MonoGS++: Fast and Accurate Monocular RGB Gaussian SLAM

Ren-Wu Li
renwu.li@amd.com
Wenjing Ke
wenjing.ke@amd.com
Dong Li
d.li@amd.com
Lu Tian
lu.tian@amd.com
Emad Barsoum
emad.barsoum@amd.com

Advanced Micro Devices, Inc.
Beijing, China

Abstract

We present MonoGS++, a novel fast and accurate Simultaneous Localization and Mapping (SLAM) method that leverages 3D Gaussian representations and operates solely on RGB inputs. While previous 3D Gaussian Splatting (GS)-based methods largely depended on depth sensors, our approach reduces the hardware dependency and only requires RGB input, leveraging online visual odometry (VO) to generate sparse point clouds in real-time. To reduce redundancy and enhance the quality of 3D scene reconstruction, we implemented a series of methodological enhancements in 3D Gaussian mapping. Firstly, we introduced dynamic 3D Gaussian insertion to avoid adding redundant Gaussians in previously well-reconstructed areas. Secondly, we introduced clarity-enhancing Gaussian densification module and planar regularization to handle texture-less areas and flat surfaces better. We achieved precise camera tracking results both on the synthetic Replica and real-world TUM-RGBD datasets, comparable to those of the state-of-the-art. Additionally, our method realized a significant 5.57x improvement in frames per second (fps) over the previous state-of-the-art, MonoGS [8].

1 Introduction

Simultaneous Localization and Mapping (SLAM) technologies play a pivotal role in the realm of robotics and augmented reality. Conventional visual SLAM systems often face challenges with sparse and incomplete scene reconstructions, which are limited by the use of sparse point clouds. This limitation has propelled the development of methods like Neural Radiance Fields (NeRF), which facilitate dense and continuous scene reconstructions. However, the high computational load and the latency in scene updates inherent in NeRF-based systems often offset their benefits.

A promising alternative to NeRF, 3D Gaussian Splatting (3D GS [7]), offers a flexible, point-based scene representation through the use of 3D Gaussian distributions. This technique has been integrated into recent neural dense visual slam system such as SplaTAM [6] and Gaussian Splatting SLAM (MonoGS [8]), which have shown significant improvements both in speed and mapping capabilities compared to NeRF-based SLAM. However, these previous 3D GS-based approaches still face two main limitations. First, since the original 3D GS is initialized by the offline reconstructed structure from motion (SFM) point cloud, how to initialize the 3D Gaussian map is challenging for an online SLAM system. As depicted in Figure 1 a), previously mentioned 3D GS-based systems heavily rely on RGB-D images as input, from which local point clouds are back-projected and by the way local 3D Gaussians are initialized from current viewpoint. The dependency on depth sensor considerably narrows their applicability. Second, to obtain the online camera poses for merging local 3D Gaussians, the existing 3D GS-based SLAM methods often integrate pose optimization into overall optimization process as shown in Figure 1 a), which burdens the system and slows down the overall latency.

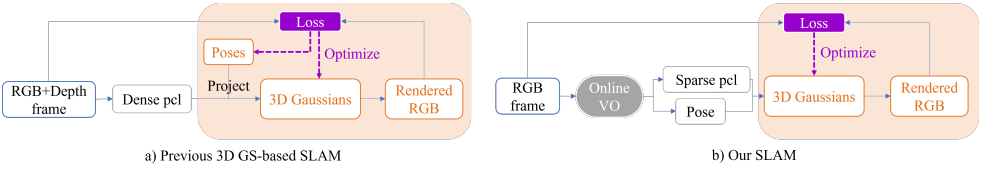

Figure 1: The pipeline of previous 3D GS-based methods and our method.

To address the above two limitations, we introduced MonoGS++, a monocular RGB Gaussian SLAM taking pure monocular RGB as input, as shown in Figure 1 b). We separated pose optimization from the overall network optimization, using visual odometry (VO) to derive online sparse point clouds and camera poses, and focused on enhancing the quality of 3D Gaussian mapping. With this design improvement and innovative enhancements in the mapping module, we developed a fast and accurate monocular RGB 3D Gaussian SLAM system.

