6DOPE-GS: Online 6D Object Pose Estimation using Gaussian Splatting

Yufeng Jin[1,2], Vignesh Prasad[1], Snehal Jauhri[1], Mathias Franzius[2], Georgia Chalvatzaki[1,3]
[1]Computer Science Department, Technische Universität Darmstadt, Germany
[2]Honda Research Institute Europe GmbH, Offenbach, Germany [3]Hessian.AI, Darmstadt, Germany
{yufeng.jin, vignesh.prasad, snehal.jauhri}@tu-darmstadt.de,
georgia.chalvatzaki@tu-darmstadt.de

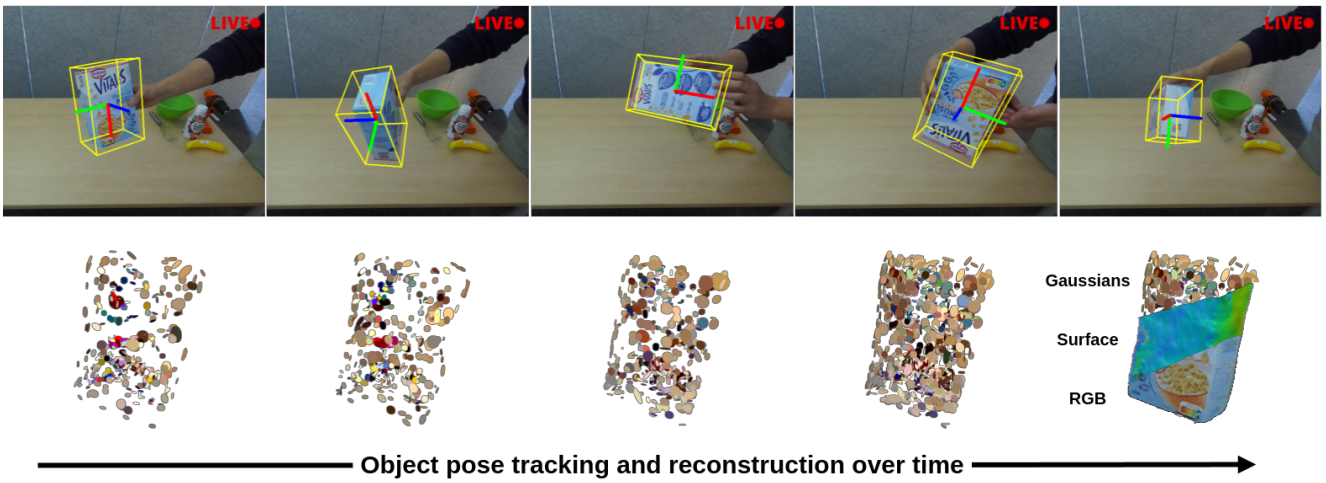

Figure 1. Demonstrating live object pose tracking and reconstruction of a test object in the real-world using 6DOPE-GS: a novel method for joint 6D object pose estimation and reconstruction using Gaussian Splatting. **Top:** 6D pose estimates of the object over time, **Bottom:** Example reconstruction over time with 2D Gaussian disks used to render the surface and appearance of the object. Our method enables live pose tracking and Gaussian Splat reconstruction of dynamic objects at 3.5Hz.

Abstract

Efficient and accurate object pose estimation is an essential component for modern vision systems in many applications such as Augmented Reality, autonomous driving, and robotics. While research in model-based 6D object pose estimation has delivered promising resu ts, model-free methods are hindered by the high computational load in rendering and inferring consistent poses of arbitrary objects in a live RGB-D video stream. To address this issue, we present 6DOPE-GS, a novel method for online 6D object pose estimation & tracking with a single RGB-D camera by effectively leveraging advances in Gaussian Splatting. Thanks to the fast differentiable rendering capabilities of Gaussian Splatting, 6DOPE-GS can simultaneously optimize for 6D object poses and 3D object reconstruction. To achieve the necessary efficiency and accuracy for live tracking, our method uses incremental 2D Gaussian Splatting with an intelligent dynamic keyframe selection procedure to achieve high spatial object coverage and prevent erroneous pose updates. We also propose an opacity statistic-based pruning mechanism for adaptive Gaussian density control, to ensure training stability and efficiency. We evaluate our method on the HO3D and YCBInEOAT datasets and show that 6DOPE-GS matches the performance of state-of-the-art baselines for model-free simultaneous 6D pose tracking and reconstruction while providing a 5× speedup. We also demonstrate the method's suitability for live, dynamic object tracking and reconstruction in a real-world setting.

1. Introduction

Computer vision systems for our 3D world are expected to advance beyond static scenes and structured multi-camera setups to more challenging real-world applications. Precise tracking and accurate reconstruction of objects allows capturing essential spatial and structural information, essential for downstream tasks such as robotic manipulation [10, 56],

augmented reality [57, 78], automation [15, 26], etc.

The majority of 6D object pose estimation and tracking methods, whether for seen or unseen objects, have primarily used model-based techniques. Several approaches [28, 30, 41, 81] use CAD models rendered from various angles during training and perform feature matching at inference time for rapid pose estimation. FoundationPose [72] leverages synthetic training data to perform state-of-the-art instance-level pose estimation using either a CAD model or a small set of reference images annotated with the object poses. Notably, there has been exciting progress in zero-shot, model-free methods over the past few years [69, 71]. BundleSDF [71] operates in a model-free manner by jointly optimizing a "neural object field" and the object poses by learning a 3D Signed Distance Field representation while concurrently running a global pose graph optimization. However, despite BundleSDF's reported near real-time pose optimization capabilities ($\sim$10Hz), the neural object field training is far from real-time, which limits the average tracking frequency to $\sim$0.4Hz. The significant computational overhead associated with training the neural object field hinders its applicability in live dynamic scenarios, where rapid pose updates are crucial.

To address this limitation, we leverage Gaussian Splatting [22, 25] which offers significantly better computational efficiency for real-time applications. We propose a novel method for online 6D object pose estimation through Gaussian Splatting, "6DOPE-GS", that enables model-free, live object tracking and reconstruction. Building upon recent advances in using Gaussian Splatting for SLAM [24], 6DOPE-GS jointly optimizes object poses from observed keyframes and reconstructs a 3D object model on the fly using incremental 2D Gaussian Splatting [22]. We propose several algorithmic enhancements to attain the required accuracy, efficiency, and training stability for live reconstruction and tracking. For accuracy, our method uses a novel dynamic keyframe selection mechanism to prioritize spatial coverage of the object and reconstruction confidence-based filtering to exclude keyframes with erroneous pose estimates. To maintain training stability and efficiency, we propose an adaptive Gaussian density control mechanism based on the opacity statistics of the Gaussians. Our contributions provide a significant speed-up in object pose estimation and tracking while maintaining high accuracy. In particular, we evaluate 6DOPE-GS on the HO3D and YCBInEOAT datasets and observe that it matches the state-of-the-art performance of competitive baselines while providing a 5$\times$ speedup. We also demonstrate the live, dynamic object tracking and reconstruction ability of 6DOPE-GS in a real-world setting. To the best of our knowledge, we are the first method to perform joint object tracking and Gaussian Splat reconstruction live at 3.5Hz from a single RGB-D camera.

Our contributions are as follows:

- We propose a novel method that effectively leverages 2D Gaussian Splatting for efficient and accurate model-free 6D object pose estimation and reconstruction.
- We leverage the computationally efficient differentiable rendering of Gaussian Splatting to jointly optimize a 2D Gaussian Splatting-based "Gaussian Object Field" along with an object-centric pose graph of observed keyframes, that provides accurate, refined keyframe pose updates.
- We propose a dynamic keyframe selection approach based on the spatial coverage of the set of keyframes and a reconstruction confidence-based filtering mechanism to exclude keyframes with erroneous pose estimates.
- We incorporate a novel adaptive Gaussian density control mechanism based on opacity percentiles to filter out "unimportant" Gaussians, thereby improving training stability and computational efficiency.

2. Related Work

2.1. Object Pose Estimation and Tracking

Instance-level 6D object pose estimation typically requires object CAD models and/or pretraining [4, 18, 19, 27, 30, 48, 63, 64, 66, 70]. Such instance-level methods can be further categorized into correspondence-based [47, 49, 62], template-based [9, 60], voting-based [18, 19, 31, 41, 64], and regression-based [14, 21] methods. For better generalization, some approaches use an object CAD model only at inference time [28, 54, 81]. Other methods [3, 17, 20, 32, 45, 59, 72] relax this assumption by utilizing only few reference images of the object instead of the CAD model.

BundleTrack [69] enables near real-time ($\sim$10Hz), model-free tracking with a SLAM-style approach. It uses keyframe point correspondences for coarse pose initialization with RANSAC, followed by object-centric pose graph optimization for refined estimates. BundleSDF [71] extends this by jointly performing pose tracking and object reconstruction through a neural object field, achieving state-of-the-art results in model-free settings. However, the neural field training is slow and computationally demanding ($\sim$6.7s per training round [71]), limiting its real-time applicability. We address this key limitation by leveraging the efficiency of Gaussian Splatting for joint object reconstruction and pose refinement, enabling effective live tracking.

