2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

AMG-SLAM: Adaptive Monocular Gaussian SLAM for Efficient Surface Reconstruction

Youqi Pan[1], Wugen Zhou[1], and Hongbin Zha[1,2]

***Abstract*— Dense SLAM with a monocular camera remains a highly challenging task. In this paper, we present AMG-SLAM, a novel dense monocular SLAM system that tightly couples sparse tracking with dense Gaussian mapping to achieve fully online and high-quality surface reconstruction. In the front-end, learning-based modules enable efficient pose tracking and Gaussian proposal with sparse depth initialization. Specifically, we propose a fidelity-aware Gaussian proposal strategy that adaptively adds new Gaussians based on reconstruction completeness, effectively avoiding redundancy. In the back-end, we propose a focus-and-balance online refinement strategy, which adaptively selects under-optimized Gaussians for focused refinement while ensuring globally balanced optimization by maximizing scene view coverage. We evaluated our method on synthetic and real-world datasets, including Replica, ScanNet, and EuRoC. Thanks to efficient system coupling and adaptive Gaussian proposal and refinement, our system achieves trajectory accuracy, rendering precision, and geometric accuracy comparable to or exceeding current state-of-the-art methods, while also demonstrating high efficiency.**

I. INTRODUCTION

Estimating self-motion while simultaneously reconstructing the surrounding environment is critical for many vision and robotics applications, such as autonomous driving and augmented reality. These technologies are often referred to as dense Simultaneous Localization and Mapping (dense SLAM), or dense Tracking and Mapping (dense TAM).

Monocular dense SLAM remains a highly challenging task due to the lack of geometric measurements. Classical methods [1], [2] attempt to directly optimize scene depth, but are typically limited to small-scale reconstructions or semi-dense maps due to high computational costs. In recent years, learning-based approaches such as [3], [4] have leveraged neural networks to assist in depth estimation. Meanwhile, methods based on implicit representations [5], such as [6], [7], model the scene as a neural radiance field. However, these approaches often suffer from limited reconstruction accuracy or slow reconstruction speed.

As an emerging representation, 3D Gaussian Splatting (3DGS) [8] offers strong scene rendering capabilities and fast rendering speed. Some methods [9], [10] integrate 3DGS with SLAM to achieve high-quality photometric reconstruction. Building upon this, other approaches [11], [12] incorporate dense depth estimation to enable Gaussian-Splatting(GS)-based geometric reconstruction SLAM. However, they typically require offline post-processing to obtain high-quality reconstruction. Moreover, dense depth estimation is resource-intensive and may suffer from alignment and generalization issues, which hinder the deployment of SLAM systems in fully online scenarios.

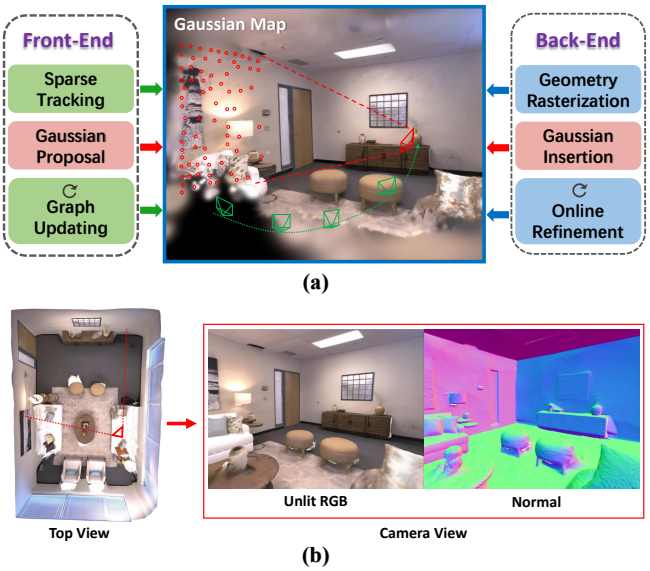

Fig. 1. Overview of **AMG-SLAM**. **(a)** The system includes a sparse tracking and Gaussian proposal front-end, as well as an online Gaussian surface reconstruction back-end. **(b)** Online reconstruction results for *Replica room0*, without any post-processing.

In this work, we aim to develop a fully online monocular Gaussian SLAM system capable of high-quality reconstruction. We argue that the key limitation of existing GS-based methods lies in their lack of scene-adaptive awareness, reflected in two main issues: 1) Gaussian proposals are typically sampled randomly from dense depth images, resulting in redundancy and a failure to adaptively complete missing regions. 2) Sliding-window-based optimization leads to uneven refinement of Gaussians across the scene.

We propose **AMG-SLAM**, an **A**daptive **M**onocular **G**aussian **SLAM** system that enables efficient and fully online surface reconstruction. AMG-SLAM features a tightly coupled design between a sparse visual tracking front-end and a GS-based mapping back-end, enabling efficient collaboration through extensive information sharing. In the front-end, we introduce a fidelity-aware adaptive Gaussian proposal strategy, which measures the completeness of the current reconstruction across different regions to adaptively propose new Gaussians. In the back-end, we establish a

*This work is supported by the NFS, China (U22A2061), and 230601GP0004.

[1]State Key Laboratory of General Artificial Intelligence, School of IST, Peking University, Beijing, China, panyouqi@stu.pku.edu.cn, zhouwugen@pku.edu.cn, zha@cis.pku.edu.cn

[2]School of Artificial Intelligence and Computer Science, Anqing Normal University, Anqing, China

Gaussian priority model that adaptively focuses optimization on high-priority Gaussians by selecting suitable views. In addition, we employ a maximal scene coverage view selection strategy to globally balance optimization efforts across the scene, effectively preventing catastrophic forgetting. Our method demonstrates that by tightly coupling tracking and mapping, and incorporating reconstruction-aware adaptive Gaussian proposal and refinement mechanisms, we can achieve fully online and efficient surface reconstruction without relying on dense depth estimation.

