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

Pocket-SLAM: Rendering-Area-Aware Pruning for Memory-Efficient 3DGS-SLAM

Leshu Li[1*], Jie Peng[2], Yang Zhao[1]

***Abstract*— 3D Gaussian Splatting (3DGS) has garnered significant attention in Simultaneous Localization and Mapping (SLAM) due to its advances in capturing fine-grained geometry features and synthesizing novel views. For SLAM in large-scale scenes, such as autonomous driving, 3DGS-SLAM faces a critical limitation. The memory consumption increases continuously over time as Gaussian points accumulate, leading to poor memory efficiency and limiting its applicability. In this work, we propose a rendering-area–aware pruning strategy that selectively removes Gaussians based on their contribution to the effective rendering area, rather than solely relying on Gaussian-level heuristics (e.g., opacity or gradient magnitude). This perspective directly targets the sources of memory redundancy, effectively reducing the peak memory footprint of 3DGS-SLAM during runtime. Evaluations on the EuRoC and KITTI datasets demonstrate that our method consistently outperforms existing pruning approaches in large-scale outdoor scenes, achieving over 60% memory reduction and more than 2× FPS improvement while preserving localization and mapping accuracy. These results highlight rendering-area–aware pruning as a promising direction for scaling 3DGS-SLAM to real-world autonomous driving scenarios. Our code is publicly available at https://github.com/UMN-ZhaoLab/Pocket-SLAM.git.**

I. Introduction

Visual SLAM is fundamental to robot perception and underpins a wide range of applications, including autonomous navigation, environment mapping, and interactive perception. 3D Gaussian Splatting (3DGS) [1] has recently attracted significant attention in SLAM as it enables fine-grained geometric representation and high-quality novel view synthesis. For instance, MonoGS [2] and GS-SLAM [3] demonstrate that 3DGS-SLAM can achieve both high-fidelity scene reconstruction and accurate camera tracking, highlighting its strong potential for applications such as autonomous driving [4], [5] and drone navigation [6]. Moreover, LGS-SLAM [7] and WildGS-SLAM [8] extend 3DGS-SLAM to large-scale outdoor scenarios, validating its feasibility in real-world driving environments. Nevertheless, a critical limitation persists: large-scale deployments introduce substantial memory redundancy. While the number of Gaussians in compact indoor scenes is relatively modest, vast and unstructured outdoor environments necessitate storing and updating millions of Gaussians in real time, resulting in prohibitively high peak memory consumption that becomes a major bottleneck for deployment. This challenge is further exacerbated by the increasing demand for 3DGS-SLAM to operate efficiently on resource-constrained edge devices

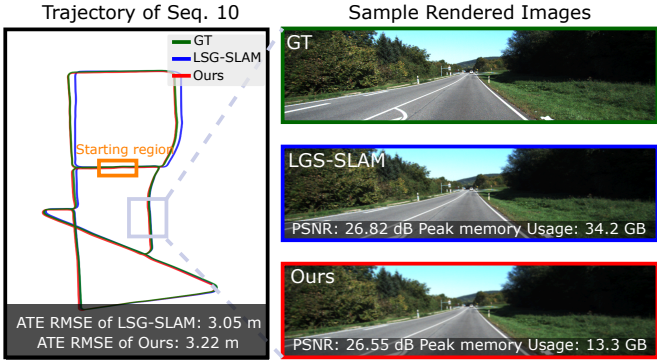


Fig. 1. Results on KITTI [14] sequence 10. Compared with LSG-SLAM [7], our method achieves comparable camera tracking accuracy (Absolute Trajectory Error Root Mean Square Error, ATE) and rendering quality (Peak Signal-to-Noise Ratio, PSNR) in large-scale scenarios, while reducing peak memory usage by 61%.

(e.g., GPUs embedded in autonomous vehicles or drones). In this context, reducing peak runtime memory consumption emerges as a central requirement for practical deployment.

Existing works have investigated Gaussian pruning in the context of 3DGS. However, most of these efforts focus primarily on small-scale indoor scenarios [9]–[12] or address only the storage of 3DGS-SLAM keyframes [13], while neglecting the critical issue of peak runtime memory consumption. This limitation is particularly important, as peak memory directly determines the feasibility of deploying 3DGS-SLAM on edge devices with limited memory capacity, and therefore has far greater practical implications for real-world large-scale applications. Furthermore, indoor and outdoor scenarios exhibit fundamentally different characteristics: while indoor environments are relatively compact and texture-rich, outdoor environments often consist of vast regions with sparse or low-texture areas. These differences result in a significantly larger number of Gaussians being generated and continuously updated during outdoor mapping, which further exacerbates the challenge of peak runtime memory consumption.

In this paper, we present Pocket-SLAM, a practical and seamlessly integrated extension to 3DGS-SLAM that incorporates a *rendering-area–aware pruning strategy*. During mapping, our method quantifies the contribution of each Gaussian by computing its effective pixel coverage on the image plane and prunes those with negligible rendering impact. Rendering-area–aware pruning measures Gaussian importance from the perspective of their contribution to the final rendered image, ensuring that in large-scale scenes, Gaussians covering critical regions are preferentially pre-

[1]University of Minnesota, Twin Cities, USA. li00385@umn.edu
[2]University of North Carolina at Chapel Hill, USA.
*Corresponding author

served. However, applying rendering-area–aware pruning alone, while effective in reducing memory, inevitably results in severe information loss in texture-dense regions. To mitigate this issue, we introduce a *tile-level budget mechanism*, which adaptively constrains the pruning ratio within each tile to prevent over-pruning in both texture-dense and texture-scarce areas, thereby maintaining balanced Gaussian distributions and preserving texture information. As illustrated in Fig. 1, our method achieves substantial memory savings in outdoor environments while maintaining both tracking and mapping accuracy. Furthermore, our approach is orthogonal to other 3DGS acceleration techniques and can be readily combined with them for further performance improvements.

In summary, the contributions of our work include:

- We propose a *rendering-area–aware pruning strategy* that assesses Gaussian importance based on effective pixel coverage, shifting the criterion from local heuristics to scene-level rendering efficiency.
- We introduce a *tile-level budget mechanism* that adaptively constrains pruning according to per-tile Gaussian allocation, preventing excessive pruning in both texture-dense and texture-sparse regions, and ensuring balanced Gaussian distributions with robust texture preservation.
- We demonstrate through evaluations on large-scale outdoor environments that our method outperforms mainstream pruning strategies, reducing peak memory consumption by over 60% and improving FPS by more than $2\times$ while preserving localization and mapping accuracy.

