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

Leshu Li[1], Jie Peng[2], Yang Zhao[1]
[1]Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities, USA
[2]Department of Computer Science, University of North Carolina at Chapel Hill, USA

Abstract

3D Gaussian Splatting (3DGS) has recently gained attention in Simultaneous Localization and Mapping (SLAM) for its ability to capture fine-grained geometry and synthesize high-quality novel views. However, in large-scale scenarios such as autonomous driving, 3DGS-SLAM suffers from continuously increasing memory consumption as Gaussians accumulate, limiting practical deployment. We propose a rendering-area–aware pruning strategy that removes Gaussians according to their effective image-plane contribution, rather than relying solely on local heuristics such as opacity or gradient magnitude. By directly targeting rendering redundancy, our method significantly reduces peak runtime memory usage. Experiments on the EuRoC and KITTI datasets show that our approach achieves over 60% memory reduction and more than 2× FPS improvement while maintaining localization and reconstruction accuracy. These results demonstrate the effectiveness of rendering-aware pruning for scalable 3DGS-SLAM in real-world autonomous driving settings. Our code is publicly available at https://github.com/UMN-ZhaoLab/Pocket-SLAM.git.

Keywords

3D Gaussian Splatting; Visual SLAM; 3D Reconstruction; Autonomous Driving; Edge Deployment

ACM Reference Format:
Leshu Li[1], Jie Peng[2], Yang Zhao[1]. 2026. Pocket-SLAM: Rendering-Area-Aware Pruning for Memory-Efficient 3DGS-SLAM. In . ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Visual SLAM is fundamental to robot perception and supports applications such as autonomous navigation and environment mapping. 3D Gaussian Splatting (3DGS) [16] has recently been integrated into SLAM due to its ability to represent fine-grained geometry and synthesize high-quality novel views. Systems such as MonoGS [22] and GS-SLAM [29] demonstrate accurate tracking and high-fidelity reconstruction, while LGS-SLAM [28] and WildGS-SLAM [31] extend 3DGS-SLAM to large-scale outdoor scenarios, highlighting its potential for autonomous driving [6, 17] and drone navigation [25].

VisArch'26, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-XXXX-X/2026/02
https://doi.org/XXXXXXX.XXXXXXX

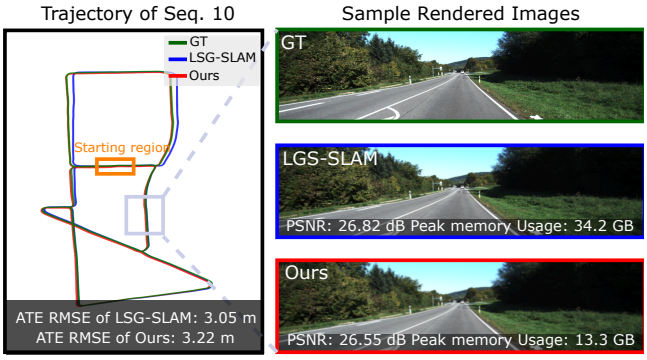

Figure 1: Results on KITTI [19] sequence 10. Compared with LSG-SLAM [28], our method achieves comparable camera tracking accuracy (Absolute Trajectory Error Root Mean Square Error, ATE) and rendering quality (Peak Signal-to-Noise Ratio, PSNR) in large-scale scenarios, while reducing peak memory usage by 61%.

Despite these advances, a key limitation remains: memory consumption grows continuously as Gaussians accumulate during mapping. While indoor scenes contain a manageable number of Gaussians, large-scale outdoor environments require millions of splats, leading to prohibitively high peak memory usage. This issue is critical for deployment on resource-constrained edge devices, where peak runtime memory directly determines feasibility.