Our main contributions can be summarized as follows:

- We propose a novel dense monocular SLAM system that tightly integrates learning-based sparse visual tracking with GS-based mapping via a continual feedback mechanism, achieving efficient and accurate surface reconstruction in a fully online manner.
- We propose a fidelity-aware adaptive Gaussian proposal strategy, which proposes new Gaussians only in under-reconstructed regions based on rendering fidelity, initializing their positions through sparse depth estimation, thereby reducing redundant optimization.
- We propose the focus-and-balance online refinement strategy, which adaptively selects the Gaussians most in need of optimization for focused refinement, while simultaneously balancing global Gaussian optimization to prevent scene degradation.

II. Related Work

A. Monocular Dense SLAM.

Monocular dense SLAM is highly challenging due to the lack of geometric information input. Early methods like DTAM [1] optimize inverse depth over all pixels for dense mapping, while LSD-SLAM [2] focuses on semi-dense reconstruction using high-gradient regions. These approaches are limited to small scenes or semi-dense maps because of computational costs.

With the rise of deep learning, some methods leverage neural networks to accelerate depth optimization. CodeSLAM and CodeMapping [4], [13] use a VAE network to compress depth maps into a low-dimensional representation, significantly reducing the computation load. DROID-SLAM [3] adopts networks to predict reprojection residuals, iteratively updating optical flow and the depth map.

In addition to explicit scene representations, implicit dense SLAM [5], [14], has also achieved promising results. NICER-SLAM [6] optimizes the neural implicit map by constructing rendering consistency constraints. Some hybrid methods [15] combine implicit representation with feature-based tracking, jointly optimizing the neural map parameters and camera poses.

B. Monocular Gaussian Splatting SLAM.

In recent years, researchers have proposed SLAM methods based on 3D Gaussian splatting [16], [17], utilizing the explicit properties of Gaussian primitives and fast rendering to obtain realistic scene representation.

In monocular settings, MonoGS [9] uses rendering photometric error as a constraint, enabling Gaussian-based tracking and mapping. To enhance tracking efficiency, Photo-SLAM [10], MGSO [18] and MonoGS++ [19] adopt visual odometry as tracking front-end, and optimize dense Gaussian map representation through differentiable rendering.

Some methods focus on improving geometric reconstruction. MGS-SLAM [20] uses sparse feature points for pose estimation and designs a multi-view stereo network to estimate depth maps. IG-SLAM [21] adopts the dense tracking and global optimization of Go-SLAM [22], simultaneously predicting image depth and its uncertainty. Splat-SLAM [11] and Hi-SLAM2 [12] combines dense tracking with learning-based depth prediction to obtain refined depth maps, and integrates loop closure for accurate localization and geometric reconstruction. However, most methods fail to achieve satisfactory reconstruction quality in a fully online setting and require additional offline pose and map optimization.

Unlike previous methods that rely on dense depth estimation and offline refinement, our approach enables high-quality surface reconstruction in a fully online manner without requiring dense depth maps.

III. Method

A. System Overview

Our method consists of a tightly shared and collaborative front-end–back-end structure, as shown in Fig. 2(a).

The front-end estimates camera poses via sparse feature tracking. Meanwhile, it adaptively proposes new Gaussians to the back-end based on current reconstruction fidelity.

The back-end incrementally inserts and optimizes new Gaussians within a sliding window. Crucially, it leverages the current Gaussian optimization status and inter-view disparity to effectively focus-and-balance the optimization targets, enabling online scene refinement.

The front-end and back-end share camera poses, Gaussian map, sparse feature depths, and inter-viewpoint disparity. Through continuous feedback and collaboration, the system achieves fully online, high-quality scene reconstruction.

B. Sparse Pose Tracking

We adopt a sparse, feature-based approach for pose tracking. Specifically, a feature encoder transforms each input frame into a feature map, from which a set of feature patches are sampled. A matching network predicts the corresponding locations of these feature patches in other frames, along with their covariance.

After obtaining the predicted matching relationships, we construct reprojection residuals to optimize the camera poses and the inverse depth of patches.

$$\min_{T,d} \sum_{i,j \in \mathcal{G}} \left\| \hat{p}_j^t - K T_s^t \mathbf{d}(p_i^s) K^{-1} p_i^s \right\|_{\Sigma_{ij}}^2, \tag{1}$$

where $(p_i^s, \hat{p}_j^t)$ represents a predicted matching pair between frame i (source) and frame j (target). T_s^t denotes the relative pose from the source to the target frame. $\mathbf{d}(\cdot)$ is the inverse depth associated with the feature patches, and K is the

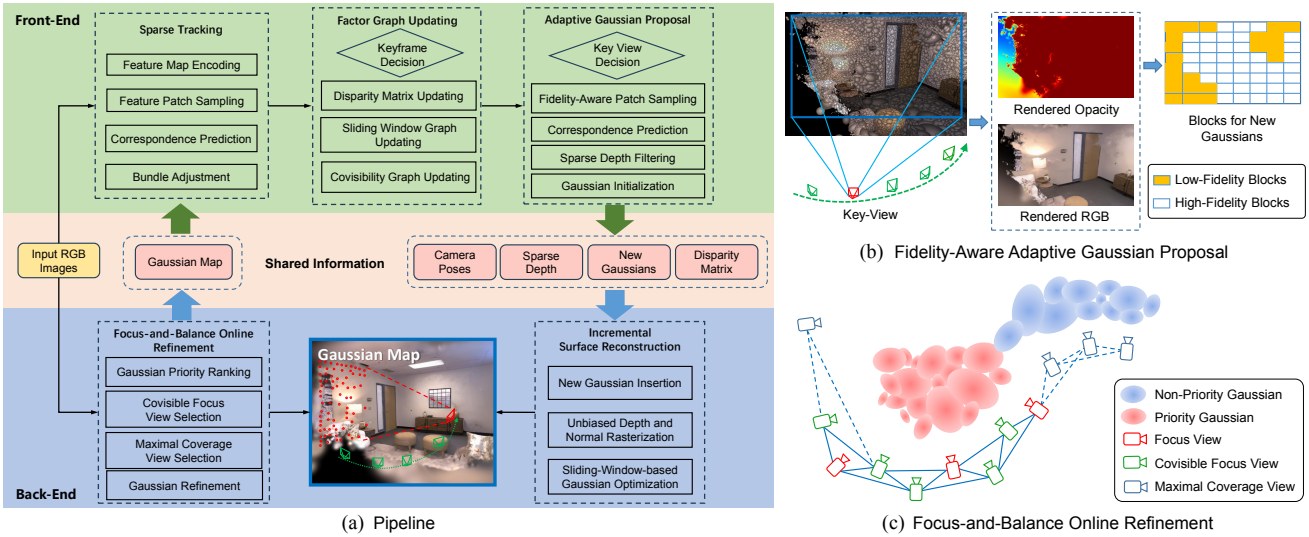

Fig. 2. a) The system consists of a front-end and a back-end, tightly coupled through a feedback mechanism enabled by information sharing. b) We identify blocks with low rendering fidelity as regions that require additional Gaussians, based on opacity and RGB error from the key view. c) We evaluate Gaussian priorities to select covisible views for focused optimization of high-priority Gaussians, while also choosing views that offer maximal scene coverage to ensure global optimization balance.