II. Related Work

A. Traditional Visual SLAM

Visual SLAM has been extensively studied in both indoor and outdoor scenarios, yet the challenges differ fundamentally. Indoor SLAM frameworks [15]–[19] operate in compact environments with abundant textures and structural regularities, enabling reliable tracking with sparse feature-based, direct, or dense methods. These approaches, however, are less effective in large-scale outdoor environments, which are often unstructured and contain vast texture-sparse regions. Outdoor SLAM frameworks, such as stereo-based [20], [21] and LiDAR-based systems [22], [23], address scale and robustness but require specialized sensors and still struggle to jointly achieve efficiency and high-fidelity reconstruction.

B. 3DGS-SLAM in Outdoor Environments

With the advent of 3DGS [1], researchers have increasingly integrated Gaussian primitives into SLAM systems. MonoGS [2] and GS-SLAM [3] show the feasibility of combining 3DGS with monocular SLAM for accurate tracking and high-fidelity reconstruction. Photo-SLAM [24] and LoopSplat [25] incorporate ORB-SLAM [15] modules or global BA to enhance robustness, but most evaluations remain limited to indoor scenes. Recent work such as LGS-SLAM [7] and WildGS-SLAM [8] extends 3DGS-SLAM to outdoor driving scenarios, highlighting its potential for large-scale deployment. However, outdoor environments require millions of Gaussians to represent unstructured regions, resulting in prohibitively high peak memory consumption and limiting deployment on resource-constrained edge devices such as those in autonomous vehicles and drones.

C. Memory-Efficient Gaussian Pruning

To mitigate redundancy in 3DGS models, recent research has focused on Gaussian pruning. LightGaussian [9] prunes Gaussians based on opacity, removing those with negligible contributions to the rendered image. LP-3DGS [11] uses gradient magnitude as an importance criterion, discarding Gaussians with small parameter updates. PUP-3DGS [10] removes Gaussians with little perceptual utility, while MaskGaussian [12] temporarily masks low-weight Gaussians before pruning them. Although effective for 3DGS rendering and training, these methods cannot be directly applied to 3DGS-SLAM, which demands per-frame rendering and tracking accuracy. GEVO [13] is among the few works addressing memory in 3DGS-SLAM, but it focuses only on storage via reduced keyframe retention rather than runtime peak memory consumption. This limitation motivates our work: we propose Pocket-SLAM, which introduces a *rendering-area–aware pruning strategy* and a *tile-level budget mechanism* to reduce peak memory consumption in large-scale outdoor environments while preserving tracking accuracy and reconstruction fidelity.

III. Proposed Methods

To achieve efficient and balanced pruning, Pocket-SLAM introduces two complementary strategies: the *rendering-area–aware pruning strategy* (Sec. III-B) and the *tile-level budget mechanism* (Sec. III-C). The *rendering-area–aware pruning strategy* assesses the contribution of each Gaussian by explicitly computing its effective pixel coverage on the image plane. However, pruning solely based on rendering area tends to eliminate Gaussians with small coverage even in texture-dense regions, leading to significant information loss. To address this issue, we introduce the *tile-level budget mechanism*, which allocates Gaussian survival budgets across different image regions. The combination of these mechanisms prevents over-pruning in texture-rich areas while still achieving substantial reductions in redundant Gaussians. Overall, this design balances SLAM accuracy with memory efficiency, making Pocket-SLAM particularly well-suited for large-scale outdoor environments.

Pocket-SLAM is built upon the standard SLAM framework, which generally comprises two fundamental stages: *tracking* and *mapping*. In the tracking stage, Pocket-SLAM allocates budgets to each tile based on the gradient characteristics of its Gaussians. In the mapping stage, it ranks Gaussians according to both the tile-level budgets and their rendering areas, pruning those deemed less important. The overall pipeline of Pocket-SLAM is illustrated in Fig. 2.

A. SLAM Pipeline

The Pocket-SLAM framework follows the standard SLAM paradigm, alternating between *tracking*, which estimates

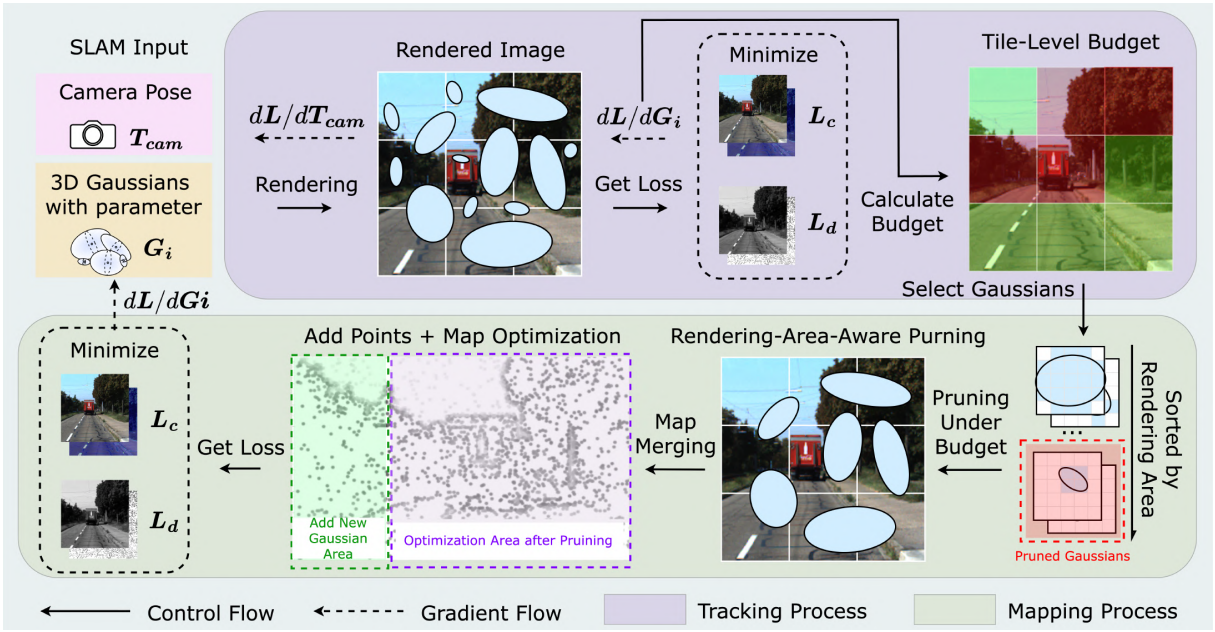


Fig. 2. The overview pipeline of Pocket-SLAM. For each incoming frame, the pose is initialized with multi-modality priors and optimized using rendering and feature warping losses (tracking). Keyframes are then selected to refine the scene and insert new Gaussians (mapping). Our pruning strategies are integrated into mapping: the *Tile-Level Budget Mechanism* assigns balanced survival budgets during tracking to guide pruning, and *Rendering-Area–Aware Pruning* removes low-contribution Gaussians after mapping optimization, enabling memory-efficient SLAM without compromising accuracy.