Existing Gaussian pruning methods primarily target 3DGS rendering in small-scale indoor scenes [9, 12, 21, 30] or focus on keyframe storage [11]. However, they largely overlook peak runtime memory consumption in large-scale outdoor SLAM. Moreover, pruning strategies designed for indoor scenes fail to account for the structural differences of outdoor environments, where vast texture-sparse regions coexist with dense local structures, resulting in severe memory redundancy.

To address this challenge, we present Pocket-SLAM, an extension to 3DGS-SLAM that incorporates a *rendering-area–aware pruning strategy*. During mapping, we quantify each Gaussian's contribution based on its effective pixel coverage and remove those with negligible rendering impact. Unlike local heuristics such as opacity or gradient magnitude, our criterion directly reflects scene-level rendering contribution, making it well suited for outdoor SLAM.

To prevent excessive information loss in texture-dense regions, we further introduce a *tile-level budget mechanism*, which allocates survival budgets across image tiles and constrains pruning locally. This design preserves both large-area Gaussians that provide global constraints and fine-grained Gaussians that encode local details. As shown in Fig. 1, our approach significantly reduces peak memory

usage while maintaining tracking and mapping accuracy. Moreover, Pocket-SLAM is orthogonal to existing 3DGS acceleration techniques and can be seamlessly integrated into standard SLAM pipelines. In summary, the contributions of our work include:

- We propose a *rendering-area–aware pruning strategy* that evaluates Gaussian importance based on effective pixel coverage, shifting pruning criteria from local heuristics to rendering contribution.
- We introduce a *tile-level budget mechanism* that constrains pruning per region, preventing information loss in both texture-dense and texture-sparse areas.
- We demonstrate on large-scale outdoor benchmarks that our method reduces peak memory consumption by over 60% and improves FPS by more than 2× while preserving localization and reconstruction accuracy.

2 Related Work

2.1 Traditional Visual SLAM and 3DGS-SLAM

Visual SLAM has been widely studied in both indoor and outdoor settings, where challenges differ significantly. Indoor SLAM frameworks [5, 10, 20, 23, 24] operate in compact, texture-rich environments and achieve reliable tracking using sparse, direct, or dense methods. In contrast, large-scale outdoor scenes are often unstructured and texture-sparse, making traditional approaches less effective. Stereo-based [8, 18] and LiDAR-based systems [3, 27] improve robustness and scale handling but require specialized sensors and still face trade-offs between efficiency and reconstruction fidelity.

With the emergence of 3DGS [16], Gaussian primitives have been incorporated into SLAM systems. MonoGS [22] and GS-SLAM [29] demonstrate accurate tracking and high-quality reconstruction using 3DGS. Photo-SLAM [14] and LoopSplat [32] enhance robustness by integrating ORB-SLAM [23] modules or global BA, though most evaluations focus on indoor scenes. LGS-SLAM [28] and WildGS-SLAM [31] extend 3DGS-SLAM to outdoor driving scenarios, revealing its scalability potential.

2.2 Memory-Efficient Gaussian Pruning

To reduce redundancy in 3DGS models, prior work has explored Gaussian pruning. LightGaussian [9] removes low-opacity Gaussians, LP-3DGS [30] relies on gradient magnitude, PUP-3DGS [12] considers perceptual importance, and MaskGaussian [21] adopts a masking-then-pruning strategy. While effective for 3DGS rendering and training, these methods are not designed for 3DGS-SLAM, which requires stable per-frame tracking. GEVO [11] addresses memory in 3DGS-SLAM but focuses on keyframe storage rather than peak runtime memory. In contrast, 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 SLAM while preserving accuracy.

3 Proposed Methods

Pocket-SLAM is designed to reduce peak runtime memory in large-scale outdoor 3DGS-SLAM while preserving tracking stability and reconstruction fidelity. To achieve this goal, we introduce two complementary components: (1) a *rendering-area–aware pruning strategy* (Sec. 3.2) that evaluates Gaussian importance from the perspective of image contribution, and (2) a *tile-level budget mechanism* (Sec. 3.3) that constrains pruning locally to prevent regional information collapse. Together, these components explicitly target runtime memory redundancy without modifying the underlying SLAM optimization framework.