camera intrinsic matrix. $\mathcal{G}$ denotes the factor graph, which dynamically maintains the associations among keyframes.

C. Disparity-Based Factor Graph Updating

We determine keyframes and update the factor graph by evaluating inter-frame disparity. Specifically, we assess the disparity between the i-th most recent frame and the $(i{+}1)$-th most recent frame. If the disparity exceeds a certain threshold, the frame is designated as a new keyframe k.

The system maintains a disparity matrix $D \in \mathbb{R}^{N_k \times N_k}$, which stores disparity between all keyframe pairs. For each newly added keyframe, we compute its disparity with all existing keyframes and update the matrix accordingly.

The factor graph $\mathcal{G}$ consists of two components: a sliding window graph $\mathcal{W}$ and a covisibility graph $\mathcal{C}$. After a new keyframe k_{new} is identified, it is connected to 8 previous keyframes k_w to form the sliding window graph. Additionally, we compute the median disparity within the sliding window and use it as a threshold to search the disparity. The covisibility graph $\mathcal{C}$ is constructed by connecting keyframe k_{new} to all other keyframes k_c whose disparity with k_{new} falls below this threshold, as shown in Eq. 2.

$$\mathcal{C} = \{k_c \notin \mathcal{W} \mid D_{k_{\text{new}},k_c} < \text{Median}\left(\{D_{k_{\text{new}},k_w} \mid k_w \in \mathcal{W}\}\right)\} \tag{2}$$

It is worth noting that the disparity matrix is shared between the front-end and back-end, and it also plays a crucial role in the back-end's focus-and-balance online refinement.

D. Fidelity-Aware Adaptive Gaussian Proposal

A key challenge in monocular Gaussian-splatting SLAM is determining when and where to propose new Gaussians, as well as how to initialize their properties, particularly depth. Existing methods [11], [12] typically identify keyframes based on inter-frame overlap, then randomly sample points on these keyframes and initialize Gaussians using dense depth estimates.

In contrast, we propose an adaptive Gaussian proposal and initialization strategy in the front-end, guided by the reconstruction fidelity.

We refer to frames that provide Gaussian proposals to the back-end as key views. After each tracking step, we select the i-th most recent frame as a candidate view and compute its disparity with the last key view. If the disparity exceeds a threshold θ, the frame is designated as a new key view.

As shown in Fig. 2(b), we render the back-end-shared Gaussian map from the new key view and evaluate rendering fidelity based on opacity and photometric error. We divide the image into $B_m \times B_n$ blocks and evaluate the minimum opacity and maximum photometric error within each block. If the minimum opacity falls below its threshold or the maximum photometric error exceeds its threshold, the block is considered low-fidelity, as shown in Eq. 3.

$$\min_{p \in P} \mathbf{O}_{\text{rend}}(p) < \sigma_o \quad \text{or} \quad \max_{p \in P} \|\mathbf{I}_{\text{rend}}(p) - \mathbf{I}_{\text{orig}}(p)\| > \sigma_i, \tag{3}$$

where p denotes a pixel in block P. $\mathbf{O}_{\text{rend}}$ and $\mathbf{I}_{\text{rend}}$ are the rendered opacity and RGB image, respectively, while $\mathbf{I}_{\text{orig}}$ denotes the original input image.

We sample feature patches from low-fidelity blocks in the key view's feature map. Meanwhile, a separate factor graph $\mathcal{G}_{kv}$ is constructed between the key view and the keyframes with low disparity. Using the matching network, we predict patch correspondences across views to establish constraints for depth filtering. The recovered depths are then projected into 3D space to initialize new Gaussians.

Our adaptive Gaussian proposal, guided by inter-view disparity and rendering fidelity, and refined via factor-graph depth filtering, enables efficient and accurate mapping without relying on dense depth estimation.

E. Incremental Gaussian Surface Reconstruction

We represent the scene using a series of 3D Gaussian primitives. Each Gaussian is defined by its center $\mathbf{x}_c \in \mathbb{R}^3$, opacity $\alpha \in [0, 1]$, scale $\mathbf{S} \in \mathbb{R}^{3\times3}$, and rotation $\mathbf{R} \in SO(3)$:

$$G(\mathbf{x}) = \alpha \exp\left(-(\mathbf{x} - \mathbf{x}_c)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \mathbf{x}_c)\right), \tag{4}$$

where the covariance matrix is given by $\mathbf{\Sigma} = \mathbf{RSS}^\top\mathbf{R}^\top$.

To achieve high-quality Gaussian surface modeling, we adopt the unbiased depth and normal rasterization method proposed in [23]. It defines the ray–Gaussian intersection as the region with the highest opacity, leading to depth rendering that more accurately aligns with the true geometry of physical surfaces.

When the front-end provides new Gaussian proposals, the back-end inserts them into the map and incrementally optimizes the scene within a sliding window.

Since incremental optimization is required only for key views and is limited to a sliding window, this leads to an imbalance between workload and Gaussian optimization. We propose adaptively selecting viewpoints for online Gaussian refinement during periods of non-key views, which we term *focus-and-balance online refinement*.

F. Focus-and-Balance Online Refinement

To address the imbalance in sliding window optimization, we propose a focus-and-balance online refinement strategy that autonomously selects suitable viewpoints to maintain scene consistency when no new Gaussians are inserted.

