

FlashSLAM: Accelerated RGB-D SLAM for Real-Time 3D Scene Reconstruction with Gaussian Splatting

flashslam.github.io

Phu Pham¹, Damon Conover², Aniket Bera¹

¹Department of Computer Science, Purdue University ²DEVCOM Army Research Laboratory
{phupham, aniketbera}@purdue.edu, damon.m.conover.civ@army.mil

Abstract

We present FlashSLAM, a novel SLAM approach that leverages 3D Gaussian Splatting for efficient and robust 3D scene reconstruction. Existing 3DGs-based SLAM methods often fall short in sparse view settings and during large camera movements due to their reliance on gradient descent-based optimization, which is both slow and inaccurate. FlashSLAM addresses these limitations by combining 3DGS with a fast vision-based camera tracking technique, utilizing a pretrained feature matching model and point cloud registration for precise pose estimation in under 80 ms - a 90% reduction in tracking time compared to SplaTAM - without costly iterative rendering. In sparse settings, our method achieves up to a 92% improvement in average tracking accuracy over previous methods. Additionally, it accounts for noise in depth sensors, enhancing robustness when using unspecialized devices such as smartphones. Extensive experiments show that FlashSLAM performs reliably across both sparse and dense settings, in synthetic and real-world environments. Evaluations on benchmark datasets highlight its superior accuracy and efficiency, establishing FlashSLAM as a versatile and high-performance solution for SLAM, advancing the state-of-the-art in 3D reconstruction across diverse applications.

1. Introduction

Building accurate 3D maps while simultaneously tracking position has become foundational in advancing autonomous systems across robotics, computer vision, and augmented reality [5, 17, 26], leading to the rapid evolution of Simultaneous Localization and Mapping (SLAM) techniques. Recent advancements in SLAM have focused on developing more accurate, efficient, and robust methods for real-time 3D reconstruction and camera tracking.

Integrating neural rendering techniques with SLAM systems has shown promise for joint localization and photore-

alistic view reconstruction [21, 22, 25, 34]. However, many of these approaches depend heavily on implicit representations, which are computationally intensive and impractical for portable device deployment [8]. This constraint has driven research toward more efficient representations that preserve high-quality reconstructions while supporting real-time performance.

One such representation that has gained significant attention is 3D Gaussian Splatting (3DGS) [12]. Originally developed for offline scene reconstruction, 3DGS has recently been adapted for real-time SLAM applications, driving a wave of innovative research in the field. Several pioneering works [11, 15, 30] have demonstrated the potential of 3DGS in unifying tracking, mapping, and rendering within a single framework, capable of handling both monocular and RGB-D inputs effectively. These methods have shown significant progress in addressing key challenges in SLAM, such as real-time performance, high-quality reconstruction, and robustness to various input modalities.

Despite these advancements, existing 3DGS-based SLAM methods still face challenges in sparse settings or scenarios with rapid camera movement. Current approaches often rely on gradient descent optimization for localization, which can lead to slow convergence and inaccurate results when dealing with large camera displacements.

Our work addresses these limitations by introducing a novel SLAM approach that leverages 3D Gaussian splatting for robust 3D scene reconstruction while incorporating an efficient and accurate vision-based camera tracking method. Unlike existing approaches that optimize camera pose using gradient descent and require rendering for each parameter update, our approach utilizes a pretrained local feature-matching model and point cloud registration. This results in faster and more accurate camera estimation, particularly in sparse settings or when dealing with large camera movements.

The main contributions of our work are as follows:

- **Efficient Vision-based Camera Tracking:** A fast and

accurate camera pose estimation method using pretrained feature matching and point cloud registration, achieving significant speed and accuracy gains, especially in sparse environments and with large camera movements.

- **High-Fidelity Reconstruction:** An optimized 3DGS representation utilizing RGB-D images to produce high-quality 3D reconstructions in real time.
- **Robustness to Depth Sensor Errors:** A truncated depth approach mitigates errors from consumer-grade depth sensors, enabling effective use on devices like smartphones.
- **Comprehensive Evaluation:** Extensive experiments on benchmark datasets highlight our approach’s accuracy, efficiency, and robustness.

These contributions advance SLAM and 3D reconstruction, offering a robust, efficient, and adaptable solution for diverse applications and hardware. The following sections detail our methodology, experimental setup, and results, demonstrating our approach’s effectiveness across multiple benchmarks.

2. Related Work

2.1. SLAM Systems

Simultaneous Localization and Mapping (SLAM) has been a central challenge in robotics and computer vision for decades, evolving from traditional sparse feature-based methods to dense volumetric representations. Early approaches like ORB-SLAM [19] utilized distinctive visual features for mapping and localization, while later techniques such as KinectFusion [20] employed depth sensors for richer geometric reconstruction. These methods, however, faced limitations in certain environments and often required specialized hardware.

The recent introduction of neural implicit representations, particularly Neural Radiance Fields (NeRF) [16], has opened new avenues for SLAM research. Systems like iMAP [25] and NICE-SLAM [34] have demonstrated the potential of these representations for high-quality scene reconstruction and localization. While promising, these neural approaches face challenges in computational efficiency, scalability, and real-time performance, especially in large-scale or dynamic environments. Current research focuses on addressing these limitations through hybrid approaches, more efficient neural architectures, and multi-modal sensor fusion, aiming to create more robust and versatile SLAM systems for real-world applications [22].

2.2. 3D Gaussian Splatting

3D Gaussian Splatting (3DGS), introduced by Kerbl et al. [12], is a robust technique for representing and rendering 3D scenes using collections of 3D Gaussian primitives to model both geometry and appearance. Compared

to traditional mesh-based methods [10, 14], 3DGS offers a more flexible representation that enables smoother surface approximations and better management of partial or noisy data. These properties make it especially suitable for real-time applications like SLAM, where data is often incomplete.

Recent advancements in 3DGS have further expanded its utility, supporting efficient integration of multimodal data such as RGB and depth, which enhances reconstruction accuracy. This method has shown significant improvements in rendering speed and quality over earlier approaches like Neural Radiance Fields (NeRF) [16], establishing it as a promising direction for research in computer graphics and computer vision [12].

2.3. 3DGS-based SLAM

The integration of 3D Gaussian Splatting (3DGS) into SLAM systems marks a significant advancement, utilizing 3DGS for real-time mapping and localization. Matsuki et al. [15] introduced a monocular Gaussian Splatting SLAM system that operates live at 3 fps, using Gaussians as the sole 3D representation for efficient tracking, mapping, and rendering.

Similarly, GS-SLAM by Yan et al. [30] proposed a dense visual SLAM system based on 3DGS, offering competitive reconstruction and localization performance at reduced computational costs. Through a real-time differentiable splatting rendering pipeline, GS-SLAM balances accuracy and efficiency, accelerating map optimization and RGB-D rendering.

SplaTAM by Keetha et al. [11] further advanced 3DGS-based SLAM by introducing a dense RGB-D system that achieves real-time performance and high-quality reconstruction, showcasing the potential of 3DGS in SLAM, especially with depth data.

Despite their strengths, these systems face challenges in sparse or dynamic real-world scenes. All current 3DGS-based SLAM methods estimate camera pose by projecting the previous pose forward using a constant velocity assumption in camera center + quaternion space. This approach struggles with large camera movements, leading to inaccurate poses and cumulative errors that cause *camera drifting*, compromising reconstruction accuracy.

Techniques like loop closure detection and bundle adjustment (BA) could refine global poses but often fail in feature-sparse environments or under lighting, viewpoint, or dynamic changes [27]. Without a loop in the camera path, loop closure cannot mitigate drift. Additionally, rapid camera movements, repetitive patterns, and limited computational resources exacerbate drift, challenging SLAM systems to maintain real-time precision in large or complex environments.