Pocket-SLAM follows the standard SLAM paradigm consisting of *tracking* and *mapping* stages. Tracking estimates camera poses while collecting gradient statistics for budget allocation, and mapping refines Gaussian parameters before applying structured pruning (Fig. 2).

3.1 SLAM Pipeline

We denote the camera pose as $T_{\text{cam}} \in SE(3)$ and the Gaussian set as $\{G_i\}_{i=1}^{N}$, where each G_i encodes position, covariance, opacity, and color. For each incoming frame, we minimize a joint photometric and depth reconstruction loss:

$$L = L_c + \lambda_d L_d, \tag{1}$$

where L_c measures photometric consistency and L_d enforces depth consistency.

Tracking. During tracking, the Gaussian set $\{G_i\}$ is fixed and only the pose T_{cam} is optimized. Although Gaussian parameters are not updated in this stage, we compute the per-Gaussian gradient magnitude:

$$g_i = \|\nabla_{G_i} L\|_2, \tag{2}$$

which reflects how strongly each Gaussian contributes to pose optimization. These gradient statistics are accumulated and later used to guide tile-level budget allocation.

Mapping. In mapping, camera poses are fixed while Gaussian parameters are optimized. New Gaussians may be inserted to represent newly observed regions. After mapping converges for a keyframe, pruning is performed to remove redundant Gaussians. Importantly, pruning is applied only after optimization to avoid interfering with convergence.

3.2 Rendering-Area–Aware Pruning

Conventional pruning methods rely on local heuristics such as opacity or gradient magnitude, which do not necessarily reflect a Gaussian's contribution to the rendered image. In contrast, we directly measure each Gaussian's effective rendering area.

For each Gaussian G_i, we compute its projected contribution on the image plane:

$$C_i = \sum_{\mathbf{p}\in\Omega} \alpha_i(\mathbf{p}), \qquad S_i = \frac{C_i}{\sum_j C_j}, \tag{3}$$

where $\alpha_i(\mathbf{p})$ denotes the pixel-wise contribution and Ω is the image domain. C_i measures total pixel coverage, and S_i normalizes it across all Gaussians.

Within each tile k, Gaussians are ranked by S_i, and only the top B_k^{trk} are retained. This criterion aligns pruning with scene-level rendering impact. In outdoor environments, Gaussians covering