We maintain real-time statistics of global Gaussian optimization to quickly focus on the Gaussians that require further optimization and allocate suitable views as supervision. To ensure global balance and avoid catastrophic forgetting from overly focused optimization, we augment the selected views by adding viewpoints that maximize scene coverage.

1) Gaussian Priority Modeling: We model the optimization priority of a Gaussian i as being proportional to its gradient norm $\|\nabla_i\|$ from the last backpropagation and the number of iterations T_i since it was last optimized, and inversely proportional to its observation frequency F_i. The relationship can be formalized in Eq. 5.

$$Priority_i = \frac{\|\nabla_i\|^2 \cdot (T_i + \sigma)}{(F_i + \delta)}, \tag{5}$$

where σ, δ are small constants to avoid division by zero. In practice, Gaussians that were recently optimized ($T_i < 3$) are excluded from the computation, and those with gradient norms below a certain threshold ϵ are ignored.

2) Covisible Focus View Selection: Once the priorities of the Gaussians are computed, we identify the viewpoints from which these Gaussians are initially observed (i.e. the proposal keyviews). We rank these keyviews based on the scores of high-priority Gaussians they contain and select the top-K as the *focus views* V^f.

However, directly supervising these Gaussians using their proposal keyviews will limit generalization capability. To mitigate this issue, we instead select a set of viewpoints that are covisible with the focus views, referred to as *covisible focus views* V^{cf}.

Algorithm 1 Maximal Coverage View Selection

Require: Disparity matrix $D \in \mathbb{R}^{N\times N}$, initial view indices $\mathcal{K}$, total number of desired views M
Ensure: Selected view set $\mathcal{M}$ with $|\mathcal{M}| = M$

```
Initialize M ← K
while |M| < M do
    for each j ∈ {1, ..., N} \ M do
        Compute marginal gain: Δ(j) ← Σ_{i∈M} D_ij
    end for
    Select j* ← arg max_j Δ(j)
    Update M ← M ∪ {j*}
end while
return M
```

Fortunately, the front-end disparity matrix provides an efficient way to search for covisible views. By measuring disparity, we can identify keyframes with the smallest disparity and their neighboring frames as covisible views.

$$\begin{aligned} V^{cf} &= \{v_k^{cf} \mid k = 1, \ldots, K\}, \\ v_k^{cf} &= \arg\min_{v \in V \setminus V^f} D(v_k^f, v). \end{aligned} \tag{6}$$

Here, $D(\cdot, \cdot)$ denotes a disparity metric measuring the geometric difference between two keyframes. The search is performed over the set of all mapped keyframes V, excluding the current local frames V^f.

Leveraging covisible focus views for cross-view supervision provides more diverse observations and facilitates more robust optimization of the selected Gaussians, thereby enhancing local scene reconstruction.

3) Maximal Coverage View Selection.: To balance the overall optimization and prevent reconstruction degradation outside the focus region, we sample a small number of additional viewpoints to achieve maximal scene coverage beyond the existing K covisible focus views.

We aim to sample a total of M views from N candidates, where $M \ll N$. Given an initial subset of K selected views, denoted as $\mathcal{K} \subset \{1, \ldots, N\}$ with $|\mathcal{K}| = K$, the objective is to select an additional $M - K$ views such that the final set $\mathcal{M}$, satisfying $\mathcal{K} \subseteq \mathcal{M}$ and $|\mathcal{M}| = M$, maximizes the overall scene coverage. Once again, we approximate this process by maximizing view disparity using the disparity matrix.

We formulate the following optimization problem:

$$\begin{aligned} \max_{x \in \{0,1\}^N} \quad & f(x) = \sum_{1 \le i < j \le N} D(v_i, v_j) x_i x_j \\ \text{s.t.} \quad & \sum_{i=1}^{N} x_i = M; x_i = 1, \forall i \in \mathcal{K}, \end{aligned} \tag{7}$$

where x_i is the binary decision variables.

Since $M \ll N$, a greedy algorithm can yield an approximate solution, as shown in Alg. 1. Specifically, we iteratively select the view that maximizes the total disparity with the current selected viewpoints. The time complexity is $\mathcal{O}((M - K) \cdot N)$, which is efficient for an online scenario.

The set $\mathcal{M}$, composed of both the covisible focus views and the maximal coverage views, serves as the optimal supervision frames for the current online refinement.

G. Loss Function

Both the incremental optimization and the adaptive online refinement share a unified loss function, jointly maintaining the quality of the reconstruction and rendering.

Depth-Normal Consistency Loss. As formulated in Eq. 8, the depth-normal consistency loss enforces consistency between the unit normal $\tilde{\mathbf{n}}$ estimated by the gradient of the rendered depth and the unit normal $\mathbf{n}$ from the directly rendered normal map.

$$\mathcal{L}_n = \sum_i (1 - \mathbf{n}_i^\top \tilde{\mathbf{n}}_i). \tag{8}$$

Sparse Depth Loss. The sparse depth loss includes the depth of feature patches sampled during front-end pose tracking, as well as the depth of patches used to initialize Gaussians from key views. This design introduces multi-view geometric constraints.

$$\mathcal{L}_d = \sum_{i \in \mathcal{D}} \left\| d_i^{rend} - d_i^{esti} \right\|. \tag{9}$$

Photometric Loss. The photometric loss constrains the difference between the rendered RGB image from the Gaussian map and the actual input image.

$$\mathcal{L}_p = \sum_i \left\| I_i^{rend} - I_i^{orig} \right\|. \tag{10}$$

The final loss function is a combination of the three terms, as shown in Eq. 11. In practice, we set $\alpha = 0.05$, $\beta = 0.2$, and $\gamma = 1.0$.

$$\mathcal{L} = \alpha \mathcal{L}_n + \beta \mathcal{L}_d + \gamma \mathcal{L}_p. \tag{11}$$

IV. Experiments

We conduct experimental evaluations of our method on both synthetic and real-world datasets, and validate the effectiveness of the innovations through ablation studies.