Our main contributions can be summarized as follows. (1) We proposed dynamic 3D Gaussian insertion to avoid adding redundant 3D Gaussians in previously well-reconstructed areas. (2) We proposed the clarity-enhancing Gaussian densification module and planar regularization to better handle texture-less areas and flat surfaces. (3) We achieved precise camera tracking results both on the synthetic Replica and real-world TUM-RGBD datasets, comparable to those of the state-of-the-art. Our tracking accuracy and rendering quality on the TUM-RGBD dataset surpass those of MonoGS [8], with improvements of +5.21dB in PSNR, +0.18 in SSIM, and a reduction of 28.85cm in ATE. Furthermore, our method achieves a 5.57x increase in fps compared to MonoGS.

2 Related Works

Implicit Neural Scene Representation. Neural Radiance Fields (NeRF)[9] and its follow-ups have transformed 3D scene representation by encoding scenes as continuous implicit fields using MLPs. These techniques excel in novel view synthesis and reconstruction, adept

at capturing intricate geometries and unseen regions. The NeRF-based SLAM approach iMap[13] integrates NeRF into SLAM, jointly optimizing camera poses and the implicit MLP. However, vanilla NeRF [9] requires excessive training time and is prone to overfitting. NICE-SLAM [17] and NICER-SLAM [18] address this by using feature-dense grids and a pretrained encoder, achieving faster convergence and more accurate reconstructions. Other methods [5, 15] accelerate training and rendering with explicit 3D voxels or hash grids, but rely on RGB-D data.

Point-based Neural Dense Visual SLAM. Point-based scene representations have shown potential in neural dense visual SLAM. Point-SLAM [10] employs a neural point cloud for concurrent tracking and mapping, ensuring accurate 3D reconstructions. 3D Gaussians Splatting (3D GS)[7], a general point-based representation, is utilized in works like SplaTAM[6], and MonoGS [8], delivering impressive results in tracking and mapping. Nonetheless, these methods often require depth data for initializing 3D Gaussians and are incompatible with monocular setups. While neural rendering in 3D Gaussian Splatting is fast, the iterative optimization in these SLAM frameworks can still slow down the overall performance.

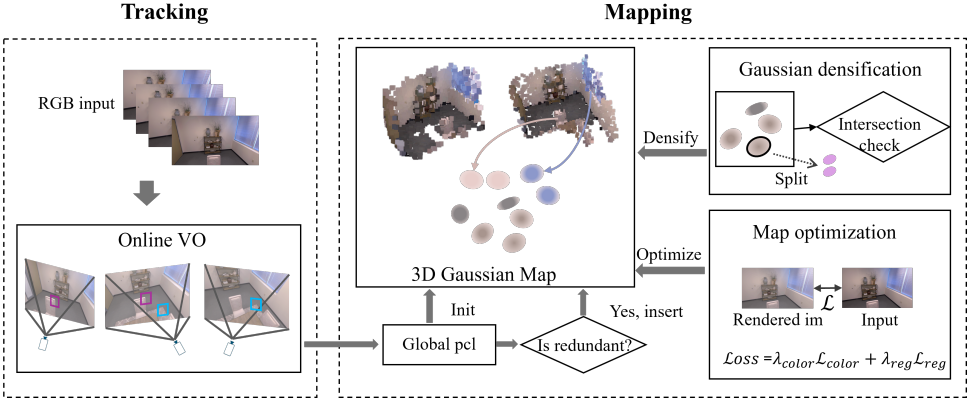

Figure 2: Overview of the SLAM system.

3 Methodology

As shown in Figure 2, our SLAM system mainly includes two parts: tracking and mapping. In tracking phase, we employ a deep patch-based visual odometry that is both efficient and lightweight, facilitating the estimation of camera poses alongside a set of optimized 3D sparse patches. In mapping phase, the scene is represented as a set of 3D Gaussians, which grows progressively and dynamically.

3.1 Scene Representation

Our system models the scene using a set of 3D Gaussians, each characterized by a mean vector $\mu \in \mathbb{R}^3$, which denotes position, and a covariance matrix Σ, defining its spatial distri-

bution:

$$G(x) = \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \quad \Sigma = RSS^TR^T \tag{1}$$

Gaussians are further parameterized by color coefficients $c \in \mathbb{R}^k$, orientation via quaternion $r \in \mathbb{R}^4$, scale vector $s = \text{diag}(S) \in \mathbb{R}^3$, and opacity $\alpha \in \mathbb{R}$.

