Enhancing Gaussian Splatting SLAM with Feature-based Tracking

Muhammad Awais, Karlo Koledić, Luka Petrović, Ivan Marković*

***Abstract*—Accurate localization and scene reconstruction are essential for the autonomous navigation of mobile agents. Simultaneous Localization and Mapping (SLAM) algorithms address both challenges by formulating a unified optimization problem, offering an integrated solution to both objectives. Recent advances in learning-based scene understanding have significantly improved accuracy and robustness, particularly in adverse scenarios that are troublesome for traditional geometric methods. However, generating an accurate dense scene reconstruction remains an open challenge, largely due to the complexity of the optimization problem, making it unsuitable for real-time requirements on resource-constrained devices. Novel advances in 3D reconstruction such as implicit representations and Gaussian Splatting present an enticing formulation enabling offline reconstruction of large-scale scenes. While these approaches have been successfully adapted for online incremental reconstruction, particularly through Gaussian Splatting SLAM methods, they are hindered by significant computational complexity and convergence challenges due to the non-convex nature of photometric optimization. In this work we rethink this approach by combining the strengths of traditional feature-based methods with innovative reconstruction capability of Gaussian splatting. Specifically, we integrate feature-based pose estimation, relocalization and loop closure with 3D Gaussian-based scene reconstruction. This results in state-of-the-art tracking and mapping performance on the EuRoC and TUM datasets, while significantly reducing convergence iterations and improving real-time performance.**

I. Introduction

Simultaneous localization and mapping (SLAM) has been a vibrant field of study aimed at achieving precise pose tracking and high-fidelity map reconstruction. With the widespread availability of data and affordability of camera sensors, these components have become integral to SLAM systems. Traditionally, sparse SLAM methods have excelled in accurately tracking camera trajectories, but they often fall short in generating high-fidelity maps needed for augmented reality (AR) or learning-based systems. In contrast, dense SLAM methods aim to produce photorealistic maps while maintaining precise camera tracking, continuously driving advancements in real-time performance and scene representation techniques.

Visual SLAM methods leverage various sensor configurations, including monocular, RGB-D, and stereo camera setups, each offering distinct challenges and advantages. The monocular setup, lacking scale information, presents particular challenges in accurate localization and mapping compared to RGB-D and stereo configurations that provide depth perception.

* Authors are with the University of Zagreb Faculty of Electrical Engineering and Computing, Laboratory for Autonomous Systems and Mobile Robotics, Croatia. {name.surname}@fer.unizg.hr

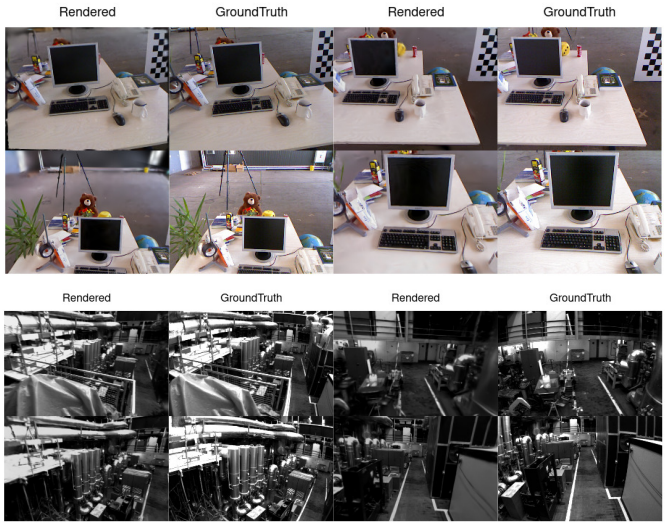


Fig. 1: Scene reconstruction achieved by our approach on EuRoC and TUM datasets. Our system combines traditional feature-based camera pose estimation with novel dense scene representation based on Gaussian splatting. This results in significant accuracy increases in pose estimation and scene reconstruction, while substantially reducing the computational complexity.

Visual SLAM can be classified by its input processing—either direct or indirect—and by its output—sparse or dense. Direct methods, like Direct Sparse Odometry (DSO) [15], optimize photometric errors across frames for real-time tracking, while indirect methods such as ORB-SLAM [3] rely on feature extraction to track and map environments. Sparse visual SLAM prioritizes computational efficiency by mapping significant points, whereas dense visual SLAM, including approaches like Kinect Fusion [5] and DROID-SLAM [16], focuses on detailed reconstructions, optimizing local frame data or employing global consistency for robustness and accuracy.