A. Experimental Setup

1) Implementation details.: In the front-end, we adopt the feature encoder and matching network from DPVO [24], with minor modifications, removing the hidden state from the network updater. This eliminates the need to store the hidden state of the factor graph during the tracking process, resulting in lower resource consumption. For key view selection and Gaussian proposal, we set $i = 4$, set the disparity threshold for key view selection to $\theta = 5$, and divide the image into 32×32 patches (i.e., $B_m = 32, B_n = 32$). We sample Gaussians amounting to approximately 1/256 of the total pixels within each patch.

In the back-end, our Gaussian parameterization and optimizer settings are consistent with those of [9] and [23]. For focus-and-balance online refinement, we select no more than 8% of the existing frames as the supervision view ($M \leq 0.08 * N$). Our method does not implement any post-processing, such as global pose optimization or color refinement in [9], [21].

To ensure a fair comparison, we set the size of the output image rendering to match that used in Splat-SLAM [11].

2) Datasets.: We evaluate our system on synthetic Replica [25] and real-world ScanNet [26] and EuRoC [27] datasets, assessing its tracking, rendering, and reconstruction performance both qualitatively and quantitatively.

3) Platform.: Our workstation is equipped with an Intel i9-14900K CPU, and an NVIDIA RTX 3090 24GB GPU.

TABLE I

Rendering and reconstruction evaluation on Replica. Methods marked with * include 2K-iter post-processing; other methods are fully **online**.

Method	Metric	O0	O1	O2	O3	O4	R0	R1	R2	**Avg.**
Photo-SLAM [10]	PSNR↑	35.23	35.11	29.70	31.20	31.27	29.07	31.02	31.22	31.73
	SSIM↑	0.95	0.94	0.91	0.92	0.93	0.85	0.90	0.92	0.91
	LPIPS↓	0.11	0.12	0.17	0.14	0.12	0.19	0.13	0.13	0.14
MGS-SLAM [20]	PSNR↑	35.51	34.25	30.83	31.86	34.38	29.91	31.06	31.49	32.41
	SSIM↑	0.94	0.93	0.91	0.92	0.95	0.89	0.90	0.91	0.92
	LPIPS↓	0.07	0.11	0.12	0.07	0.08	0.08	0.09	0.08	0.09
Splat-SLAM* [11]	PSNR↑	40.81	40.64	35.91	35.03	37.40	32.25	34.31	35.95	36.45
	SSIM↑	**0.98**	0.97	0.96	0.95	**0.98**	0.91	0.93	0.95	0.95
	LPIPS↓	0.05	0.05	0.07	0.06	**0.04**	0.10	0.09	0.06	0.06
	D-L1↓	2.44	1.97	**2.46**	2.62	2.53	**2.90**	**2.16**	**2.18**	**2.41**
IG-SLAM* [21]	PSNR↑	41.68	41.30	34.68	34.92	34.80	32.33	34.64	35.29	36.21
	SSIM↑	**0.98**	**0.98**	0.95	0.96	0.96	0.93	0.95	0.96	0.96
	LPIPS↓	**0.02**	0.03	0.06	0.05	0.07	0.07	0.06	0.05	0.05
	D-L1↓	2.23	**1.94**	6.40	7.67	4.45	4.79	3.04	4.15	4.33
HI-SLAM2 (Full)* [12]	PSNR↑	42.28	43.16	**37.31**	**36.99**	**38.95**	35.48	**36.93**	**38.53**	**38.71**
	SSIM↑	**0.98**	**0.98**	**0.97**	**0.97**	0.97	0.96	**0.97**	**0.97**	**0.97**
	LPIPS↓	**0.02**	0.03	0.04	0.04	**0.04**	0.04	0.04	**0.04**	**0.03**
HI-SLAM2 (Online) [12]	PSNR↑	33.89	39.04	31.33	33.10	29.48	29.75	29.49	29.47	31.94
	SSIM↑	0.92	0.97	0.93	0.94	0.91	0.89	0.87	0.87	0.91
	LPIPS↓	0.09	0.05	0.10	0.07	0.11	0.08	0.10	0.14	0.09
Ours	PSNR↑	**42.53**	**43.19**	36.24	35.10	37.58	**36.06**	35.37	36.52	37.82
	SSIM↑	**0.98**	**0.98**	0.96	**0.97**	0.97	**0.97**	0.96	**0.97**	**0.97**
	LPIPS↓	**0.02**	**0.02**	**0.03**	**0.03**	0.05	**0.03**	**0.03**	**0.04**	**0.03**
	D-L1↓	**2.21**	2.49	3.96	**2.60**	**2.44**	3.31	3.08	3.36	2.93

TABLE II

Rendering evaluation on EuRoC. Methods marked with * include 2K-iter post-processing; others are fully **online**.

Method	Metric	MH01	MH02	V101	V201	**Avg.**
Photo-SLAM [10]	PSNR↑	13.95	14.20	17.07	15.68	15.23
	SSIM↑	0.42	0.43	0.62	0.62	0.52
	LPIPS↓	0.37	0.36	0.27	0.32	0.33
IG-SLAM* [21]	PSNR↑	22.33	22.31	20.55	24.59	22.44
	SSIM↑	0.78	0.77	0.79	0.85	0.80
	LPIPS↓	0.22	0.23	0.29	0.18	0.23
Ours	PSNR↑	**25.74**	**25.46**	**25.73**	**27.09**	**26.05**
	SSIM↑	**0.89**	**0.87**	**0.90**	**0.90**	**0.89**
	LPIPS↓	**0.06**	**0.11**	**0.08**	**0.07**	**0.08**

B. Evaluation

1) Evaluation of Rendering.: We compare the rendering accuracy of our system with state-of-the-art methods across multiple datasets, with results averaged over three trials. It is worth noting that the results from [9], [11], [12], [21]

