-consuming to acquire.

Ideally, a new object should be learned from a casually captured set of RGB reference images without requiring any expensive model parameter optimization. Recently, Liu *et al*. [27] introduced a method called Gen6D in this direction. Gen6D works by extracting 2D feature maps from the reference images, which are subsequently utilized for various sub-tasks, including object localization, initial pose estimation, and pose refinement. However, relying only on 2D representation often leads to sub-optimal performance. Alternatively, OnePose [46]

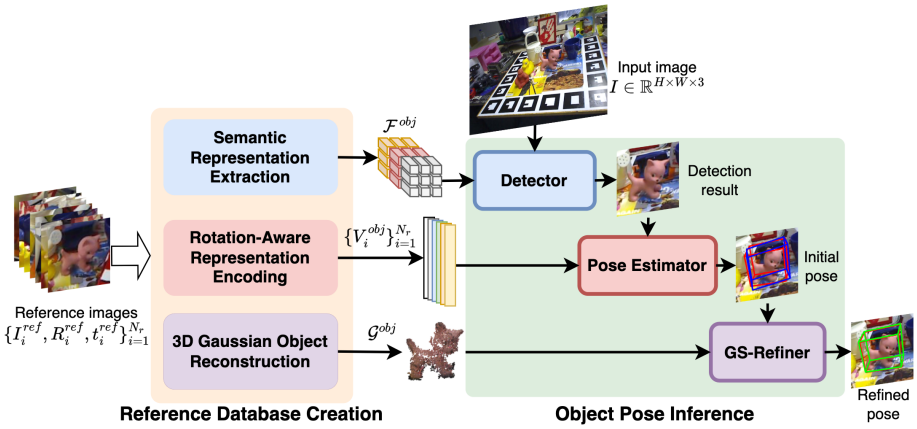

Fig. 1: Overview of GS-Pose. GS-Pose involves two distinct phases to achieve pose estimation for a novel object, *i.e.*, reference database creation and object pose inference. The first phase operates offline and occurs only once per object to construct multiple representations of the object. These representations include an object semantic representation ($\mathcal{F}^{obj}$), a set of rotation-aware embedding vectors ($\{V_i^{obj}\}_{i=1}^{N_r}$), and a 3D Gaussian Object ($\mathcal{G}^{obj}$). During inference, GS-Pose first employs an object detector to detect the object in a query image using the semantic information $\mathcal{F}^{obj}$. Then, GS-Pose adopts a pose estimator to produce an initial pose (blue box) from the detection result with the rotation-aware embeddings $\{V_i^{obj}\}_{i=1}^{N_r}$, Finally, GS-Pose leverages a pose refinement module (GS-Refiner) with $\mathcal{G}^{obj}$ to obtain a refined pose (green box). We indicate the ground-truth pose in red.

and OnePose++ [14] explicitly reconstruct a 3D point cloud from the reference images via local feature matching. The 6D pose is obtained using 2D-3D correspondence matching between the test image and the reference point cloud. The practical challenge is to obtain an accurate 3D point cloud representation, particularly for texture-less and symmetric objects. Furthermore, both approaches still assume an external object detector for cropping out the object of interest, limiting the real generalizability of these methods.

The key ingredient in the pose estimation is the object representation generated from the input images. Popular choices include 2D feature maps [27], 3D point clouds [14, 46], latent 3D models [35], and 3D CAD models [42], to name a few. Generally, each representation exhibits strengths in one aspect, e.g. object localization or fast initial 6D pose approximation, but performs poorly on other parts of the pipeline. With these insights, we propose a framework that applies multiple representations optimized for the three key steps: 1) object localization, 2) fast initial 6D pose estimation, and 3) iterative pose refinement. In particular, we leverage the recent advancements in so-called Foundation models and co-segmentation paradigms to construct powerful representations for object localization using only a handful of reference images. Secondly, we estimate a rough 6D pose using optimized template retrieval. Finally, the pose estimate is

refined using an iterative render-and-compare technique. To this end, we rely on a novel inverse rendering method called 3D Gaussian Splatting (3DGS) [17], which represents a scene by many differentiable 3D Gaussian primitives with optimizable geometric and appearance properties. This explicit representation enables real-time photorealistic rendering capabilities, ideal for 6D pose optimization.

We evaluate the proposed framework, called GS-Pose, on the LINEMOD [15] and OnePose-LowTexture [14] datasets and obtain the new state-of-the-art results on both benchmarks. Finally, we demonstrate how the framework's components can be utilized for object pose tracking over video frames.

The contributions of our work can be summarized as:

- We propose a new multi-stage framework for object localization and 6D pose estimation. For each stage, we propose an optimized representation obtained from a set of posed RGB images of previously unseen objects.
- We present a generalizable co-segmentation approach for extracting object segmentation masks from the reference RGB images. The masks can be further used for representation learning.
- We present an efficient and accurate 3D Gaussian Splatting-based pose refinement and tracking methods.
- We experimentally confirm that the proposed framework achieves state-of-the-art performance on the LINEMOD and OnePose-LowTexture datasets.

2 Related Works

Object-Specific Pose Estimation. Most existing pose estimation methods [2, 3, 5, 12, 16, 18, 24, 37, 44, 49, 52] are object-specific pose estimators, which are specialized for pre-defined objects and cannot generalize to previously unseen objects without retraining. Some of them [2,3,5,18,49,52] directly regress the 6D pose parameters from RGB images by training deep neural networks on a large number of labeled images. While other approaches [5,12,16,24,36,37,44] establish 2D-3D correspondences between 2D images and 3D object models to estimate the 6D pose by solving the Perspective-n-Point (PnP) [21] problem. To relax the assumptions about each object instance, category-level methods [4, 6, 7, 50] have recently been proposed to handle unseen object instances of the same trained category by assuming that objects within the same category share similar shape priors. However, they are still incapable of estimating the object pose of unknown categories.

Generalizable Object Pose Estimation. This type of work [1, 14, 27, 34, 42, 46, 47] removes the requirement of the object specific-training and can perform pose estimation for previously unseen objects during inference. There are two mainstreams, *i.e.*, object model-based and object model-free. The model-based approaches [1, 42, 47] assume access to the 3D CAD models for rendering the object pose-conditioned images that are often utilized for template matching [1, 47, 53], pose refinement [23], or correspondence establishment [42]. To avoid

3D CAD models, recent works [14, 27, 34, 46] resort to capturing object multi-view images with known poses as reference data for pose estimation. OnePose series [14,46] utilize the posed RGB images to reconstruct 3D object point clouds and establish explicit 2D-3D correspondences between 2D query images and the reconstructed 3D point clouds to solve the 6D pose. However, reliance on correspondences becomes fragile when applied to objects with visual ambiguities, such as symmetry. Besides, the above methods often assume that the 2D object detection or segmentation mask is available given a query image. In contrast, Gen6D [27] leverages the labeled reference images to detect the object in query images, initialize its pose, and then construct a 3D feature volume for pose refinement, which is the first work to simultaneously satisfy the requirements of being fully generalizable, model-free, and RGB-only. The follow-up works [34,54] revisit the Gen6D pipeline and improve the performance and robustness in object localization and pose estimation.

2D Object Detection. Commonly used object detection methods [13, 40, 41] are category-specific detectors and cannot generalize to untrained categories. To tackle this issue, some approaches [22, 27, 33, 42, 55] leverage object reference images to detect previously unseen objects through template matching or feature correlation. However, they often show limited generalization ability to new domains.

3D Object Representation. Most generalizable pose estimators [1, 25, 30, 32, 42] often assume that the 3D object representations are available, such as 3D CAD models. OnePose family [14, 46] explicitly reconstructs 3D object point clouds from object multi-view RGB images, which can easily fail with challenging symmetric or textureless objects. Moreover, LatentFusion [35] and Gen6D series [27, 34] utilize the 2D image features to build the 3D object feature volumes for pose refinement. In this work, we instead exploit the differentiable 3D Gaussian Splatting [17] technique to create 3D Gaussian Object representations for pose estimation. To the best of our knowledge, GS-Pose is the first work that leverages 3D Gaussian Splatting for 6D object pose estimation.

3 Approach

This section presents GS-Pose for estimating the 6D pose of novel objects from RGB images. An overview of GS-Pose is provided in Fig. 1. GS-Pose operates in two distinct phases: object reference database creation and object pose inference. The creation phase, requiring RGB images of a novel object with known poses (*e.g.*, captured with commodity devices like mobile phones), is performed offline once per object. During inference, GS-Pose leverages the pre-built object reference database to facilitate 6D pose estimation of a novel object in a cascaded manner. In the subsequent subsections, we first present the reference database creation process in Sec. 3.1. Next, we describe the inference workflow for estimating the 6D object pose in Sec. 3.2. Finally, we present the objective functions for training GS-Pose in Sec. 3.3.