In the Gaussian Splatting process, these 3D Gaussians are projected onto the image plane, transitioning to 2D representations. The 2D covariance matrix Σ^{2D} is calculated as:

$$\Sigma^{2D} = JW\Sigma W^TJ^T \tag{2}$$

where J is the Jacobian matrix, and W denotes the viewing transformation matrix.

The image is then synthesized by alpha-blending the colors and opacities of intersected Gaussians at each pixel $\mathbf{p} = (u,v)$:

$$C(\mathbf{p}) = \sum_{i=1}^{N} c_i(\mathbf{p})\alpha_i(\mathbf{p})\prod_{j=1}^{i-1}(1-\alpha_j(\mathbf{p})) \tag{3}$$

Here, $c_i(\mathbf{p})$ and $\alpha_i(\mathbf{p})$ represent the color and opacity of each Gaussian intersected by the ray from pixel $\mathbf{p}$, respectively.

3.2 SLAM

3.2.1 Tracking

We utilize Deep Patch Visual Odometry (DPVO [14]), a learning-based, sparse monocular odometry method inspired by DSO [2]. For each RGB keyframe I_i, DPVO samples K square patches of size p, each parameterized in homogeneous coordinates as $P_k^i = [u,v,1,d]^\top$, where d represents the inverse depth, assumed constant across the patch.

DPVO constructs a patch graph with edges $\mathcal{E} = (i,j,k)$, where each edge indicates a trajectory of patch P_k^i from I_i to I_j. I_j is within the local optimization window $\mathcal{N}(i)$. The projection of P_k^i to I_j, denoted as $P_k^{j\leftarrow i}$, is derived from:

$$P_k^{j\leftarrow i} \sim KT_jT_i^{-1}K^{-1}P_k^i, \tag{4}$$

Camera poses and inverse depths are refined by differentiable bundle adjustment. A recurrent network predicts patch trajectory updates $\delta_{kj} \in \mathbb{R}^2$ and confidence weights $\Sigma_{kj} \in \mathbb{R}^2$ for each edge (i,j,k) in the graph, aiming to minimize the Mahalanobis distance:

$$\sum_{(i,j,k)\in\mathcal{E}} \left\| KT_jT_i^{-1}K^{-1}\hat{P}k^i - \hat{P}k^{j\leftarrow i} \right\|^2_{\Sigma_{kj}}, \tag{5}$$

For each new coming keyframe, the tracking module will optimize current camera pose and update the inverse depths of available patches.

3.2.2 Mapping

Map Initialization. The initialization of the 3D Gaussian map, denoted as $\mathcal{G}$, commences following the completion of the differential visual odometry's initialization stage, which spans N frames. Contrary to traditional methods that employ depth sensors to back-project

depth maps for initializing 3D Gaussians, our approach utilizes the centers of optimized patches from the initial N frames. These centers are back-projected to form a global point cloud $\mathcal{P}$ in the world coordinate system, as described by the following equation:

$$\mathcal{P} = \{T_i^{-1} K^{-1} \hat{P}_k^i \mid i \leq N, k \leq K\}. \tag{6}$$

Subsequently, the map $\mathcal{G}$ is generated from $\mathcal{P}$, with the total number of 3D Gaussians given by $|\mathcal{G}| = N \times K$.

Dynamic 3D Gaussian Insertion. The Gaussian map $\mathcal{G}$ undergoes progressive enlargement by integrating each new keyframe I_j along with its optimized camera pose T_j, as derived from the tracking module. This integration process involves back-projecting patches to update and augment the existing point cloud $\mathcal{P}$ by the updated point cloud $\mathcal{P}' = \{T_i^{-1} K^{-1} \hat{P}_k^i \mid i \leq j, k \leq K\}$. Instead of generating a new 3D Gaussian for every point in the updated point cloud $\mathcal{P}'$, the necessity of each point is evaluated. Points situated within well-reconstructed regions are considered redundant and excluded. The inclusion of a point as the center of a new 3D Gaussian is determined by measuring the distance between each point in $\mathcal{P}'$ and the means of existing 3D Gaussians. Points with distances surpassing a threshold τ are selected for the creation of new 3D Gaussians:

$$\mathcal{P} = \mathcal{P} \cup \{q \mid d(q, \mathcal{P}) < \tau, q \in \mathcal{P}'\}, \quad d(q, \mathcal{P}) = \min_{p \in \mathcal{P}} \|q - p\|_2. \tag{7}$$