3. Method

The core of our method is an efficient tracking system designed to perform accurately in both dense and sparse settings. In this section, we describe our approach in detail. Figure 1 provides an overview, highlighting how our method integrates precise feature matching, reliable pose estimation, and refinement techniques to achieve high-accuracy tracking. This enables robust SLAM performance across varying levels of data sparsity and environmental complexity.

3.1. 3D Gaussian representation

Following recent advances, we adopt the 3D Gaussian splatting (3DGS) approach to represent the scene as a continuous field of Gaussian splats. Each splat G_i has a 3D center \mathbf{c}_i , covariance Σ_i , color \mathbf{v}_i , and opacity α_i . The Gaussian function $G_i(\mathbf{x})$ for a point \mathbf{x} in space is defined as:

$$G_i(\mathbf{x}) = \alpha_i \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{c}_i)^T \Sigma_i^{-1} (\mathbf{x} - \mathbf{c}_i)\right), \quad (1)$$

where Σ_i controls the size and orientation, and α_i modulates opacity.

To render, each splat's color \mathbf{v}_i is scaled by its intensity and opacity, contributing to the final image. We project each 3D center \mathbf{c}_i to the 2D image plane using the camera intrinsics K and extrinsics $[R|t]$, yielding:

$$\mathbf{p}_i = K[R|t]\mathbf{c}_i, \quad (2)$$

with projected covariance $\Sigma_i^{\text{proj}} = J_i \Sigma_i J_i^T$ for adapting the splat's shape in the image plane.

This differentiable rendering process allows us to optimize Gaussian parameters by comparing rendered images from multiple viewpoints with ground-truth images, refining the splats for accurate scene representation. We use this 3DGS representation in our SLAM framework to achieve high-fidelity, adaptable 3D reconstructions without relying on explicit meshes.

3.2. Camera Pose Tracking

Recent Gaussian-based SLAM methods, such as Splatam [11, 15, 30], rely on an assumption of constant velocity and small discrepancies in camera movement, optimizing the camera position based on photometric and depth losses between rendered and ground-truth images. While effective for dense SLAM, this approach struggles in sparse settings where large camera movements and limited frame overlap challenge the model's ability to accurately estimate pose. In these scenarios, the assumption of gradual motion breaks down, leading to failure in aligning frames with significant disparity.

Classic approaches for estimating rigid body transformations, such as the least-squares fitting method for 3D point sets [1, 6, 7], provide a foundational method for determining camera transformations by aligning corresponding points between frames. In our setup, we utilize LightGlue [13], a robust feature-matching framework designed to produce highly reliable correspondences, even under challenging conditions. LightGlue generates high-confidence matches between frames I_{t-1} and I_t , forming a strong basis for accurate transformation estimation, especially in scenarios with significant disparity between frames.

To utilize these correspondences, we first extract keypoints and descriptors from the RGB frames I_{t-1} and I_t using SuperPoint [4], with LightGlue yielding high-quality matched points $(\mathbf{p}_{t-1}, \mathbf{p}_t)$. After filtering for reliable matches, we back-project these 2D points to 3D space using the camera's intrinsic parameters, resulting in two point sets, \mathbf{P}_{t-1} and \mathbf{P}_t , representing frames I_{t-1} and I_t . We then estimate the optimal rigid transformation, consisting of a rotation matrix $\mathbf{R} \in \text{SO}(3)$ and translation vector $\mathbf{t} \in \mathbb{R}^3$, by minimizing the least-squares error:

$$\min_{\mathbf{R}, \mathbf{t}} \sum_{i=1}^N \|\mathbf{p}_t^{(i)} - (\mathbf{R}\mathbf{p}_{t-1}^{(i)} + \mathbf{t})\|^2. \quad (3)$$

To estimate the rotation matrix \mathbf{R} and translation vector \mathbf{t} , we first calculate the centroids of each point set:

$$\mathbf{c}_{t-1} = \frac{1}{N} \sum_{i=1}^N \mathbf{p}_{t-1}^{(i)}, \quad \mathbf{c}_t = \frac{1}{N} \sum_{i=1}^N \mathbf{p}_t^{(i)}. \quad (4)$$

Centering each point set around its centroid, we obtain $\mathbf{Q}_{t-1} = \mathbf{P}_{t-1} - \mathbf{c}_{t-1}$ and $\mathbf{Q}_t = \mathbf{P}_t - \mathbf{c}_t$, and compute the covariance matrix $\mathbf{H} = \mathbf{Q}_{t-1}^T \mathbf{Q}_t$.

The optimal rotation matrix \mathbf{R} is given by:

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T, \quad (5)$$

where \mathbf{U} and \mathbf{V} are obtained from the Singular Value Decomposition (SVD) of \mathbf{H} , such that $\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^T$. If $\det(\mathbf{R}) < 0$, we adjust by negating the last column of \mathbf{V} to ensure \mathbf{R} is a valid rotation. The translation vector is computed as:

$$\mathbf{t} = \mathbf{c}_t - \mathbf{R}\mathbf{c}_{t-1}. \quad (6)$$

The rigid transformation provides a good initial estimate of the camera pose; however, it can be further refined through gradient-based optimization with significantly fewer iterations. Although the transformation is generally accurate when depth values are reliable, performance can degrade in the presence of depth sensor noise, which may distort the transformation. To mitigate this, we apply a dynamically adjusted depth truncation threshold, set at the 70th percentile of the depth distribution for each frame, to

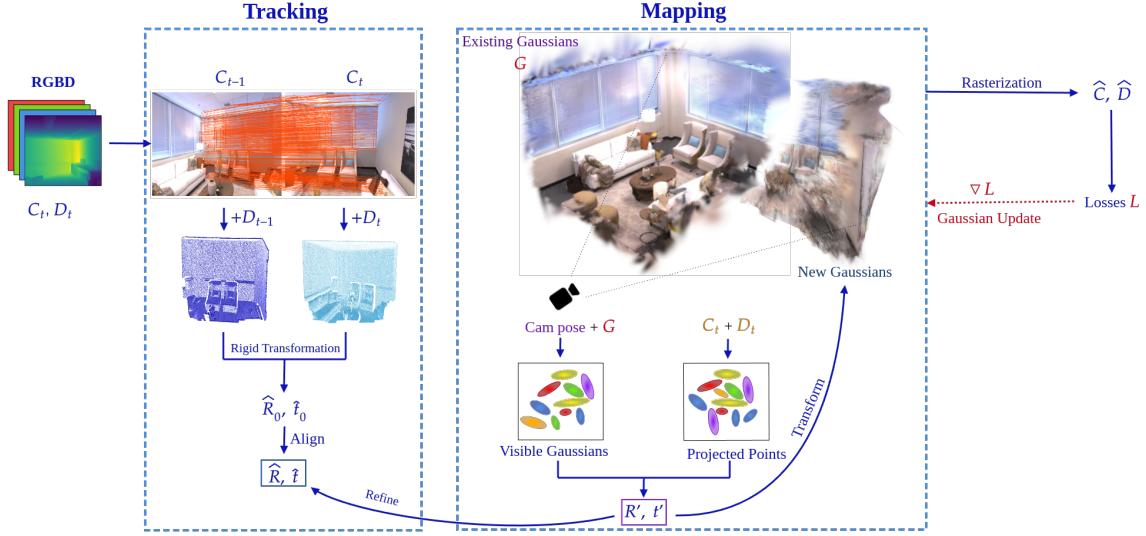


Figure 1. Overview of FlashSLAM: Our approach takes RGB-D inputs to perform accurate 3D scene reconstruction. Initially, precise matches between consecutive frames are detected, which enables tracking of the camera pose through a rigid transformation. This pose is further refined using gradient-based optimization, leveraging Gaussian alignment to ensure accurate registration of new frames with the existing 3D model. The mapping process updates and transforms existing Gaussian splats in the 3D scene, producing high-quality reconstructions with efficient alignment and optimization steps.

filter out excessively noisy points. This approach avoids setting the threshold too low, thereby retaining valid matches while excluding unreliable depths. By ensuring that only reliable depth values contribute to the final transformation, we improve robustness in challenging conditions.