2.2. 3D Reconstruction

3D Reconstruction is a well-studied problem in Photogrammetry. Structure from Motion (SfM) [43] is a commonly used approach to estimate camera poses and a sparse 3D structure from images without prior pose knowledge in an offline manner. Multi-View Stereo approaches (MVS) [13, 68] build upon such pose estimates to refine a dense 3D reconstruction. For enabling more real-time reconstruction and pose tracking, Simultaneous Localization and Map-

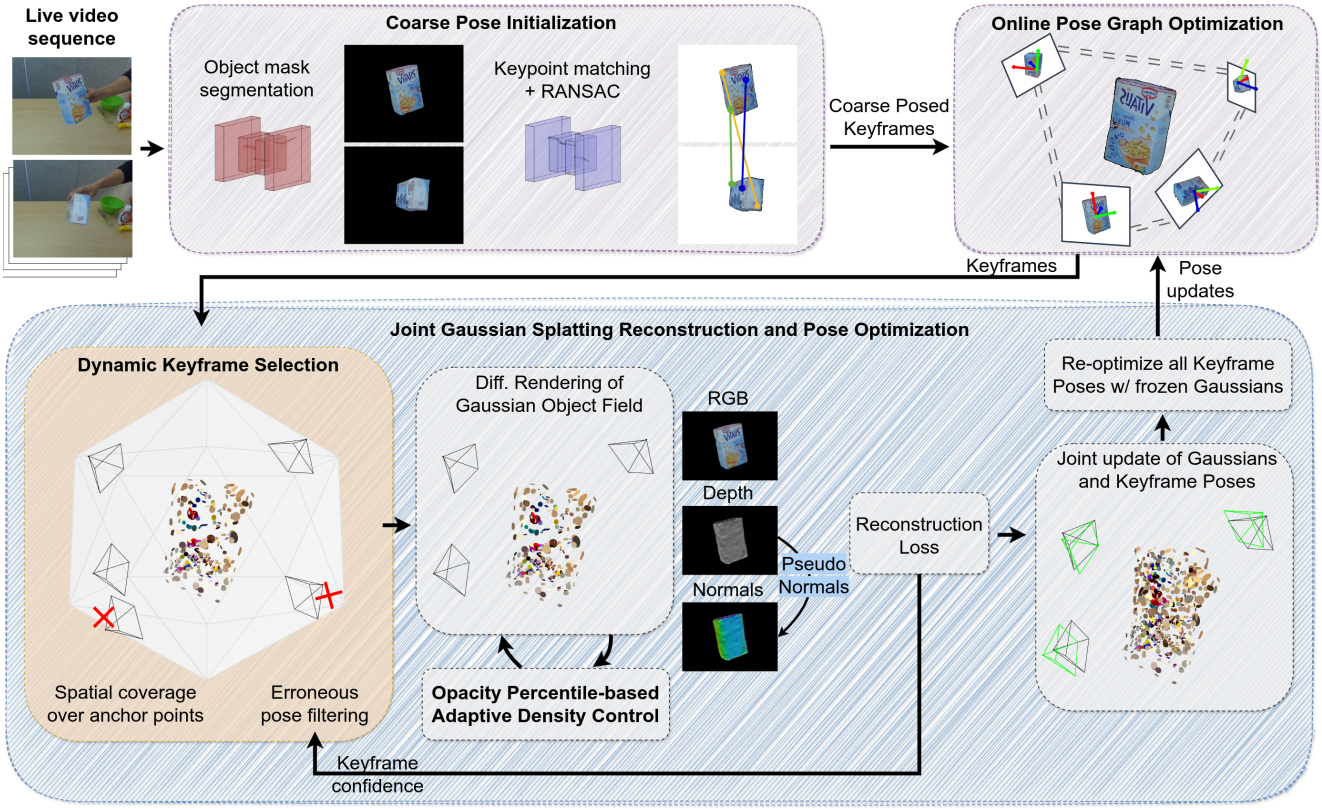

Figure 2. Overview of our approach: 6DOPE-GS. Given a live input RGB-D video stream, we obtain object segmentation masks using SAM2 [50] on the incoming video frames. We then use LoFTR [58], a transformer-based feature matching approach, to obtain pairwise correspondences between multiple views. We initialize a set of "keyframes" based on the density of matched features, for which we establish initial coarse pose estimates using RANSAC. To obtain refined pose updates for the keyframes, we use a 2D Gaussian Splatting-based "Gaussian Object Field" that is jointly optimized with the keyframe poses in a concurrent thread. We filter out erroneous keyframes for accurate pose refinement updates using a novel dynamic keyframe selection mechanism based on spatial coverage and reconstruction confidence. Moreover, we incorporate an opacity percentile-based adaptive density control mechanism to prune out inconsequential Gaussians, thus improving training stability and efficiency. Once the Gaussian Object Field is updated, it is temporarily frozen and the poses of keyframes that were filtered out are also updated. The object pose estimate at each timestep is then obtained by performing an online pose graph optimization using the incoming keyframe with the current set of keyframes.

ping (SLAM) methods [5, 23, 34] approach the problem by jointly optimizing the camera poses and the environment reconstruction. Emerging methods that leverage neural representations have enhanced the fidelity of 3D reconstructions [51, 65, 67, 77]. Along similar lines, the use of Neural Radiance Fields (NeRFs) [38] and Signed Distance Fields (SDFs) [7, 37, 42, 44], with their volumetric rendering approach provide highly photorealistic reconstructions.

Gaussian Splatting [25] is a particle-based alternative that models scene density with Gaussian distributions and achieves significantly faster rendering speeds with similar levels of photorealism by using rasterization of explicit Gaussian particles, thereby avoiding the ray-marching steps used by volumetric rendering methods. Recently, 2D Gaussian Splatting [22] has improved the surface rendering capabilities of Gaussian Splatting by optimizing oriented planar 2D Gaussian disks close to the scene surfaces. However, all these methods still depend on pre-established camera poses. Coming from a SLAM perspective, recent approaches explore jointly optimizing camera pose estimates as well as the map reconstruction that uses Gaussian Splatting [24, 33, 36, 75]. In this work, we propose a novel method that extends scene-level approaches to object-level tracking and reconstruction by leveraging the SLAM-inspired capabilities for object tracking [69, 71] and Gaussian Splatting [24, 36, 75] with the precise surface rendering capabilities offered by 2D Gaussian Splatting [22].

3. Method

We introduce a novel method for real-time 6D Object Pose Estimation using the representation capabilities of 2D Gaussian Splatting. Fig. 2 presents a schematic overview

of our approach. To accurately track the 6DoF pose of an object captured by a single RGB-D camera, we start by segmenting the object in the first frame using SAM2 [50] to ensure precise object segmentation throughout the video sequence. With the object segmented across multiple frames, we apply LoFTR [58] to establish point correspondences, identifying keyframes for a Coarse Pose Initialization via Bundle Adjustment [69] (Sec. 3.1). This initial set of coarsely estimated keyframes is then refined through a joint optimization with 2D Gaussians using differentiable rendering, yielding accurate pose corrections and an improved object model for the keyframes (Sec. 3.2). To improve the quality of the generated 3D model and to subsequently enable a more accurate pose refinement, we propose a dynamic keyframe selection technique for selecting the best keyframes for optimizing the 2D Gaussians based on their estimated spatial coverage around the object and their reconstruction accuracy (Sec. 3.3). During this phase, we iteratively employ a novel pruning/adaptive density control mechanism to stabilize the number of Gaussian particles required, to balance computational efficiency with reconstruction accuracy (Sec. 3.4). Once the joint optimization converges, all the keyframe poses are subsequently optimized and help guide the Online Pose Graph Optimization (Sec. 3.5) in continuously refining the object pose at each subsequent timestep for robust and precise tracking.

3.1. Coarse Pose Initialization

To enable real-time 6D pose tracking and reconstruction of arbitrary objects, we first use SAM2 [50] for facilitating effective segmentation and tracking of the object in question. Specifically, we use a fixed-length window of past frames combined with prompted images as input. We then use LoFTR [58], a transformer-based dense feature matcher, to estimate feature point correspondences between neighboring images. Using these matches, we compute a coarse pose estimate between pairs of RGB-D frames with nonlinear least-squares optimization [1] in a RANSAC fashion [12]. Subsequently, a keyframe memory pool is created wherein if an incoming frame is deemed to be spatially diverse compared to the existing pool, it is added as a new keyframe. Further details regarding the feature matching and the keyframe memory pool initialization are in [71].

3.2. Gaussian Object Field

To build an internal model that captures the visual and geometric properties of the object in an efficient and accurate manner, we build a Gaussian Object Field using 2D Gaussian Splatting (2DGS) [22] to achieve precise surface geometry reconstruction. Unlike 3D Gaussian Splatting (3DGS) [25], which primarily emphasizes on redering realistic visual effects, 2DGS ensures accurate geometric alignment by converting each Gaussian into a disk-like surfel. This surfel-based approach, combined with our novel dynamic keyframe selection (Sec. 3.3) and opacity quartile-based pruning (Sec. 3.4), enables 2DGS to precisely model the rendered surface, thereby delivering reliable depth estimates and addressing the limitations observed in 3DGS.