camera poses, and *mapping*, which refines the Gaussian scene representation. We denote the camera pose as $T_{\text{cam}} \in SE(3)$, which consists of a 3D rotation and translation that transform world coordinates into the camera coordinate system. The set of 3D Gaussians is represented as $\{G_i\}_{i=1}^{N}$, where each G_i encodes position, covariance, opacity, and color parameters. For an incoming frame, the rendered color and depth at pixel $\mathbf{p}$ are denoted by $C(\mathbf{p}; G_i, T_{\text{cam}})$ and $Z(\mathbf{p}; G_i, T_{\text{cam}})$, respectively. We then define the photometric loss L_c and depth loss L_d as:

$$L_c = \sum_{\mathbf{p}\in\Omega_t} \rho\Big(I_t(\mathbf{p}) - C_t(\mathbf{p}; G_i, T_{\text{cam}})\Big)^2, \tag{1}$$

$$L_d = \sum_{\mathbf{p}\in\Omega_t} \psi\Big(D_t(\mathbf{p}) - Z_t(\mathbf{p}; G_i, T_{\text{cam}})\Big)^2, \tag{2}$$

where I_t and D_t denote the observed color and depth, and ρ, ψ are robust penalty functions. Ω_t represents the set of all pixel coordinates in the image domain of frame t.

Tracking. In the tracking stage, the Gaussian set G_i is fixed, and only the current estimated camera pose T_{cam} is optimized by minimizing:

$$L = L_c + \lambda_d\, L_d, \tag{3}$$

where $\lambda_d \geq 0$ balances the two losses. During backpropagation, gradients are first propagated to each Gaussian and then to the camera pose:

$$\frac{\partial L}{\partial G_i} \longrightarrow \frac{\partial L}{\partial T_{\text{cam}}} \longrightarrow T_{\text{cam}} \text{ update}. \tag{4}$$

The per-frame gradient magnitude of each Gaussian is:

$$g_i = \left\|\nabla_{G_i} L\right\|_2, \tag{5}$$

which is not used to update G_i during tracking but is later reused to compute tile-level Gaussian budgets. The pose update is performed in Lie algebra form:

$$T_{\text{cam}} \leftarrow \exp\big(\widehat{\delta\boldsymbol{\xi}}\big)\, T_{\text{cam}}, \quad \delta\boldsymbol{\xi} = -\eta\, \nabla_{T_{\text{cam}}} L, \tag{6}$$

where η is the learning rate and $\widehat{(\cdot)}$ is the $\mathfrak{se}(3)$ hat operator.

Mapping. In the mapping stage, the camera poses T_{cam} are fixed, and the Gaussian set $\{G_i\}$ is optimized by minimizing

$$L = L_c + \lambda_d^{\star} L_d. \tag{7}$$

Here, $\lambda_d^{\star}$ denotes a potentially different weighting coefficient used in mapping to balance photometric and depth terms. Gradients are propagated to each Gaussian parameter and used for updates:

$$\nabla_{G_i} L = \nabla_{G_i}\big(L_c + \lambda_d L_d\big), \tag{8}$$

$$G_i \leftarrow G_i - \alpha\, \nabla_{G_i} L, \tag{9}$$

where α is the learning rate. The mapping optimization continues until convergence, after which the rendering-area–aware pruning is performed to remove redundant Gaussians.

B. Rendering-Area–Aware Pruning

After the mapping stage has converged, we prune Gaussians based on their rendering contribution in the current frame. For each Gaussian G_i, the pixel-level contribution is:

$$\alpha_i(\mathbf{p}) = o_i \cdot \exp\big(-\tfrac{1}{2}(\mathbf{p} - \boldsymbol{u}_i)^{\top} \Lambda_i^{-1} (\mathbf{p} - \boldsymbol{u}_i)\big), \tag{10}$$

where $\boldsymbol{u}_i$ and Λ_i denote the projected mean and covariance. Aggregating over all pixels yields the coverage

$$C_i = \sum_{\mathbf{p}\in\Omega} \alpha_i(\mathbf{p}), \qquad S_i = \frac{C_i}{\sum_j C_j}. \tag{11}$$

Within each tile k, we rank Gaussians by S_i and retain only the top B_k^{trk}, where B_k^{trk} is determined by the Tile-Level Budget Mechanism to be defined in the next subsection. Since newly added Gaussians in the current mapping round have not yet undergone tracking-based budget allocation, they are exempt from pruning until subsequent iterations.

In outdoor SLAM scenarios, particularly in autonomous driving datasets, Gaussians with large rendering areas frequently correspond to critical structures such as roads and sky. These regions are essential for robust localization and navigation, making Gaussians with larger rendering areas especially important in outdoor environments. However, relying solely on rendering contribution introduces several challenges in complex outdoor scenes, which motivates our Tile-Level Budget Mechanism(Sec. III-C).

C. Tile-Level Budget Mechanism

Outdoor SLAM scenes are often highly complex [26]–[28]: some regions are texture-rich and information-dense, while others are relatively sparse. If pruning were conducted solely based on rendering area, two key issues would emerge. First, texture-rich regions are typically represented by a large number of small Gaussians. Uniform pruning by area would risk eliminating nearly all Gaussians in these regions, leaving them empty. Second, texture-sparse regions can often be represented by only a few Gaussians. As the global pruning ratio increases, removing these few Gaussians would result in the complete loss of information in such regions.

As illustrated in Fig. 3(a), when the pruning ratio gradually increases, both tracking accuracy (ATE) and mapping accuracy (PSNR) degrade significantly. The root cause is that pruning based solely on rendering area tends to prioritize the removal of small Gaussians. However, these small Gaussians are predominantly concentrated in texture-rich regions, and deleting them without restriction eliminates most of the Gaussians in these areas. Consequently, the rendered images in these regions experience a substantial loss of fidelity, as shown in Fig. 3(b), ultimately compromising overall SLAM accuracy. Therefore, introducing a budget mechanism to prevent regions from being entirely depleted is essential for ensuring reliable SLAM performance.