Scene representations are an integral part of SLAM systems, playing a pivotal role in the success of these pipelines by serving as the cornerstone for accurate localization, mapping, and navigation. They facilitate the fusion of sensor data, enabling the system to build and update a coherent model of the environment in real-time. The choice of representation is particularly important for dense reconstruction. Often, there is a delicate balance between reconstruction quality and com-

putational complexity.

Traditional explicit representations, such as feature-based, grid-based, and point-based maps directly describe the environment using keypoints, landmarks, or edges extracted from sensor data. Grid-based representations divide the environment into cells, indicating obstacle occupancy and are often used in robotics for spatial information representation. Despite their simplicity, the fixed resolution of grids can lead to inefficient memory use and loss of detail. Point-based maps, including surfels that approximate surfaces with attributes like position and normal direction, represent the environment through collections of 3D points derived from sensors like lidar or depth cameras. These maps are well-suited for detailed environmental reconstruction and localization. Unfortunately, these representations are often unsuitable for dense 3D reconstruction, particularly for real-time resource-constrained devices.

Implicit representations, on the other hand, describe scenes through mathematical functions without directly defining features. Techniques such as KinectFusion [5], BundleFusion [6], and InfiniTAM [7] use Truncated Signed Distance Function (TSDF) for real-time dense surface reconstruction on GPUs. Recent innovations in neural rendering, such as NeRF [1] demonstrate significant advancements in novel view synthesis and scene reconstruction. Instead of using the hand-crafted functions, NeRF models complex, high-fidelity scenes by training a multi-layer perceptron (MLP) on a dataset of ground truth images. It learns a continuous volumetric function that encapsulates the scene's appearance and geometry, which can be queried at any point in space to generate novel views with remarkable realism. Furthermore, similar neural models have been proposed to learn various representations like SDFs [12] or sparse voxel fields [13]. Consequently, there has been a sudden emergence of implicit SLAM algorithms such as iMAP [8] or ESLAM [14]. Furthermore, approaches like Nice-SLAM [9] and Co-SLAM [10] incorporate multi-resolution grids and advanced tensor representations to enhance mapping speed and reduce computational demands, providing a promising direction for future SLAM systems.

Unfortunately, neural implicit representations require expensive per-pixel model execution and raycasting, hindering real-time performance. Conversely, Gaussian Splatting [2] provides a computationally efficient alternative, particularly suitable for real-time applications such as SLAM. This method represents the scene with a collection of 3D Gaussian points—each defined by properties such as anisotropic covariance and opacity, optimized using ground truth images through a gradient descent-based optimizer. Unlike NeRF, Gaussian Splatting does not rely on neural networks for inference, facilitating faster training and rendering. Recent Gaussian Splatting SLAM methods such as MonoGS [17] and SplaTAM [18] formulate a joint localization and scene reconstruction objective relying on photometric error. While such an approach results in impressive scene reconstruction quality, real-time performance is severely constrained, especially on resource-constrained devices.

This work aims to merge the strengths of conventional SLAM techniques, known for their prowess in real-time, precise pose estimation, with cutting-edge developments in computer graphics. The ultimate aim is to forge a system that not only operates swiftly and robustly but also yields high-fidelity maps crucial for subsequent tasks. To achieve this objective, we present an integrated approach employing ORBSLAM3 [3] for camera tracking and Gaussian splatting [2] for dense map representation.

The rationale behind the proposed system is the partitioning of the process to ensure critical tracking functions operate in real time, facilitated by feature-based SLAM. Meanwhile, the mapping process runs concurrently, to optimize the Gaussians to achieve photo-realistic scene reconstruction. This enables the pipeline to steer clear of the computational bottleneck associated with direct pose optimization. This leads to a substantial improvement in tracking and reconstruction performance, while significantly lowering the computational complexity.

II. Method

The pipeline is divided into two separate processes running in parallel: camera tracking via a feature-based method, and dense mapping, which utilizes estimated camera poses to provide accurate scene reconstruction. This section elucidates the process of camera tracking, followed by a detailed explanation of 3D reconstruction utilizing 3D Gaussians.