In 3DGS, a scene is represented as a set of 3D Gaussian particles, each of which represents a 3D distribution and is defined by its 3D centroid (mean) $\boldsymbol{\mu} \in \mathbb{R}^3$ and a covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ which can be decomposed into a diagonalized scaling matrix $\boldsymbol{S} = diag([s_x, s_y, s_z])$ and a rotation matrix $\boldsymbol{R} \in SO(3)$ as $\boldsymbol{\Sigma} = \boldsymbol{RSS}^\top\boldsymbol{R}^\top$, which denotes the volume (spread) of the particle in 3D space. Along with the mean and covariance, each Gaussian is further characterized by spherical harmonic coefficients $c \in \mathbb{R}^k$ to represent view-dependent appearance, and an opacity value $\alpha \in [0, 1]$. For rendering, each 3D Gaussian is converted to camera coordinates using the world-to-camera transformation matrix $\boldsymbol{W}$ and mapped to the image plane via a local affine transformation $\boldsymbol{J}$, $\boldsymbol{\Sigma}' = \boldsymbol{JW\Sigma W}^\top\boldsymbol{J}^\top$. Once the 3D Gaussian is "splatted" onto the image plane, excluding the third row and column of $\boldsymbol{\Sigma}'$ results in a 2D covariance matrix $\boldsymbol{\Sigma}^{2D}$ that represents a 2D Gaussian G^{2D} in the image plane. The Gaussians are first ordered in ascending order based on their distance to the camera origin. Using volumetric rendering, we calculate the per-pixel color estimates $\hat{c}(\boldsymbol{p})$ of a pixel $\boldsymbol{p} = [u, v]^T$ as the α-blending of N ordered Gaussians from front to back along the view direction

$$\hat{c}(\boldsymbol{p}) = \sum_{i\in N} c_i\alpha_i G_i^{2D}(\boldsymbol{p}) \prod_{j=1}^{i-1}(1 - \alpha_j G_j^{2D}(\boldsymbol{p})), \quad (1)$$

where α_i and c_i denote the opacity and the view-dependent appearance of the ith Gaussian, respectively. The depth image can be similarly rendered by replacing c_i with the z-depth coordinate of the ith Gaussian in the camera frame.

For 2DGS [22], the z-component of the scaling matrix is set to zero $\boldsymbol{S} = diag([s_u, s_v, 0])$ for each Gaussian, thereby collapsing the 3D volume into a set of 2D oriented planar Gaussian disks with two principal axes $\boldsymbol{t}_u$ and $\boldsymbol{t}_v$. The normal to the 2D Gaussian can then be defined as $\boldsymbol{t}_w = \boldsymbol{t}_u \times \boldsymbol{t}_v$, which allows us to define the rotation matrix for the Gaussian particle as $\boldsymbol{R} = [\boldsymbol{t}_u, \boldsymbol{t}_v, \boldsymbol{t}_w]$. Moreover, along with photometric reconstruction, 2DGS additionally incorporates depth distortion and normal consistency to further enhance the quality of the reconstructions. For further details regarding 2DGS, please refer to [22].

In our approach, along with optimizing the parameters of each 2D Gaussian, we aim to jointly refine the keyframe poses as well. We do so by propagating the gradients of losses through the projection operation of the 2D Gaussians onto the image plane of each keyframe, as done in [24, 36, 75]. We use automatic differentiation via PyTorch [46] for calculating the gradients and updating the keyframe poses. Further details are in Appendix.

3.3. Dynamic Keyframe Selection for Gaussian Splatting Optimization

Once we obtain a coarse pose initialization of keyframes, we aim to construct a 2DGS model of the object. However, errors in the pose initialization can cause a divergence in the Gaussian Splatting optimization. Unlike BundleSDF's ray-casting method [71], which renders individual pixels, Gaussian Splatting uses tile-based rasterization, rendering entire images one at a time, thereby increasing the computational cost linearly as the number of keyframes increases. To mitigate these issues, we introduce a dynamic keyframe selection approach to filter out erroneous keyframes.

To acquire a reliable Gaussian Object Field, we strategically sparsely select keyframes to optimize keyframe poses and object Gaussians. We first establish a series of "anchor points" at varying resolution levels, using the vertices and face centers of an icosahedron (as shown in Fig. 2-bottom left) to approximate evenly distributed points on a sphere centered on the object [53]. We then cluster the initial coarse keyframe pose estimates around these anchor points along the icosahedron. To maximize information from all viewpoints around the object, we select the keyframe with the largest object mask in the cluster of each anchor point, effectively training under sparse-view conditions with the aid of depth information [29]. This ensures that we minimize instances where the object is largely occluded and consider views where we have better visibility of the object.

While jointly optimizing the 2D Gaussians and the selected keyframe poses, we further remove outliers with erroneous pose estimates based on the reconstruction error obtained during the 2D Gaussian optimization. This approach is necessary because reconstruction residuals can impede pose optimization during the joint optimization of 2D Gaussians and keyframe poses. Specifically, we estimate the median absolute deviation (MAD) of the reconstruction loss at each iteration, which represents the typical "spread" around the median value, to identify and remove outlier views. The rationale for using MAD lies in its robustness; as a median-based metric, MAD is less influenced by extreme values than other measures, such as the mean or standard deviation, making it more reliable in the presence of outliers. Views with absolute deviations exceeding three times the MAD are classified as outliers.

3.4. Opacity Percentile-based Adaptive Density Control

During the optimization of the Gaussian Object Field, we perform periodic pruning and densification to maintain both the number and compactness of Gaussians. However, the vanilla Adaptive Density Control proposed in 3DGS has several limitations [2], since it demands considerable engineering work to adjust densification intervals and fine-tune opacity thresholds to stabilize training. Prior work [8] demonstrates that a gradual iterative pruning strategy can yield significantly sparser models while preserving high fidelity. Similarly, Fan et al. [11] propose an importance weighting based on the scale percentile and the opacity of the Gaussians. However, they mainly focus on efficient compression of Gaussians. Inspired by [11], and since we have an object-centric focus, we limit the scale of the Gaussians and instead use a percentile-based pruning strategy based on opacity for stabilizing the number of Gaussians.

After a fixed number of optimization steps, we prune the Gaussians with opacity in the bottom 5th percentile until the opacity of the 95th percentile of the Gaussian particles exceeds a given threshold. This allows us to ensure that during the forward rendering (Eq. 1), a good number of high-quality Gaussian particles remain and those that are more inconsequential get pruned out. We empirically verify that our approach compared to naive absolute thresholding [25], improves the performance of our method. We further trigger splitting and cloning of the Gaussian particles when the positional gradient exceeds a predefined threshold, similar to [25]. Notably, the variation of the positional gradient remains relatively stable and does not continuously increase during training. Once the optimization of the Gaussian Object Field converges, the poses of all the keyframes are refined using the reconstruction of the RGB, depth, and normals, by temporarily freezing the 2D Gaussians.

3.5. Online Pose Graph Optimization

When we receive the updated poses for the keyframes from the Gaussian Object Field, we establish a global object-centric coordinate system and a keyframe memory pool, which stores key correspondences. To balance computational efficiency with memory usage and reduce long-term tracking drift when a new frame is observed, a set of overlapping frames from the memory pool is selected for graph optimization based on the view frustum of the incoming frame. For each frame in this pool, we generate a point-normal map and compute the dot-product of the normals with the camera-ray direction of the new frame, to assess visibility. Frames are selected if the visibility ratio of the new incoming frame exceeds a defined threshold. We choose the best keyframes from the pool to construct the pose graph along with the incoming frame. We optimize the pose graph using pairwise geometric consistency by minimizing a dense pixel-wise re-projection error as in [69].

4. Experiments

4.1. Datasets

4.1.1. YCBInEOAT Dataset

The YCBInEOAT dataset [73] offers ego-centric RGB-D video recordings of a dual-arm Yaskawa Motoman SDA10f robot manipulating YCB objects. Using an Azure Kinect

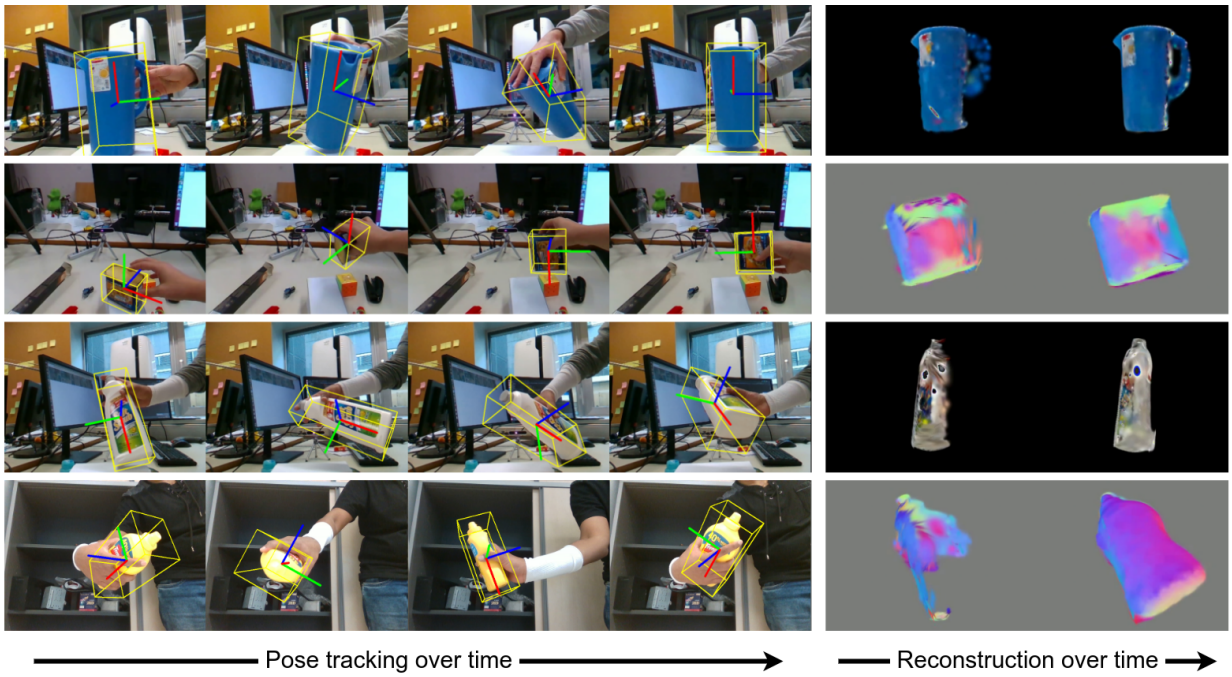

Figure 3. Qualitative results of our method, 6DOPE-GS, tested on video sequences from the HO3D dataset, namely AP13, MPM14, SB13, and SM1 (from top to bottom). **Left:** Our method tracks the 6D object pose over time with high accuracy, **Right:** 6DOPE-GS is effective at reconstructing both the appearance (rows 1 and 3) and surface geometry (rows 2 and 4) of the object over time. The first image shows the initial reconstruction at the beginning of the sequence, the second image shows the refined reconstruction over time.

camera positioned at mid-range, the dataset captures three types of manipulation tasks: single-arm pick-and-place, within-hand manipulation, and handoff between arms. In total, it includes 5 YCB objects [74] across 9 video sequences, amounting to 7,449 frames. Each frame is annotated with accurate 6D object poses, calibrated with the camera's extrinsic parameters.