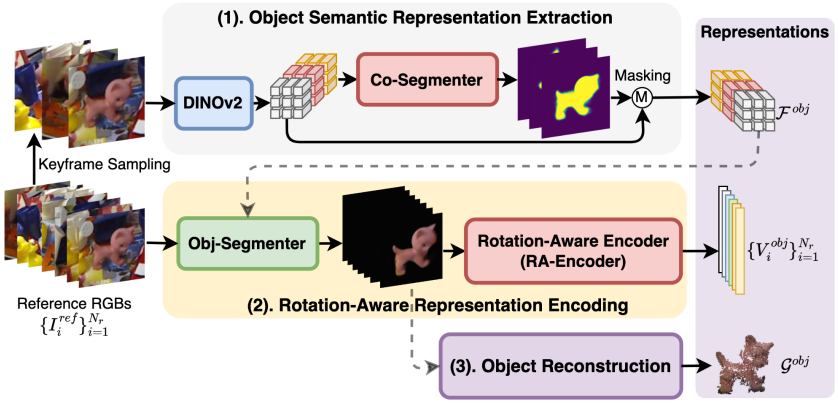

Fig. 2: Overview of the reference database creation process. We begin by selecting a group of keyframes from reference images. (1). These keyframes are processed through DINOv2 and Co-Segmenter to jointly predict object segmentation masks. These predicted masks are then utilized to extract the object semantic tokens ($\mathcal{F}^{obj}$) from the keyframe feature tokens. (2). We perform image segmentation for all reference images $\{I_i^{ref}\}_{i=1}^{N_r}$ using an Obj-Segmenter with the obtained semantic information $\mathcal{F}^{obj}$. Subsequently, we employ an RA-Encoder to extract the rotation-aware embeddings $\{V_i^{obj}\}_{i=1}^{N_r}$ from the segmented images. (3). Finally, we create a 3D Gaussian Object representation $\mathcal{G}^{obj}$ (viewed as a 3D point cloud for simplicity) using all segmented images with poses.

3.1 Reference Database Creation

This section describes the process for creating the reference database of a novel object based on its reference data. This database is primarily comprised of object semantic representation $\mathcal{F}^{obj}$, a set of 3D object rotation-aware embedding vectors $\{V_i^{obj}\}_{i=1}^{N_r}$, and a 3D Gaussian Object representation $\mathcal{G}^{obj}$. The creation process involves three sub-steps: (1) semantic representation extraction, (2) 3D object rotation-aware representation encoding, and (3) 3D Gaussian Object (3DGO) model reconstruction, as depicted in Fig. 2. In the following subsections, we elaborate on each sub-step.

Semantic Representation Extraction. To equip GS-Pose for 2D object detection and segmentation, we first extract a set of feature representation tokens that can effectively capture the semantic information of the target object from reference images. We leverage DINOv2 [31] to extract these tokens from RGB images. Essentially, a Co-Segmenter is employed to segment the object from the background, ensuring that only relevant feature tokens within the object region are considered (see Fig. 2 top). Given N_r reference images, we first select N_k ($\ll N_r$) keyframes using farthest point sampling (FPS) [39] based on their corresponding 3D rotation labels. Then, we extract image feature tokens $\mathcal{F}^{fps} \in \mathbb{R}^{N_k \times L \times C}$ from these keyframes using DINOv2, where L and C denote the token number and feature dimension of each frame. Next, we feed these

feature tokens into the proposed Co-Segmenter, consisting of a transformer-like module and a mask decoding head, to jointly predict the object segmentation masks. Specifically, we reshape the keyframe feature tokens as feature maps (denoted as $\hat{\mathcal{F}}^{fps}$), from which we sample a set of frame-wise center tokens $\hat{\mathcal{F}}_c^{fps} \in \mathbb{R}^{N_k \times C}$ located at the 2D center of these feature maps. Next, the transformer-like module takes $\mathcal{F}^{fps}$ and $\hat{\mathcal{F}}_c^{fps}$ as input and sequentially performs the self- and cross-attention computation, which can be formulated as

$$L_m \times \begin{cases} \mathcal{F}^{fps} = \text{SelfAttn}(F^{fps}) \in \mathbb{R}^{N_k \times L \times C} \\ \mathcal{F}^{fps} = \text{Reshape}(\mathcal{F}^{fps}) \in \mathbb{R}^{1 \times N_k L \times C} \\ \mathcal{F}^{fps} = \text{CrossAttn}(\mathcal{F}^{fps}, \hat{\mathcal{F}}_c^{fps}) \\ \mathcal{F}^{fps} = \text{SelfAttn}(\mathcal{F}^{fps}) \in \mathbb{R}^{1 \times N_k L \times C} \\ \mathcal{F}^{fps} = \text{Reshape}(\mathcal{F}^{fps}) \in \mathbb{R}^{N_k \times L \times C} \end{cases}, \tag{1}$$

where $L_m = 4$ is the depth of this module. The transformed $\mathcal{F}^{fps}$ is then fed into the mask decoding head (two 3×3 convolutional layers followed by an upsampling layer) to produce the keyframe segmentation masks (see Fig. 3 top). Finally, we extract the object-aware semantic feature tokens $\mathcal{F}^{obj}$ from the keyframe feature maps $\hat{\mathcal{F}}^{fps}$ using the predicted masks.

Rotation-Aware Representation Encoding. This step focuses on extracting the 3D object rotation-aware embedding vectors from reference images, which enables GS-Pose to estimate an initial pose via template retrieval. To achieve this, we first adopt an Obj-Segmenter to segment the object from each reference image and then utilize a Rotation-Aware Encoder (RA-Encoder) to extract an image-level embedding vector from the segmented image (see Fig. 2 middle). Obj-Segmenter includes the DINOv2 backbone, a transformer-like module, and a mask decoding head (identical to the one in Co-Segmenter). Concretely, Obj-Segmenter first extracts the DINOv2 feature tokens $F_i^{ref} \in \mathbb{R}^{L \times C}$ from the i^{th} reference image. Then, the image feature tokens (F_i^{ref}) along with the object semantic tokens ($\mathcal{F}^{obj}$) are fed into the transformer-like module to perform self- and cross-attention computation, which can be formulated as

$$L_m \times \begin{cases} F_i^{ref} = \text{SelfAttn}(F_i^{ref}) \\ F_i^{ref} = \text{CrossAttn}(F_i^{ref}, \mathcal{F}^{obj}) \end{cases}. \tag{2}$$

Subsequently, the mask decoding head is utilized to produce a segmentation mask M_i^{ref} from the transformed image features F_i^{ref} (see Fig. 3 middle). Finally, we extract an image-level representation vector $V_i^{ref} \in \mathbb{R}^{64}$ from the segmented image using RA-Encoder. RA-Encoder includes the DINOv2 backbone, four 3×3 convolutional layers with stride 2, a generalized average pooling layer, and a fully connected layer with an output dimension of 64 (see Fig. 3 bottom).

3D Gaussian Object Reconstruction. The last step is to create a 3DGO representation $\mathcal{G}^{obj}$ for pose refinement (see Fig. 2 bottom). 3D Gaussian Splatting [17] represents a 3D structure as a set of 3D Gaussians. Each 3D Gaussian is