3.3. Mapping

Adding New Gaussians. Our mapping step dynamically updates the 3D Gaussian map to maintain an accurate representation of the scene. At each frame, we render the scene using the current estimated camera pose and compare the rendered depth values, D_r , with the ground-truth depth image, D_{gt} . For each pixel (u, v) in D_{gt} where $D_{gt}(u, v) < D_r(u, v)$, we identify the need for additional Gaussians to represent that region of the scene, as existing Gaussians fail to capture closer surfaces.

After identifying regions that require new Gaussians, we project the pixels of the new frame into 3D space using depth values and the camera's intrinsic parameters. Let P_o denote the set of projected points that have already been mapped and are represented by existing Gaussians, and let P_n represent the set of points corresponding to the newly identified regions that need additional Gaussians. Due to potential noise in the depth sensor, the projected positions in P_n may be inaccurate. To address this, we apply a transformation correction process to align P_o with the current Gaussians before integrating P_n .

To align P_o with the existing Gaussians, we first select a subset $G_v \subset G$ of Gaussians that are visible from the current camera position, where G represents all Gaussians in the map. The visibility of a Gaussian $g \in G$ is determined by checking if its center falls within the frustum of the current camera view. We then downsample both G_v and P_o to reduce computational complexity, before applying Iterative Closest Point (ICP) to find the optimal rotation \mathbf{R} and translation \mathbf{t} . The transformation (\mathbf{R}, \mathbf{t}) is accepted if the ICP alignment achieves a fitness score $f > 0.2$ and an error $e < 0.1$. If these conditions are not met, the transformation is rejected to prevent the introduction of inaccuracies.

Once a valid transformation (\mathbf{R}, \mathbf{t}) is obtained, we apply it to the points in P_n to correct their positions:

$$\mathbf{p}_n^{(i)} \leftarrow \mathbf{R}\mathbf{p}_n^{(i)} + \mathbf{t}, \quad (7)$$

where $\mathbf{p}_n^{(i)} \in P_n$. The transformed points in P_n are then used as the centers for new Gaussians to be added to the map.

Additionally, we adjust the current estimated camera pose by incorporating the computed transformation, ensuring that the newly added Gaussians are seamlessly integrated with the existing map. This alignment process helps maintain coherence between new and existing Gaussians, bolstering the robustness of subsequent optimizations.

Loss Functions. To optimize the Gaussian map, we adopt a strategy similar to recent RGB-D SLAM methods by combining photometric and depth losses to ensure both color fidelity and geometric accuracy in the reconstructed scene.

The photometric loss function, L_{color} , measures the L_1 difference between the ground-truth image C and the reconstructed image \hat{C} from the same estimated camera pose:

$$L_{\text{color}} = \|C - \hat{C}\|_1. \quad (8)$$

In addition to the photometric loss, we apply a depth loss, L_{depth} , to ensure geometric consistency in the scene reconstruction. The depth loss is defined as the L_1 distance between the ground-truth depth map D and the reconstructed depth map \hat{D} :

$$L_{\text{depth}} = \|D - \hat{D}\|_1, \quad (9)$$

where D and \hat{D} represent the depth values for each pixel in the ground-truth and reconstructed images, respectively.

The overall loss function, L , combines the photometric and depth losses with a weighting factor λ that controls the relative importance of each term:

$$L = \lambda L_{\text{color}} + (1 - \lambda) L_{\text{depth}}. \quad (10)$$

The parameter λ allows us to balance color and depth consistency according to the specific requirements of the scene reconstruction.

Keyframe Selection. In dense SLAM settings, where camera movement is minimal, many frames have substantial overlap with their neighboring frames, leading to redundancy. Optimizing the Gaussian map using all these frames would be computationally expensive and unnecessary. To address this, we employ a simple keyframe selection strategy, as proposed in prior work [15]. A new frame is added to the keyframe list if it overlaps with the previous keyframe and falls below a specified threshold. This overlap is measured as the intersection over union (IoU) of the visible Gaussians within the camera frustums of the two frames. In sparse settings, we simplify the selection by designating every k^{th} frame as a keyframe.

3.4. Color Refinement with Priority Sampling

To enhance the appearance of the reconstructed scene, we employ a refinement step to improve the alignment of Gaussian positions and colors. Unlike initial optimization, this step does not add or remove Gaussians; rather, it adjusts the existing Gaussians for more precise alignment, making the refinement computationally efficient as the number of Gaussians remains constant.

For this process, we use a list of keyframes and apply a priority sampling strategy, loss-weighted sampling, where

frames with higher losses are more likely to be sampled for refinement. The probability of sampling a frame i is proportional to its loss magnitude, L_i . Specifically, we compute the probability $P(i)$ of selecting frame i as follows:

$$P(i) = \frac{L_i}{\sum_{j=1}^N L_j}, \quad (11)$$

where N is the total number of keyframes, and L_j represents the loss for each frame j . This ensures that frames with greater discrepancies (higher L_i) are sampled more frequently, directing optimization efforts toward frames where alignment is weakest.

We also experimented with an alternative strategy, worst-first sampling, which samples only the frame with the highest loss by maintaining a heap of frame losses. However, we found that loss-weighted sampling combined with random sampling yielded better results. Specifically, to avoid ‘forgetting’ well-aligned frames, we apply priority sampling with a probability $p = 0.4$ and use random sampling with probability $1-p = 0.6$, selecting frames uniformly from the keyframe list. This balance ensures that high-error frames are refined more often while still maintaining optimization across well-aligned frames, preserving global consistency in the reconstructed scene.

4. Experiments and Results

4.1. Experiment Setup

Datasets. For performance evaluation, we use two widely recognized datasets in 3D computer vision and robotics: Replica [23] and TUM RGB-D [24]. The Replica dataset consists of synthetic 3D indoor scenes with dense geometry and high-resolution HDR textures. The TUM RGB-D dataset provides synchronized RGB and depth images from a handheld camera, suitable for real-world indoor environments. For comparison with existing methods, we evaluate on a subset of each dataset, specifically using 8 scenes from Replica [23] and all 5 scenes from TUM RGB-D [24].

Evaluation Metrics. To evaluate rendering quality, we employ three metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [28], and Learned Perceptual Image Patch Similarity (LPIPS) [33]. These metrics are computed by comparing the rendered image frames against the corresponding sensor images. For assessing camera tracking precision, we utilize the Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE) calculated on the keyframes.

Baseline Methods. We evaluate our SLAM method in comparison to state-of-the-art SLAM approaches based on NeRF and 3D Gaussian Splatting. For rendering quality, we compare against Point-SLAM [22], MonoGS [15],

SplatAM [11], and GS-SLAM [30]. Furthermore, we assess our tracking performance relative to iMAP [25], NICE-SLAM [34], and ESSLAM [9] across various datasets. In this work, we place greater emphasis on Gaussian-based methods, as NeRF-based approaches have been extensively studied in recent Gaussian-based SLAM methods. It has been demonstrated that Gaussian-based approaches offer significant advantages in terms of both rendering quality and speed.

Implementation Details. We implement our method using PyTorch and evaluate it on a single NVIDIA A100 GPU with 80GB of memory. Further details will be provided in the supplementary material.