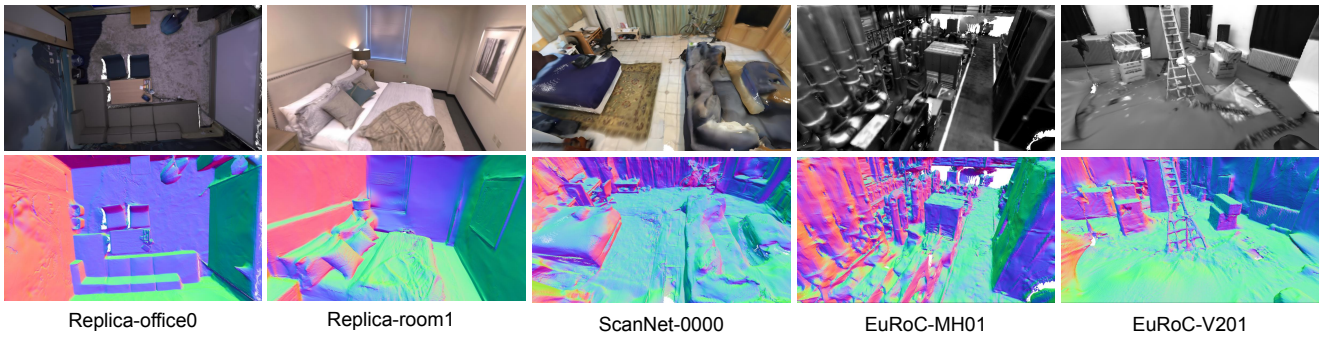

Fig. 3. Reconstruction mesh of AMG-SLAM. Shown in unlit RGB and normal.

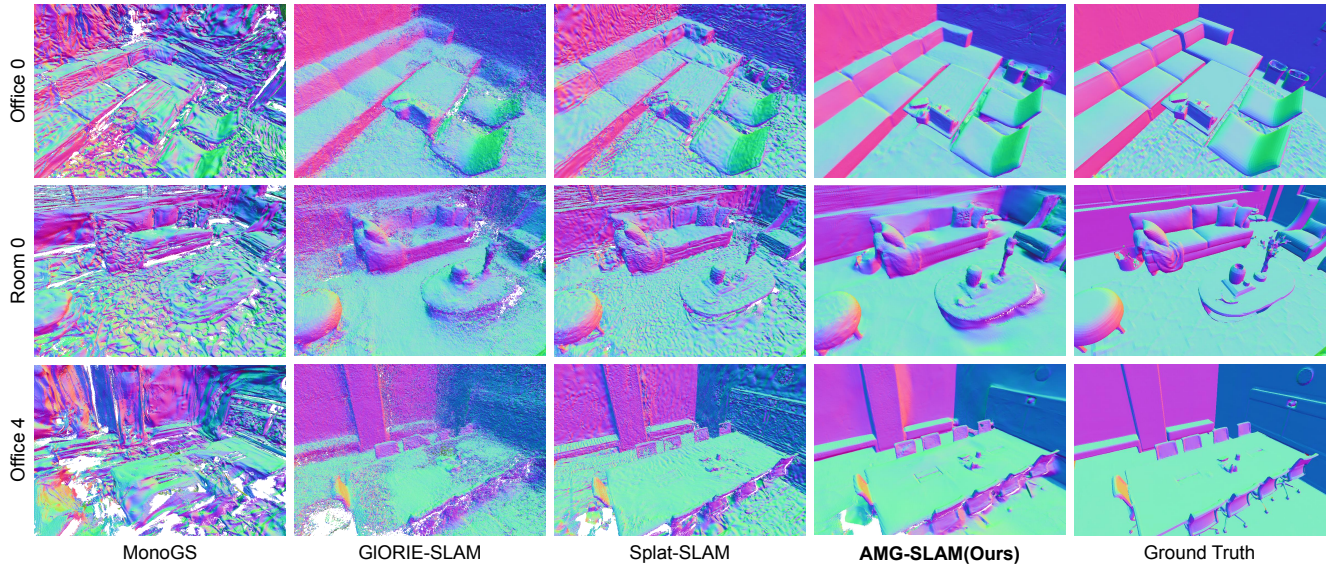

Fig. 4. Comparison of normal maps on the Replica dataset. We take images from [11] except ours.

include post-processing with 2k or 26k iterations offline map optimization, while our method is fully online.

The evaluation results on Replica are shown in Tab. I. Since HI-SLAM2 [12] is a hybrid online–offline method, its post-processing not only includes map refinement but also post-keyframe selection and joint global pose–map optimization. We evaluate both the online and the full versions separately. Our method significantly outperforms other online approaches and achieves performance comparable to the current state-of-the-art method (HI-SLAM2 (Full)).

The evaluation results on EuRoC, a large-scale scene dataset with fast motions, are shown in Tab. II. We selected the same sequences as in [10], and our method outperforms others by a large margin. Results on ScanNet are presented in Tab. III. We selected the same scenes as in [11] and computed the average results across all scenes, where our performance is second only to Splat-SLAM [11].

2) Evaluation of Reconstruction.: We conduct a qualitative evaluation of the reconstruction results by using a TSDF-fusion to reconstruct mesh from the Gaussian map. Fig. 3 shows the reconstruction mesh and normal of our method across various datasets. Our approach is capable of recovering the overall structure of the scene, whether in small environments like Replica or large, complex scenes like the EuRoC and Scannet. We compare our surface reconstruction results with other dense SLAM systems. As shown in Fig. 4, our method produces smoother and more realistic surfaces.

TABLE III

RENDERING EVALUATION ON SCANNET. METHODS MARKED WITH * INCLUDE 26K-ITER POST-PROCESSING; OTHERS ARE FULLY **ONLINE**.