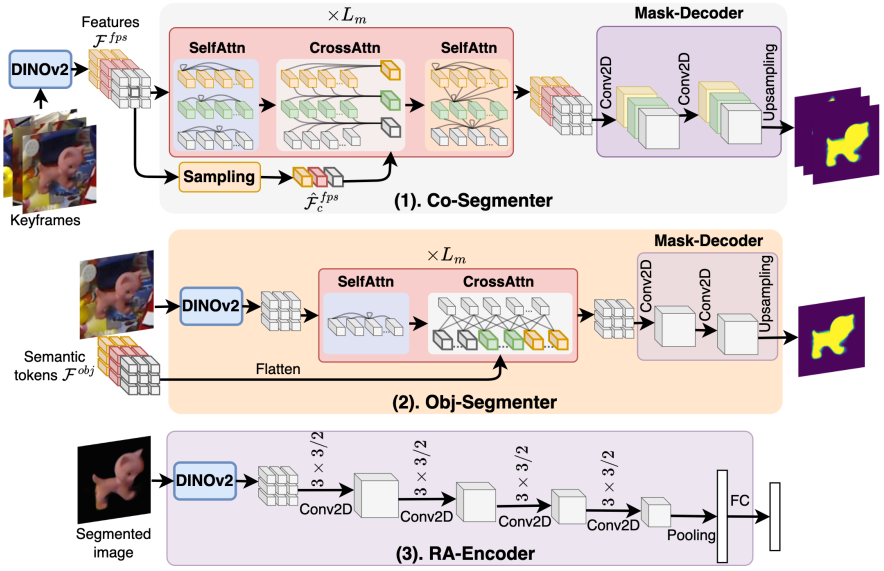

Fig. 3: (1). Co-Segmenter includes a transformer-like module and a mask decoding head (Mask-Decoder). The transformer-like module takes the keyframe feature tokens ($\mathcal{F}^{fps}$) and the sampled center tokens ($\hat{\mathcal{F}}_c^{ref}$) as input and then sequentially performs the frame-wise self-attention, cross-attention, and group-wise self-attention for L_m times. The transformed features are lastly fed into the mask decoder to produce the keyframe co-segmentation masks. **(2). Obj-Segmenter** consists of a feature extraction backbone (DINOv2), a transformer-like module, and a mask decoding head (identical to the one in Co-Semgenter). Likewise, the transformer-like module performs sequential frame-wise self-attention and cross-attention with the object semantic tokens ($\mathcal{F}^{obj}$) for L_m times. Then, the mask decoder predicts a segmentation mask from the transformed image features. **(3). RA-Encoder** contains the DINOv2 backbone, four 3×3 2D convolutional (Conv2D) layers with stride 2, a generalizable average pooling layer, and a fully connected (FC) layer. RA-Encoder extracts the 3D object rotation-aware representation embedding from a segmented RGB image.

parameterized with a 3D coordinate $\mu \in \mathbb{R}^3$, a 3D rotation quaternion $r \in \mathbb{R}^4$, a scale vector $s \in \mathbb{R}^3$, an opacity factor $\alpha \in \mathbb{R}$, and spherical harmonic coefficients $h \in \mathbb{R}^k$, where k is the degrees of freedom. Consequently, a 3DGO model is represented as $\mathcal{G}^{obj} = \{\mu_i, r_i, s_i, \alpha_i, h_i\}_{i=1}^{U}$, where U is the number of 3D Gaussians. All segmented reference images with known poses are utilized to build this 3DGO model. We refer to [17] for more details.

3.2 Object Pose Inference

This section outlines the inference pipeline of GS-Pose, a cascaded process consisting of three core components. Firstly, GS-Pose employs an object detector for detection. Secondly, GS-Pose obtains an initial pose using a pose estimator

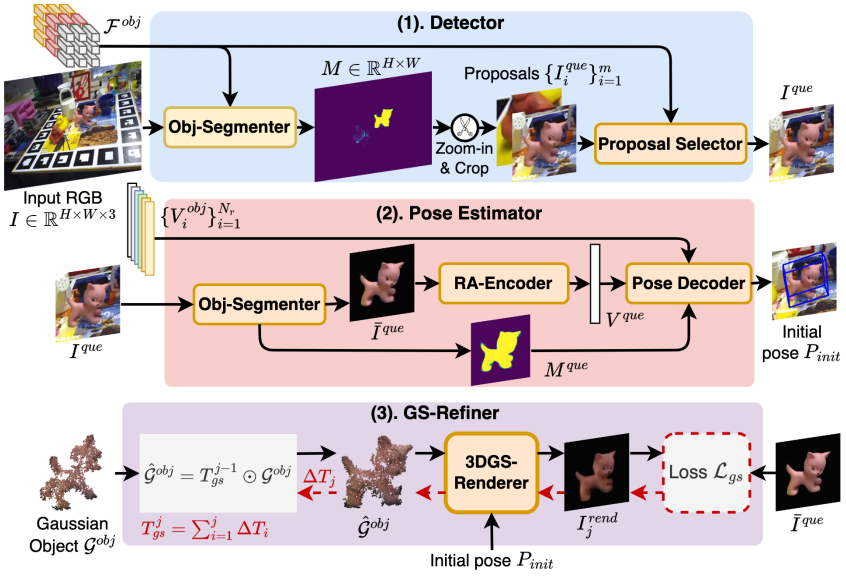

Fig. 4: (1). Detector first employs an Obj-Segmenter to produce a mask from the input image using the semantic information ($\mathcal{F}^{obj}$). Then, connected components are computed from the predicted mask to generate proposals, which are further processed by a proposal selector to determine the final output. **(2). Pose Estimator** utilizes an Obj-Segmenter to predict an object mask M^{que} ($\mathcal{F}^{obj}$ is omitted for clarity). An embedding vector V^{que} is then extracted from the segmented image using RA-Encoder, followed by a pose decoder for estimating an initial pose (P_{init}) using both V^{que} and M^{que}. **(3). GS-Refiner** starts by applying an optimizable transformation T_{gs}^{j-1} to the 3D coordinates of the 3D Gaussian Object (3DGO) $\mathcal{G}^{obj}$, where $j \geq 1$ is the refinement step. Then, the 3D Gaussian Splatting-based renderer (3DGS-Renderer) generates an RGB image (I_j^{rend}) using the initial pose (P_{init}) and the transformed 3DGO ($\hat{\mathcal{G}}^{obj}$). Finally, the gradient ΔT_i is used to update the transformation parameter T_{gs}^{j}, minimizing the difference ($\mathcal{L}_{gs}$) between the rendered image and the segmented image.

based on the detection. Finally, a 3D Gaussian Splatting-based pose refinement module (GS-Refiner) is adopted to optimize the initial pose. Fig. 4 illustrates these components, and we describe each one in detail below.

Detector. We leverage a segmentation-based detector to localize the target object using the object semantic representation (see Fig. 4 top). The detector consists of an Obj-Segmenter (as described in Sec. 3.1) and a proposal selector. Specifically, given an input image, we first apply Obj-Segmenter to predict a segmentation mask, from which we generate a set of mask proposals $\{M_i^{que}\}_{i=1}^{m}$ by finding the connected components, where m represents the number of proposals. Subsequently, a set of object-centric RGB images $\{I_i^{que}\}_{i=1}^{m}$ are cropped from the input image using the 2D bounding boxes derived from these mask pro-

posals. Next, we feed these RGB images into the proposal selector to obtain the final detection result. Within the proposal selector, we first extract the DINOv2 feature tokens $\{F_i^{que} \in \mathbb{R}^{L\times C}\}_{i=1}^{m}$ from these cropped images and then compute the image-level cosine similarities between these image features and the object semantic representation $\mathcal{F}^{obj}$. We select the one with the highest similarity score as the output, denoted as I^{que}.

Pose Estimator. In the second stage, we estimate an initial pose using a template retrieval-based pose estimator (see Fig. 4 middle). This pose estimator is comprised of an Obj-Segmenter (identical to the one in Detector), an RA-Encoder (as described in Sec. 3.1), and a pose decoder. We first obtain a segmentation mask M^{que} using Obj-Segmenter as well as an image-level representation vector $V^{que} \in \mathbb{R}^{64}$ using RA-Encoder from the detection. We then input M^{que} and V^{que} into the pose decoder to compute an initial 6D pose $P_{init} = [R_{init}, t_{init}]$. More specifically, the pose decoder first computes the cosine similarity scores $\{c_i = ||V^{que}|| \cdot ||V_i^{obj}||\}_{i=1}^{N_r}$ between the query vector V^{que} and the reference vectors within the set of 3D object rotation-aware representation $\{V_i^{obj}\}_{i=1}^{N_r}$. Consequently, the reference rotation matrix R_j^{ref} with the highest similarity score is retrieved as the initial 3D rotation estimate (R_{init}), where j denotes the index of the closest reference template. We then analytically infer the initial 3D translation t_{init} using the query mask (M^{que}) and the j^{th} reference mask (M_j^{ref}). The details of the 3D translation estimation are provided in the supplementary materials.

GS-Refiner. The initial pose is further refined using the 3D object representation $\mathcal{G}^{obj}$. GS-Refiner relies on a differentiable 3D Gaussian Splatting-based rendering [17], which allows us to refine the pose through a render-and-compare approach (see Fig. 4 bottom). Essentially, we iteratively optimize a learnable transformation T_{gs} to minimize the difference between the rendered object and the actual observation in the query image,

$$T_{gs} = \arg\min_{T_{gs}^*} \mathcal{L}_{gs}(\mathcal{R}_{gs}(T_{gs}^* \odot \mathcal{G}^{obj}, P_{init}), \bar{I}^{que}), \tag{3}$$

where $\mathcal{R}_{gs}$ denotes the rendering procedure, $T_{gs}^* \in SE(4)$ is the learnable transformation parameter, and $\odot$ indicates applying a rigid transformation to the 3D coordinates of $\mathcal{G}^{obj}$, and $\bar{I}^{que}$ denotes the segmented query image. The loss $\mathcal{L}_{gs}$ is defined as

$$\mathcal{L}_{gs} = \mathcal{L}_{D-SSIM} + \mathcal{L}_{D-MSSIM}, \tag{4}$$

where $\mathcal{L}_{D-SSIM}$ and $\mathcal{L}_{D-MSSIM}$ denote the SSIM-based and Multi-scale SSIM-based [51] losses, respectively. We initialize T_{gs}^* with an identity transformation (i.e., $R^* = \mathbb{I}$, $t^* = [0,0,0]^T$) and iteratively update it using the AdamW [28] solver with cosine annealing learning rate, starting from 5×10^{-3} and decaying to 1×10^{-6} over a maximum of 100 iteration steps. We also employ an early-stopping strategy based on loss convergence. The refined pose is finally obtained by $P = P_{init}T_{gs}$.