A. Tracking

Dense visual mapping with Gaussian splatting representation requires accurate $SE(3)$ poses to facilitate optimization of 3D Gaussians. In seminal work [2], 3D Gaussians are optimized after the initial structure-from-motion (SfM) [20] optimization. In contrast to full bundle-adjustment via SfM, we propose to estimate the camera poses incrementally via a feature-based odometry or Visual SLAM. In this work, we use a modified ORB-SLAM3 tracker [3] due to the provided flexibility of the chosen multi-sensor setup, enabling monocular, stereo, RGB-D and visual-inertial tracking. Here, we provide a brief overview of certain ORB-SLAM3 pipelines which are integrated into our system.

1) Feature detection: Upon receiving a frame from the camera, the initial step involves detecting keypoints that are not only present in the current frame, but also trackable across subsequent frames. Additionally, to establish correspondence between these keypoints, a choice of adequate descriptor is essential. It is imperative that this keypoint detection and description process is not only real-time but also robust to scale, rotation, and changes in viewpoint. ORB (Oriented FAST and Rotated BRIEF) [21] satisfy these requirements, while being extremely efficient to compute due to the simple 256-bit descriptor.

2) Local Map Initialization: For the monocular case, map initialization is done using triangulation from multiple views. To estimate the pose of two different frames, two geometric models i.e. a homography and a fundamental matrix, are computed in parallel and the best one is chosen using a heuristic.

3) Pose Estimation: Once the map has been initialized, subsequent frames are tracked by minimizing the reprojection error between the map points and their corresponding image coordinates.

$$\{R_{\mathrm{CW}}, P_W^C\} = \arg\min_{R_{\mathrm{CW}}, P_W^C} \sum_j \rho\left(\left\| x_C^j - \pi_m\left(R_{\mathrm{CW}} X_W^j + P_W^C\right)\right\|_{\Sigma_j}^2\right) \quad (1)$$

4) Local Mapping: Local sparse mapping is performed over a window of keyframes. This optimizes the keyframes which are covisible with the current keyframe and the map points seen by those keyframes. This sparse map, containing previously matched ORB features, is used only for purpose of camera pose estimation.

5) Loop Closing: The current keyframe is matched against a database of keyframes using a bag-of-words approach based on DBoW2 [22]. In the monocular case, a similarity transformation $S \in Sim(3)$ with seven degrees of freedom—six for pose and one for scale—is computed between the current keyframe and the loop-closing keyframe:

$$S = \begin{bmatrix} sR & t \\ 0_{1\times 3} & 1 \end{bmatrix} \quad (2)$$

After computing the similarity transform, pose graph optimization is performed over similarity transformation to correct the scale drift.

B. Dense mapping

For dense mapping, the scene is modeled using 3D Gaussians, where each Gaussian is characterized by its 3D position (the mean of the Gaussian), opacity α, anisotropic covariance, and spherical harmonics (SHs) to capture view-dependent radiance. However, in this work, RGB color is optimized instead of SHs to reduce computational complexity and enhance processing speed.

1) Gaussian Initialization: In the monocular case, once the first frame is successfully tracked, Gaussian means are initialized randomly by generating depths from a range of random values. The covariance is set similarly to 3DGS [2], using an isotropic Gaussian where the axes correspond to the mean distance to the three nearest points. The color is initialized with the RGB value of the pixel coordinate that matches the corresponding location in the depth map, and opacity is activated using a sigmoid function.

For the RGB-D case, a similar procedure is followed, with the key difference being that the depths are initialized using the depth map rather than random values.

2) Gaussian Optimization: Once the map has been initialized, the Gaussian optimization process begins. The Gaussians are first projected on the image plane from a given viewpoint. The projection of these 3D Gaussian in the image space is governed by the following expressions:

$$\mu_I = \pi(T_{\mathrm{CW}} \cdot \mu_W) \qquad \Sigma_I = JW\Sigma_W W^T J^T \quad (3)$$

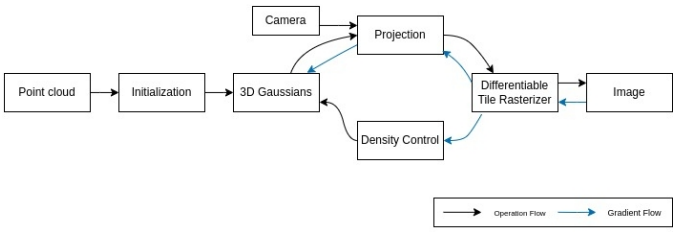


Fig. 2: Gaussian Optimization

where μ_I is the mean of the Gaussian in the image coordinates, π is the projection matrix, T_{CW} is the pose of the viewpoint and μ_W is the mean of the Gaussian in the world coordinates. The 2D covariance in the image coordinates Σ_I is represented as a product of J which is the Jacobian of the linear approximation of the projective transformation, W is the rotational component of T_{CW} and Σ_W is the covariance in the world coordinates.