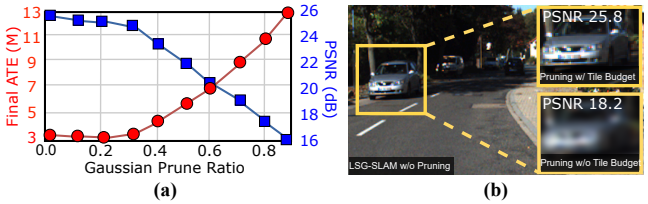
To this end, we leverage the Gaussian gradients obtained in tracking to assign a survivor budget for each image tile. For the set of Gaussians $\{G_i\}$ projected into tile $\mathcal{T}_k$, we compute the average gradient magnitude:

$$G_k = \frac{1}{N_k} \sum_{i \in \mathcal{T}_k} g_i, \tag{12}$$

where N_k is the number of Gaussians in tile k. A higher G_k indicates that the region is texture-dense and carries strong constraints, thus requiring a larger budget; conversely, a lower G_k suggests weaker constraints. Given a global target N_{tar}, we allocate per-tile budgets as:

$$B_k^{\text{trk}} = \text{clip}\left(\left\lceil N_{\text{tar}} \cdot \frac{G_k}{\sum_j G_j} \right\rceil, B_{\min}, B_{\max} \right), \tag{13}$$

where $B_{\min}$ and $B_{\max}$ ensure that no tile is completely depleted or overly concentrated.



(a) (b)

Fig. 3. Results on KITTI [14] sequence 3. Experiments are conducted using LSG-SLAM [7] as the baseline. (a) illustrates the variation of ATE and PSNR under different pruning ratios when applying Rendering-Area–Aware Pruning without the Tile-Level Budget. (b) shows a sample scene from sequence 13 at a pruning ratio of 0.6, comparing results with and without the Tile-Level Budget. This visualization highlights why increasing the pruning ratio leads to a significant degradation in SLAM accuracy.

This process is performed during the tracking stage. Since G_i are not updated in tracking, the resulting tile-level budgets remain relatively stable across iterations, providing consistent and reliable guidance for subsequent rendering-area–aware pruning. In particular, texture-rich regions with larger gradients are allocated higher budgets, whereas texture-sparse regions are assigned lower budgets. The key principle of our design is to preserve information across all regions, ensuring that neither dense nor sparse areas suffer complete information loss. By combining this budget allocation with rendering-area–aware pruning, our method effectively balances memory efficiency and reconstruction fidelity.

IV. Experimental Results

In this section, we present extensive real-world experiments to evaluate the effectiveness of the proposed Pocket-SLAM. We further conduct systematic comparisons with state-of-the-art 3DGS pruning algorithms, and the results demonstrate that in complex large-scale outdoor scenarios, Pocket-SLAM delivers superior performance and higher efficiency compared to existing methods.

Datasets. We conduct evaluations on two widely used outdoor benchmarks. The EuRoC MAV dataset [29] includes outdoor sequences MH01–MH05, recorded by a micro aerial vehicle in challenging environments with significant viewpoint and illumination variations. The KITTI odometry dataset [14] provides large-scale outdoor driving sequences (00–10, except 01), captured by a vehicle equipped with stereo cameras and GPS/IMU sensors, covering diverse urban, residential, and rural scenes. Results for sequence 01 are omitted, as all methods perform poorly on it.

Metrics. We evaluate Pocket-SLAM from two perspectives and compare it with competitive baselines. The first perspective is ***Accuracy***. For trajectory accuracy, we report the root-mean-square error (RMSE) of the absolute trajectory error (ATE) [30]. For reconstruction quality, we measure peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [31], and learned perceptual image patch similarity (LPIPS) [32]. The second perspective is ***Performance***. We use frames per second (FPS) and peak memory consumption as evaluation metrics. Notably, peak memory consumption accounts for all intermediate parameters involved in 3DGS-SLAM computation, reflecting the minimum memory required for a device to execute 3DGS-SLAM.

Baselines. Our proposed Pocket-SLAM is implemented on top of the LSG-SLAM [7] pipeline, while its pruning strategy is generic and can be seamlessly integrated into other SLAM frameworks. To validate its effectiveness, we compare it against several representative pruning-based methods in the 3DGS literature, including LightGaussian [9], which prunes Gaussians primarily based on opacity, removing those with negligible contributions to the rendered image; LP-3DGS [11], which employs gradient magnitude as an importance signal and discards Gaussians with limited updates during optimization; and MaskGaussian [12], which adopts a masking–then–pruning scheme, where Gaussians with low rendering weights are first deactivated before being permanently pruned. These pruning methods, as well as Pocket-SLAM, are applied within the 3DGS-SLAM pipeline for comparison. It is worth noting that PUP-3DGS [10] is designed from the perspective of perceptual image quality, which does not directly align with tracking accuracy, and thus is not included in our comparison. Our method is orthogonal to other 3DGS acceleration techniques [33]–[35] and can be combined with them for further performance improvement. For fair evaluation, all methods are compared under the same pruning ratio on the NVIDIA A6000 [36] platform.

Implementation Details. We perform 50 iterations per frame in the tracking stage and 100 iterations per keyframe in the mapping and loop optimization stages. For loss weighting, we set $\lambda_d = 1.0$ in Eq. 3 and $\lambda_d^\star = 1.5$ in Eq. 7. For pruning, the global target Gaussian count N_{tar} is set to $0.4N_{\text{init}}$, where N_{init} denotes the number of Gaussians before pruning. The minimum per-tile budget $B_{\min}$ is fixed at 5 to prevent regions from being completely emptied, while the maximum budget $B_{\max}$ is set to 200 to avoid excessive concentration of Gaussians within a single tile. These hyperparameters ensure balanced pruning across dense and sparse regions while preserving reconstruction fidelity.

A. Evaluation on EuRoC

Tracking and Mapping Accuracy. Comprehensive evaluation results are presented in Tab. I. Under a unified pruning ratio, only our method achieves tracking and mapping accuracy comparable to LSG-SLAM [7]. In contrast, LightGaussian [9] and LP-3DGS [11] often fail on challenging sequences (e.g., MH04, MH05) because their pruning relies on local Gaussian heuristics (e.g., opacity or gradient magnitude) that do not capture each Gaussian's image contribution, frequently removing essential Gaussians in texture-sparse regions and causing drift. MaskGaussian [12], while completing all five sequences, still shows a notable accuracy drop. It prunes by global Gaussian importance but neglects the need to preserve Gaussians critical for reconstructing the *current* frame, where per-frame fidelity is fundamental for stable tracking. By contrast, our rendering-area–aware pruning with the tile-level budget mechanism evaluates each Gaussian's effective pixel coverage on the current frame and ensures balanced preservation across texture-rich and texture-sparse regions, thereby maintaining stable and reliable tracking and high-fidelity reconstruction.