4.1.2. HO3D Dataset

The HO3D dataset [16] features 27 multi-view (68 single-view) sequences showing hand-object interactions involving 10 YCB objects [74] with annotated 3D poses. The RGB-D video data, captured at close range with an Intel RealSense camera, provides detailed records of hand manipulations of objects. Ground-truth 3D poses are generated via multi-view registration, facilitating evaluations of pose, reconstruction, and texture accuracy. We use the latest version, HO3D_v3, and conduct evaluations on the official test set, which comprises 4 objects and 13 sequences.

4.2. Metrics & Baselines

We assess pose estimation accuracy using the area under the curve percentage for the ADD and ADD-S (ADD-Symmetric) metrics [20, 74]. While these are typically computed using the object model, similarly to [69, 71], we use the ground-truth pose of the object in the first frame for calculating the metrics. For object reconstruction, we measure the Chamfer distance between the reconstructed and ground-truth object meshes in object coordinates. We compare our approach against other fully model-free RGB-D baselines, namely BundleTrack [69] and BundleSDF [71], that we ran using their open-source implementations[1]. We additionally report the performance of other approaches mentioned in the leaderboard of [71]. We use the same object masks as input to all methods for a fair comparison.

4.3. Results

As seen in Tables 1 and 2, we outperform previous SLAM-based approaches as well as BundleTrack [69]. Since the sequences in YCBInEOAT (Table 1) have smoother motions with less diverse viewpoints, most approaches fare equally well as we do not have large occlusions or jumps which can cause erroneous coarse pose initialization. However, we achieve better reconstruction at a sub-centimeter level compared to other approaches on the YCBInEOAT dataset. Some examples of the pose estimation and reconstruction are shown in Fig. 3. It can be seen that 6DOPE-GS achieves accurate pose tracking and accurately reconstructs the surface appearance and geometry as more of the object becomes visible over time.

[1] https://github.com/NVlabs/BundleSDF

Method	Pose		Reconstruction
	ADD-S (%) ↑	ADD (%) ↑	CD (cm) ↓
NICE-SLAM [80]	23.41	12.70	6.13
SDF-2-SDF [55]	28.20	14.04	2.61
DROID-SLAM [61]	46.39	34.68	4.63
MaskFusion [52]	41.88	35.07	2.34
BundleTrack [69]	93.01	87.26	2.81
BundleTrack*	92.54	84.91	-
BundleSDF [71]	93.77	86.95	1.16
BundleSDF*	92.82	84.28	1.23
6DOPE-GS	**93.79**	**87.82**	**0.83**

Table 1. Comparison on the YCBInEOAT Dataset. ADD and ADD-S are reported as AUC percentages (0 to 0.1m), and reconstruction accuracy is measured by Chamfer loss. All baseline results are the officially reported ones, except for the ones marked with an asterisk (*), which we reproduced from the original implementation available on GitHub by the authors.

Method	Pose		Reconstruction
	ADD-S (%) ↑	ADD (%) ↑	CD (cm) ↓
NICE-SLAM [80]	22.29	8.97	52.57
SDF-2-SDF [55]	35.88	16.08	9.65
KinectFusion [40]	25.81	16.54	15.49
DROID-SLAM [61]	64.64	33.36	30.84
BundleTrack [69]	92.39	66.01	52.05
BundleTrack*	93.96	77.75	-
BundleSDF [71]	**96.52**	**92.62**	0.57
BundleSDF*	94.86	89.56	0.58
6DOPE-GS	95.07	84.33	**0.43**

Table 2. Comparison on the HO3D Dataset. ADD and ADD-S are reported as AUC percentages (0 to 0.1m), and reconstruction accuracy is measured by Chamfer loss. All baseline results are the officially reported ones, except for the ones marked with an asterisk (*), which we reproduced from the original implementation available on GitHub by the authors.

The HO3D dataset is more challenging, as there are more occlusions from human hands as well as a large range of rotational motions which cause ambiguities in the coarse pose initialization. Although we perform better than BundleSDF [71] in terms of the ADD-symmetric score, we do not compare favorably in terms of the absolute score.

This shortcoming mainly stems from the different rendering pipelines, and subsequently the gradient updates for the pose estimation. The ray-tracing-based approach of learning the neural object field via an SDF in BundleSDF enables sampling the rays in an i.i.d. manner that is more conducive for optimization. The differentiable rendering in Gaussian Splatting, on the other hand, requires the entire image to be rendered and subsequently optimized for each keyframe pose individually. While in theory, this limitation can be improved with a mini-batch-styled processing, doing so, would increase both the computational time and memory usage, potentially causing issues in the gradient accumulation due to the large number of Gaussian particles.

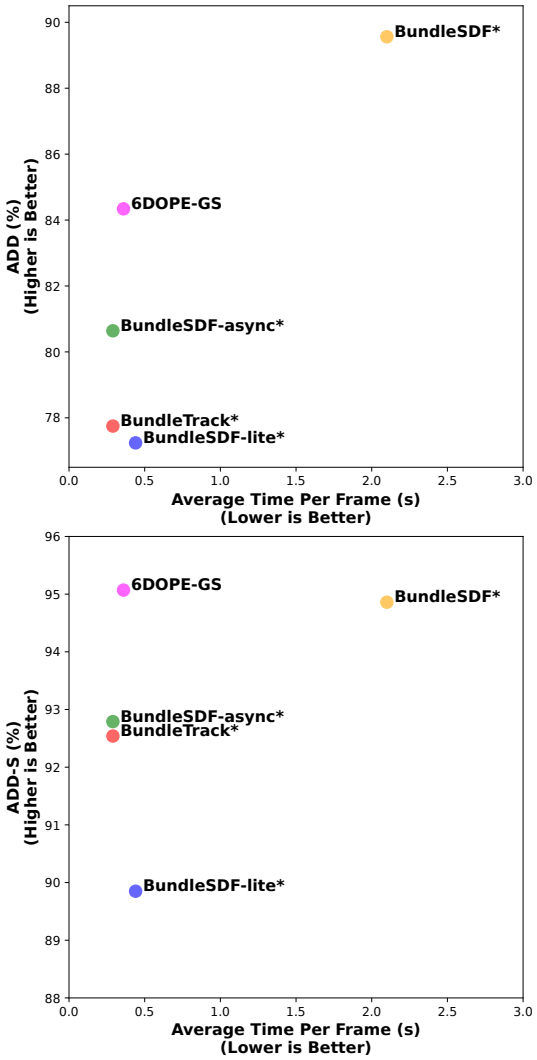

Figure 4. Comparison between speed and performance for different approaches on the HO3D dataset. While BundleSDF achieves high performance, it comes at the cost of speed. On the other hand, 6DOPE-GS achieves a favorable tradeoff between speed and performance. (*) implies that results are based on the implementation available on GitHub by the authors.

4.4. Temporal Efficiency

To evaluate the temporal efficiency of different approaches, we compare the tradeoff between the performance and the average processing time per frame for the different approaches on the HO3D dataset. We test the approaches on a desktop with a 12th Gen Intel(R) Core(TM) i9-12900KF CPU, 64GB RAM, equipped with an NVIDIA GeForce RTX 4090 GPU. We explore the performance of two more versions of BundleSDF [71] that reduce the processing time. In the first version, named "BundleSDF-async", we deactivate the synchronization between the threads performing neural object field learning and the online pose

graph optimization. Although this approach introduces potential variability in pose accuracy, it reduces latency and achieves more real-time performance. In the second version, which we call "BundleSDF-lite", we reduce the number of optimization steps while learning the neural object field, enabling faster synchronization between the threads.

From Fig. 4, we observe that the high pose tracking accuracy of BundleSDF [71] comes at a high computational cost. Since the pose tracking thread waits to get synchronized with the neural object field thread, it requires an average processing time of 2.1 seconds. One surprising result is the fact that BundleSDF-async achieves more accurate performance than BundleSDF-lite even though BundleSDF-async runs the pose estimation without waiting for the neural object field. This result exhibits the dependence of the pose graph optimization on having accurate keyframe poses. While the neural object field training in BundleSDF-async eventually yields more accurate poses (although at a delayed timestep) than BundleSDF-lite, the latter is unable to provide accurate pose estimates given the premature termination of the optimization to achieve faster speeds. In contrast, 6DOPE-GS provides a more balanced tradeoff between speed and accuracy. We achieve competitive performance without having to compromise on speed ($\sim$5$\times$ speedup compared to BundleSDF) as a result of the rapid convergence of the Gaussian Object Field optimization.