3.3 Training Objective Functions

We employ the widely adopted Binary Cross Entropy (BCE) loss to train both Co-Segmenter ($\mathcal{L}_{coseg}$) and Obj-Segmenter ($\mathcal{L}_{objseg}$) for pixel-wise segmentation prediction, *i.e.*,

$$\mathcal{L}_{coseg} = \mathcal{L}_{BCE}(\mathcal{M}, \bar{\mathcal{M}}), \quad \mathcal{L}_{objseg} = \mathcal{L}_{BCE}(M, \bar{M}), \tag{5}$$

where $\mathcal{M}$ and $\bar{\mathcal{M}}$ separately denote the predicted and ground truth group-wise segmentation masks, M and $\bar{M}$ are the predicted and ground-truth frame-wise segmentation masks, respectively. Additionally, we adopt the Negative Log-Likelihood (NLL) loss to train RA-Encoder for learning the 3D object rotation-aware representation, defined as:

$$\mathcal{L}_{rot} = -log \frac{exp(||V^{que}|| \cdot ||V_p||/\tau)}{\sum_{j=1}^{S_r} exp(||V^{que}|| \cdot ||V_j||/\tau)}, \tag{6}$$

where S_r is the number of the reference samples in a batch, V^{que} and V_j are the representation vectors of the query and the j^{th} reference samples, respectively, τ is the temperature, and p is the index of the positive training sample determined by measuring the geodesic distance of the 3D rotation matrices, calculated as:

$$p = \underset{0 \le j \le S_r}{\arg\min} \quad \arccos \frac{trace(R^{que} R_j^T) - 1}{2}, \tag{7}$$

where R^{que} is the ground truth rotation matrix of the query sample, and R_j is for the j^{th} training sample. Consequently, the entire framework is optimized through a combined loss in an end-to-end manner,

$$\mathcal{L}_{total} = \lambda_c \mathcal{L}_{coseg} + \lambda_o \mathcal{L}_{objseg} + \lambda_r \mathcal{L}_{rot}, \tag{8}$$

where $\lambda_{\{c,o,r\}}$ represent the balance weights.

4 Experiments

Datasets. We utilize the synthetic MegaPose dataset [19] for training and the real-world datasets LINEMOD [15] and OnePose-LowTexture [14] for evaluation. The MegaPose dataset was generated using BlendProc [10] and 1000 diverse objects from the Google Scanned Objects dataset [11] and includes one million synthetic RGB images. The LINEMOD dataset [15] contains 13 objects and is commonly used for 6D object pose evaluation. Following [14, 27, 34, 46], the training split of LINEMOD is selected as reference data, while the testing split is used for evaluation. OnePose-LowTexture [14] is a challenging dataset with low-texture or texture-less objects, from which eight scanned objects are utilized for evaluation. Each object was captured by two video sequences with different backgrounds. We follow OnePose++ [14] to select the first sequence as reference data and the other as query data for testing.

Table 1: Quantitative results of the 2D object detection on **LINEMOD [15]** in terms of the mAP@[0.5:0.95](%) metric. All methods include a subset of objects of LINEMOD for training, except ours, which is trained using the synthetic images from MegaPose [19]. We highlight the best in **Bold**.

Method	cat	duck	bvise	cam	driller	Avg.
SRPN-P [22]	11.85	1.62	18.94	2.44	8.91	8.76
SRPN [22]	9.72	4.56	22.47	13.43	10.97	12.23
SRPN-D [22]	22.97	1.85	49.14	17.76	18.89	22.12
OSOP [42]	32.10	34.81	26.68	24.33	21.36	27.86
Gen6D [27]	76.99	42.15	63.33	72.92	48.78	60.83
Cas6D [34]	79.46	67.44	66.32	76.39	59.35	69.79
LocPoseNet [54]	81.68	61.80	**79.45**	**80.50**	68.31	74.35
GS-Pose (Ours)	**84.10**	**82.85**	71.45	79.20	**79.41**	**79.40**

Baseline Methods. For comparison, we assess GS-Pose against several state-of-the-art methods: Gen6D [27], Cas6D [34], OnePose [46], and OnePose++ [14]. They take RGB reference images of novel objects with known poses as input to define the object coordinate system and then estimate the 6D pose of these objects from query images without retraining the network parameters.

Metrics. We adopt the widely used ADD [15] metric that measures the average distance between points after being transformed by the ground truth and predicted poses. The ADDS metric is used for symmetric objects, which measures the average distance to the closest point instead of the ground truth point. Following the protocol [15], we report the average recall rate of ADD(S) within 10% of the object diameter, denoted as **ADD(S)@0.1d**. We also compute the 2D projection errors of the points after being transformed by the ground truth and predicted poses. We report the average recall rate within 5 pixels, denoted as **Proj@5pix**. In addition, the mean Average Precision (**mAP**)[0.5:0.95](%) [26] is reported for evaluating the 2D object localization performance.

Experimental Configurations. We set $N_k = 8$, $\tau = 0.1$, $\lambda_c = 1$, $\lambda_r = 1$, $\lambda_o = 10$, $S_r = 32$ in our experiments. We use the AdamW [28] solver with the cosine annealing learning rate, starting from 1×10^{-4} to 1×10^{-6}, to train our framework for 100,000 steps on an Nvidia RTX3090 GPU with a batch size of 2.

4.1 Object Detection

Experiment Setups. Given a set of object-centric RGB images as reference data, the task is to localize the object in query images without fine-tuning the model parameters.

Results on LINEMOD. We first report the quantitative results of the 2D object detection on LINEMOD [15] regarding the mAP@[0.5:0.95](%) metric in Tab. 1. We primarily compare GS-Pose against Gen6D [27], Cas6D [34], and

Table 2: Quantitative results on the subset of objects in **LINEMOD [15]** regarding ADD(S)@0.1d. † indicates that the method is trained with another subset of objects in LINEMOD. We highlight the best in **Bold**.

Method	Pose Refiner	cat	duck	bvise	cam	driller	Avg.
Gen6D [27]†	✗	15.97	7.89	25.48	22.06	17.24	17.73
LocPoseNet [54]†	✗	-	-	-	-	-	27.27
OSOP [42]	✗	34.43	20.08	50.41	32.30	43.94	36.23
GS-Pose (Ours)	✗	**47.80**	**30.70**	**63.47**	**44.61**	**47.27**	**46.77**
OSOP [42]	OSOP [42]	42.54	22.16	55.59	36.21	49.57	42.21
Gen6D [27]	Vol-Refiner [27]	40.92	16.24	62.11	45.59	48.76	42.72
Gen6D [27]†	Vol-Refiner [27]	60.68	40.47	77.03	66.67	67.39	62.45
LocPoseNet [54]†	Vol-Refiner [27]	-	-	-	-	-	68.58
Cas6D [34]†	Cas-Refiner [34]	60.58	51.27	86.72	70.10	84.84	70.72
GS-Pose (Ours)	Vol-Refiner [27]	60.78	50.99	86.34	68.04	82.16	69.66
GS-Pose (Ours)	**GS-Refiner (Ours)**	**84.53**	**67.89**	**96.32**	**88.04**	**87.12**	**84.78**

LocPoseNet [54], which are the most similar works. Overall, GS-Pose achieves 79.40% mAP performance, surpassing all baseline approaches. It is worth noting that all methods utilize a subset of hold-out objects in LINEMOD for training, except ours (trained using the synthetic images [19]).

4.2 Object Pose Estimation

Experiment Setups. Given a set of reference RGB images of a novel object with known poses, the task is to estimate the 6D pose of the object in query images without fine-tuning the network parameters. We conduct experiments under two settings: 1) end-to-end pose estimation without requiring 2D bounding boxes, and 2) pose estimation with 2D bounding boxes. The latter involves estimating the pose from cropped object-centric images acquired either using the YOLOv5 detector [48], or by projecting the 3D object bounding boxes using ground truth object poses.