TABLE I. Camera tracking and rendering results on EuRoC [29]. Tracking is evaluated by ATE RMSE ↓ [m], Rendering is evaluated by PSNR ↑ [dB], SSIM ↑ [0-1], and LPIPS ↓ [0-1]. "-" means tracking lost.

Method	Metric	MH01	MH02	MH03	MH04	MH05	Avg.
LSG-SLAM [7]	ATE↓	**0.05**	**0.02**	**0.08**	0.06	0.11	0.06
	PSNR↑	**31.23**	**33.31**	30.26	**30.72**	**28.54**	30.81
	SSIM↑	**0.98**	**0.99**	**0.98**	**0.98**	**0.96**	0.98
	LPIPS↓	**0.04**	**0.03**	**0.05**	**0.05**	**0.07**	0.05
LightGaussian [9]	ATE↓	1.32	0.94	-	1.67	-	1.31
	PSNR↑	13.24	12.85	-	10.94	-	12.34
	SSIM↑	0.54	0.55	-	0.50	-	0.53
	LPIPS↓	0.41	0.39	-	0.45	-	0.41
LP-3DGS [11]	ATE↓	0.44	0.35	1.34	0.54	-	0.67
	PSNR↑	15.65	16.25	13.84	12.83	-	14.64
	SSIM↑	0.64	0.68	0.60	0.60	-	0.63
	LPIPS↓	0.33	0.32	0.37	0.40	-	0.35
MaskGaussian [12]	ATE↓	0.24	0.18	0.38	0.24	2.38	1.08
	PSNR↑	20.30	21.84	19.26	18.66	17.23	19.4
	SSIM↑	0.76	0.77	0.72	0.71	0.68	0.72
	LPIPS↓	0.29	0.28	0.32	0.33	0.35	0.31
Ours (w/o Tile Budget)	ATE↓	0.16	0.09	0.26	0.15	1.24	0.38
	PSNR↑	24.28	25.49	23.84	22.36	20.18	23.23
	SSIM↑	0.84	0.85	0.83	0.82	0.77	0.82
	LPIPS↓	0.21	0.20	0.23	0.24	0.29	0.23
Ours (w/ Tile Budget)	ATE↓	**0.05**	0.03	**0.08**	**0.05**	0.13	0.07
	PSNR↑	31.05	32.45	**30.28**	30.55	28.33	30.53
	SSIM↑	**0.98**	**0.99**	**0.98**	0.97	**0.96**	0.98
	LPIPS↓	**0.05**	**0.03**	**0.05**	**0.05**	0.08	0.05

TABLE II. Performance on EuRoC [29], reported as Peak Memory ↓ [GB] and FPS ↑ (lower is better for memory, higher is better for FPS).

Method	Metric	MH01	MH02	MH03	MH04	MH05	Avg.
LSG-SLAM [7]	Memory↓	24.2	21.4	25.6	30.2	25.6	25.4
	FPS↑	1.2	1.5	1.2	1.1	1.3	1.3
MaskGaussian [12]	Memory↓	16.3	13.8	17.9	22.2	17.8	17.6
	FPS↑	1.5	1.9	1.6	1.5	1.6	1.6
Ours	Memory↓	**9.6**	**8.4**	**10.2**	**12.1**	**10.2**	10.1
	FPS↑	**3.3**	**4.2**	**3.5**	**3.3**	**3.6**	3.6

Peak Memory and Runtime Performance. We compare three algorithms that successfully complete all five EuRoC sequences as baselines. As shown in Tab. II, our method reduces peak memory consumption by 61.3% on average across the five sequences and achieves a 2.7× improvement in FPS. In contrast, MaskGaussian [12], although adopting the same pruning ratio as ours, employs a mask-based strategy in which low-importance Gaussians are not immediately removed but retained in memory for a period of time. This deferred deletion results in much weaker memory reduction, yielding only about 30% savings compared to the baseline. Moreover, by substantially lowering peak memory consumption, our method significantly improves runtime efficiency, achieving a 2.2× speedup over MaskGaussian.

B. Evaluation on KITTI

Tracking and Mapping Accuracy. Detailed evaluation results are presented in Tab. III. Our rendering-area–aware pruning strategy consistently achieves tracking accuracy comparable to LSG-SLAM [7] across all sequences. Moreover, in challenging cases such as sequences 00, 04, 05, 06, and 08, by effectively eliminating redundant Gaussians,

TABLE III. Camera tracking and rendering results on KITTI [14]. Tracking is evaluated by ATE RMSE ↓ [m], Rendering is evaluated by PSNR ↑ [dB], SSIM ↑ [0-1], and LPIPS ↓ [0-1]. ”-” means tracking lost.The results of sequence 01 are not reported, as all methods perform poorly on it.