4.5. Ablations

We assessed our design choices on the HO3D dataset, chosen for its variety of scenarios, the results of which are shown in Table 3. Performance was reduced without dynamic keyframe selection (Ours w/o KF selection) due to the retention of inaccurate pose estimates during training, which introduces residual errors in the reconstruction loss and hinders pose optimization. Applying the vanilla adaptive density control (Ours w/o Pruning), where all Gaussians below a predefined threshold are removed, causes abrupt changes in the number of Gaussians. This results in significant rendering fluctuations, slowing the convergence of training. We find that our approach with the proposed additions, namely Dynamic Keyframe Selection and the Opacity Percentile-based Adaptive Density Control performs the best among all.

A visual comparison of the different ablations in reconstructing a water pitcher from HO3D can be seen in Fig. 5. This example is particularly difficult as there are large rotational motions and large occlusions. In this example, we find that neither of the ablations can accurately reconstruct the pitcher, especially the handle. Rather, they end up with two partially reconstructed handles as a result of inaccurate keyframe estimates which cause points to be added from a new keyframe with the handle visible. However, by combining the dynamic keyframe selection and the opacity percentile-based adaptive density control, our final approach reconstructs the handle more accurately.

Method	Pose		Reconstruction
	ADD-S (%) ↑	ADD (%) ↑	CD (cm) ↓
Ours (basic)	93.52	80.25	0.44
Ours w/o KF Selection	94.44	82.40	0.41
Ours w/o Pruning	92.48	80.87	0.42
Ours (final)	**95.07**	**84.33**	0.43

Table 3. Ablation study of 6DOPE-GS on the HO3D dataset. "Ours w/o KF Selection" removes the dynamic keyframe selection strategy (Sec. 3.3) and performs joint optimization using all keyframes. "Ours w/o Pruning" replaces the Opacity Percentile-based Adaptive Density Control 3.4 and employs vanilla adaptive density control [25]. "Ours (basic)" is a basic version of 6DOPE-GS that naively uses all the keyframes and uses the vanilla adaptive density control.

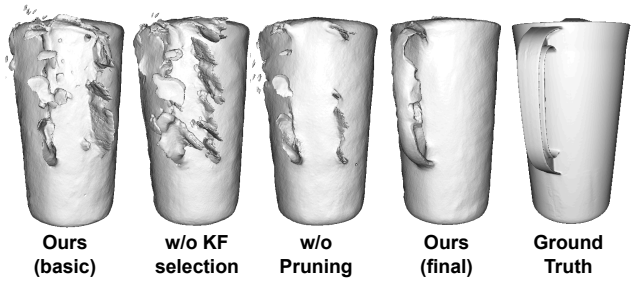

Figure 5. Object reconstruction example using the different ablations over our approach.

5. Conclusion

In this paper, we proposed "6DOPE-GS", a novel method for model-free 6D object pose estimation and reconstruction that leveraged 2D Gaussian Splatting for jointly optimizing object pose estimates and 3D reconstruction in an iterative manner. Key to our method's efficiency were a novel dynamic keyframe selection mechanism based on spatial coverage, as well as a confidence-based filtering mechanism to remove erroneous keyframes, followed by an opacity percentile-based adaptive density control for pruning out inconsequential Gaussians. These contributions enabled 6DOPE-GS to achieve competitive performance in a computationally efficient manner ($\sim$5$\times$ speedup), as validated on the HO3D and YCBInEOAT datasets, successfully capturing a practical balance of speed, accuracy, and stability for dynamic tracking scenarios in near real-time.

However, there are still some shortcomings that we aim to tackle in our future work. While Gaussian rasterization rendering enables high efficiency and allows for quick refinement of small translation and in-plane rotation errors, it has limitations in gradient computations compared to differentiable ray casting used by neural radiance fields. In future

work, we aim to explore ray casting for rendering Gaussian representations [39], which can enable improvements in both performance and computational efficiency. Another key shortcoming is that, currently the optimized 2D Gaussians model itself is not used in the online pose graph optimization, but rather the optimized poses. In future work, we would also explore how to more tightly couple the trained object representation to the pose graph optimization.

References

[1] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987. 4

[2] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising Densification in Gaussian Splatting. 5

[3] Jianqiu Chen, Zikun Zhou, Mingshan Sun, Rui Zhao, Liwei Wu, Tianpeng Bao, and Zhenyu He. Zeropose: Cad-prompted zero-shot object 6d pose estimation in cluttered scenes. *IEEE Transactions on Circuits and Systems for Video Technology*, 2024. 2

[4] Kai Chen and Qi Dou. Sgpa: Structure-guided prior adaptation for category-level 6d object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2773–2782, 2021. 2

[5] Weifeng Chen, Guangtao Shang, Aihong Ji, Chengjun Zhou, Xiyang Wang, Chonghui Xu, Zhenxiong Li, and Kai Hu. An overview on visual slam: From tradition to semantic. *Remote Sensing*, 14(13):3010, 2022. 3

[6] Ho Kei Cheng and Alexander G. Schwing. XMem: Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model. 2

[7] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020. 3

[8] Chenxi Lola Deng and Enzo Tartaglione. Compressing explicit voxel grid representations: fast nerfs become also small. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1236–1245, 2023. 5

[9] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking. 37(5):1328–1342. 2

[10] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3665–3671. IEEE, 2020. 1

[11] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023. 5

[12] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 4

[13] Yasutaka Furukawa, Carlos Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015. 2

[14] Ge Gao, Mikko Lauri, Yulong Wang, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. 6d object pose regression via supervised learning on point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3643–3649. IEEE, 2020. 2

[15] Felix Gorschlüter, Pavel Rojtberg, and Thomas Pöllabauer. A survey of 6d object detection based on 3d models for industrial applications. *Journal of Imaging*, 8(3):53, 2022. 2

[16] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3196–3206, 2020. 6

[17] Xingyi He, Jiaming Sun, Yuang Wang, Di Huang, and Xiaowei Zhou. OnePose++: Keypoint-Free One-Shot Object Pose Estimation without CAD Models. . 2

[18] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. FFB6D: A Full Flow Bidirectional Fusion Network for 6D Pose Estimation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3002–3012. IEEE, . 2

[19] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. PVN3D: A Deep Point-Wise 3D Keypoints Voting Network for 6DoF Pose Estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11629–11638. IEEE, . 2

[20] Yisheng He, Yao Wang, Haoqiang Fan, Jian Sun, and Qifeng Chen. FS6D: Few-Shot 6D Pose Estimation of Novel Objects. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6804–6814. IEEE, . 2, 6

[21] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-stage 6d object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2930–2939, 2020. 2

[22] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. 2, 3, 4, 1

[23] Iman Abaspur Kazerouni, Luke Fitzgerald, Gerard Dooly, and Daniel Toal. A survey of state-of-the-art on visual slam. *Expert Systems with Applications*, 205:117734, 2022. 3

[24] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21357–21366, 2024. 2, 3, 4

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. 2, 3, 4, 5, 8

[26] Kilian Kleeberger, Christian Landgraf, and Marco F Huber. Large-scale 6d object pose estimation dataset for industrial

bin-picking. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2573–2578. IEEE, 2019. 2

[27] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pages 574–591. Springer, 2020. 2

[28] Yann Labbé, Lucas Manuelli, Arsalan Mousavian, Stephen Tyree, Stan Birchfield, Jonathan Tremblay, Justin Carpentier, Mathieu Aubry, Dieter Fox, and Josef Sivic. Megapose: 6d pose estimation of novel objects via render & compare. *arXiv preprint arXiv:2212.06870*, 2022. 2

[29] Yanyan Li, Chenyu Lyu, Yan Di, Guangyao Zhai, Gim Hee Lee, and Federico Tombari. Geogaussian: Geometry-aware gaussian splatting for scene rendering. In *European Conference on Computer Vision*, pages 441–457. Springer, 2025. 5

[30] Jiehong Lin, Lihua Liu, Dekun Lu, and Kui Jia. SAM-6D: Segment Anything Model Meets Zero-Shot 6D Object Pose Estimation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27906–27916. IEEE. 2

[31] Xingyu Liu, Shun Iwase, and Kris M Kitani. Kdfnet: Learning keypoint distance field for 6d object pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4631–4638. IEEE, 2021. 2

[32] 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. 2

[33] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. 3

[34] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual slam algorithms. *Robotics*, 11(1):24, 2022. 3

[35] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. 1

[36] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024. 3, 4

[37] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 3

[38] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 3

[39] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *arXiv preprint arXiv:2407.07090*, 2024. 9

[40] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011. 7

[41] Van Nguyen Nguyen, Thibault Groueix, Mathieu Salzmann, and Vincent Lepetit. GigaPose: Fast and Robust Novel Object Pose Estimation via One Correspondence. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9903–9913. IEEE. 2

[42] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. In *Robotics: Science and Systems*, 2022. 3

[43] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion*. *Acta Numerica*, 26:305–364, 2017. 2

[44] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 3

[45] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10710–10719, 2020. 2

[46] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 4

[47] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G. Derpanis, and Kostas Daniilidis. 6-DoF object pose from semantic keypoints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2011–2018. 2

[48] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. pages 4561–4570. 2

[49] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE international conference on computer vision*, pages 3828–3836, 2017. 2

[50] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos. 3, 4, 2

[51] Xinlin Ren, Xingkui Wei, Zhuwen Li, Yanwei Fu, Yinda Zhang, and Xiangyang Xue. Deepsfm: Robust deep iterative refinement for structure from motion. *IEEE Transac-*

tions on Pattern Analysis and Machine Intelligence, 46(6): 4058–4074, 2023. 3

[52] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20. IEEE, 2018. 7

[53] Edward B Saff and Amo BJ Kuijlaars. Distributing many points on a sphere. *The mathematical intelligencer*, 19:5–11, 1997. 5