4.2. Tracking and Mapping Accuracies

Methods	Metrics	r0	r1	r2	o0	o1	o2	o3	o4	Avg.	FPS
Point-SLAM [22]	PSNR ↑	32.04	34.08	35.50	38.26	39.16	33.99	33.48	33.49	35.00	
	SSIM ↑	0.974	0.977	0.982	0.983	0.986	0.960	0.960	0.979	0.975	1.33
	LPIPS ↓	0.113	0.116	0.111	0.100	0.118	0.156	0.132	0.142	0.124	
GS-SLAM [30]	PSNR ↑	31.56	32.86	32.59	38.70	41.17	32.36	32.03	32.92	34.27	
	SSIM ↑	0.968	0.973	0.971	0.986	0.993	0.978	0.970	0.968	0.976	387
	LPIPS ↓	0.094	0.075	0.093	0.050	0.033	0.094	0.110	0.112	0.083	
SplaTAM [11]	PSNR ↑	32.86	33.89	35.25	38.26	39.17	31.97	29.70	31.81	34.11	
	SSIM ↑	0.980	0.970	0.980	0.980	0.980	0.970	0.950	0.950	0.970	400
	LPIPS ↓	0.070	0.100	0.080	0.090	0.090	0.100	0.120	0.150	0.100	
MonoGS [15]	PSNR ↑	34.83	36.43	37.49	39.95	42.09	36.24	36.70	37.50	37.65	
	SSIM ↑	0.954	0.959	0.965	0.971	0.977	0.964	0.963	0.963	0.964	769
	LPIPS ↓	0.068	0.076	0.075	0.072	0.055	0.078	0.065	0.070	0.070	
Ours	PSNR ↑	36.78	38.85	38.45	43.32	43.73	37.39	37.56	37.45	39.21	
	SSIM ↑	0.970	0.974	0.976	0.985	0.986	0.976	0.973	0.970	0.976	899
	LPIPS ↓	0.042	0.045	0.057	0.028	0.030	0.040	0.038	0.054	0.042	

Table 1. Comparison of different methods based on PSNR, SSIM, and LPIPS metrics across various scenes. For each scene, the top three performances for each metric are highlighted as **first**, **second**, and **third**.

Rendering quality Table 1 presents a quantitative comparison of our approach on the Replica dataset [23] against four baseline methods: Point-SLAM [22], GS-SLAM [30], SplaTAM [11], and MonoGS [15]. The performance metrics—PSNR, SSIM, and LPIPS—are evaluated across eight scenes (room 0–2 and office 0–4). For each metric, the top three results per scene are highlighted, with the highest score marked as **first**, **second**, and **third**. Results from other methods are sourced directly from their original papers.

Our method consistently surpasses the baselines, achieving the highest average scores across all metrics. Specifically, we achieve an average PSNR of **38.45**, SSIM of **0.978**, and LPIPS of **0.042**, showing notable improvements over the other methods. On a per-scene basis, our method ranks first for all but one PSNR score in the office 4

scene. For SSIM, our approach also achieves top scores in most scenes, indicating superior structural similarity in reconstructions compared to baselines. In terms of LPIPS, which reflects perceptual quality, our method consistently produces the lowest scores across all scenes.

Moreover, our approach achieves the highest frame rate (FPS) of **899**, substantially outperforming the competing methods. This high FPS highlights the computational efficiency of our approach, making it well-suited for real-time applications.

The qualitative results shown in Figure 2 further illustrate the effectiveness of our approach compared to baseline methods on the Replica dataset. Across various scenes, our method demonstrates superior rendering fidelity, accurately capturing fine details and structure in challenging areas, as seen in the boxed regions. For example, in complex textures like window blinds and furniture contours, our approach produces clearer and sharper reconstructions than the baselines. Competing methods exhibit noticeable blurring, artifacts, or misalignment in these regions, while our results more closely resemble the ground truth.

Methods	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
iMAP [25]	3.12	2.54	2.31	1.69	1.03	3.99	4.05	1.93	2.58
Vox-Fusion [31]	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94	3.09
NICE-SLAM [34]	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.07
ESLAM [9]	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63	0.63
Point-SLAM [22]	0.61	0.41	0.37	0.38	0.48	0.54	0.69	0.72	0.53
GS-SLAM [30]	0.48	0.53	0.33	0.52	0.41	0.59	0.46	0.70	0.50
SplaTAM [11]	0.31	0.40	0.29	0.47	0.27	0.29	0.32	0.55	0.36
MonoGS [15]	0.44	0.32	0.31	0.44	0.52	0.23	0.17	2.25	0.58
Ours	0.31	0.34	0.28	0.41	0.40	0.26	0.22	2.18	0.55

Table 2. Tracking performance on the Replica dataset [23], measured by Absolute Trajectory Error (ATE) in centimeters.

Tracking Performance. Table 2 compares the tracking performance of our SLAM method with several state-of-the-art methods on the Replica dataset [23]. The metrics presented are tracking errors measured by the average absolute trajectory error (ATE). For each scene, the three best-performing methods are highlighted.

Our method consistently ranks among the top performers, achieving an average tracking error of **0.55**, which places it alongside the leading methods in terms of tracking accuracy. Specifically, our approach is either the first or second-best in all room scenes (r0, r1, r2), and it performs competitively across office scenes. The performance drop in the Office 4 scene can be attributed to the lack of texture in consecutive frames, which results in poor correspondence detection and matching. This limitation highlights the challenges of maintaining high tracking accuracy in low-texture environments, where reliable feature matching becomes more difficult.

Table 3 further demonstrates the robustness of our

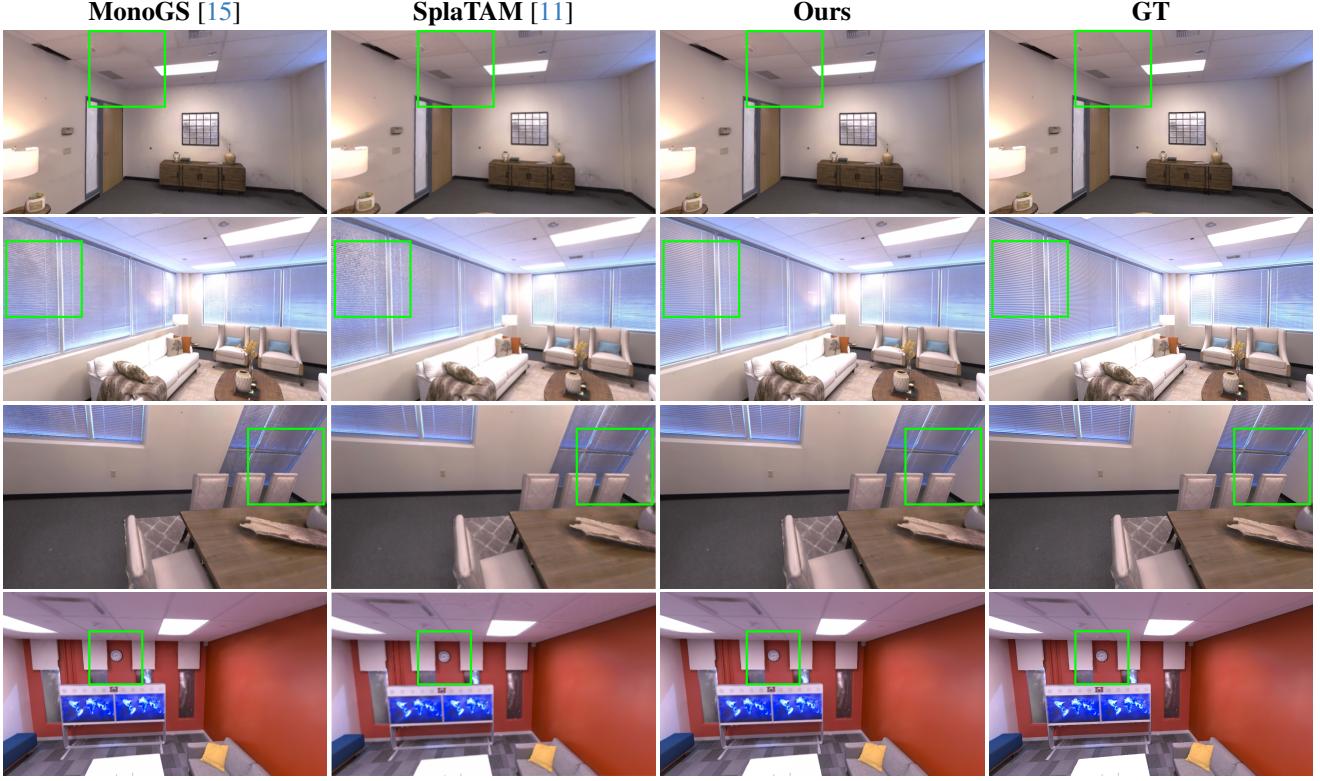


Figure 2. Rendering comparison on Replica dataset.

method on the TUM-RGBD dataset [24], where it achieves competitive results across sequences with an average tracking error of **4.17**, consistently ranking among the top three performers.