Results on LINEMOD. Following Gen6D [27], we first evaluate GS-Pose on a subset of objects in LINEMOD and report the quantitative results in Tab. 2. GS-Pose achieves 46.77% ADD(S)@0.1d without pose refinement and 84.78% 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 (46.77%). As an additional experiment, we also leverage the feature volume-based pose refiner (Vol-Refiner) proposed in Gen6D [27] for pose refinement. Vol-Refiner improves the initial accuracy to 69.66%, lagging behind 84.78% achieved with GS-Refiner.

Finally, we evaluate GS-Pose using all objects in LINEMOD [15] to compare with OnePose [46], OnePose++ [14], and MFOS [20]. Tab. 3 shows the

Table 3: Quantitative results on **LINEMOD [15]** regarding the ADD(S)@0.1d and Proj@5pix metrics. ✓ indicates using YOLOv5 [48] as the object detector and ✗ for using our built-in object detector. * indicates symmetric objects. We highlight the best in **bold**.

Method	YOLO	ape	bwise	cam	can	cat	driller	duck	ebox*	glue*	holep	iron	lamp	phone	Avg.
		ADD(S)@0.1d													
OnePose [46]	✓	11.8	92.6	88.1	77.2	47.9	74.5	34.2	71.3	37.5	54.9	89.2	87.6	60.6	63.6
OnePose++ [14]	✓	31.2	**97.3**	88.0	89.2	70.4	**92.5**	42.3	**99.7**	48.0	69.7	**97.4**	**97.8**	**76.0**	76.9
MFOS [20]	✓	47.2	73.5	87.5	85.7	80.2	92.4	60.8	99.6	69.7	**93.5**	82.4	95.8	51.0	78.4
GS-Pose (Ours)	✗	52.2	96.3	88.0	96.9	84.5	87.1	67.9	98.7	80.6	74.6	93.0	81.7	60.4	81.7
GS-Pose (Ours)	✓	**65.1**	95.7	**89.4**	**97.2**	**84.6**	90.7	**72.3**	99.2	**88.9**	78.6	91.7	94.0	70.8	**86.0**
		Proj@5pix													
OnePose [46]	✓	35.2	94.4	96.8	87.4	77.2	76.0	73.0	89.9	55.1	79.1	92.4	88.9	69.4	78.1
OnePose++ [14]	✓	97.3	**99.6**	**99.6**	**99.2**	98.7	**93.1**	**97.7**	**98.7**	51.8	**98.6**	**98.9**	**98.8**	**94.5**	94.3
GS-Pose (Ours)	✗	84.3	98.6	96.7	97.2	98.9	87.4	96.8	96.2	87.8	97.1	95.6	69.1	85.4	91.6
GS-Pose (Ours)	✓	**97.5**	98.5	99.0	97.6	**99.0**	91.9	97.6	96.9	**96.8**	98.2	96.8	90.0	91.1	**96.2**

quantitative results for the ADD(S)@0.1d and Proj@5pix metrics. Notably, all methods except ours necessitate 2D object bounding boxes, typically provided by YOLOv5 [48]. Overall, GS-Pose achieves an impressive 81.7% average performance, outperforming all baselines regarding the ADD(S)@0.1d metric. When using the 2D detection results predicted by YOLOv5 [48], as in OnePose [46] and OnePose++ [14], GS-Pose further improves the ADD(S)@0.1d metric to 86.0% and Proj@5pix to 96.2%, setting new state-of-the-art performance on LINEMOD. This advantage is largely attributed to the symmetric or textureless objects (*e.g.*, *ape, cat, duck, glue*), where the correspondence-based methods like OnePose [46], OnePose++ [14], and MFOS [20] inherently struggle.

Results on OnePose-LowTexture. We evaluate and compare GS-Pose against the baselines [14,27,46] on OnePose-LowTexture [14]. In addition, we also include PVNet [38], which trains a single network per object using approximately 5000 rendered images. Tab. 4 reports the quantitative results regarding ADD(S)@0.1d and shows a new state-of-the-art performance (89.5%) achieved by GS-Pose. The keypoint-based approach OnePose [46] obtains 59.7% average recall, which lags behind GS-Pose by ∼30%. OnePose relies on local feature matching to establish the keypoint-based 2D-3D correspondences, making it unreliable for low-texture or texture-less objects in this dataset. To alleviate this, OnePose++ [14] employs the keypoint-free LoFTR [45] for feature matching and significantly improves the result to 88.9%. Even though OnePose++ necessitates ground-truth 2D object bounding boxes for evaluation, GS-Pose still outperforms it using our built-in detector. Compared to the object-specific pose estimator PVNet [38], GS-Pose outperforms it by a substantial margin.

4.3 Additional Experiments

We conduct additional experiments on the subset of LINEMOD and report the results in Tab. 5 and Tab. 6.

Table 4: Quantitative results on **OnePose-LowTexture [14]**. "GT-BBox" indicates that 2D object bounding boxes are obtained by projecting the ground truth 3D object bounding boxes.

Method	GT-BBox	ADD@0.1d
PVNet [38]	✗	62.7
Gen6D [27]	✗	38.9
OnePose [46]	✓	59.7
OnePose++ [14]	✓	88.9
GS-Pose (Ours)	✗	**89.5**

Table 5: Ablation studies on **LINEMOD [15]**.

Variant	ADD(S)@0.1d
w/o Proposal Selector	81.13
w/o $\mathcal{L}_{D-SSIM}$	81.88
w/o $\mathcal{L}_{D-MSSIM}$	84.12
Full pipeline	**84.78**

Table 6: Results on **LINEMOD [15]** regarding the varying number of reference images.

Method	Number of reference images (N_r)				
	16	32	64	128	All (∼180)
Gen6D [27]	29.07	49.41	-	-	62.45
Cas6D [34]	32.43	53.90	-	-	70.72
OnePose++ [14]	31.38	54.98	-	-	78.10
GS-Pose (Ours)	**43.83**	**57.97**	73.68	82.46	**84.78**

Ablation studies. To assess the efficacy of the connected components-based proposal selector in object detection, we remove it from our object detector and then utilize the 2D bounding box derived from the entire segmentation mask as the output. As a result, the ADD(S)@0.1d metric decreases by 3.5%, indicating the effectiveness of the proposal selector. Besides, when either $\mathcal{L}_{D-SSIM}$ or $\mathcal{L}_{D-MSSIM}$ is removed from GS-Refiner, the performance decreases, indicating that both terms contribute positively to the pose refinement.

Number of reference images. GS-Pose consistently achieves the best performance in all settings regarding using different numbers of reference images. When using only 64 reference images, GS-Pose obtains 73.68%, already outperforming the results achieved by Gen6D [27] (62.45%) and Cas6D [34] (70.72%) using all reference images.

Runtime. GS-Pose costs about one second per image (with resolution 480×640) on a desktop with an AMD 835 Ryen 3970X CPU and an Nvidia RTX3090 GPU, in which ∼ 0.16s for object detection, ∼ 0.03s for initial pose estimation, and ∼ 0.8s for pose refinement. GS-Pose employs an iterative, gradient-based optimization process for pose refinement, which improves accuracy but at the cost of computational efficiency. In future work, we plan to explore more efficient optimization algorithms, such as the Levenberg-Marquardt algorithm, to accelerate the pose refinement process for GS-Pose.

Object Pose Tracking. We further leverage GS-Pose for 6D object pose tracking with minimal adjustments. The complete GS-Pose inference pipeline is first applied to estimate an initial pose of the object in the first frame of an RGB

video. This initial pose serves as the starting point for subsequent frames. For each subsequent frame, we leverage the 6D pose from the previous frame to crop an object-centric RGB image. We then utilize Obj-Segmenter to segment the object and GS-Refiner to refine this pose based on the segmentation. We refer to the supplementary materials for tracking examples and implementation details.

5 Discussion and Conclusion

This work presents GS-Pose, a novel end-to-end framework for estimating the 6D pose of novel objects in RGB images. GS-Pose exploits multiple representations of a novel object to facilitate cascaded sub-tasks, object detection, initial pose estimation, and pose refinement. GS-Pose is trained once using synthetic RGB images and then evaluated on two real-world datasets, LINEMOD and OnePose-LowTexture. The experimental results showcase that GS-Pose achieves state-of-the-art performance on the benchmark datasets and demonstrates decent generalization capabilities to new datasets. Moreover, GS-Pose can also be effortlessly adapted for robust 6D object pose tracking without significant adjustments. However, objects with slender or thin structures may pose challenges to GS-Pose due to their poor segmentation.