Method	Metrics	00	02	03	04	05	06	07	08	09	10	Avg.
LSG-SLAM [7]	ATE↓	3.15	**9.22**	**1.88**	1.62	**1.97**	2.62	**1.52**	8.43	**6.06**	**3.05**	3.85
	PSNR↑	**27.09**	**26.64**	26.18	**25.46**	26.90	**27.79**	25.98	**26.17**	**26.81**	**26.82**	26.58
	SSIM↑	**0.97**	**0.97**	**0.97**	**0.95**	**0.97**	**0.97**	**0.96**	0.96	**0.97**	**0.97**	0.97
	LPIPS↓	**0.07**	**0.07**	**0.07**	**0.09**	**0.07**	**0.06**	**0.07**	0.08	**0.07**	**0.07**	0.07
LightGaussian [9]	ATE↓	-	-	15.23	16.84	19.85	20.21	19.94	-	-	21.23	18.88
	PSNR↑	-	-	15.83	15.27	16.02	16.20	15.28	-	-	16.43	15.83
	SSIM↑	-	-	0.79	0.78	0.80	0.80	0.80	-	-	0.79	0.79
	LPIPS↓	-	-	0.33	0.31	0.33	0.32	0.32	-	-	0.32	0.32
LP-3DGS [11]	ATE↓	9.20	-	8.98	7.26	10.83	11.29	9.23	-	-	12.45	9.89
	PSNR↑	18.57	-	18.34	17.39	18.84	19.22	18.43	-	-	17.76	18.36
	SSIM↑	0.84	-	0.83	0.81	0.83	0.83	0.82	-	-	0.82	0.83
	LPIPS↓	0.24	-	0.26	0.24	0.26	0.25	0.25	-	-	0.24	0.25
MaskGaussian [12]	ATE↓	5.42	14.23	4.29	3.88	4.29	5.32	4.53	13.87	12.32	6.94	7.51
	PSNR↑	21.72	20.83	20.98	19.29	20.31	21.97	20.57	19.99	20.03	19.98	20.56
	SSIM↑	0.92	0.91	0.92	0.90	0.90	0.91	0.91	0.90	0.89	0.92	0.91
	LPIPS↓	0.18	0.20	0.18	0.22	0.19	0.18	0.18	0.16	0.19	0.19	0.19
Ours (w/o Tile Budget)	ATE↓	4.23	12.48	2.47	2.19	2.95	3.34	3.26	10.63	7.62	4.56	5.37
	PSNR↑	24.63	23.81	23.54	22.83	23.61	24.62	23.84	22.45	23.91	22.29	23.55
	SSIM↑	0.95	0.94	0.95	0.93	0.93	0.94	0.94	0.93	0.92	0.95	0.94
	LPIPS↓	0.12	0.14	0.12	0.16	0.13	0.12	0.12	0.11	0.13	0.13	0.13
Ours (w/ Tile Budget)	ATE↓	**3.13**	9.24	1.92	**1.58**	**1.97**	**2.58**	1.55	**7.32**	6.83	3.22	3.81
	PSNR↑	27.01	26.31	**26.49**	25.31	**26.95**	27.33	**26.31**	26.10	26.33	26.55	26.46
	SSIM↑	**0.97**	**0.97**	**0.97**	**0.95**	**0.97**	**0.97**	**0.96**	**0.97**	0.96	**0.97**	0.97
	LPIPS↓	**0.07**	**0.07**	**0.07**	**0.09**	**0.07**	**0.06**	**0.07**	**0.07**	0.08	0.08	0.07

TABLE IV. Performance on KITTI [14], reported as Peak Memory ↓ [GB] and FPS ↑ (lower is better for memory, higher is better for FPS).

Method	Metrics	00	02	03	04	05	06	07	08	09	10	Avg.
LSG-SLAM [7]	Memory ↓	40.2	36.6	32.1	37.1	30.6	31.2	38.5	36.2	30.4	34.2	34.7
	FPS ↑	0.7	0.8	0.9	0.7	1.1	0.9	0.7	0.8	1.0	0.8	0.8
MaskGaussian [12]	Memory ↓	35.2	31.6	26.3	32.3	24.8	25.2	32.4	30.5	24.3	29.6	29.2
	FPS ↑	0.9	1.0	1.1	0.9	1.4	1.1	0.9	1.0	1.3	1.1	1.1
Ours	Memory ↓	**13.9**	**12.7**	**10.8**	**12.9**	**10.2**	**10.5**	**14.0**	**12.4**	**10.1**	**13.3**	11.9
	FPS ↑	**2.0**	**2.2**	**2.5**	**2.0**	**3.1**	**2.4**	**1.8**	**2.3**	**2.9**	**2.3**	2.4

our method even surpasses LSG-SLAM, demonstrating that carefully designed pruning can not only preserve but also enhance robustness in large-scale outdoor environments. By contrast, since the KITTI dataset [14] targets autonomous driving and its scenes are larger and more expansive than EuRoC [29], pruning methods based on gradient or opacity, including LightGaussian [9] and LP-3DGS [11], experience severe tracking loss due to overly Gaussian removal.

Peak Memory and Runtime Performance. Detailed evaluation results are presented in Tab. IV. We compare three algorithms that successfully track all KITTI 00–10 sequences: LSG-SLAM [7], MaskGaussian [12], and our Pocket-SLAM. As shown in the table, Pocket-SLAM reduces peak memory consumption by 65.7% while achieving a 2.9× speedup in FPS. Notably, the improvement on KITTI is even more pronounced than on EuRoC, not only in runtime performance but also in tracking accuracy. This is because KITTI, as an autonomous driving benchmark, contains much larger outdoor environments where most of the important Gaussians are concentrated in wide rendering areas such as roads and sky. Our rendering-area–aware pruning strategy can more effectively distinguish essential from redundant Gaussians in such settings, yielding substantial memory savings and improved accuracy. A more detailed explanation of this mechanism is provided in Sec. IV-C.

C. Result Profiling

Accuracy Result Analysis. Detailed evaluation results are presented in Fig. 4, highlighting the limitations of existing pruning strategies in 3DGS-SLAM. LightGaussian [9] prunes Gaussians based on opacity, whereas LP-3DGS [11] relies on gradient magnitude. Both criteria are problematic in outdoor scenarios, since Gaussians covering large but critical regions such as sky and road often exhibit low opacity or gradients and are therefore aggressively removed. This eliminates wide-coverage Gaussians that provide global photometric constraints, leading to unstable pose estimation and frequent drift. In contrast, Pocket-SLAM evaluates Gaussian importance by rendering area, which more accurately reflects their contribution to the image and is particularly well suited for outdoor SLAM and autonomous driving tasks.

MaskGaussian [12] introduces rendering area as a measure but applies it in a globally area-driven manner, thereby overlooking texture-dense regions. Consequently, fine-grained Gaussians encoding trees, bushes, and façades are masked out, weakening the local consistency on which SLAM losses directly rely. Pocket-SLAM addresses this limitation by augmenting rendering-area pruning with a tile-level budget

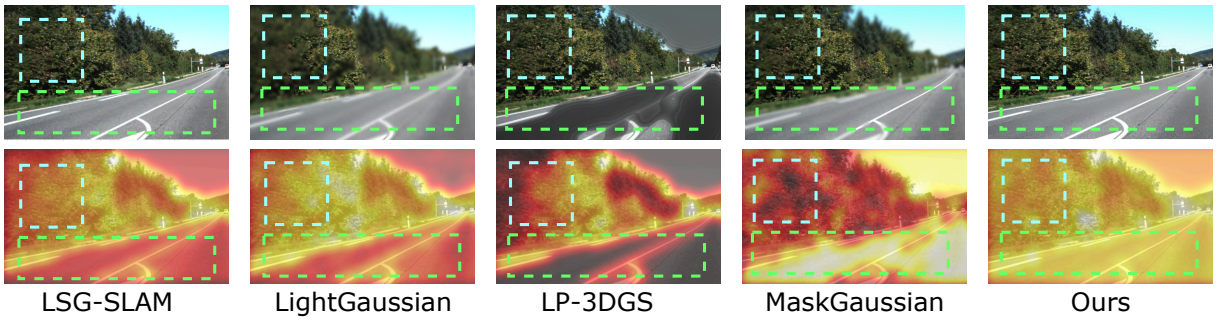


Fig. 4. Results on KITTI [14] sequence 10. We compare five methods: LSG-SLAM [7], LightGaussian [9], LP-3DGS [11], MaskGaussian [12], and our approach. For each method, the top row shows the rendered image, while the bottom row presents the Gaussian density heatmap, where brighter colors indicate regions with higher Gaussian counts. Texture-dense and texture-sparse regions are highlighted with blue and green boxes, respectively.