[54] 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

[55] Miroslava Slavcheva, Wadim Kehl, Nassir Navab, and Slobodan Ilic. Sdf-2-sdf: Highly accurate 3d object reconstruction. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 680–696. Springer, 2016. 7

[56] Stefan Stevšić, Sammy Christen, and Otmar Hilliges. Learning to assemble: Estimating 6d poses for robotic object-object manipulation. *IEEE Robotics and Automation Letters*, 5(2):1159–1166, 2020. 1

[57] Yongzhi Su, Jason Rambach, Nareg Minaskan, Paul Lesur, Alain Pagani, and Didier Stricker. Deep multi-state object pose estimation for augmented reality assembly. In *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 222–227. IEEE, 2019. 2

[58] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-Free Local Feature Matching with Transformers. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8918–8927. IEEE. 3, 4

[59] 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. 2

[60] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the european conference on computer vision (ECCV)*, pages 699–715, 2018. 2

[61] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021. 7

[62] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-Time Seamless Single Shot 6D Object Pose Prediction. pages 292–301. 2

[63] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints, . 2

[64] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martin-Martin, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion. pages 3343–3352, . 2

[65] Hengyi Wang and Lourdes Agapito. 3d reconstruction with spatial memory. *arXiv preprint arXiv:2408.16061*, 2024. 3

[66] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation, . 2

[67] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024. 3

[68] Xiang Wang, Chen Wang, Bing Liu, Xiaoqing Zhou, Liang Zhang, Jin Zheng, and Xiao Bai. Multi-view stereo in the deep learning era: A comprehensive review. *Displays*, 70: 102102, 2021. 2

[69] Bowen Wen and Kostas Bekris. BundleTrack: 6D Pose Tracking for Novel Objects without Instance or Category-Level 3D Models. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8067–8074. 2, 3, 4, 5, 6, 7

[70] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E. Bekris. Se(3)-TrackNet: Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10367–10373, . 2

[71] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Muller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects, . 2, 3, 4, 5, 6, 7, 8

[72] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects, . 2

[73] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E Bekris. se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10367–10373. IEEE, 2020. 5

[74] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 6

[75] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, 2024. 3, 4

[76] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting. 1

[77] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-

Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. *arXiv preprint arXiv:2410.03825*, 2024. 3

[78] Yan Zhao, Shaobo Zhang, Wanqing Zhao, Ying Wei, and Jinye Peng. Augmented reality system based on real-time object 6d pose estimation. In *2023 2nd International Conference on Image Processing and Media Computing (ICIPMC)*, pages 27–34. IEEE, 2023. 2

[79] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 2

[80] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12786–12796, 2022. 7

[81] Evin Pınar Örnek, Yann Labbé, Bugra Tekin, Lingni Ma, Cem Keskin, Christian Forster, and Tomas Hodan. FoundPose: Unseen Object Pose Estimation with Foundation Features. 2

6DOPE-GS: Online 6D Object Pose Estimation using Gaussian Splatting

Supplementary Material

6. Joint Optimization of 2D Gaussians and Keyframe Poses

Once we obtain an initial set of coarse keyframe poses, we jointly optimize the 2D Gaussians and the keyframe poses based on the photometric and structural losses used in in [22]. These losses include a color consistency loss $\mathcal{L}_c$, a depth consistency loss $\mathcal{L}_d$, a depth distortion loss $\mathcal{L}_{dd}$, and a normal alignment loss $\mathcal{L}_n$. The losses are computed between the rendered and observed images within the pixels covered in the object segmentation mask $\mathcal{M}$. To further reduce camera depth noise, we additionally use the Huber loss.

The color consistency loss $\mathcal{L}_c$ and the depth consistency loss $\mathcal{L}_d$ is the L_1 loss between the observed and the rendered images. The losses are calculated after projecting the Gaussians onto the image frame. Specifically, the Gaussians are ordered according to their ascending z-depth in the camera frame after which the contribution of each Gaussian to the loss is weighted based on their opacity and Gaussian density. This process is called α-blending. We represent the blending weight of each of the N ordered Gaussians as

$$\omega_i(\boldsymbol{p}) = \alpha_i G_i^{2D}(\boldsymbol{p}) \prod_{j=1}^{i-1}(1-\alpha_j G_j^{2D}(\boldsymbol{p})) \quad ; \quad \forall i \in N \tag{2}$$

We then calculate the color consistency loss $\mathcal{L}_c$ and the depth consistency loss $\mathcal{L}_d$ as

$$\mathcal{L}_c = \frac{1}{|\mathcal{M}|} \sum_{\boldsymbol{p}\in\mathcal{M}} |\hat{c}(\boldsymbol{p}) - c(\boldsymbol{p})| \tag{3}$$

$$\mathcal{L}_d = \frac{1}{|\mathcal{M}|} \sum_{\boldsymbol{p}\in\mathcal{M}} \rho(|\hat{d}(\boldsymbol{p}) - d(\boldsymbol{p})|) \tag{4}$$

where $\rho(\cdot)$ is the Huber loss, $c(\boldsymbol{p})$ and $d(\boldsymbol{p})$ are the observed color and depth, and $\hat{c}(\boldsymbol{p}) = \sum_{i\in N} c_i\omega_i(\boldsymbol{p})$ and $\hat{d}(\boldsymbol{p}) = \sum_{i\in N} d_i\omega_i(\boldsymbol{p})$ are the rendered color and depth at a pixel $\boldsymbol{p}$.

The depth distortion loss clusters the 2D Gaussians along the ray path, minimizing gaps between intersected Gaussians and enhancing depth accuracy.

$$\mathcal{L}_{dd} = \sum_{i,j\in N} \omega_i\omega_j|z_i - z_j| \tag{5}$$

where ω_i is the blending weight for the i^{th} Gaussian intersection, and z_i denotes the depth of each intersection point. Adjusting the intersection depth z_i to encourage the concentration of splats along the ray.

The normal loss $\mathcal{L}_{\text{n}}$ further refines object shape by aligning each Gaussian's normal with the local surface gradient.

$$\mathcal{L}_n = \sum_{\boldsymbol{p}\in\mathcal{M}} 1 - \hat{\boldsymbol{n}}(\boldsymbol{p})^\top \boldsymbol{n}(\boldsymbol{p}) \tag{6}$$

where $\hat{\boldsymbol{n}}(\boldsymbol{p})$ and $\boldsymbol{n}(\boldsymbol{p})$ are the normals at pixel $\boldsymbol{p}$ estimated by the gradient of the rendered and observed depth images respectively. By aligning the splat normal with the estimated normal, we ensure that the 2D splats accurately approximate the local object surface.

For refining the keyframe poses, a learnable affine transformation $\boldsymbol{T}_k \in SE(3)$ of the kth keyframe is applied to the 2D Gaussians similar to [35, 76]. The transformation $\boldsymbol{T}_k$ is represented by a rotation $\boldsymbol{R}_k \in SO(3)$ and a translation $t_k \in \mathbb{R}^3$. We learn the transformations along with the Gaussians by minimizing the above-mentioned losses obtained by projecting each Gaussian $G_i \in \mathcal{G}$ onto each selected keyframe pose $\boldsymbol{T}_k \in \mathcal{K}$ as

$$\begin{aligned}\mathcal{G}^*, \mathcal{K}^* = \arg\min_{\mathcal{G},\mathcal{K}} \sum_{k\in|\mathcal{K}|} & \lambda_c\mathcal{L}_c(\hat{I}(\boldsymbol{T}_k \odot \mathcal{G}), I_k) \\ & +\lambda_d\mathcal{L}_d(\hat{D}(\boldsymbol{T}_k \odot \mathcal{G}), D_k) \\ & +\lambda_{dd}\mathcal{L}_{dd} + \lambda_n\mathcal{L}_n\end{aligned} \tag{7}$$

where $\hat{I}$ and $\hat{D}$ are the rendered color and depth images obtained by projecting the Gaussians $\mathcal{G}$ to the k^{th} keyframe pose $\boldsymbol{T}_k$ and I_k, D_k represent the observed color and depth images of the k^{th} keyframe. λ_c, λ_d, λ_{dd} and λ_n are the relative weights for the color, depth, depth distortion and normal losses respectively.

7. Metrics

To evaluate 6-DoF object pose estimation, we calculate the Area Under Curve (AUC) percentage based on the ADD and ADD-S metrics. The ADD metric determines the average Euclidean distance between corresponding points on the 3D object model after transformation by the predicted and ground truth poses.

$$\text{ADD} = \frac{1}{N}\sum_{i=1}^{N} \|(Rx_i + t) - (R_{\text{gt}}x_i + t_{\text{gt}})\|, \tag{8}$$

where N represents the number of points in the 3D object model, x_i denotes a point on the model, R and t are the predicted rotation matrix and translation vector, and R_{gt} and t_{gt}

are their ground truth counterparts. A lower ADD value indicates a more accurate pose estimation. The estimation is considered successful if the ADD is within a specific threshold. For symmetric objects, where distinct poses may appear identical (e.g., a cylindrical object rotated by 180°), the ADD-S metric is more appropriate. Instead of directly pairing corresponding points, ADD-S measures the average distance between a transformed model point and its nearest neighbor in the ground truth-transformed model.

$$\text{ADD-S} = \frac{1}{N}\sum_{i=1}^{N}\min_{x_j\in\mathcal{M}}\|(Rx_i+t)-(R_{\text{gt}}x_j+t_{\text{gt}})\|, \quad (9)$$

where $x_j \in \mathcal{M}$ denotes the set of all 3D points on the object model. This formulation accounts for the ambiguities inherent to symmetric objects, providing a more robust evaluation of pose estimation.