Methods	fr1/ desk	fr1/ desk2	fr1/ room	fr2/ xyz	fr3/ office	Avg.
ElasticFusion [29]	2.53	6.83	21.49	1.17	2.52	6.91
ORB-SLAM2 [18]	1.60	2.20	4.70	0.40	1.00	1.98
NICE-SLAM [34]	4.26	4.99	34.49	31.73	3.87	15.87
Vox-Fusion [31]	3.52	6.00	9.43	1.59	26.01	11.31
Point-SLAM [22]	4.34	4.54	30.92	3.23	1.84	8.92
MonoGS * [15]	1.50	6.57	5.21	1.44	1.49	2.70
SplaTAM [11]	3.35	5.64	11.13	1.24	5.16	5.48
Ours	1.48	3.85	11.75	1.81	1.98	4.17

Table 3. Tracking performance on TUM-RGBD [24] (ATE [cm]). Results of the based lines are reported directly from their paper. For MonoGS [15], since performance on **fr1/ desk2** and **fr1/ room** are not reported, we reproduced the numbers by running their official released code.

Tracking in sparse setting Our approach demonstrates high tracking accuracy in sparse environments, where reduced frame density and rapidly changing scenes pose significant challenges. Tables 4 and 5 show tracking perfor-

mance results on the Replica and TUM-RGBD datasets, where our method consistently achieves the lowest average absolute trajectory error (ATE) across various strides compared to baseline methods such as SplaTAM [11] and MonoGS [15]. These baseline results were obtained by running their publicly released code.

In these experiments, the stride represents the number of frames skipped in the original dataset. For example, if a dataset contains 2000 frames and a stride of 20 is used, only the 1st, 20th, 40th frames, and so on are selected, effectively reducing the frame count and increasing sparsity. This setup tests the ability of SLAM methods to handle scenarios with limited frame data.

On the Replica dataset, our method demonstrates superior accuracy, achieving an average ATE of **0.84** with a stride of 10 and maintaining low ATE values as stride increases, showing minimal performance degradation even at high sparsity levels. Similarly, on the TUM-RGBD dataset, our method maintains low ATE values across all strides, achieving an average ATE of **3.07** with a stride of 20 and **3.49** with a stride of 40, significantly outperforming MonoGS [15] and SplaTAM [11], which exhibit higher errors under the same conditions. These baseline methods fail in most cases, with the exception of the **fr2/xyz** scene on TUM-RGBD [24], where the higher frame count and sim-

pler camera motion make tracking easier.

The key strength of our approach lies in its ability to maintain precise tracking in sparse settings, unlike competing methods that experience significant accuracy degradation as frame sparsity increases. This capability makes our SLAM method highly suitable for real-time applications in bandwidth-constrained environments, where computational resources may limit the frequency of frame capture. The consistent performance across both the Replica and TUM-RGBD datasets confirms our approach as a reliable solution for sparse environments in SLAM applications.

Methods	Stride	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
SpliTAM [11]	10	166.4	147.9	78.4	168.2	63.8	68.9	962.6	466.0	265.3
	20	195.0	311.9	231.2	400.4	90.9	112.9	170.1	618.5	744.3
	40	767.9	430.3	117.3	815.8	870.0	506.8	508.7	323.8	542.6
MonoGS [15]	10	37.86	25.02	15.49	14.44	9.68	33.58	29.63	36.05	25.22
	20	92.07	80.95	53.34	67.80	20.93	101.69	89.44	113.69	77.49
	40	93.14	84.24	94.03	78.44	42.13	123.12	121.31	109.36	93.22
Ours	10	0.46	2.17	0.88	0.56	0.84	0.46	0.52	0.85	0.84
	20	0.49	8.14	1.57	2.46	2.80	5.21	0.71	4.96	3.29
	40	0.45	9.53	1.25	23.58	6.10	1.04	0.85	6.29	6.14

Table 4. Tracking performance with different strides on various scenes from the Replica dataset.

Methods	Stride	fr1/desk	fr1/desk2	fr1/room	fr2/xyz	fr3/office	Avg.
SpliTAM [11]	10	295.40	593.82	2206.30	1.38	187.95	656.97
	20	194.59	328.94	1091.26	1.40	3116.74	946.59
	40	185.69	250.32	439.97	1.44	1074.38	390.36
MonoGS [15]	10	73.75	90.90	97.29	0.95	117.88	76.15
	20	81.34	85.08	96.98	0.93	189.52	90.77
	40	84.05	84.65	89.76	9.13	191.63	91.84
Ours	10	1.48	6.87	7.24	1.86	1.76	3.84
	20	1.44	3.60	6.76	1.77	1.78	3.07
	40	1.48	4.25	8.07	1.68	1.98	3.49

Table 5. Tracking performance with sparse setting on the TUM-RGBD [24] dataset.

5. Ablation Studies

Gradient-based pose refinement. We evaluate gradient-based camera tracking refinement by varying the number of tracking iterations, where each iteration involves rendering the scene to optimize the camera pose. Results in Table 6 show that increasing iterations improves accuracy (lower ATE and higher PSNR) but also increases tracking time per frame.

Without refinement, we achieve an ATE of 0.65 cm and PSNR of 35.32 dB in 78 ms per frame. The optimal balance between accuracy and speed occurs at **50** iterations, achieving the lowest ATE (0.55 cm) and highest PSNR (39.21 dB) in 485 ms per frame. Increasing to 70 iterations offers minimal gains in accuracy while significantly raising processing time. Notably, other 3DGS-based methods require 100-200 tracking iterations to reach similar quality in dense settings but still fail completely in sparse settings.

Pose refinement	Track. iter	ATE [cm]↓	PSNR [dB]↑	Track. time [ms / frame]
✗	0	0.65	35.32	78
✓	10	0.62	36.25	310
✓	30	0.59	38.15	401
✓	50	0.55	39.21	485
✓	70	0.56	38.10	512

Table 6. Camera tracking refinement ablations on Replica [23]. The results are averaged.

	Sampling	Replica	TUM
Random		37.62	22.15
Worst first		38.56	22.64
Loss-weighted		39.21	22.85

Table 7. Ablation on color refinement with sampling strategies.

Priority sampling strategies. Table 7 presents an ablation study on color refinement using different sampling strategies. The three strategies evaluated are Random Sampling, Worst-First Sampling, and Loss-Weighted Sampling. Results show that the Loss-Weighted Sampling strategy achieves the highest PSNR on both the Replica and TUM datasets, with values of 39.21 and 22.85 dB, respectively. This indicates that prioritizing frames based on their loss magnitude improves color alignment and overall reconstruction quality more effectively than random or worst-first approaches.