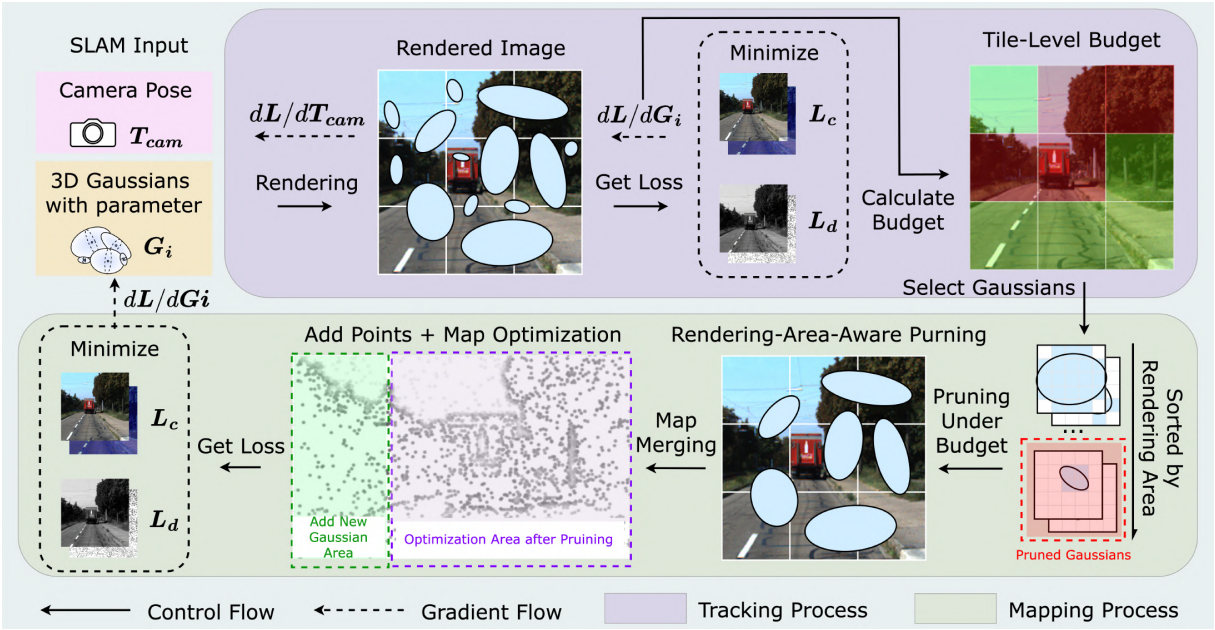

Figure 2: Overview of Pocket-SLAM. Tracking estimates camera poses and computes tile budgets, while mapping refines Gaussians. *Tile-Level Budget Mechanism* guides pruning allocation, and *Rendering-Area–Aware Pruning* removes low-contribution Gaussians to reduce memory without degrading accuracy.

large regions such as roads and sky provide global photometric constraints and thus naturally receive higher priority.

However, coverage-based pruning tends to remove small Gaussians in texture-rich areas, which may degrade local consistency. This motivates tile-level budget control.

3.3 Tile-Level Budget Mechanism

Outdoor scenes exhibit spatial heterogeneity: some regions are texture-dense with many small Gaussians, while others are sparse and dominated by large splats. Uniform pruning based solely on rendering area may remove too many Gaussians in dense regions or deplete sparse tiles, leading to instability (Fig. 3).

To enforce balanced survival across regions, we allocate budgets per image tile using gradient statistics collected during tracking. For Gaussians projected into tile $\mathcal{T}_k$, we compute:

$$G_k = \frac{1}{N_k} \sum_{i \in \mathcal{T}_k} g_i, \tag{4}$$

where N_k is the number of Gaussians in tile k. G_k reflects how strongly the region influences pose optimization.

Given a global target Gaussian count N_{tar}, we assign per-tile survival budgets:

$$B_k^{\text{trk}} = \text{clip}\left(\left\lceil N_{\text{tar}} \cdot \frac{G_k}{\sum_j G_j} \right\rceil, B_{\min}, B_{\max}\right). \tag{5}$$

Tiles with larger gradients (texture-rich, constraint-dense regions) receive higher budgets, while less informative regions receive fewer. The clipping bounds prevent tiles from being either completely depleted or overly concentrated.

By combining tile-level budgets with rendering-area ranking, Pocket-SLAM preserves both global structures and fine-grained local details. This structured pruning mechanism effectively limits peak runtime memory while maintaining tracking stability and reconstruction quality.

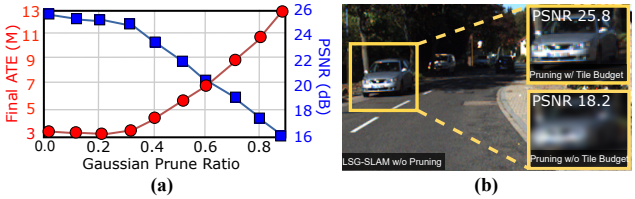

Figure 3: Effect of pruning without and with the Tile-Level Budget on KITTI [19]. Increasing pruning ratio without local constraints significantly degrades ATE and PSNR, while our budget mechanism maintains stable performance.