Clarity-Enhancing Gaussian Densification. In the original 3DGS [7], the 3D Gaussians are densified with the guidance of pixel rendering gradient, 3D Gaussians with high gradients are cloned or split depending on their 3D scales. As discussed in [3], the gradient-based densification may fail in areas with smooth textures and the rendered pixel is dominated by the 3D Gaussian with the largest alpha blending weight. Therefore, we further perform 3D Gaussian densification guided by rendering the most dominated Gaussian of each pixel to enhance the clarity of smooth colored regions. Specifically, for each pixel $\mathbf{p} = (u, v)$ in the current frame I_i, let $\mathcal{G}_{\mathbf{m}}(\mathbf{p})$ denote the 3D Gaussian with the largest alpha blending weight:

$$w_m = \alpha_m(\mathbf{p}) \prod_{n=1}^{m-1} (1 - \alpha_n(\mathbf{p})), \quad \mathbf{m} = \max_m w_m. \tag{8}$$

Subsequently, we compute the intersected rendered pixels between $\mathcal{O}_i(n)$ and $\mathcal{O}_i(\mathbf{m})$ to produce a split mask $\mathbb{M}$:

$$\mathcal{M}(n) = \mathbb{1}(|\mathcal{O}_i(n) \cap \mathcal{O}i(\mathbf{m})| > \sigma), \tag{9}$$

where σ is the split threshold. Gaussians $\mathcal{G}_n(\mathbf{p})$ for which $\mathcal{M}(n) = 1$ are then split to enhance the representation's fidelity and clarity in texture-smooth areas.

Map Optimization with Planar Regularization. The Gaussian map is optimized by minimizing the discrepancy between the rendered image $\hat{I}_i$ and the original I_i. According to GaussianShader [4] and GaussianPro [1], 3D Gaussians naturally tend to flatten and approximate planar surfaces during optimization. This characteristic is particularly advantageous for representing thin and flat structures such as walls, tables, and floors. Consequently, beyond the standard photometric loss $\mathcal{L}color$, we have integrated a planar regularization term

$\mathcal{L}reg$. This term promotes the flattening of 3D Gaussian planes by specifically minimizing the smallest scale dimension across the three axes. And the objective functions is defined as:

$$\mathcal{L}_{color} = (1-\lambda_{photo}) \cdot \mathcal{L}_{photo}(\hat{I}_i, I_i) + \lambda_{photo} \cdot \mathcal{L}_{SSIM}(\hat{I}_i, I_i), \quad \mathcal{L}_{reg} = |\max(0.01, \min(s))| \tag{10}$$

$$\mathcal{L} = \lambda_{color} \cdot \mathcal{L}_{color} + \lambda_{reg} \cdot \mathcal{L}_{reg} \tag{11}$$

where $\mathcal{L}_{photo}$ denotes the photometric loss (i.e., the L1 loss), and $\mathcal{L}_{SSIM}$ signifies the structural similarity index measure. The weighting parameter λ_{photo} is empirically set to 0.2, consistent with the Gaussian Splatting approach as delineated in [7].

4 Experiments

4.1 Experimental Setting

Datasets. We evaluate our method using both synthetic and real datasets, specifically Replica [11] and TUM-RGBD [12]. The Replica dataset presents fewer challenges in the RGB-D context due to its high-quality depth maps and minimal frame-to-frame displacement. However, it introduces significant challenges for monocular setups due to its textureless surfaces and purely rotational movements. In contrast, TUM-RGBD is a more challenging real-world dataset that poses more difficulties due to motion blur and noise resulting from the use of outdated, low-quality cameras. Additionally, the captured depth images are noisy and contain holes, which complicates processing with RGB-D based methods.