References

1. Cai, D., Heikkilä, J., Rahtu, E.: Ove6d: Object viewpoint encoding for depth-based 6d object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6803–6813 (2022) 3, 4
2. Cai, D., Heikkilä, J., Rahtu, E.: Sc6d: Symmetry-agnostic and correspondence-free 6d object pose estimation. In: 2022 International Conference on 3D Vision (3DV). pp. 536–546. IEEE (2022) 3
3. Cai, D., Heikkilä, J., Rahtu, E.: Msda: Monocular self-supervised domain adaptation for 6d object pose estimation. In: Scandinavian Conference on Image Analysis. pp. 467–481. Springer (2023) 3
4. Chen, D., Li, J., Wang, Z., Xu, K.: Learning canonical shape space for category-level 6d object pose and size estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11973–11982 (2020) 3
5. Chen, H., Wang, P., Wang, F., Tian, W., Xiong, L., Li, H.: Epro-pnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2781–2790 (2022) 3
6. Chen, W., Jia, X., Chang, H.J., Duan, J., Shen, L., Leonardis, A.: Fs-net: Fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1581–1590 (2021) 3
7. Chen, X., Dong, Z., Song, J., Geiger, A., Hilliges, O.: Category level object pose estimation via neural analysis-by-synthesis. In: European Conference on Computer Vision. pp. 139–156. Springer (2020) 3
8. Collet, A., Martinez, M., Srinivasa, S.S.: The moped framework: Object recognition and pose estimation for manipulation. The international journal of robotics research **30**(10), 1284–1306 (2011) 1
9. Deng, X., Xiang, Y., Mousavian, A., Eppner, C., Bretl, T., Fox, D.: Self-supervised 6d object pose estimation for robot manipulation. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 3665–3671. IEEE (2020) 1
10. Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., Katam, H.: Blenderproc. arXiv preprint arXiv:1911.01911 (2019) 10
11. Downs, L., Francis, A., Koenig, N., Kinman, B., Hickman, R.M., Reymann, K., McHugh, T.B., Vanhoucke, V.: Google scanned objects: A high-quality dataset of 3d scanned household items. 2022 International Conference on Robotics and Automation (ICRA) pp. 2553–2560 (2022) 10
12. Haugaard, R.L., Buch, A.G.: Surfemb: Dense and continuous correspondence distributions for object pose estimation with learnt surface embeddings. arXiv preprint arXiv:2111.13489 (2021) 3
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017) 4
14. He, X., Sun, J., Wang, Y., Huang, D., Bao, H., Zhou, X.: Onepose++: Keypoint-free one-shot object pose estimation without cad models. Advances in Neural Information Processing Systems **35**, 35103–35115 (2022) 2, 3, 4, 10, 11, 12, 13, 14, 21, 22
15. 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. pp. 548–562. Springer (2012) 3, 10, 11, 12, 13, 14, 21

16. Hodan, T., Barath, D., Matas, J.: Epos: Estimating 6d pose of objects with symmetries. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11703–11712 (2020) 3
17. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023) 3, 4, 6, 7, 9
18. Labbé, Y., Carpentier, J., Aubry, M., Sivic, J.: Cosypose: Consistent multi-view multi-object 6d pose estimation. In: European Conference on Computer Vision. pp. 574–591. Springer (2020) 3
19. Labbé, Y., Manuelli, L., Mousavian, A., Tyree, S., Birchfield, S., Tremblay, J., Carpentier, J., Aubry, M., Fox, D., Sivic, J.: Megapose: 6d pose estimation of novel objects via render & compare. arXiv preprint arXiv:2212.06870 (2022) 10, 11, 12
20. Lee, J., Cabon, Y., Brégier, R., Yoo, S., Revaud, J.: Mfos: Model-free & one-shot object pose estimation. arXiv preprint arXiv:2310.01897 (2023) 12, 13, 21
21. Lepetit, V., Moreno-Noguer, F., Fua, P.: Epnp: An accurate o (n) solution to the pnp problem. International journal of computer vision **81**(2), 155 (2009) 3
22. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8971–8980 (2018) 4, 11
23. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 683–698 (2018) 3
24. Li, Z., Wang, G., Ji, X.: Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7678–7687 (2019) 3
25. Lin, J., Liu, L., Lu, D., Jia, K.: Sam-6d: Segment anything model meets zero-shot 6d object pose estimation. arXiv preprint arXiv:2311.15707 (2023) 1, 4
26. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014) 11
27. Liu, Y., Wen, Y., Peng, S., Lin, C., Long, X., Komura, T., Wang, W.: Gen6d: Generalizable model-free 6-dof object pose estimation from rgb images. In: European Conference on Computer Vision. pp. 298–315. Springer (2022) 1, 2, 3, 4, 10, 11, 12, 13, 14, 22
28. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) 9, 11
29. Marchand, E., Uchiyama, H., Spindler, F.: Pose estimation for augmented reality: a hands-on survey. IEEE transactions on visualization and computer graphics **22**(12), 2633–2651 (2015) 1
30. Nguyen, V.N., Hu, Y., Xiao, Y., Salzmann, M., Lepetit, V.: Templates for 3d object pose estimation revisited: Generalization to new objects and robustness to occlusions. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 6771–6780 (2022) 4
31. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.Y., Xu, H., Sharma, V., Li, S.W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: Dinov2: Learning robust visual features without supervision (2023) 5

32. Örnek, E.P., Labbé, Y., Tekin, B., Ma, L., Keskin, C., Forster, C., Hodan, T.: Foundpose: Unseen object pose estimation with foundation features. arXiv preprint arXiv:2311.18809 (2023) 1, 4
33. Osokin, A., Sumin, D., Lomakin, V.: Os2d: One-stage one-shot object detection by matching anchor features. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16. pp. 635–652. Springer (2020) 4
34. Pan, P., Fan, Z., Feng, B.Y., Wang, P., Li, C., Wang, Z.: Learning to estimate 6dof pose from limited data: A few-shot, generalizable approach using rgb images. arXiv preprint arXiv:2306.07598 (2023) 3, 4, 10, 11, 12, 14
35. Park, K., Mousavian, A., Xiang, Y., Fox, D.: Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020) 2, 4
36. Park, K., Patten, T., Vincze, M.: Pix2pose: Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2019) 3
37. Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnet: Pixel-wise voting network for 6dof pose estimation. In: CVPR (2019) 3
38. Peng, S., Zhou, X., Liu, Y., Lin, H., Huang, Q., Bao, H.: Pvnet: pixel-wise voting network for 6dof object pose estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020) 13, 14, 22
39. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017) 5
40. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016) 4
41. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28**, 91–99 (2015) 4
42. Shugurov, I., Li, F., Busam, B., Ilic, S.: Osop: A multi-stage one shot object pose estimation framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6835–6844 (2022) 1, 2, 3, 4, 11, 12
43. Su, Y., Rambach, J., Minaskan, N., Lesur, P., Pagani, A., Stricker, D.: Deep multi-state object pose estimation for augmented reality assembly. In: 2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct). pp. 222–227. IEEE (2019) 1
44. Su, Y., Saleh, M., Fetzer, T., Rambach, J., Navab, N., Busam, B., Stricker, D., Tombari, F.: Zebrapose: Coarse to fine surface encoding for 6dof object pose estimation. arXiv preprint arXiv:2203.09418 (2022) 3
45. Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X.: Loftr: Detector-free local feature matching with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8922–8931 (2021) 13
46. Sun, J., Wang, Z., Zhang, S., He, X., Zhao, H., Zhang, G., Zhou, X.: Onepose: One-shot object pose estimation without cad models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6825–6834 (2022) 1, 2, 3, 4, 10, 11, 12, 13, 14, 21, 22
47. Sundermeyer, M., Marton, Z.C., Durner, M., Brucker, M., Triebel, R.: Implicit 3d orientation learning for 6d object detection from rgb images. In: The European Conference on Computer Vision (ECCV) (September 2018) 3

48. Ultralytics: Yolov5: Real-time object detection (2023), https://github.com/ultralytics/yolov5 12, 13, 21
49. Wang, G., Manhardt, F., Tombari, F., Ji, X.: GDR-Net: Geometry-guided direct regression network for monocular 6d object pose estimation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16611–16621 (June 2021) 3
50. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2642–2651 (2019) 3
51. Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003. vol. 2, pp. 1398–1402. Ieee (2003) 9
52. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In: Proceedings of Robotics: Science and Systems (RSS) (2018) 3
53. Xiao, Y., Qiu, X., Langlois, P.A., Aubry, M., Marlet, R.: Pose from shape: Deep pose estimation for arbitrary 3d objects. arXiv preprint arXiv:1906.05105 (2019) 3
54. Zhao, C., Hu, Y., Salzmann, M.: Locposenet: Robust location prior for unseen object pose estimation. arXiv preprint arXiv:2211.16290 (2022) 4, 11, 12
55. Zhao, Y., Guo, X., Lu, Y.: Semantic-aligned fusion transformer for one-shot object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7601–7611 (2022) 4

A Appendix

A.1 Preprocessing

In this part, we describe the image pre-processing step for creating the reference database. Given the reference data $\{\hat{I}_i^{ref}, K_i^{ref}, R_i^{ref}, t_i^{ref}\}_{i=1}^{N_r}$ of the target object, where $\hat{I}_i^{ref} \in \mathbb{R}^{H\times W\times 3}$ denotes the i^{th} RGB image along with the corresponding camera intrinsic K_i^{ref}, 3D rotation matrix R_i^{ref}, and 3D translation vector t_i^{ref}, we prepocess 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_i^{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_i^{ref} and then compute the scale factor by $S_i^{ref} = d_{obj} \cdot f_i^{ref}/\hat{tz}_i^{ref}$, where d_{obj} is the diameter of the 3D object bounding box, f_i^{ref} is the camera focal length, and $\hat{tz}_i^{ref} \in \mathbb{R}$ 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 the normalized image (I_i^{ref}) is assumed to be located along the camera optical axis $[0, 0, t_i^z]^T$, where $tz_i^{ref} = \hat{tz}_i^{ref} \cdot S_i^{ref}/S$ is the object z-axis translation after rescaling. We set $S = 224$ in all experiments.