6. Conclusion

We propose a novel SLAM approach that utilizes 3D Gaussian Splatting (3DGS) for real-time 3D scene reconstruction, paired with an efficient vision-based camera tracking strategy. This method delivers high-fidelity reconstructions and robust performance against noisy depth sensor data, making it suitable for consumer-grade cameras. By incorporating pretrained feature matching and point cloud registration, FlashSLAM achieves significant improvements in tracking speed and accuracy, excelling in challenging scenarios such as sparse settings and rapid camera movements. Comprehensive evaluations on benchmark datasets confirm FlashSLAM’s superior accuracy and efficiency compared to existing methods.

Limitations. Our method may fail in scenarios with excessive noise in depth sensor data or non-textured images, as these conditions hinder our vision-based camera tracking approach.

References

- [1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):739–753, 1984.

- tern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987. 3
- [2] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 11, 12
- [3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 11
- [4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 337–33712, 2017. 3
- [5] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006. 1
- [6] David W. Eggert, Adele Lorusso, and Robert B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9:272–290, 1997. 3
- [7] Berthold K. P. Horn, Hugh M. Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opt. Soc. Am. A*, 5(7):1127–1135, 1988. 3
- [8] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photo-realistic mapping for monocular stereo and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21584–21593, 2024. 1
- [9] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17408–17419, 2023. 6
- [10] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, page 61–70, Goslar, DEU, 2006. Eurographics Association. 2
- [11] 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 (CVPR)*, pages 21357–21366, 2024. 1, 2, 3, 6, 7, 8, 11, 12, 13, 14, 15, 16
- [12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1, 2
- [13] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023. 3
- [14] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery. 2
- [15] Hidenobu Matsuki, Riku Murai, Paul H.J. Kelly, and Andrew J. Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18039–18048, 2024. 1, 2, 3, 5, 6, 7, 8, 12, 13, 14, 15, 16
- [16] 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. In *ECCV*, 2020. 2
- [17] Stefan Milz, Georg Arbeiter, Christian Witt, Bassam Abdallah, and Senthil Yogamani. Visual slam for automated driving: Exploring the applications of deep learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 360–36010, 2018. 1
- [18] Raul Mur-Artal and Juan D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33:1255–1262, 2016. 7
- [19] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 2
- [20] 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, 2011. 2
- [21] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3437–3444. IEEE, 2023. 1
- [22] Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 1, 2, 5, 6, 7, 11, 12
- [23] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 5, 6, 8, 12, 13
- [24] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012. 5, 7, 8, 13

- [25] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 1, 2, 6
- [26] Charalambos Theodorou, Vladan Velisavljevic, Vladimir Dyo, and Fredi Nonyelu. Visual slam algorithms and their application for ar, mapping, localization and wayfinding. *Ar-ray*, 15:100222, 2022. 1
- [27] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, page 298–372, Berlin, Heidelberg, 1999. Springer-Verlag. 2
- [28] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 5
- [29] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems*, 2015. 7
- [30] 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 (CVPR)*, pages 19595–19604, 2024. 1, 2, 3, 6
- [31] Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 499–507, 2022. 6, 7, 11
- [32] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 11, 12
- [33] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 5
- [34] 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 (CVPR)*, 2022. 1, 2, 6, 7, 11

FlashSLAM: Accelerated RGB-D SLAM for Real-Time 3D Scene Reconstruction with Gaussian Splatting

flashslam.github.io

Supplementary Material

7. Novel View Synthesis

Figures 3 and 4 showcase the novel view synthesis results on the ScanNet++ dataset [32], highlighting both RGB and depth outputs for two representative scenes. In Figure 3, we present qualitative comparisons for scene IDs 8b5caf3398 (first three columns) and b20a261fdf (last three columns). The top, middle, and bottom rows correspond to results from the baseline method (SplATAM [11]), our proposed method, and the ground truth (GT), respectively. Compared to SplATAM, our method achieves enhanced fidelity in rendering fine details, including edges, textures, and object boundaries, resulting in closer alignment with the ground truth. For instance, in scene 8b5caf3398, our results accurately reconstruct the textures of the red chair and white table, while the baseline suffers from noticeable artifacts and blurring. Similarly, for scene b20a261fdf, our method maintains the structural integrity of the monitors and desks, providing better spatial consistency.

Figure 4 further evaluates novel view synthesis performance by including depth map comparisons for scene b20a261fdf. The left columns depict RGB images, while the right columns show the corresponding depth maps for the same viewpoints. Our method demonstrates robustness to depth noise present in the ground truth. For example, in the first view, there is a discontinuity in the ground truth depth at the pillar, and in the second and last views, noisy depth values are observed on top of the chair. These artifacts result in missing details in SplATAM [11], as evident in the incomplete reconstruction of these regions. In contrast, our method retains the integrity of the reconstruction, producing depth maps with smoother transitions and fewer missing regions. Additionally, our depth predictions better capture object boundaries and spatial coherence, as evident in regions such as the edges of desks and chairs. These results highlight the effectiveness and reliability of our approach in generating both high-fidelity RGB images and accurate depth maps, making it well-suited for novel view synthesis in 3D scene reconstruction.

8. Additional Tracking Results

A comparative evaluation of tracking performance for various SLAM methods is provided on the ScanNet [3] and ScanNet++ [32] datasets, as shown in Tables 8 and 9. These tables highlight the accuracy and robustness of our approach

in challenging scenarios, demonstrating its ability to outperform baseline methods under varying conditions.

Table 8 reports tracking performance on the ScanNet dataset across multiple sequences, including 0000, 0059, 0106, 0169, 0181, and 0207. Our approach achieves the best average ATE (10.33 cm), outperforming competing methods such as NICE-SLAM (10.73 cm) and SplATAM (11.88 cm). Moreover, our method consistently ranks in the top three for all sequences, underlining its robustness and generalizability across diverse scenes with varying complexities.

Table 9 presents the tracking performance on the ScanNet++ dataset, focusing on two scenes (8b5caf3398 and b20a261fdf). SplATAM only reports results for the first 360 frames of b20a261fdf, and we include these results here for reference. In addition to evaluating the first 360 frames of b20a261fdf, we also report results on the full dataset to demonstrate our method’s robustness in handling scenarios with large jumps in camera movement between consecutive frames.

To account for the differing sequence lengths, the average performance is calculated using results for scene 8b5caf3398 combined with the corresponding setting (“360” or “full”) for b20a261fdf. Across both settings, our method consistently delivers strong results, achieving an average ATE of 1.56 cm in the “360” setting and 1.60 cm in the “full” setting. In contrast, SplATAM’s performance degrades significantly when using the full dataset, as it fails to maintain accurate camera tracking under large camera movements. These results underscore our method’s ability to handle both limited-frame scenarios and more complex cases involving irregular camera motion, outperforming SplATAM [11], ORB-SLAM3 [2], and Point-SLAM [22] in both settings.

Methods	0000	0059	0106	0169	0181	0207	Avg.
Vox-Fusion [31]	68.84	24.18	8.41	27.28	23.30	9.41	26.90
NICE-SLAM [34]	12.00	14.00	7.90	10.90	13.40	6.20	10.73
Point-SLAM [22]	10.24	7.81	8.65	22.16	14.77	9.54	12.20
SplATAM [11]	12.83	10.10	17.72	12.08	11.10	7.46	11.88
Ours	9.86	7.80	8.02	14.25	13.36	8.71	10.33

Table 8. Tracking performance on the ScanNet dataset [3], reported using the Absolute Trajectory Error (ATE) metric (cm).