Metrics. We assess the performance of our method using RMSE of ATE for tracking accuracy and photometric metrics like PSNR, SSIM, and LPIPS for reconstruction quality.

Baselines. We compare our approach with state-of-the-art neural dense visual slam methods, including NICE-SLAM [17], Vox-Fusion [16], and Point-SLAM [10], as well as recent 3D GS-based methods such as SplaTAM [6] and MonoGS [8]. Notably, MonoGS [8], which supports both monocular RGB and RGB-D modes, serves as our primary baseline for comparison. For a fair comparison, we reproduced the experiments for SplatAM, PointSLAM, and MonoGS and recorded the results in the experimental tables and visualization figures. All baseline methods rely exclusively on RGB-D data inputs, while only MonoGS (RGB) [8] and our method accept monocular RGB data as input.

4.2 Evaluation

Results on Replica dataset. The results on Replica dataset are presented in Table 1. This table shows that our method not only achieves the most accurate camera tracking outcomes but also obtains comparable results in rendering quality. Our approach achieves better PSNR scores on average. Additionally, our method delivers competitive performance in SSIM and LPIPS. Compared to MonoGS (RGB), which uses the same RGB input as we do, our performance metrics significantly surpass theirs across all indicators. Since most methods achieve good accuracy, we focused on detailed comparisons with MonoGS, which has the highest accuracy among them. As shown in Figure 3, our method performed better on planar details like table edges and blinds due to effective planar regularization. In weakly textured areas such as carpets, MonoGS's results were blurry, whereas ours retained more detail.

Method	Modality	Metric	room0	room1	room2	office0	office1	office2	office3	office4	Avg.
NICE-SLAM	RGB-D	PSNR[dB]↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42
		SSIM↑	0.68	0.75	0.81	0.87	0.88	0.79	0.80	0.85	0.80
		LPIPS↓	0.33	0.27	0.20	0.22	0.18	0.23	0.20	0.19	0.233
		ATE-MSE (cm)↓	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.07
Vox-Fusion	RGB-D	PSNR[dB]↑	22.39	22.36	23.92	27.79	29.83	20.33	23.47	25.21	24.41
		SSIM↑	0.68	0.75	0.79	0.85	0.87	0.79	0.80	0.84	0.80
		LPIPS↓	0.30	0.26	0.23	0.24	0.18	0.24	0.21	0.19	0.236
		ATE-MSE (cm)↓	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94	3.09
Point-SLAM	RGB-D	PSNR[dB]↑	32.40	34.08	35.5	38.26	39.16	33.39	33.48	33.49	35.17
		SSIM↑	0.97	0.97	0.98	0.98	0.98	0.96	**0.96**	**0.97**	0.97
		LPIPS↓	0.11	0.11	0.11	0.1	0.11	0.15	0.13	0.14	0.12
		ATE-MSE (cm)↓	0.61	0.41	0.37	0.38	0.48	0.54	0.69	0.72	0.53
SplaTAM	RGB-D	PSNR[dB]↑	32.86	33.89	35.25	38.26	39.17	31.97	29.70	31.81	34.11
		SSIM↑	**0.98**	**0.97**	**0.98**	**0.98**	**0.98**	**0.97**	0.95	0.95	**0.97**
		LPIPS↓	0.07	0.10	0.08	0.09	0.09	0.10	0.12	0.15	0.10
		ATE-MSE (cm)↓	0.31	0.40	0.29	0.47	0.27	**0.29**	0.32	0.55	0.36
MonoGS	RGB-D	PSNR[dB]↑	**34.83**	36.43	**37.49**	39.95	42.09	**36.24**	**36.70**	37.06	37.50
		SSIM↑	0.95	0.95	0.96	0.97	0.97	0.96	0.96	0.95	0.96
		LPIPS↓	**0.068**	**0.076**	**0.075**	0.072	**0.055**	**0.078**	**0.065**	0.099	**0.070**
		ATE-MSE (cm)↓	0.47	0.43	0.31	0.70	0.57	0.31	0.31	3.2	0.79
MonoGS	RGB	PSNR[dB]↑	28.94	26.12	31.82	32.73	34.47	27.01	30.76	27.29	29.89
		SSIM↑	0.88	0.80	0.92	0.92	0.93	0.88	0.91	0.90	0.89
		LPIPS↓	0.18	0.32	0.16	0.21	0.19	0.26	0.16	0.25	0.22
		ATE-MSE (cm)↓	5.87	29.47	6.53	23.02	15.93	20.89	3.98	43.85	18.69
Ours	RGB	PSNR[dB]↑	33.75	**36.47**	37.01	**42.31**	**43.05**	36.11	36.34	**37.28**	**37.79**
		SSIM↑	0.94	0.96	0.96	0.98	0.97	0.95	0.96	0.96	0.96
		LPIPS↓	0.092	0.076	0.077	**0.052**	0.064	0.090	0.078	**0.086**	0.077
		ATE-MSE (cm)↓	**0.20**	**0.17**	**0.22**	**0.29**	**0.13**	0.42	**0.20**	**0.42**	**0.26**