A.2 Initial 3D Translation Estimation

We analytically infer the initial 3D translation t_{init} using the query mask $M^{que} \in \mathbb{R}^{S\times S}$ with the retrieved reference mask $M_j^{ref} \in \mathbb{R}^{S\times S}$, where j denotes the index of the reference template with the highest feature similarity score. Specifically, we calculate a relative scale factor $\delta_s \in \mathbb{R}$ and a relative 2D center offset ratio $\Delta_{xy} \in \mathbb{R}^2$ between M^{que} and M^{ref} as follows:

$$\delta_s = \sqrt{\frac{Area(M^{que})}{Area(M_j^{ref})}}, \quad \Delta_{xy} = \frac{C_{bbox}(M^{que}) - C_{bbox}(M_j^{ref})}{S}, \tag{9}$$

where $Area(M) = \sum_{j=0}^{S}\sum_{i=0}^{S} M_{[i,j]}$ denotes the mask area, and $C_{bbox}(M)$ denotes the 2D center of the bounding box tightly surrounding the mask M. We then compute the distance $t_z^{que} \in \mathbb{R}$ and the 2D center $P_{xy}^{que} \in \mathbb{R}^2$ of the object in the query image by

$$t_z^{que} = \frac{tz_j^{ref} \cdot S}{\delta_s \cdot S_{box}^{que}}, \quad P_{xy}^{que} = S_{box}^{que}\Delta_{xy} + C_{box}, \tag{10}$$

where tz_j^{ref} is the retrieved object reference distance, C_{box}^{que} and S_{box}^{que} are the 2D center and scale of the 2D object bounding box predicted from the input query

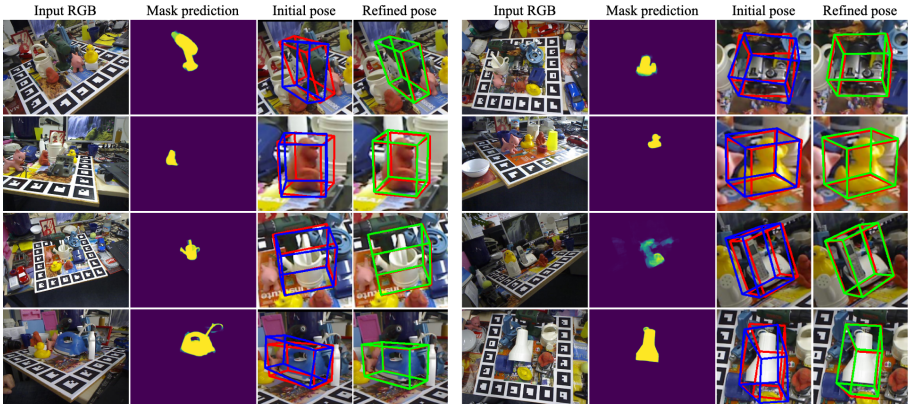

Fig. 5: Qualitative examples from LINEMOD. We present the intermediate segmentation predictions (for localization) as well as the estimated 6D poses. Blue, green, and red boxes represent initial, refined, and ground truth poses, respectively.

Table 7: Quantitative results on **LINEMOD [15]** regarding the ADD(S)@0.1d metric. ✓ indicates using YOLOv5 [48] as the object detector and ✗ for using our built-in object detector. * indicates symmetric objects. "$_{\text{init}}$" indicates the initial pose estimation results of GS-Pose. We highlight the best in **bold**.

Method	yolo	ape	bwise	cam	can	cat	driller	duck	ebox*	glue*	holep	iron	lamp	phone	Avg.
OnePose [46]	✓	11.8	92.6	88.1	77.2	47.9	74.5	34.2	71.3	37.5	54.9	89.2	87.6	60.6	63.6
OnePose++ [14]	✓	31.2	**97.3**	88.0	89.2	70.4	**92.5**	42.3	**99.7**	48.0	69.7	**97.4**	**97.8**	**76.0**	76.9
MFOS [20]	✓	47.2	73.5	87.5	85.7	80.2	92.4	60.8	99.6	69.7	**93.5**	82.4	95.8	51.0	78.4
GS-Pose $_{\text{init}}$	✗	32.7	63.5	44.6	58.4	47.8	47.3	30.7	94.6	66.9	48.0	53.5	36.7	33.2	50.6
GS-Pose $_{\text{init}}$	✓	39.2	58.0	48.3	65.6	53.7	52.0	39.3	93.5	76.3	52.0	53.7	46.0	39.8	55.2
GS-Pose (Ours)	✗	52.2	96.3	88.0	96.9	84.5	87.1	67.9	98.7	80.6	74.6	93.0	81.7	60.4	81.7
GS-Pose (Ours)	✓	**65.1**	95.7	**89.4**	**97.2**	**84.6**	90.7	**72.3**	99.2	**88.9**	78.6	91.7	94.0	70.8	**86.0**

image I. Consequently, we obtain the initial translation estimate by $t_{init} = t_z^{que} K^{-1} \bar{P}_{xy}^{que}$, where $\bar{P}_{xy}^{que} \in \mathbb{R}^3$ is the homogeneous form of P_{xy}^{que}, K is the camera intrinsic of the query image I.

A.3 Additional Results

We show qualitative examples from LINEMOD in Fig. 5, and report detailed quantitative results in Tab. 7 and Tab. 8, including the initial pose estimation results for each object.

A.4 6D Pose Tracking with GS-Pose

We extend the utilization of GS-Pose to facilitate 6D object pose tracking. The proposed tracking module, referred to as GS-Tracker, consists of two core components: Obj-Segmenter and GS-Refiner, depicted in Fig. 6. Given the current

Table 8: Quantitative results on each object in **OnePose-LowTexure [14]** regarding the ADD(S)@0.1d metric. "$_{\text{init}}$" indicates the initial pose estimation results of GS-Pose. We highlight the best in **bold**.

Method	Toyrobot	Teabox	Catmodel	Camera	Shiningscan	Moliere	David	Marseille	Avg.
PVNet [38]	12.3	90.0	68.1	67.6	95.6	57.3	49.6	61.3	62.7
Gen6D [27]	55.5	40.0	70.0	42.2	62.7	16.6	15.8	8.1	38.9
OnePose [46]	65.6	89.0	39.7	90.9	87.9	31.2	42.7	30.4	59.7
OnePose++ [14]	**89.5**	**99.1**	97.2	**92.6**	**98.5**	79.5	**97.2**	57.6	88.9
GS-Pose $_{\text{init}}$	51.8	75.8	74.1	65.3	92.8	68.4	65.4	57.0	68.8
GS-Pose (Ours)	88.3	87.1	**99.5**	86.0	98.4	**92.8**	86.8	**77.3**	**89.5**