4 Experimental Results

Datasets. We evaluate on the EuRoC MAV dataset [4] using sequences MH01–MH05, which contain challenging viewpoint and illumination changes. While EuRoC is compact compared to autonomous driving datasets, it provides a benchmark to verify that pruning does not degrade SLAM accuracy. Large-scale outdoor evaluations on KITTI [19] are deferred to Appendix A.1. **Metrics**. We evaluate from two perspectives. ***Accuracy*** is measured by ATE RMSE [15], which reflects trajectory consistency. ***Performance*** is measured by FPS and peak runtime memory consumption. Peak memory includes all Gaussian-related intermediate parameters during tracking and mapping, reflecting the minimum hardware requirement for deployment.

Baselines. Pocket-SLAM is implemented on top of LSG-SLAM [28]. We compare against LightGaussian [9], LP-3DGS [30], and MaskGaussian [21]. All methods are evaluated under the same pruning ratio on NVIDIA A6000 [2] to ensure fairness.

Implementation Details. We use 50 tracking iterations per frame and 100 mapping iterations per keyframe. Loss weights are set to $\lambda_d = 1.0$ and $\lambda_d^\star = 1.5$. The global target Gaussian count is $N_{\text{tar}} = 0.4N_{\text{init}}$, with per-tile bounds $B_{\min} = 5$ and $B_{\max} = 200$. These settings ensure stable optimization while enabling significant memory reduction.

4.1 Evaluation on EuRoC

Tracking Accuracy (ATE). Tab. 1 reports ATE RMSE under a unified pruning ratio. Pocket-SLAM achieves tracking accuracy comparable to LSG-SLAM across all sequences, indicating that rendering-area–aware pruning does not compromise pose estimation stability. In contrast, LightGaussian and LP-3DGS frequently lose tracking on challenging sequences (e.g., MH04 and MH05), as their local pruning criteria remove Gaussians that provide global photometric constraints. MaskGaussian completes all sequences but shows consistent accuracy degradation, reflecting insufficient protection of texture-critical regions.

Table 1: Camera tracking and rendering results on EuRoC [4]. 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 [28]	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 [30]	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 [21]	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 2: Performance on EuRoC [4], 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 [28]	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 [21]	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

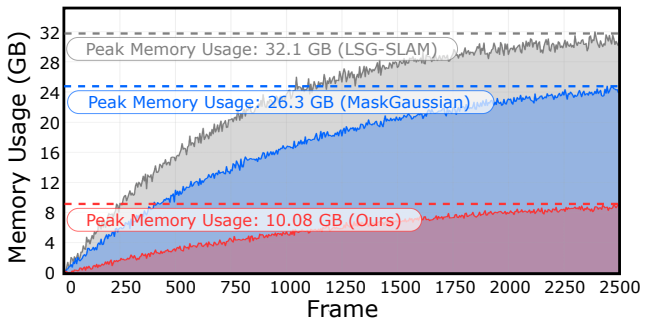

Figure 4: Results on KITTI [19] sequence 3, showing the trend of memory usage growth with frames for LSG-SLAM [28], MaskGaussian [21], and Pocket-SLAM. All Gaussian-related parameters are included.

Peak Memory and FPS. Tab. 2 presents runtime efficiency comparisons. Pocket-SLAM reduces peak memory consumption by 61.3% on average relative to LSG-SLAM while achieving a 2.7× FPS improvement. Compared with MaskGaussian, our method achieves substantially stronger memory reduction and a 2.2× speedup. This difference stems from the mask-based strategy in MaskGaussian, which temporarily retains low-importance Gaussians in memory, whereas Pocket-SLAM directly eliminates redundant splats.

Memory Growth Trend. To further analyze runtime behavior, Fig. 4 illustrates memory usage growth over frames. Memory increases at each keyframe due to Gaussian insertion during mapping. Pocket-SLAM consistently maintains a lower peak memory footprint than LSG-SLAM and MaskGaussian throughout the sequence. This demonstrates that direct rendering-area pruning with tile-level budget control is more effective in limiting peak runtime memory than deferred mask-based pruning.