mechanism, which preserves small Gaussians in texture-rich areas while still pruning redundant ones. This balanced design maintains both global constraints and local detail, enabling significant memory reduction without compromising tracking or reconstruction accuracy.

In summary, Pocket-SLAM addresses limitations of pruning methods by introducing Rendering-Area–Aware Pruning, which resolves the mismatch in Gaussian importance estimation for outdoor scenarios. Additionally, the Tile-Level Budget mechanism mitigates information loss in texture-dense regions caused by purely area-driven pruning. With these designs, Pocket-SLAM achieves stable tracking and accurate mapping while substantially reducing memory consumption.

Peak Memory Usage Result Analysis. We observe that memory consumption fluctuates at each keyframe, since mapping operations periodically introduce new Gaussians. Consequently, the change in the number of Gaussians at each keyframe directly determines the peak memory consumption. Although MaskGaussian [12] adopts the same pruning ratio as our method, it relies on a mask-pruning strategy in which low-importance Gaussians are temporarily deactivated rather than immediately removed. As a result, a large number of redundant Gaussians remain stored in memory for longer throughout the process. In practice, this means that at each keyframe MaskGaussian removes far fewer Gaussians than our method, which directly eliminates redundant splats. Consequently, its peak memory consumption is reduced only marginally, whereas our approach achieves substantially larger memory savings. This observation is consistent with the trends shown in Fig. 5.

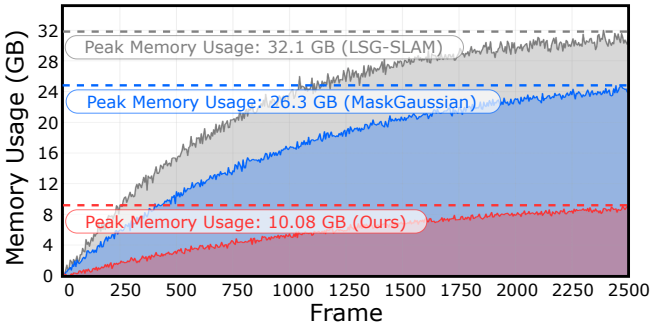


Fig. 5. Results on KITTI [14] sequence 3, showing the trend of memory usage growth with frames for LSG-SLAM [7], MaskGaussian [12], and Pocket-SLAM. All Gaussian-related parameters are included.

V. Conclusions and Future Work

In this paper, we present Pocket-SLAM, a memory-efficient 3DGS-SLAM framework that integrates a *rendering-area–aware pruning strategy* with a *tile-level budget mechanism*. While rendering-area–aware pruning reduces redundant Gaussians by evaluating their pixel coverage, it inevitably removes small Gaussians in texture-dense regions, leading to local information loss and degraded mapping quality. To address this issue, Pocket-SLAM incorporates the tile-level budget mechanism, which allocates survival budgets across tiles according to texture distribution. This ensures that both large-area Gaussians providing global constraints (e.g., roads and sky) and small Gaussians in texture-rich regions (e.g., trees, bushes, façades) are preserved. By combining these two complementary strategies, Pocket-SLAM achieves over 60% memory reduction and more than 2× FPS improvement on the EuRoC [29] and KITTI [14] datasets, while maintaining robust tracking and high-fidelity reconstruction.

Future directions include designing adaptive pruning schedules that dynamically adjust pruning ratios based on scene complexity and temporal variations; exploring long-term, dynamic outdoor scenarios such as autonomous driving at night, across seasonal changes, or under severe weather conditions; and deploying Pocket-SLAM on resource-constrained hardware platforms such as embedded GPUs [37] and edge accelerators [38]–[40] to enable practical and reliable robotic applications.

REFERENCES

[1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[2] H. Matsuki, R. Murai, P. H. J. Kelly, and A. J. Davison, "Gaussian Splatting SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[3] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Wang, and X. Li, "Gs-slam: Dense visual slam with 3d gaussian splatting," 2024. [Online]. Available: https://arxiv.org/abs/2311.11700

[4] P.-C. Kung, X. Zhang, K. A. Skinner, and N. Jaipuria, "Lihi-gs: Lidar-supervised gaussian splatting for highway driving scene reconstruction," 2024. [Online]. Available: https://arxiv.org/abs/2412.15447

[5] L. Cheng, Z. Qi, Z. Zhou, C. Lu, and G. Xiong, "Lt-gaussian: Long-term map update using 3d gaussian splatting for autonomous driving," 2025. [Online]. Available: https://arxiv.org/abs/2508.01704

[6] J. Tang, Y. Gao, D. Yang, L. Yan, Y. Yue, and Y. Yang, "Dronesplat: 3d gaussian splatting for robust 3d reconstruction from in-the-wild drone imagery," 2025. [Online]. Available: https://arxiv.org/abs/2503.16964

[7] Z. Xin, C. Wu, P. Huang, Y. Zhang, Y. Mao, and G. Huang, "Large-scale gaussian splatting slam," 2025. [Online]. Available: https://arxiv.org/abs/2505.09915

[8] J. Zheng, Z. Zhu, V. Bieri, M. Pollefeys, S. Peng, and I. Armeni, "Wildgs-slam: Monocular gaussian splatting slam in dynamic environments," 2025. [Online]. Available: https://arxiv.org/abs/2504.03886

[9] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang, "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps," 2024. [Online]. Available: https://arxiv.org/abs/2311.17245

[10] A. Hanson, A. Tu, V. Singla, M. Jayawardhana, M. Zwicker, and T. Goldstein, "Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting," 2025. [Online]. Available: https://arxiv.org/abs/2406.10219

[11] Z. Zhang, T. Song, Y. Lee, L. Yang, C. Peng, R. Chellappa, and D. Fan, "Lp-3dgs: Learning to prune 3d gaussian splatting," 2024. [Online]. Available: https://arxiv.org/abs/2405.18784