A pixel color in the image is synthesized as:

$$C_p = \sum_{i\in N} c_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \quad (4)$$

where C_p represents the color at pixel p, c_i is the color associated with the i'th Gaussian, α_i is the associated opacity. The Gaussians are sorted according to their depths before projecting them on to the image plane and then rasterized using alpha compositing.

Similarly, a depth map can also be rasterized as:

$$D_p = \sum_{i\in N} z_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \quad (5)$$

where z_i is the depth of i'th Gaussian. Once a rendered RGB image and a depth map has been obtained, the loss is computed as:

$$E_{pho} = \|I_r - I_{gt}\|_1 . \quad (6)$$

When depth measurements are available, an additional depth loss is computed as:

$$E_{depth} = \|D_r - D_{gt}\|_1 \quad (7)$$

Furthermore, the SSIM [19] loss is computed as:

$$E_{SSIM} = SSIM(I_r, I_{gt}) \quad (8)$$

In an incremental mapping process, the Gaussians can stretch in the viewing direction, so to prevent them from stretching in different directions they are encouraged to remain isotropic by incorporating an isotropic loss term:

$$E_{iso} = \Sigma_i \|s_i - \tilde{s}_i\| , \quad (9)$$

where s_i is the covariace and $\tilde{s}_i$ is the mean of the covariance. The final loss for the monocular camera is:

$$L = \lambda_{pho}((1-\lambda_{ssim})E_{pho} + \lambda_{ssim}E_{SSIM}) + (1-\lambda_{pho})E_{iso}. \quad (10)$$

TABLE I: Tracking and Mapping performance on EuRoc Monocular dataset

Sequence	Method	RMSE[m]↓	SSIM↑	PSNR↑	LPIPS↓	FPS↑	Iterations↓
MH01	MonoGS	2.44	0.56	16.86	0.45	1.60	19581
	Ours	**0.037**	**0.68**	**18.78**	**0.40**	**6.51**	**7170**
MH02	MonoGS	0.76	0.65	19.43	0.34	1.55	16247
	Ours	**0.074**	**0.67**	**19.51**	**0.36**	**6.33**	**5949**
MH03	MonoGS	3.87	0.47	13.50	0.60	1.39	15247
	Ours	**0.039**	**0.68**	**18.67**	**0.38**	**6.29**	**5179**

With available depth measurements, the loss is computed as:

$$L = \lambda_{pho}((1-\lambda_{depth})E_{pho} + \lambda_{depth}E_{depth}) + (1-\lambda_{pho})E_{iso} \tag{11}$$

3) Keyframing: The Gaussians are optimized using a window of keyframes. These keyframes are obtained from the ORBSLAM3 keyframe database with the difference that once a keyframe is addded to the database, it is not removed. The window for optimization contains the last n keyframes and k keyframes chosen randomly to ensure that the previous map does not deteriorate.

4) Map Expansion: The map is expanded at every keyframe and Gaussians are added to the already existing map. For the monocular case, the rendered depth is used to initialize the new Gaussians while in the RGB-D case depth from the sensor is used to initialize new Gaussians.

5) Geometry aware densification: In the RGB-D case, Gaussians are only added in areas where the geometry is not represented accurately. This is done by generating a mask for the areas where rendered depth has a large error when compared to the ground-truth depth. The mask is generated as:

$$M_{dens} = \|D_{ren} - D_{gt}\|_1 > \lambda_{depth} \tag{12}$$

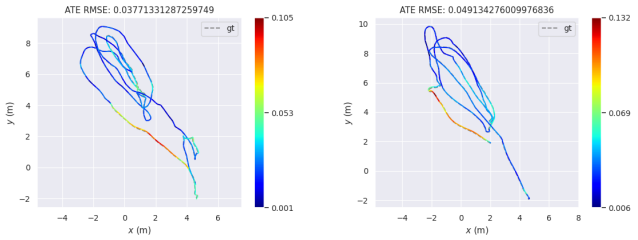
This prevents Gaussians from being added in areas which are already well represented and ensures that Gaussian addition is focused to under-represented areas.

6) Density Control: During Gaussian optimization, some areas may be under-reconstructed while others may be over-reconstructed. In addition to this, opacity of some Gaussians become so small that they don't contribute to the rendered image. To overcome these issues a similar procedure as in [2] is adopted, where Gaussians below a certain opacity are removed, Gaussians in over-reconstructed areas are split, while Gaussians in under-reconstructed areas are cloned.