5 Conclusions and Future Work

We present Pocket-SLAM, a memory-efficient 3DGS-SLAM framework that combines rendering-area–aware pruning with a tile-level budget mechanism. This design removes redundant Gaussians while preserving both global structures and local details, achieving over 60% peak memory reduction and more than 2× FPS improvement on EuRoC [4] without sacrificing tracking accuracy.

Future work includes adaptive pruning strategies and deployment on resource-constrained platforms such as embedded GPUs [1] and edge accelerators [7, 13, 26].

References

[1] [n. d.]. Jetson Orin for Next-Gen Robotics | NVIDIA. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/. (Accessed on 04/02/2024).

[2] 2020. NVIDIA RTX A6000 Graphics Card. NVIDIA, Workstation GPU. https://www.nvidia.com/en-us/products/workstations/rtx-a6000/ 48 GB GDDR6 ECC memory, Ampere architecture, PCIe 4.0, Workstation graphics accelerator.

[3] Jose Luis Blanco-Claraco. 2025. A flexible framework for accurate LiDAR odometry, map manipulation, and localization. *The International Journal of Robotics Research* 44, 9 (Feb. 2025), 1553–1599. doi:10.1177/02783649251316881

[4] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. 2016. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research* (2016). arXiv:http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.full.pdf+html doi:10.1177/0278364915620033

[5] Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez, Jose M. M. Montiel, and Juan D. Tardos. 2021. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Transactions on Robotics* 37, 6 (Dec. 2021), 1874–1890. doi:10.1109/tro.2021.3075644

[6] Luqi Cheng, Zhangshuo Qi, Zijie Zhou, Chao Lu, and Guangming Xiong. 2025. LT-Gaussian: Long-Term Map Update Using 3D Gaussian Splatting for Autonomous Driving. arXiv:2508.01704 [cs.CV] https://arxiv.org/abs/2508.01704

[7] Sankeerth Durvasula, Adrian Zhao, Fan Chen, Ruofan Liang, Pawan Kumar Sanjaya, and Nandita Vijaykumar. 2023. DISTWAR: Fast Differentiable Rendering on Raster-based Rendering Pipelines. arXiv:2401.05345 [cs.CV] https://arxiv.org/abs/2401.05345

[8] Daniela Esparza and Gerardo Flores. 2021. The STDyn-SLAM: A stereo vision and semantic segmentation approach for SLAM in dynamic outdoor environments. arXiv:2010.09857 [cs.RO] https://arxiv.org/abs/2010.09857

[9] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. 2024. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. arXiv:2311.17245 [cs.CV] https://arxiv.org/abs/2311.17245

[10] Maksim Filipenko and Ilya Afanasyev. 2018. Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment. In *2018 International Conference on Intelligent Systems (IS)*. IEEE, 400–407. doi:10.1109/is.2018.8710464

[11] Dasong Gao, Peter Zhi Xuan Li, Vivienne Sze, and Sertac Karaman. 2025. GEVO: Memory-Efficient Monocular Visual Odometry Using Gaussians. *IEEE Robotics and Automation Letters* 10, 3 (March 2025), 2774–2781. doi:10.1109/lra.2025.3534683

[12] Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. 2025. PUP 3D-GS: Principled Uncertainty Pruning for 3D Gaussian Splatting. arXiv:2406.10219 [cs.CV] https://arxiv.org/abs/2406.10219

[13] Houshu He, Gang Li, Fangxin Liu, Li Jiang, Xiaoyao Liang, and Zhuoran Song. 2025. GSArch: Breaking Memory Barriers in 3D Gaussian Splatting Training via Architectural Support. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 366–379. doi:10.1109/HPCA61900.2025.00037