[12] Y. Liu, Z. Zhong, Y. Zhan, S. Xu, and X. Sun, "Maskgaussian: Adaptive 3d gaussian representation from probabilistic masks," 2025. [Online]. Available: https://arxiv.org/abs/2412.20522

[13] D. Gao, P. Z. X. Li, V. Sze, and S. Karaman, "Gevo: Memory-efficient monocular visual odometry using gaussians," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, p. 2774–2781, Mar. 2025. [Online]. Available: http://dx.doi.org/10.1109/LRA.2025.3534683

[14] Y. Liao, J. Xie, and A. Geiger, "Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d," 2022. [Online]. Available: https://arxiv.org/abs/2109.13410

[15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, p. 1147–1163, Oct. 2015. [Online]. Available: http://dx.doi.org/10.1109/TRO.2015.2463671

[16] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, p. 1874–1890, Dec. 2021. [Online]. Available: http://dx.doi.org/10.1109/TRO.2021.3075644

[17] Z. Liao, W. Wang, X. Qi, X. Zhang, L. Xue, J. Jiao, and R. Wei, "Object-oriented slam using quadrics and symmetry properties for indoor environments," 2020. [Online]. Available: https://arxiv.org/abs/2004.05303

[18] M. Filipenko and I. Afanasyev, "Comparison of various slam systems for mobile robot in an indoor environment," in *2018 International Conference on Intelligent Systems (IS)*. IEEE, Sep. 2018, p. 400–407. [Online]. Available: http://dx.doi.org/10.1109/IS.2018.8710464

[19] R. Mur-Artal and J. D. Tardos, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, p. 1255–1262, Oct. 2017. [Online]. Available: http://dx.doi.org/10.1109/TRO.2017.2705103

[20] X. Li, Z. Gong, F. Tosi, M. Poggi, S. Mattoccia, D. Liu, and J. Wu, "Stereo 3d gaussian splatting slam for outdoor urban scenes," 2025. [Online]. Available: https://arxiv.org/abs/2507.23677

[21] D. Esparza and G. Flores, "The stdyn-slam: A stereo vision and semantic segmentation approach for slam in dynamic outdoor environments," 2021. [Online]. Available: https://arxiv.org/abs/2010.09857

[22] R. Xiao, W. Liu, Y. Chen, and L. Hu, "Liv-gs: Lidar-vision integration for 3d gaussian splatting slam in outdoor environments," 2024. [Online]. Available: https://arxiv.org/abs/2411.12185

[23] J. L. Blanco-Claraco, "A flexible framework for accurate lidar odometry, map manipulation, and localization," *The International Journal of Robotics Research*, vol. 44, no. 9, p. 1553–1599, Feb. 2025. [Online]. Available: http://dx.doi.org/10.1177/02783649251316881

[24] H. Huang, L. Li, H. Cheng, and S.-K. Yeung, "Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras," 2024. [Online]. Available: https://arxiv.org/abs/2311.16728

[25] L. Zhu, Y. Li, E. Sandström, S. Huang, K. Schindler, and I. Armeni, "Loopsplat: Loop closure by registering 3d gaussian splats," 2024. [Online]. Available: https://arxiv.org/abs/2408.10154

[26] F. Schmidt, C. Blessing, M. Enzweiler, and A. Valada, "Visual-inertial slam for unstructured outdoor environments: Benchmarking the benefits and computational costs of loop closing," 2025. [Online]. Available: https://arxiv.org/abs/2408.01716

[27] F. Schmidt, J. Daubermann, M. Mitschke, C. Blessing, S. Meyer, M. Enzweiler, and A. Valada, "Rover: A multiseason dataset for visual slam," *IEEE Transactions on Robotics*, vol. 41, p. 4005–4022, 2025. [Online]. Available: http://dx.doi.org/10.1109/TRO.2025.3577026

[28] O. Álvarez Tuñón, Y. Brodskiy, and E. Kayacan, "Monocular visual simultaneous localization and mapping: (r)evolution from geometry to deep learning-based pipelines," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 5, p. 1990–2010, May 2024. [Online]. Available: http://dx.doi.org/10.1109/TAI.2023.3321032

[29] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract

[30] A. Kasar, "Benchmarking and comparing popular visual slam algorithms," 2018. [Online]. Available: https://arxiv.org/abs/1811.09895

[31] Z. 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*, vol. 13, no. 4, pp. 600–612, 2004.

[32] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," 2018. [Online]. Available: https://arxiv.org/abs/1801.03924

[33] X. Wang, R. Yi, and L. Ma, "Adr-gaussian: Accelerating gaussian splatting with adaptive radius," in *SIGGRAPH Asia 2024 Conference Papers*. ACM, Dec. 2024, p. 1–10. [Online]. Available: http://dx.doi.org/10.1145/3680528.3687675

[34] Z. Liao, J. Ding, R. Fu, S. Cui, R. Gong, L. Wang, B. Hu, Y. Wang, H. Li, X. Zhang, and H. Wang, "Tc-gs: A faster gaussian splatting module utilizing tensor cores," 2025. [Online]. Available: https://arxiv.org/abs/2505.24796

[35] Y. Wang, S. Chen, and R. Yi, "Sg-splatting: Accelerating 3d gaussian splatting with spherical gaussians," 2024. [Online]. Available: https://arxiv.org/abs/2501.00342

[36] "Nvidia rtx a6000 graphics card," NVIDIA, Workstation GPU, 2020, 48 GB GDDR6 ECC memory, Ampere architecture, PCIe 4.0, Workstation graphics accelerator. [Online]. Available: https://www.nvidia.com/en-us/products/workstations/rtx-a6000/

[37] "Jetson orin for next-gen robotics — nvidia," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/, (Accessed on 04/02/2024).

[38] L. Wu, H. Zhu, S. He, J. Zheng, C. Chen, and X. Zeng, "Gauspu: 3d gaussian splatting processor for real-time slam systems," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2024, pp. 1562–1573.

[39] H. He, G. Li, F. Liu, L. Jiang, X. Liang, and Z. Song, "Gsarch: Breaking memory barriers in 3d gaussian splatting training via architectural support," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 366–379.

[40] S. Durvasula, A. Zhao, F. Chen, R. Liang, P. K. Sanjaya, and N. Vijaykumar, "Distwar: Fast differentiable rendering on raster-based rendering pipelines," 2023. [Online]. Available: https://arxiv.org/abs/2401.05345