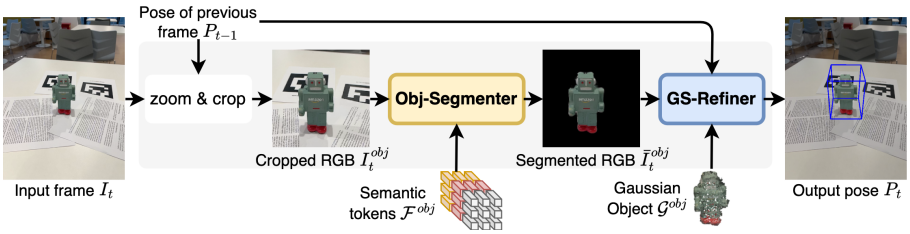

Fig. 6: Overview of GS-Tracker. GS-Tracker consists of two essential components: Obj-Segmenter and GS-Refiner. Given the current input frame $I_t \in \mathbb{R}^{H\times W\times 3}$ with resolution $H \times W$, where $t \geq 1$ denotes the current time step, we first utilize the pose estimate $P_{t-1} \in SE(4)$ of the previous frame to obtain a cropped object-centric image $I_t^{obj} \in \mathbb{R}^{S\times S\times 3}$. Subsequently, This object-centric image is fed into Obj-Segmenter along with the semantic representation tokens $\mathcal{F}^{obj}$ to produce a segmented image $\bar{I}_t^{obj}$. Finally, GS-Refiner takes the previous pose P_{t-1} and the segmented image $\bar{I}_t^{obj}$ along with the 3D Gaussian Object (3DGO) model $\mathcal{G}^{obj}$ as input and predicts a pose estimate P_t for the current frame.

frame $I_t \in \mathbb{R}^{H\times W\times 3}$ and the pose estimate $P_{t-1} \in SE(4)$ of the previous frame I_{t-1}, where $t \geq 1$ denotes the current time step, GS-Tracker follows a three-step process. First, it extracts an object-centric image $I_{t-1}^{obj} \in \mathbb{R}^{S\times S\times 3}$ from I_t using P_{t-1} (as described in Appendix A.1). Then, GS-Tracker leverages Obj-Segmenter to segment the object and subsequently utilizes GS-Refiner to update the pose P_{t-1} based on the segmented image $\bar{I}_t^{obj}$.

We initialize the process by employing GS-Pose to estimate the initial pose P_0 of the object in the first frame of the input RGB video, and then track the 6D object pose of the subsequent frames using GS-Tracker. GS-Tracker runs ~ 2.3 frame per second (FPS), and no re-initialization is required during tracking. Tracking examples from OnePose-LowTexture [14] are shown in Figs. 7 to 14.

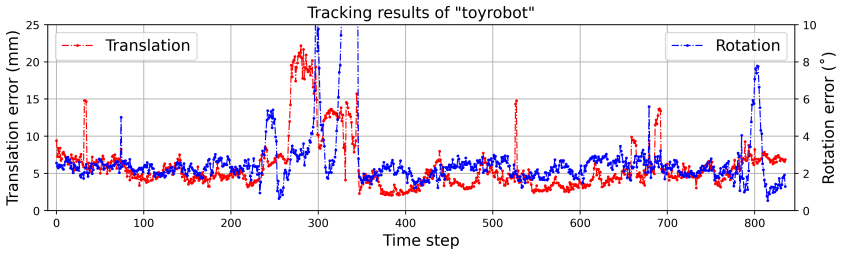

Fig. 7: Tracking results on the "Toyrobot" object. We plot the translation error (mm) and rotation error (°) of the tracking process along the time steps.

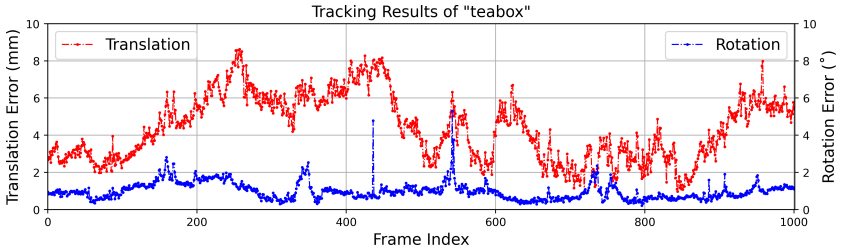

Fig. 8: Tracking results on the "Teabox" object. We plot the translation error (mm) and rotation error (°) of the tracking process along the time steps.

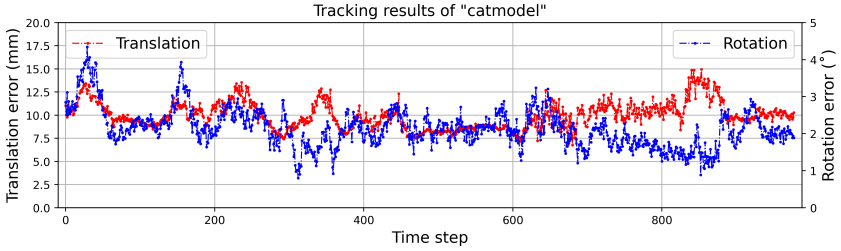

Fig. 9: Tracking results on the "Catmodel" object. We plot the translation error (mm) and rotation error (°) of the tracking process along the time steps.

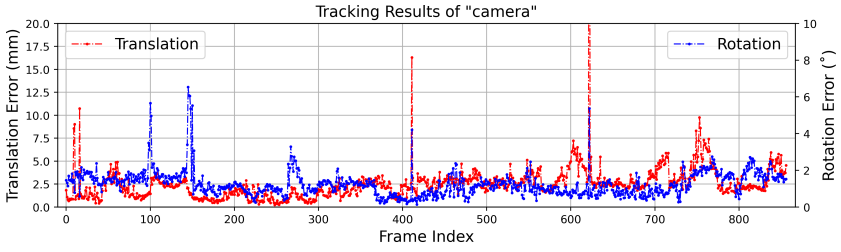

Fig. 10: Tracking results on the "Camera" object. We plot the translation error (mm) and rotation error (°) of the tracking process along the time steps.

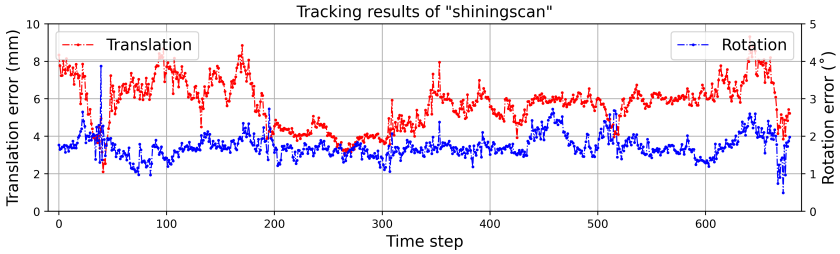

Fig. 11: Tracking results on the "Shiningscan" object. We plot the translation error (mm) and rotation error (°) of the tracking process along the time steps.

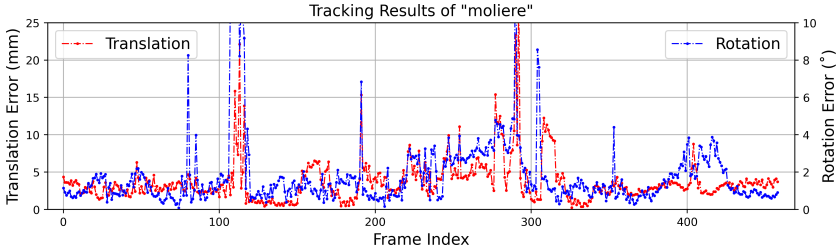

Fig. 12: Tracking results on the "Moliere" object. We plot the translation error (mm) and rotation error (°) of the tracking process along the time steps.

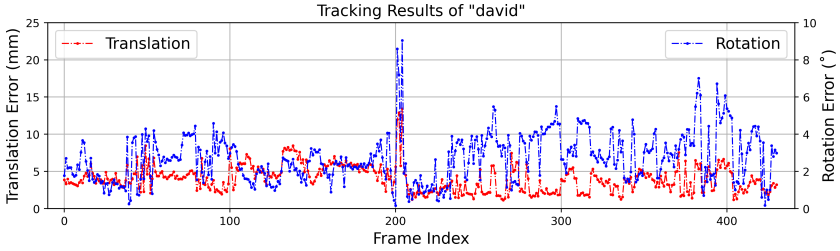

Fig. 13: Tracking results on the "David" object. We plot the translation error (mm) and rotation error (°) of the tracking process along the time steps.

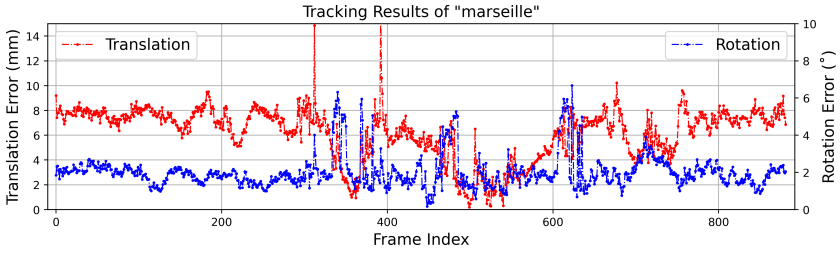

Fig. 14: Tracking results on the "Marseille" object. We plot the translation error (mm) and rotation error (°) of the tracking process along the time steps.