[14] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. 2024. Photo-SLAM: Real-time Simultaneous Localization and Photorealistic Mapping for Monocular, Stereo, and RGB-D Cameras. arXiv:2311.16728 [cs.CV] https://arxiv.org/abs/2311.16728

[15] Amey Kasar. 2018. Benchmarking and Comparing Popular Visual SLAM Algorithms. arXiv:1811.09895 [cs.RO] https://arxiv.org/abs/1811.09895

[16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[17] Pou-Chun Kung, Xianling Zhang, Katherine A. Skinner, and Nikita Jaipuria. 2024. LiHi-GS: LiDAR-Supervised Gaussian Splatting for Highway Driving Scene Reconstruction. arXiv:2412.15447 [cs.CV] https://arxiv.org/abs/2412.15447

[18] Xiaohan Li, Ziren Gong, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, Dong Liu, and Jun Wu. 2025. Stereo 3D Gaussian Splatting SLAM for Outdoor Urban Scenes. arXiv:2507.23677 [cs.RO] https://arxiv.org/abs/2507.23677

[19] Yiyi Liao, Jun Xie, and Andreas Geiger. 2022. KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D. arXiv:2109.13410 [cs.CV] https://arxiv.org/abs/2109.13410

[20] Ziwei Liao, Wei Wang, Xianyu Qi, Xiaoyu Zhang, Lin Xue, Jianzhen Jiao, and Ran Wei. 2020. Object-oriented SLAM using Quadrics and Symmetry Properties for Indoor Environments. arXiv:2004.05303 [cs.RO] https://arxiv.org/abs/2004.05303

[21] Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. 2025. MaskGaussian: Adaptive 3D Gaussian Representation from Probabilistic Masks. arXiv:2412.20522 [cs.CV] https://arxiv.org/abs/2412.20522

[22] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. 2024. Gaussian Splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[23] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. 2015. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* 31, 5 (Oct. 2015), 1147–1163. doi:10.1109/tro.2015.2463671

[24] Raul Mur-Artal and Juan D. Tardos. 2017. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics* 33, 5 (Oct. 2017), 1255–1262. doi:10.1109/tro.2017.2705103

[25] Jiadong Tang, Yu Gao, Dianyi Yang, Liqi Yan, Yufeng Yue, and Yi Yang. 2025. DroneSplat: 3D Gaussian Splatting for Robust 3D Reconstruction from In-the-Wild Drone Imagery. arXiv:2503.16964 [cs.CV] https://arxiv.org/abs/2503.16964

[26] Lizhou Wu, Haozhe Zhu, Siqi He, Jiapei Zheng, Chixiao Chen, and Xiaoyang Zeng. 2024. GauSPU: 3D Gaussian Splatting Processor for Real-Time SLAM Systems. In *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1562–1573. doi:10.1109/MICRO61859.2024.00114

[27] Renxiang Xiao, Wei Liu, Yushuai Chen, and Liang Hu. 2024. LiV-GS: LiDAR-Vision Integration for 3D Gaussian Splatting SLAM in Outdoor Environments. arXiv:2411.12185 [cs.RO] https://arxiv.org/abs/2411.12185

[28] Zhe Xin, Chenyang Wu, Penghui Huang, Yanyong Zhang, Yinian Mao, and Guoquan Huang. 2025. Large-Scale Gaussian Splatting SLAM. arXiv:2505.09915 [cs.CV] https://arxiv.org/abs/2505.09915

[29] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. 2024. GS-SLAM: Dense Visual SLAM with 3D Gaussian Splatting. arXiv:2311.11700 [cs.CV] https://arxiv.org/abs/2311.11700

[30] Zhaoliang Zhang, Tianchen Song, Yongjae Lee, Li Yang, Cheng Peng, Rama Chellappa, and Deliang Fan. 2024. LP-3DGS: Learning to Prune 3D Gaussian Splatting. arXiv:2405.18784 [cs.CV] https://arxiv.org/abs/2405.18784