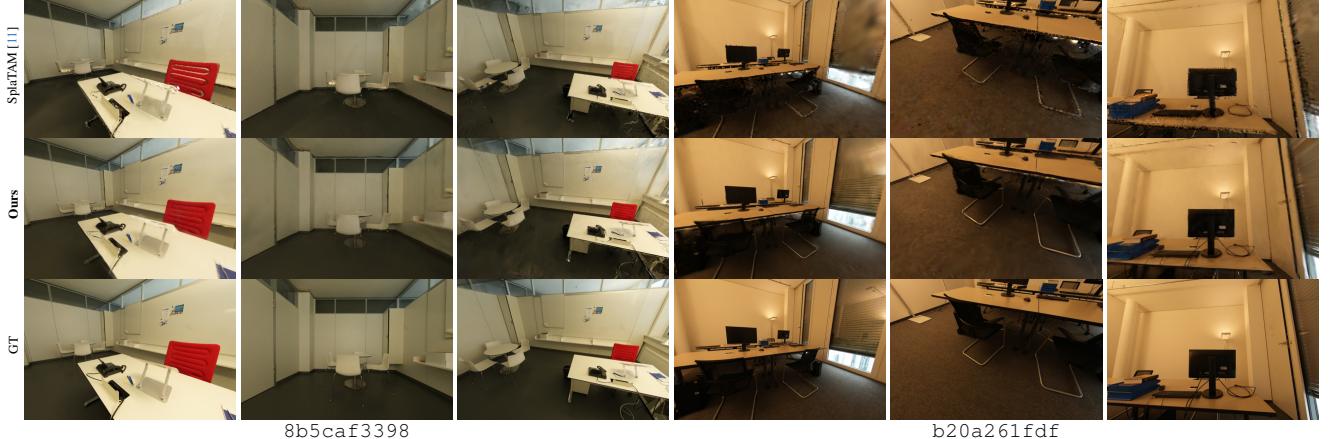


Figure 3. Novel view synthesis results for two scenes from the ScanNet++ dataset [32]. The first three columns show the results for scene ID 8b5caf3398, and the last three columns correspond to scene ID b20a261fdf.



Figure 4. Novel view synthesis results with depth for scene b20a261fdf from the ScanNet++ dataset [32]. The left columns display RGB images, and the right columns show the corresponding depth maps.

Methods	8b5caf3398 full	b20a261fdf 360	b20a261fdf full	Avg. 360	full
Point-SLAM [22]	296.7	390.8	-	343.75	-
ORB-SLAM3 [2]	156.8	159.7	-	158.25	-
Splatam [11]	0.62	1.94	16.09	1.28	8.36
Ours	1.74	1.38	1.46	1.56	1.60

Table 9. Tracking performance on ScanNet++ [32], measured by ATE [cm]. Baseline numbers are taken from [11]. For SplaTAM, results on the full dataset for scene b20a261fdf were computed using their released code. “-” indicates missing or unreported values.

9. Tracking Time Analysis

The results in Tables 10, 11, and 12 highlight the efficiency and robustness of our method in reducing tracking time. On the Replica dataset (Table 10), our method reduces the average tracking time by up to 56% compared to MonoGS [15] and maintains a consistent low tracking time across all scenes. On the TUM dataset (Table 11), our approach achieves an average time reduction of 87% com-

pared to SplaTAM and an impressive 90% reduction in the fr3_office scene.

In sparse settings (Table 12), our method outperforms SplaTAM [11], maintaining low and stable tracking times across varying frame strides while accurately tracking the camera pose. This demonstrates the scalability and reliability of our method in challenging scenarios, as further evidenced by its ability to maintain accuracy in pose tracking (see Tables 4 and 5).

Methods	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
SplaTAM [11]	2.07	2.03	1.68	1.87	1.70	1.36	2.03	2.29	1.88
MonoGS [15]	1.27	1.16	1.16	1.07	0.83	1.00	1.10	1.06	1.08
Ours	0.54	0.48	0.52	0.43	0.42	0.49	0.54	0.46	0.48

Table 10. Tracking time on the Replica dataset [23], measured in seconds.

10. Further Implementation Details

All experiments are conducted on an NVIDIA A100 80GB PCIe GPU to ensure quick and efficient experimentation.

Methods	fr1/ desk	fr1/ desk2	fr1/ room	fr2/ xyz	fr3/ office	Avg.
SplaTAM [11]	3.46	3.02	3.57	3.85	5.16	3.81
MonoGS [15]	0.89	0.93	0.89	0.59	0.81	0.82
Ours	0.50	0.48	0.50	0.51	0.52	0.50

Table 11. Tracking time on the TUM dataset [24], measured in seconds.

Methods	Stride	Replica	TUM
SplaTAM [11]	10	1.77	6.09
	20	2.51	5.35
	40	1.88	3.09
Ours	10	0.45	0.51
	20	0.49	0.51
	40	0.51	0.50

Table 12. Comparison of average tracking time of all scenes in *sparse settings* on the Replica [23] and TUM [24] datasets.

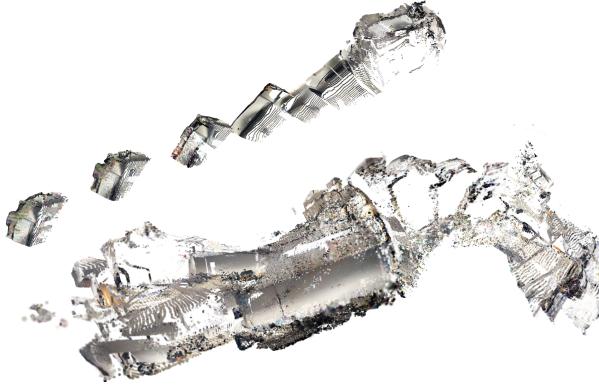


Figure 5. Example of a failed reconstruction by SplaTAM [11] on our custom dataset, showing severe distortion and misaligned geometry due to tracking failures.

However, our method is also compatible with a standard desktop setup equipped with an Intel Core i7-12700K processor and an NVIDIA GeForce RTX 3050 GPU.

For the final reconstruction, we perform 30,000 refinement iterations, which add approximately 3-8 minutes of computation time depending on the number of Gaussians in the scene. The tracking process involves 30-70 steps, while the mapping process requires 100-150 steps, ensuring robust and accurate scene reconstruction.

To facilitate further research and reproducibility, we will release the code along with all configuration details.

11. Rendering Comparisons

The additional visual results presented in Figures 6, 7, and 8 illustrate the rendering quality of different methods on the TUM dataset, specifically on the fr1/desk, fr1/xyz, and fr1/office scenes. Each figure compares the out-

puts of SplaTAM [11], MonoGS [15], and our method against the ground truth (GT).

Our method consistently delivers sharper and more accurate reconstructions across all scenes. In Figure 6 (fr1/desk), our approach captures fine details of objects such as the cables, laptop, and monitor, which are blurred or distorted in the outputs of SplaTAM and MonoGS. Similarly, in Figure 7 (fr1/xyz), our method preserves object boundaries and textures, such as the red soda can, the keyboard and desktop monitor, closely matching the GT. Finally, in Figure 8 (fr1/office), our method demonstrates higher detail retention, as seen in the books, objects on the desk, and smaller items in the scene, where other methods show noticeable blurring or inaccuracies.

12. Evaluation on Self-Captured Dataset

We evaluate our method on a self-captured dataset consisting of 296 images taken with the depth camera of an iPhone 13 Pro Max. The images were captured individually rather than extracted from a video, making the dataset well-suited for sparse settings where the overlap between consecutive frames is limited. Notably, the depth measurements from the iPhone 13 Pro Max are only reliable within a range of up to 3.5 meters, as determined empirically. This limitation introduces significant challenges for reconstruction due to inherent depth noise in the sensor.