Method	Metric	0000	0059	0106	0169	0181	0207	**Avg.**
MonoGS* [9]	PSNR ↑	16.91	19.15	18.57	20.21	19.51	18.37	18.79
	SSIM ↑	0.62	0.69	0.74	0.74	0.75	0.70	0.71
	LPIPS ↓	0.70	0.51	0.55	0.54	0.63	0.58	0.59
Splat-SLAM* [11]	PSNR ↑	**28.68**	**27.69**	**27.70**	**31.14**	**31.15**	**30.49**	**29.48**
	SSIM ↑	0.83	**0.87**	0.86	0.87	0.84	0.84	0.85
	LPIPS ↓	0.19	**0.15**	**0.18**	0.15	0.23	**0.19**	0.18
IG -SLAM* [21]	PSNR ↑	24.68	20.09	25.30	27.85	25.80	26.69	25.07
	SSIM ↑	0.74	0.68	0.83	0.82	0.83	0.78	0.78
	LPIPS ↓	0.29	0.39	0.22	0.19	0.27	0.27	0.27
Ours	PSNR ↑	26.18	22.98	25.39	29.11	25.83	27.53	26.17
	SSIM ↑	**0.84**	0.82	**0.87**	**0.88**	**0.87**	**0.85**	**0.86**
	LPIPS ↓	**0.16**	0.19	**0.18**	**0.13**	**0.20**	**0.17**	**0.17**

We quantitatively evaluated the geometric accuracy of the Gaussian map using *Depth L1*. We present a comparison in Replica [25] with [11] and [21] in Tab. I. Without relying on dense depth estimation or post-processing, our depth map

TABLE IV

TRAJECTORY ACCURACY (ATE ↓, CM) ON REPLICA

Method	O0	O1	O2	O3	O4	R0	R1	R2	**Avg.**
DPVO [24]	0.77	0.36	0.57	0.46	0.57	0.49	0.54	0.54	0.54
Photo-SLAM [10]	0.58	0.32	5.03	0.47	0.58	0.35	1.18	**0.23**	1.09
MGS-SLAM [20]	0.35	0.28	**0.26**	0.32	**0.34**	0.36	0.35	0.32	0.32
Splat-SLAM [11]	0.29	0.35	0.34	0.42	0.43	**0.29**	**0.33**	0.25	0.34
IG-SLAM [21]	0.33	0.50	0.39	0.47	0.68	0.45	0.39	0.31	0.44
Ours (w/o CG)	0.28	0.27	0.42	0.39	0.55	0.47	0.49	0.43	0.41
Ours (w CG)	**0.19**	**0.24**	0.32	**0.26**	0.41	0.31	0.40	0.27	**0.30**

TABLE V

TRAJECTORY ACCURACY (ATE ↓, CM) ON EUROC

Method	MH1	MH2	MH3	V101	V102	V201	V202	**Avg.**
DSO [28]	4.6	4.6	17.2	8.9	10.7	4.4	13.2	9.1
DROID-VO [3]	16.3	12.1	24.2	10.3	16.5	10.2	11.5	14.4
DPVO [24]	8.7	5.5	15.8	5.0	14.0	5.7	**4.9**	8.5
Photo-SLAM [10]	4.1	4.4	-	8.8	-	26.6	-	11.0
LSG-SLAM [29]	-	2.0	8.0	6.0	4.0	5.0	-	5.0
Ours	**2.8**	**1.8**	**2.9**	**3.3**	**1.3**	**3.9**	6.4	**3.2**

accuracy is close to state-of-the-art methods.

3) Evaluation of Trajectory.: We evaluated the absolute trajectory error (ATE) on Replica [25], as shown in Tab. IV. **Ours (w/o CG)** refers to our tracking method without the use of a Covisibility Graph. Although our front-end is based on a modified DPVO, the tracking accuracy remains higher than the original DPVO even without the Covisibility Graph, demonstrating the importance of the feedback Gaussian map from the back-end. The full system (**Ours (w CG)**) achieves the highest average accuracy, even outperforming those that incorporate global optimization [11], [21].

We also evaluate tracking performance on the EuRoC dataset, which features rapid motion and illumination changes. As shown in Tab. V, our method outperforms existing visual odometry approaches and Photo-SLAM [10] using ORB-SLAM3 [30] for pose estimation, demonstrating superior tracking accuracy and robustness.

C. Ablation

1) Ablation of System: First, we conducted ablation studies on system coupling, Fidelity-aware Adaptive Gaussian Proposal (FAGP), Covisibility Graph Updating (CGU), and Unbiased Depth Rasterization (UDR). In the decoupled system, the tracking and mapping operate independently. The front-end only provides pose estimation, while Gaussian primitives are proposed by the back-end in the same way as in [9]. The results are averaged over 3 trials on Replica Room 0, as shown in Tab. VI.

The results show that tight coupling of the system is essential. In a decoupled system, the front-end cannot access the Gaussian map for depth rendering, and the back-end cannot receive adaptive Gaussian proposals. This not only degrades mapping accuracy but also negatively impacts tracking performance. Besides, all components contribute to the overall system accuracy, and their combination yields the best performance. The FAGP improves Gaussian optimization efficiency by avoiding redundant new Gaussians,

TABLE VI

ABLATION OF SYSTEM ON REPLICA R0.

Coupling	FAGP	UDR	CGU	ATE[cm]↓	PSNR[dB]↑	Depth L1[cm]↓
✗	–	✗	✗	0.62	31.07	14.31
✗	–	✓	✓	0.47	32.59	8.47
✓	✗	✗	✗	0.58	33.96	6.59
✓	✓	✗	✗	0.57	34.83	5.28
✓	✓	✓	✗	0.47	34.78	3.91
✓	✓	✗	✓	0.34	35.20	4.33
✓	✓	✓	✓	**0.31**	**35.37**	**3.27**

enhancing rendering and geometric accuracy. The UDR effectively constrains geometric consistency, improving reconstruction accuracy. The CGU explores long-term inter-frame associations, greatly enhancing pose tracking accuracy.

TABLE VII

ABLATION OF ONLINE REFINEMENT ON REPLICA R0.

IO	OR			OFR	PSNR[dB]↑	Depth L1 [cm]↓
	Sliding	Random	F&B			
✗	✗	✗	✗	✓	20.79	81.46
✓	✗	✗	✗	✗	24.16	26.83
✓	✗	✗	✗	✓	34.65	3.35
✓	✓	✗	✗	✗	30.26	9.74
✓	✗	✓	✗	✗	35.04	3.37
✓	✗	✗	✓	✗	36.06	3.31
✓	✗	✗	✓	✓	**36.12**	**3.30**

2) Ablation of Online Refinement: To evaluate the effectiveness of online refinement, we compare the results of online refinement with those of offline refinement and examine different online strategies. The results are shown in Tab. VII, where IO denotes Incremental Optimization after Gaussian insertion, OR denotes Online Refinement, and OFR denotes Offline Refinement. Within online refinement, we consider three view-selection strategies: sliding window, random sampling, and Focus-and-Balance.