We assess 3D shape reconstruction performance by calculating the chamfer distance between the reconstructed and ground-truth points, adopting the symmetric formulation.

$$\text{CD}(P,Q) = \frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}\|p-q\|^2 + \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P}\|q-p\|^2 \quad (10)$$

To extract meshes from reconstructed 2D splats, we render depth maps of the training views by projecting the depth values of the splats onto the pixels. Truncated Signed Distance Fusion (TSDF) is then used to fuse the reconstructed depth maps, implemented using Open3D [79]. During TSDF fusion, we set the voxel size to 0.002 and the truncation threshold to 0.02. Additionally, we extend BundleSDF to render depth maps and apply the same surface reconstruction technique to ensure a fair comparison.

8. Implementation Details

For the object masks in the YCBInEOAT dataset, we utilize the original masks provided in the dataset. For the HO3D dataset, we use the masks extracted XMem [6], as done in BundleSDF [71]. In real-time scenarios, we manually input the initial object location as a prompt for a random frame from the camera. Based on the prompted image, SAM2 [50] segments the target object across video frames in real-time.

During coarse pose estimation, a new frame is designated as a keyframe if it has more than 10 feature correspondences with the frames in memory. For online graph optimization, we retain the configuration of BundleSDF [71], restricting the number of frames involved in pose graph optimization to a maximum of 10.

For the Gaussian object field, we transform all graph-optimized camera poses into the OpenGL camera representation, which is utilized for Gaussian splatting. OpenGL adopts a right-handed coordinate system with the camera facing the negative z-axis, whereas OpenCV uses a right-handed system with the camera facing the positive z-axis. Using the information from the first frame, we estimate the object size and rescale and translate all camera poses and point clouds to ensure the generated Gaussians fit within a canonical space ranging from -1 to 1. Starting from the 10th keyframe, we optimize the Gaussian object. We begin by fusing color point clouds from the keyframes, followed by downsampling with a voxel size of 0.01 m. We then cluster the points with a maximum distance of 0.06m between points to remove outlier points. Subsequently, we uniformly upsample the point cloud until the number of points exceeds 5000.

We initialize the Gaussian means with the point cloud positions and random rotations. To simplify training and mitigate floating artifacts caused by oversized splats, Gaussian scales are clipped between 0.005 and 0.01. We use Spherical harmonics to represent the color, beginning at level 0 and incrementing every 200 steps, up to level 2. We initialize the opacity of each Gaussian with 0.1. Pose gradients for each keyframe are initialized with six parameters, (3 for translations and 3 for the rotation in an axis-angle representation). Before optimization, we group the keyframe poses using different anchors based on a default setting of an icosahedron at level 1, yielding a total of 42 anchors. Keyframes with the largest masks in each anchor cluster are selected for joint optimization, which runs for 1000 steps.

After 500 steps, the opacity percentile-based Adaptive Density Control is applied every 100 steps. Gaussians with opacity values in the bottom 5th percentile are removed until the 95th percentile opacity exceeds 0.5. Similarly, every 100 steps, keyframes with reconstruction losses exceeding twice the median absolute deviation (MAD) are removed.

In the training loss, λ_c = 0.5, λ_d = 0.5, λ_{dd} = 0.05, λ_n = 0.05. To mitigate depth noise, we apply morphological erosion to the depth using a kernel size of 5. The Adam optimizer is utilized for optimizing both the pose and Gaussian parameters, with a decay rate set to 0.5. The initial learning rate for Gaussians is consistent with the 2DGS [22] configuration. After the joint optimization is run for 1000 steps, the Gaussians' attributes are fixed, and only the keyframe poses are further refined for 500 additional steps.

All experiments were performed on a standard desktop equipped with an AMD Ryzen 9 7950X3D 16-core Processor and a single NVIDIA RTX 4090 GPU. Only the temporal efficiency experiments in Sec. 4.4 used a PC with a different configuration. Our method utilizes two concurrently running threads. The online tracking thread processes frames at approximately 4.1 Hz, while the Gaussian object field thread operates in the background, requiring an average of 0.23 seconds per keyframe.

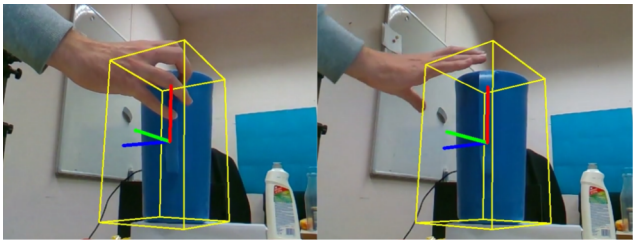

Figure 6. Rotation angle estimation from a side view is often inaccurate for symmetric objects without texture information, as exemplified by the AP10 object from the HO3D dataset.

9. Limitations

Gaussian rasterization rendering is highly efficient and allows for the rapid correction of minor translation and in-plane rotation errors. However, it is less effective in gradient computation compared to the differentiable ray casting employed by neural radiance fields. This limitation stems from the fact that pose gradients in Gaussian rasterization are approximated using the covariance matrix projected onto the 2D plane. As a result, gradient-based optimization for non-in-plane rotations or substantial pose corrections becomes problematic. For example, our method struggles to resolve rotational errors around the symmetric axis of a water pitcher, as shown in Fig 6.

10. Realtime Results

Figure 7. Example of real-time object tracking. **Top row:** Live video, object segmentation results, and pose tracking results. **Bottom row:** Rendered outputs, including color, depth, and surface normals derived from the Gaussian models.

We utilized the ZED 2 camera operating in the standard depth sensing mode to maintain a balance between frame rate and accuracy. The camera captures video at a resolution of 1080p with a frame rate of 30 FPS. An initial mask for the target object was manually created through human annotation. The SAM2 system also operates at 28 FPS. Pose tracking, when running in visualization mode, achieves a processing frequency of 3-4 Hz, primarily due to the computational overhead introduced by the GUI and the rendering of Gaussian models in the background. Without the GUI, the system can operate at a slightly higher frequency of 4-5 Hz. The Gaussian model updates approximately every 8 seconds, as illustrated in Fig 7. For a more comprehensive understanding of the system's performance, we encourage readers to refer to the supplementary video provided.

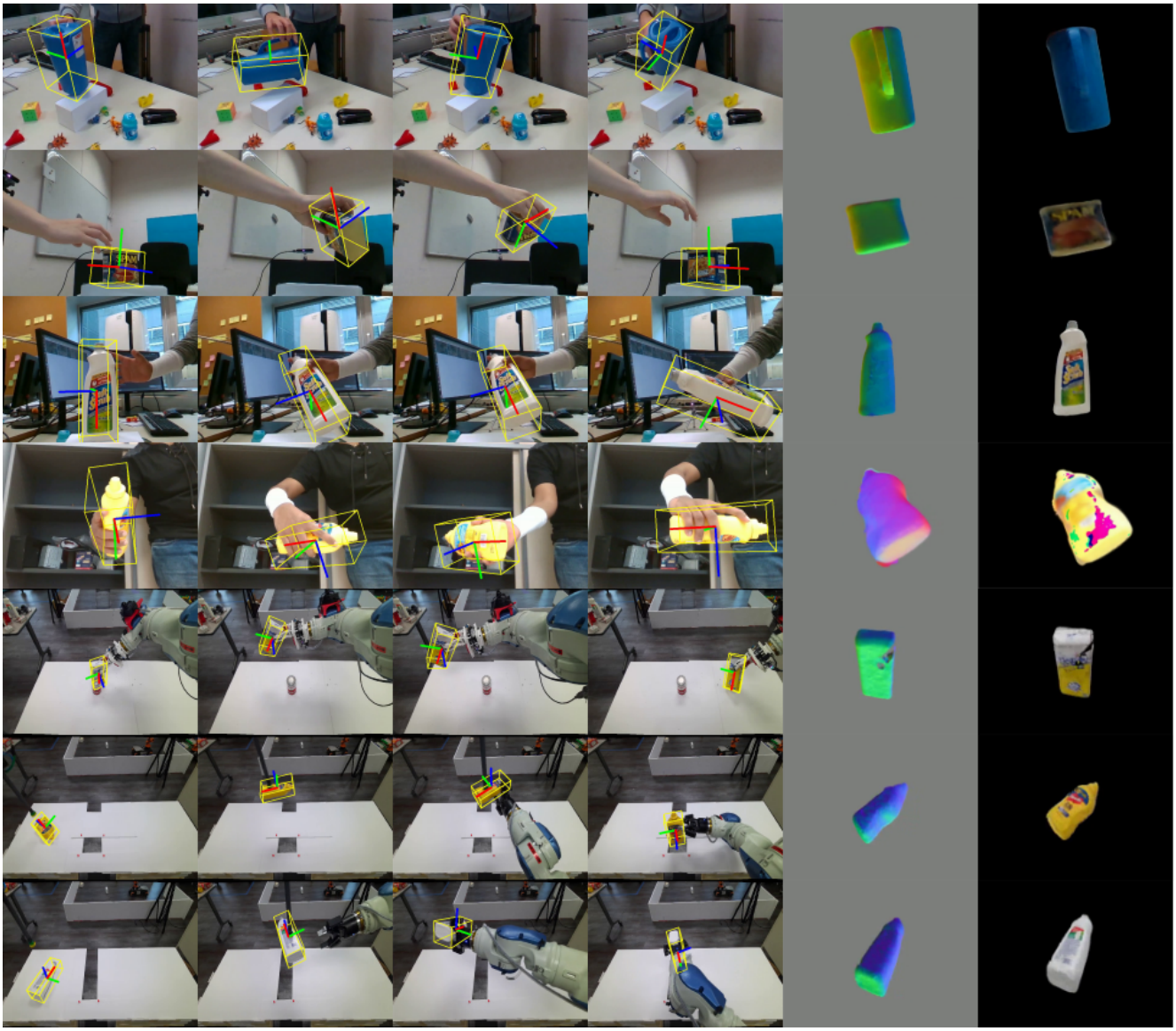

Figure 8. Qualitative results of our method on video sequences from the HO3D and YCBInEOAT datasets

Table 4. Comparison of ADD-S, ADD, and CD metrics, along with the Average Time Per Frame, across different methods on the HO3D dataset. ↑ indicates higher values are better, ↓ indicates lower values are better.