Table 1: Quantitative results and comparison of SLAM method metrics on Replica dataset. The best results are shown in **bold**, while the second-best results are underlined.

This demonstrates the efficacy of our clarity-enhancing Gaussian densification in splitting 3D Gaussians on such regions, enhancing texture detail reproduction.

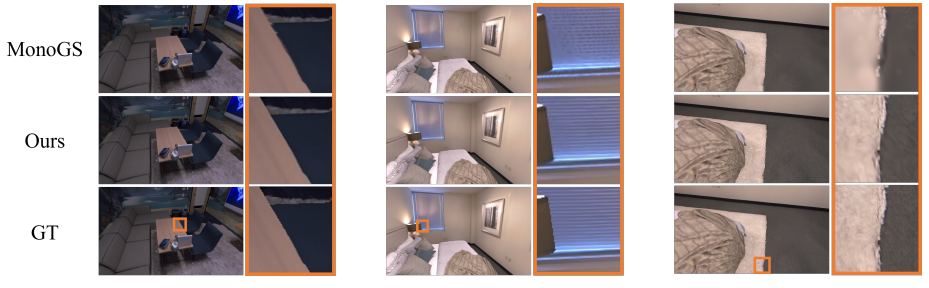

Figure 3: Rendering results on Replica dataset. Orange boxes highlight magnified details to emphasize quality differences.

Results on TUM-RGBD dataset. The results on Replica dataset are presented in Table 2. Our method achieves higher PSNR and LPIPS scores on all sequences and the second highest SSIM scores minimally lower than SplaTAM by 0.02. Regarding camera tracking accuracy, our method achieves better results on 3 sequences. However, it showed a significant discrepancy compared to SplaTAM on the fr1/room. Upon analysis, this scene involved a dynamically moving pedestrian and rapid camera motion, which led to instability in the tracking with purely RGB-based VO. It's worth noting that although MonoGS supports both RGB and RGB-D inputs, the experimental results indicate that it exhibits significant instability across different data qualities and input configurations. In Figure 4, SplaTAM displays imperfections along object edges like chessboards and metal rods. PointSLAM often show excessive noise, especially in small background objects. Meanwhile, MonoGS (RGB) gen-