[31] Jianhao Zheng, Zihan Zhu, Valentin Bieri, Marc Pollefeys, Songyou Peng, and Iro Armeni. 2025. WildGS-SLAM: Monocular Gaussian Splatting SLAM in Dynamic Environments. arXiv:2504.03886 [cs.CV] https://arxiv.org/abs/2504.03886

[32] Liyuan Zhu, Yue Li, Erik Sandström, Shengyu Huang, Konrad Schindler, and Iro Armeni. 2024. LoopSplat: Loop Closure by Registering 3D Gaussian Splats. arXiv:2408.10154 [cs.CV] https://arxiv.org/abs/2408.10154

A Additional Experimental Results

A.1 Evaluation on KITTI

We further evaluate Pocket-SLAM on the KITTI odometry benchmark [19]. Tab. 3 reports tracking and rendering results, and Tab. 4 reports peak memory and FPS. Our rendering-area–aware pruning consistently achieves tracking accuracy comparable to LSG-SLAM across all sequences. In challenging sequences, Pocket-SLAM even surpasses LSG-SLAM due to effective elimination of redundant Gaussians. Pruning methods based solely on gradient or opacity often suffer from tracking failure in large-scale outdoor scenes.

A.2 Result Profiling

Detailed evaluation results in Fig. 5 highlight the limitations of existing pruning strategies in 3DGS-SLAM. LightGaussian [9] (opacity-based) and LP-3DGS [30] (gradient-based) are problematic in outdoor scenarios, as Gaussians covering critical regions such as sky and road often exhibit low opacity or gradients and are thus removed. This eliminates wide-coverage Gaussians that provide essential global constraints, leading to unstable pose estimation and drift. In contrast, Pocket-SLAM evaluates Gaussian importance by rendering area, which better captures their contribution to the image and is more suitable for outdoor SLAM. MaskGaussian [21] also considers rendering area but applies it globally, overlooking texture-dense regions. Consequently, fine-grained Gaussians encoding trees and façades are removed, weakening local consistency. By integrating rendering-area pruning with a tile-level budget mechanism, Pocket-SLAM preserves small Gaussians in texture-rich regions while removing redundant ones, maintaining both global structure and local detail with significant memory reduction.

Table 3: Camera tracking and rendering results on KITTI [19]. 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 [28]	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 [30]	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 [21]	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 4: Performance on KITTI [19], 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 [28]	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 [21]	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

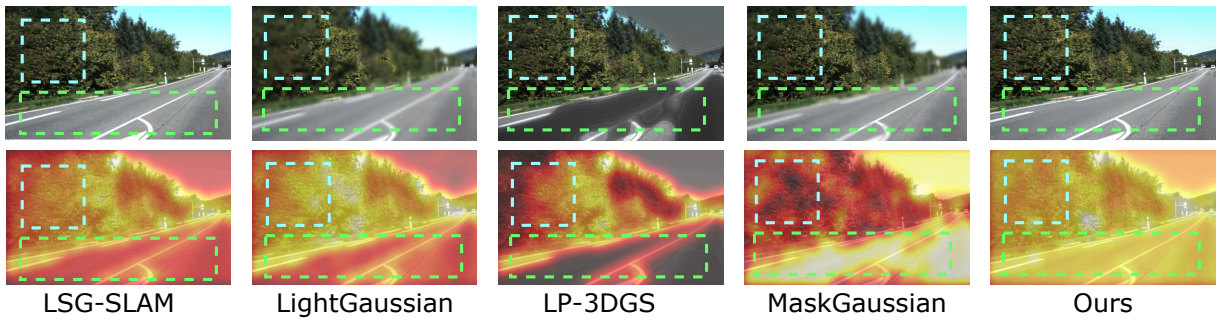

Figure 5: Results on KITTI [19] sequence 10. Comparison across five methods with rendered images and Gaussian density heatmaps.