Using only incremental optimization yields limited performance, while its combination with offline refinement brings significant gains—an approach followed by most existing methods, but it is not fully online. Our strategy improves mapping performance in an online manner, and the viewpoints selected by Focus-and-Balance are superior to those from sliding windows or random sampling. Remarkably, Focus-and-Balance Online Refinement (IO+F&B) even surpasses the offline refinement (IO+OFR) and already achieves near-optimal performance, leaving little room for further improvement through offline refinement (IO+F&B+OFR).

3) Ablation of Focus-and-Balance Strategy: We further evaluate the relationship between Gaussian Priority in Eq. 5 and reconstruction performance, as shown in Tab. VIII. If we consider only gradients, newly added Gaussians tend to have larger gradients, causing the system to always optimize the newly inserted ones, which limits the improvement of online refinement. By incorporating observation interval (Iter.) and observation frequency(Obs.), Gaussian selection becomes more biased toward high-gradient Gaussians that lack sufficient optimization, leading to a clear performance gain. Furthermore, combining with the Maximal Coverage

TABLE VIII

ABLATION OF FOCUS-AND-BALANCE STRATEGY.

Gaussian Priority			Balance	PSNR[dB]↑	Depth L1 [cm]↓
Grad.	Iter.	Obs.			
✗	✗	✗	✓	34.40	3.94
✓	✗	✗	✗	34.28	4.10
✓	✓	✗	✗	35.57	3.37
✓	✓	✓	✗	<u>35.62</u>	**3.31**
✓	✓	✓	✓	**36.06**	**3.31**

balancing view leads to optimal performance.

D. Runtime Analysis

We evaluate the runtime of our method on Replica R0, and all experiments are performed on an NVIDIA RTX 3090. The front-end achieves a frame rate of approximately 32 FPS, while the full system runs at around 9 FPS. The comparison of system speed and resource consumption is shown in Tab. IX. Compared with other methods, especially SLAM systems with geometric reconstruction, our method is highly competitive in terms of both runtime frame rate and GPU memory usage.

TABLE IX

EVALUATION OF FPS AND GPU MEMORY USAGE ON REPLICA R0.

Items	Go-SLAM [22]	Mono-GS [9]	Glorie-SLAM [15]	Splat-SLAM [11]	Mono-GS++ [19]	**Ours**
GPU Mem. [GiB]	18.5	<u>14.62</u>	15.22	17.57	-	**13.39**
Frame Rate [FPS]	<u>8.36</u>	0.32	0.23	1.24	2.48	**8.75**

V. CONCLUSIONS

In this paper, we present AMG-SLAM, an Adaptive Monocular Gaussian SLAM system for efficient surface reconstruction. Our approach adaptively introduces new Gaussians in low-fidelity regions identified from the rendered map in the back-end, and focuses refinement on under-optimized Gaussians based on a priority metric. With a tightly coupled front-end and back-end design, AMG-SLAM enables fully online reconstruction without offline post-processing, achieving competitive performance across multiple datasets.

REFERENCES

[1] R. A. Newcombe et al., "Dtam: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, 2011, pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513.

[2] J. Engel et al., "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*, Springer, 2014, pp. 834–849.

[3] Z. Teed et al., "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.

[4] H. Matsuki et al., "Codemapping: Real-time dense mapping for sparse slam using compact scene representations," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7105–7112, 2021.

[5] B. Mildenhall et al., "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[6] Z. Zhu et al., "Nicer-slam: Neural implicit scene encoding for rgb slam," in *International Conference on 3D Vision (3DV)*, Mar. 2024.

[7] A. Rosinol et al., "Nerf-slam: Real-time dense monocular slam with neural radiance fields," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 3437–3444.

[8] B. Kerbl et al., "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, Jul. 2023.

[9] H. Matsuki et al., "Gaussian Splatting SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[10] H. Huang et al., "Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[11] E. Sandström et al., "Splat-slam: Globally optimized rgb-only slam with 3d gaussians," *arXiv preprint arXiv:2405.16544*, 2024.

[12] W. Zhang et al., "Hi-slam2: Geometry-aware gaussian slam for fast monocular scene reconstruction," *arXiv preprint arXiv:2411.17982*, 2024.

[13] M. Bloesch et al., "Codeslam—learning a compact, optimisable representation for dense visual slam," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2560–2568.

[14] Z. Zhu et al., "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 12 786–12 796.

[15] G. Zhang et al., "Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam," *arXiv preprint arXiv:2403.19549*, 2024.

[16] N. Keetha et al., "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, pp. 21 357–21 366.

[17] C. Yan et al., "Gs-slam: Dense visual slam with 3d gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 595–19 604.

[18] Y. S. Hu et al., "Mgso: Monocular real-time photometric slam with efficient 3d gaussian splatting," *arXiv preprint arXiv:2409.13055*, 2024.

[19] R. Li et al., "Monogs++: Fast and accurate monocular rgb gaussian slam," *arXiv preprint arXiv:2504.02437*, 2025.

[20] P. Zhu et al., "Mgs-slam: Monocular sparse tracking and gaussian mapping with depth smooth regularization," *arXiv preprint arXiv:2405.06241*, 2024.

[21] F. A. Sarikamis et al., "Ig-slam: Instant gaussian slam," *arXiv preprint arXiv:2408.01126*, 2024.

[22] Y. Zhang et al., "Go-slam: Global optimization for consistent 3d instant reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3727–3737.

[23] B. Zhang et al., "Rade-gs: Rasterizing depth in gaussian splatting," *arXiv preprint arXiv:2406.01467*, 2024.

[24] Z. Teed et al., "Deep patch visual odometry," *Advances in Neural Information Processing Systems*, 2023.

[25] J. Straub et al., "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[26] A. Dai et al., "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5828–5839.

[27] M. Burri et al., "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016.

[28] J. Engel et al., "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018. DOI: 10.1109/TPAMI.2017.2658577.

[29] Z. Xin et al., "Large-scale gaussian splatting slam," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 8478–8485. DOI: 10.1109/ICRA55743.2025.11128359.

[30] C. Campos et al., "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021. DOI: 10.1109/TRO.2021.3075644.