Method	Modality	Metric	fr1/desk	fr1/desk2	fr1/room	fr2/xyz	fr3/office	Avg.
NICE-SLAM	RGB-D	PSNR[dB]↑	13.83	12.00	11.39	17.87	12.89	13.59
		SSIM↑	0.56	0.51	0.37	0.71	0.55	0.54
		LPIPS↓	0.48	0.52	0.62	0.34	0.49	0.49
		ATE-MSE (cm)↓	4.26	4.99	34.49	31.73	3.87	15.87
Vox-Fusion	RGB-D	PSNR[dB]↑	15.79	14.12	14.20	16.32	17.27	15.54
		SSIM↑	0.64	0.56	0.56	0.70	0.67	0.63
		LPIPS↓	0.52	0.54	0.55	0.43	0.45	0.50
		ATE-MSE (cm)↓	3.52	6.00	19.53	1.49	26.01	11.31
Point-SLAM	RGB-D	PSNR[dB]↑	13.87	14.12	14.16	17.56	18.43	15.63
		SSIM↑	0.62	0.59	0.64	0.70	0.75	0.66
		LPIPS↓	0.54	0.56	0.54	0.58	0.44	0.53
		ATE-MSE (cm)↓	4.34	**4.54**	30.92	1.31	3.48	8.92
SplaTAM	RGB-D	PSNR[dB]↑	21.16	19.26	18.73	23.11	19.92	20.44
		SSIM↑	**0.87**	**0.80**	**0.78**	**0.90**	0.82	**0.83**
		LPIPS↓	0.24	0.33	0.33	0.21	0.34	0.29
		ATE-MSE (cm)↓	3.35	6.54	**11.86**	1.34	5.41	**5.70**
MonoGS	RGB-D	PSNR[dB]↑	9.99	8.90	8.95	12.46	15.95	11.25
		SSIM↑	0.36	0.31	0.46	0.71	0.46	0.46
		LPIPS↓	0.70	0.71	0.60	0.30	0.74	0.61
		ATE-MSE (cm)↓	20.21	90.92	104.32	1.47	104.88	64.36
MonoGS	RGB	PSNR[dB]↑	17.31	14.06	14.76	22.06	23.02	18.24
		SSIM↑	0.65	0.50	0.52	0.72	0.78	0.63
		LPIPS↓	0.38	0.62	0.60	0.27	0.32	0.43
		ATE-MSE (cm)↓	3.05	79.45	84.78	4.31	1.85	34.68
Ours	RGB	PSNR[dB]↑	**22.85**	**20.64**	**22.16**	**26.52**	**25.08**	**23.45**
		SSIM↑	0.82	0.77	0.77	0.86	**0.85**	0.81
		LPIPS↓	**0.20**	**0.29**	**0.31**	**0.13**	**0.18**	**0.22**
		ATE-MSE (cm)↓	**1.79**	5.18	21.44	**0.38**	**0.36**	5.83

Table 2: Quantitative results and comparison of SLAM method metrics on TUM-RGBD dataset. The best results are shown in **bold**, while the second-best results are underlined.

erally results in blurry effects. In contrast, our method delivered superior rendering quality, excelling in weakly textured areas such as desktops and floors, and capturing fine details like the serrations on metal rods.

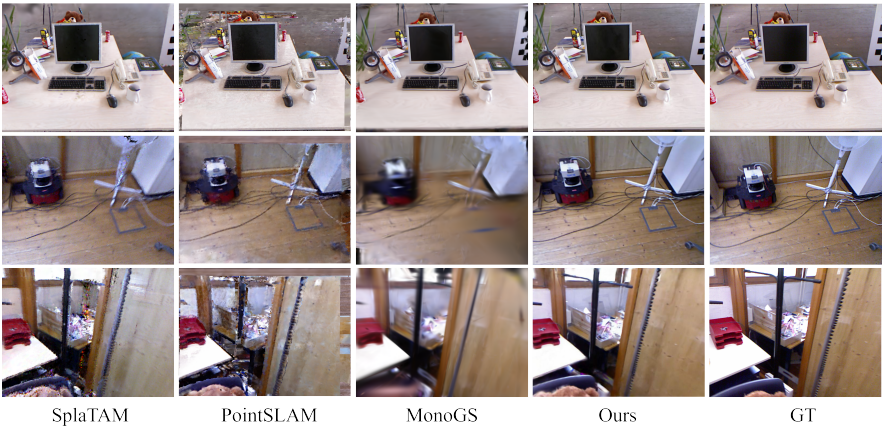

Figure 4: Rendering results on the TUM-RGBD dataset.

4.3 Runtime and Efficiency

In Table 3, we present a comprehensive analysis of runtime statistics, comparing our method with several baselines. All experimental results were obtained running on the same hardware configuration. Among these baselines, SplaTAM, Point-SLAM and MonoGS all

Method	Single Process	Tracking / Frame	Mapping / Frame	FPS ↑	ATE ↓
Point-SLAM		8.20s	40.80s	0.060	0.61
SplaTAM		3.13s	5.33s	0.115	0.31
MonoGS		1.47s	3.05s	0.445	0.47
MonoGS(RGB)		3.40s	11.60s	0.152	5.87
MonoGS*	✓	0.74s	15.62s	0.246	0.39
Ours	✓	**0.067s**	**0.27s**	**2.48**	**0.20**

Table 3: Runtime analysis on Replica/Room0. All experimental results were obtained running on the same hardware configuration.

utilize multi-processing to enhance performance, whereas MonoGS* operates within a single process. As shown in Table 3, our method not only supports real-time camera tracking but also achieves the most accurate tracking results. Significantly, although our approach is implemented in a single process fashion the same as MonoGS*, our system achieves the highest FPS, operating 5.57 times faster than MonoGS, 16.32 times faster than MonoGS taking RGB as input and 10.08 times faster than MonoGS*. This performance demonstrates our approach's superior efficiency and prospects in real-time applications.