Table 13 presents the performance of different methods on this custom dataset. Since both SplaTAM and MonoGS fail to track camera poses, their greyed-out results are invalid and included solely for reference. Figure 5 provides an example of a failed reconstruction by SplaTAM, where the scene is heavily distorted, and the geometry is misaligned, emphasizing the challenges caused by depth noise and tracking failures. In contrast, our method successfully tracks the camera poses and delivers significantly better reconstruction metrics.

Methods	Success	ATE (cm) ↓	PSNR ↑	SSIM ↑	LPIPS ↓
SplaTAM	✗	1207	16.15	0.572	0.622
MonoGS	✗	1970	13.61	0.584	0.634
Ours	✓	19.59	23.73	0.781	0.277

Table 13

Figure 9 highlights a rendering comparison on the self-captured dataset. Although SplaTAM and MonoGS produce rendered images that appear visually reasonable, their inability to accurately track the camera poses leads to completely incorrect scene reconstructions. Our method, however, achieves accurate camera pose tracking, resulting in a reconstruction that closely aligns with the ground truth. For more visual results and videos of the reconstructed scenes, please visit our project page at flashslam.github.io.



Figure 6. Rendering comparison on TUM fr1/desk.

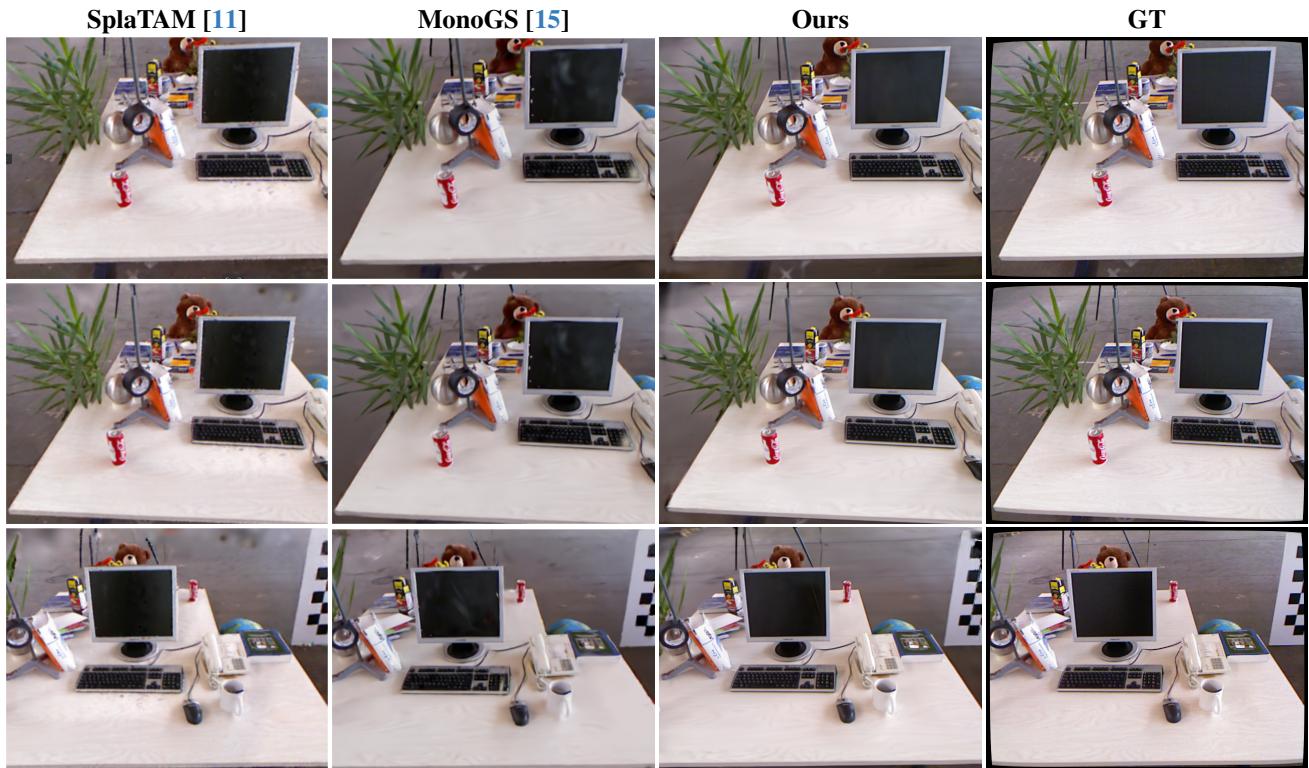


Figure 7. Rendering comparison on TUM fr1/xyz.

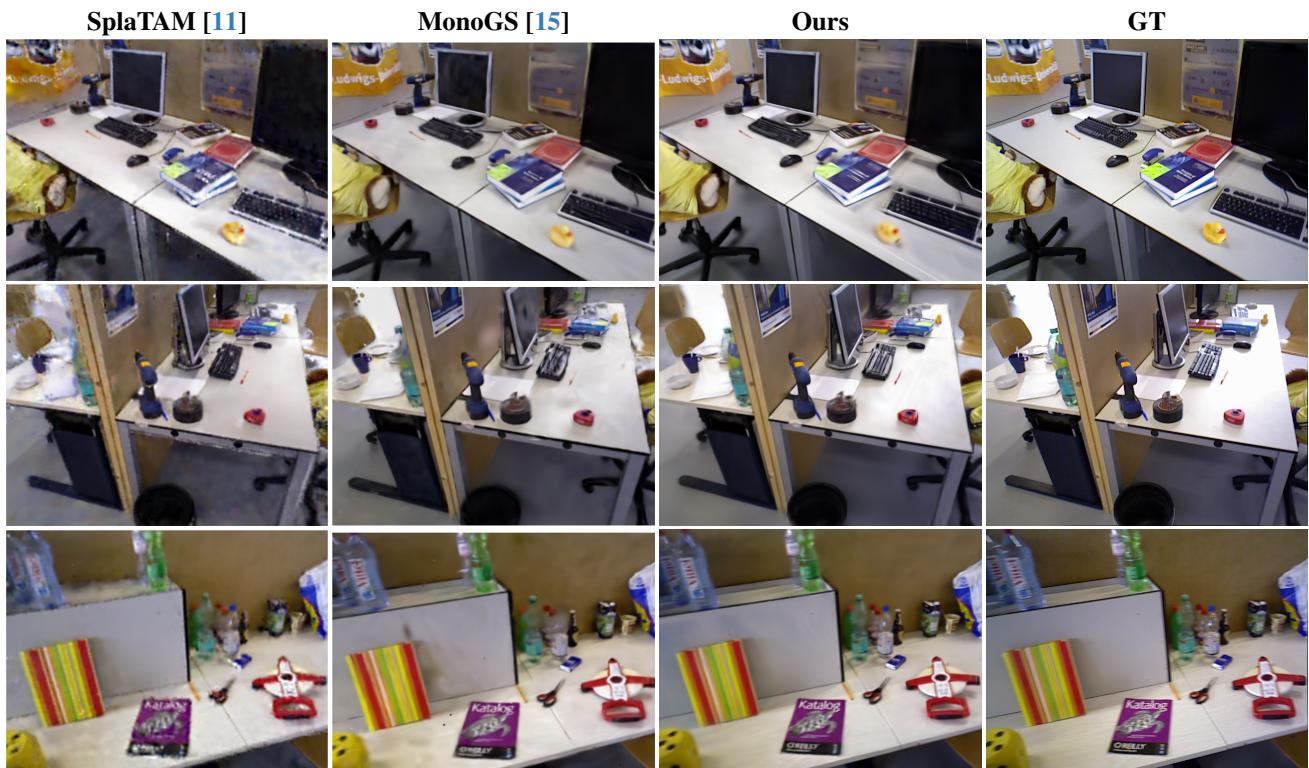


Figure 8. Rendering comparison on TUM fr1/office.