Video	Metric	BundleTrack [69]	BundleSDF [71]	BundleTrack*	BundleSDF*	BundleSDF-async*	BundleSDF-Lite	Ours
AP10	ADD-S(%)↑	91.68	96.1	91.24	**95.83**	91.78	93.03	95.34
	ADD(%)↑	36.60	91	47.98	**89.66**	66.38	78.66	85.88
	CD(cm)↓	1.88	0.47	-	0.13	0.55	0.76	**0.20**
	ATPF(s)↓	-	-	0.27	1.55	0.26	0.39	**0.23**
AP11	ADD-S(%)↑	91.45	96.18	94.14	96.01	95.47	93.34	**96.92**
	ADD(%)↑	41.28	91.76	84.53	90.91	89.32	81.05	**93.78**
	CD(cm)↓	129.18	0.56	-	0.10	0.13	0.65	**0.06**
	ATPF(s)↓	-	-	0.28	1.46	0.26	0.38	**0.23**
AP12	ADD-S(%)↑	90.79	97.06	95.42	**96.98**	96.51	93.87	96.77
	ADD(%)↑	50.82	94.76	88.09	**94.53**	92.32	83.34	93.35
	CD(cm)↓	2.47	0.59	-	**0.04**	0.09	0.74	0.06
	ATPF(s)↓	-	-	0.29	1.40	0.28	0.41	**0.25**
AP13	ADD-S(%)↑	90.68	96.16	95.65	96.19	95.96	92.66	**96.46**
	ADD(%)↑	49.03	92.73	89.95	92.77	92.11	80.14	**92.90**
	CD(cm)↓	2.77	0.63	-	**0.03**	0.05	0.78	0.06
	ATPF(s)↓	-	-	0.28	1.40	0.29	0.41	**0.24**
AP14	ADD-S(%)↑	96.02	96.01	96.09	**97.09**	96.89	96.50	95.84
	ADD(%)↑	90.30	91.25	91.07	**94.65**	94.11	92.23	91.56
	CD(cm)↓	72.40	1.28	-	**0.03**	0.05	0.71	0.07
	ATPF(s)↓	-	-	0.29	1.46	0.29	0.41	**0.25**
MPM10	ADD-S(%)↑	94.94	95.05	93.01	**95.28**	93.79	93.31	94.29
	ADD(%)↑	87.45	88.92	75.64	**89.48**	83.75	83.78	84.55
	CD(cm)↓	0.97	0.56	-	**0.13**	0.29	0.57	0.33
	ATPF(s)↓	-	-	0.29	2.39	0.29	0.44	**0.23**
MPM11	ADD-S(%)↑	89.94	96.2	96.06	**96.19**	96.53	94.75	96.09
	ADD(%)↑	53.20	91.51	91.07	91.51	92.33	88.44	**91.91**
	CD(cm)↓	88.97	0.49	-	0.09	**0.09**	0.53	0.12
	ATPF(s)↓	-	-	0.28	2.17	0.28	0.44	**0.27**
MPM12	ADD-S(%)↑	95.66	96.98	97.88	96.17	97.18	80.20	**97.76**
	ADD(%)↑	90.96	93.13	95.12	91.36	93.64	60.98	**95.48**
	CD(cm)↓	121.33	0.46	-	0.08	**0.06**	1.30	0.07
	ATPF(s)↓	-	-	0.34	1.93	0.33	0.49	**0.22**
MPM13	ADD-S(%)↑	89.42	95.8	85.37	74.70	65.79	54.19	**88.00**
	ADD(%)↑	38.78	90.62	32.03	51.91	26.39	27.51	**32.50**
	CD(cm)↓	81.39	0.57	-	**0.90**	1.67	0.92	2.06
	ATPF(s)↓	-	-	0.27	1.83	0.27	0.36	**0.25**
MPM14	ADD-S(%)↑	95.49	97.33	95.49	**97.19**	95.44	96.63	95.55
	ADD(%)↑	90.16	94.52	88.02	**94.18**	87.75	92.43	88.88
	CD(cm)↓	94.99	0.47	-	**0.07**	0.30	0.50	0.24
	ATPF(s)↓	-	-	**0.30**	2.41	0.30	0.44	0.59
SB11	ADD-S(%)↑	94.44	97.27	94.54	**97.07**	94.40	95.17	94.55
	ADD(%)↑	84.64	94.39	78.34	**93.82**	79.13	86.53	78.23
	CD(cm)↓	75.83	0.46	-	**0.12**	0.58	0.60	0.63
	ATPF(s)↓	-	-	0.23	2.18	0.23	0.37	0.59
SB13	ADD-S(%)↑	95.66	97.67	97.28	**97.71**	97.20	96.99	97.08
	ADD(%)↑	85.47	95.24	92.68	**95.31**	92.90	92.09	92.30
	CD(cm)↓	2.49	0.47	-	**0.08**	0.16	0.75	0.17
	ATPF(s)↓	-	-	0.25	1.82	0.25	0.39	**0.22**
SM1	ADD-S(%)↑	84.94	96.9	89.36	**96.87**	89.39	87.44	91.31
	ADD(%)↑	59.41	94.24	56.13	**94.19**	58.27	57.03	75.05
	CD(cm)↓	2.04	0.44	-	**0.08**	1.43	0.91	0.77
	ATPF(s)↓	-	-	0.33	5.25	0.34	0.74	**0.27**
Mean	ADD-S(%)↑	92.39	96.52	93.96	94.87	92.80	89.85	**95.07**
	ADD(%)↑	66.01	92.62	77.75	**89.56**	80.65	77.25	84.34
	CD(cm)↓	52.05	0.57	-	**0.15**	0.42	0.75	0.37
	ATPF(s)↓	-	-	0.29	2.10	0.28	0.44	**0.24**

Table 5. Comparison of ADD-S, ADD, and CD metrics, along with the Average Time Per Frame, across different methods on the YCBI-nEOAT dataset. ↑ indicates higher values are better, ↓ indicates lower values are better.

Object	Metric	BundleTrack [69]	BundleSDF [71]	BundleTrack*	BundleSDF*	BundleSDF-async*	BundleSDF-Lite	Ours
cracker_box	ADD-S(%)↑	90.2	90.63	89.41	90.23	91.78	90.52	**95.33**
	ADD(%)↑	85.08	85.37	63.24	80.29	66.38	81.99	**91.3**
	CD(cm)↓	1.36	0.76	-	0.53	0.55	0.21	**0.1**
	ATPF(s)↓	-	**-**	0.20	0.59	0.20	0.21	**0.18**
bleach_cleanser	ADD-S(%)↑	95.22	94.28	82.45	93.48	95.47	92.18	**93.81**
	ADD(%)↑	89.34	87.46	61.83	85.48	89.32	82.56	**85.78**
	CD(cm)↓	1.31	0.53	-	0.44	**0.13**	0.16	0.25
	ATPF(s)↓	-	**-**	**0.21**	1.01	0.22	0.23	**0.21**
sugar_box	ADD-S(%)↑	90.68	93.81	81.42	**96.58**	96.51	89.48	96.21
	ADD(%)↑	85.49	88.62	51.91	92.08	92.32	82.33	**92.34**
	CD(cm)↓	2.25	0.46	-	0.23	0.09	0.22	**0.07**
	ATPF(s)↓	-	**-**	**0.20**	0.70	0.22	0.26	0.21
tomato_soup_can	ADD-S(%)↑	95.24	95.24	71.61	79.19	**95.96**	94.84	94.26
	ADD(%)↑	85.78	83.1	41.36	57.3	**92.11**	82.45	86.14
	CD(cm)↓	7.36	3.57	-	1.16	**0.05**	0.35	0.28
	ATPF(s)↓	-	**-**	**0.20**	1.07	0.20	0.25	0.23
mustard_bottle	ADD-S(%)↑	95.84	95.75	88.53	95.85	96.89	95.07	**95.94**
	ADD(%)↑	92.15	89.87	71.92	90.15	**94.11**	86.95	92.22
	CD(cm)↓	1.76	0.45	-	0.31	**0.05**	0.18	0.12
	ATPF(s)↓	-	**-**	**0.21**	0.74	0.22	0.26	0.25
Mean	ADD-S(%)↑	93.01	93.77	81.17	92.55	**93.79**	93.66	**93.79**
	ADD(%)↑	87.26	86.95	57.91	84.91	83.75	86.41	**87.83**
	CD(cm)↓	2.81	1.16	-	0.53	0.29	0.25	**0.15**
	ATPF(s)↓	-	**-**	**0.21**	0.82	**0.21**	0.24	0.22