	Dynamic Gaussian Insertion	Clarity-Enhancing Densification	Planar Regularization	PSNR↑	SSIM↑	LPIPS↓	# of Gaussians	FPS↑
(A)		✓	✓	36.85	0.963	0.080	1680278	1.76
(B)	✓		✓	35.88	0.955	0.110	769568	2.71
(C)	✓	✓		**37.07**	0.963	0.083	1123479	2.85
(D)	✓	✓	✓	<u>37.01</u>	**0.963**	**0.077**	1143534	2.84

Table 4: Ablation study on Replica/Room2. We introduce four variants of our approach, designated as (A), (B), (C) and (D). A deterioration in accuracy is observed upon the removal of any proposed component.

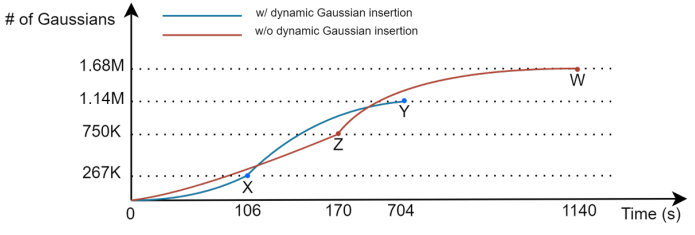

Figure 5: Estimated number of Gaussians in the system over the time series. Variant (A) in red and (D) in blue.

4.4 Ablation Study

To validate the effectiveness of each component of our method, we conducted experiments with four variations: (A) without the dynamic Gaussian insertion strategy; (B) without the clarity-enhancing Gaussian densification module; (C) without the planar regularization term; and (D) with all components enabled. The results validated on the Replica-Room2 sequence are presented in Table 4. The dynamic Gaussian insertion primarily enhances efficiency, the clarity-enhancing Gaussian densification module significantly improves overall rendering quality, and the planar regularization term mainly boosts the LPIPS value. Besides, Figure 5 further analyzes the dynamic Gaussian insertion's impact. With dynamic Gaussian insertion, the number of Gaussians prior to refinement is nearly one-third of that without it, facilitating faster optimization and achieving higher frame rates (2.84 fps vs. 1.76 fps).

5 Conclusion

We have proposed MonoGS++, a fast and accurate monocular RGB Gaussian SLAM taking pure RGB as input and performing 3D Gaussian mapping. This system achieves comparable accuracy with SOTA RGB-D-dependent methods and surpasses them on system efficiency by a large margin, promising broader applications in robotics and augmented reality. Future efforts will be focused on enhancing its robustness and adaptability in more challenging scenes with motion blur and dynamic objects moving.

References

[1] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. arXiv preprint arXiv:2402.14650, 2024.

[2] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In arXiv:1607.02565, July 2016.

[3] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians, 2024.

[4] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. arXiv preprint arXiv:2311.17977, 2023.

[5] M. M. Johari, C. Carta, and F. Fleuret. ESLAM: Efficient dense slam system based on hybrid representation of signed distance fields. In Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR), 2023.

[6] 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, 2024.

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics, 42(4), July 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[8] 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, 2024.

[9] 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.

[10] 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.

[11] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797, 2019.

[12] 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), Oct. 2012.

[13] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6229–6238, 2021.

[14] Zachary Teed, Lahav Lipson, and Jia Deng. Deep patch visual odometry. Advances in Neural Information Processing Systems, 2023.

[15] 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. In 2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pages 499–507, 2022.

[16] 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. In 2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pages 499–507. IEEE, 2022.

[17] 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.

[18] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. arXiv preprint arXiv:2302.03594, 2023.