III. Experiments

To test our approach, we perform extensive experiments, ablating localization error, scene reconstruction error and convergence iterations. We use both EuRoC [23] and TUM [24] datasets containing monocular and RGB-D sequences, respectively. For Gaussian optimization, we follow [17], accelerating optimization with custom CUDA kernels executed on a NVIDIA RTX A5000 GPU.

For evaluation of pose estimation, we use Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE).



(a) MH01 (b) MH02

Fig. 3: Estimated trajectories on EuRoC dataset

For scene reconstruction evaluation, we report Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) and Learned Perceptual Image Patch Similarity (LPIPS) on every fifth frame excluding the keyframes. Since the process runs in multiple threads, the results are non deterministic in nature, which is why we average the metrics across three different runs. To evaluate real-time performance, we report tracking FPS and mapping iterations until convergence.

A. EuRoC Monocular Dataset

First, we evaluate our approach on the EuRoC dataset. For this experiment, we compare with MonoGS [17], a current state-of-the-art Gaussian splatting SLAM. Right image and IMU data are discarded, resulting in monocular sequences requiring ground-truth alignment during evaluation.

Tracking and mapping metrics are reported in Table I. Results demonstrate clear superiority of our approach on all metrics, while achieving convergence on significantly less iterations, resulting in substantial increase in FPS. This improvement is amplified on larger sequences, where our method benefits from additional bundle adjustment and loop closing.

Additonally, we visualize estimated trajectories in Fig. 3, demonstrating drift-free pose estimation. Overall, the results show the efficacy and robustness of feature-based tracking, compared to the high-dimensional photometric optimization in MonoGS that leads to slower convergence.

B. TUM RGB-D Dataset

Furthermore, we evaluate our method on TUM RGB-D sequences to ablate the accuracy of our method with additional depth information, which should reduce the trajectory drift and reconstruction uncertainty. As shown in Table II, contrary to the monocular sequences, the mapping performance is comparable to MonoGS in terms of accuracy. Additional

TABLE II: Tracking and Mapping performance on TUM RGB-D dataset

Sequence	Method	RMSE[m]↓	SSIM↑	PSNR↑	LPIPS↓	FPS↑	Iterations↓
fr1_desk	MonoGS	**0.014**	**0.71**	**18.71**	**0.31**	1.22	6649
	Ours	0.021	0.68	18.47	0.38	**6.5**	**2164**
fr2_xyz	MonoGS	0.013	0.71	15.90	0.31	1.87	21928
	Ours	**0.0037**	**0.70**	**16.02**	**0.30**	**12.4**	**5936**
fr3_office	MonoGS	0.017	0.73	19.06	0.33	1.44	19955
	Ours	**0.014**	**0.73**	**21.12**	**0.33**	**8.55**	**6548**

depth information is highly useful for photometric error minimization, resulting in a smaller accuracy gap between our approach and MonoGS. However, the most notable difference is the tracking speed and the number of iterations it takes to achieve a similar reconstruction performance, demonstrating the efficiency of our method for resource-constrained systems.

IV. Conclusion

In this work, we have demonstrated a dense SLAM system, combining feature-based tracking and dense mapping via Gaussian splatting. We show that combining feature based pose estimation and Gaussian splatting results in better tracking performance with superior reconstruction quality at higher frame rates. Furthermore, our approach is scalable to larger sequences, where previous methods experience convergence issues due to large optimization problems. Potential for future work includes reducing the memory footprint, employing masking techniques to make it work in dynamic scenes, and last but not least, using a tighter coupling between feature-based SLAM and Gaussian splatting.

References

[1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: representing scenes as neural radiance fields for view synthesis. Commun. ACM 65, 1 (January 2022), 99–106. https://doi.org/10.1145/3503250

[2] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G., "3D Gaussian Splatting for Real-Time Radiance Field Rendering", Art. no. arXiv:2308.04079, 2023. doi:10.48550/arXiv.2308.04079.

[3] Campos, C., Elvira, R., Gómez Rodríguez, J. J., Montiel, J. M. M., and Tardós, J. D., "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM", Art. no. arXiv:2007.11898, 2020. doi:10.48550/arXiv.2007.11898.

[4] Alatan, A. Aydin, et al. "Scene representation technologies for 3DTV—A survey." IEEE transactions on circuits and systems for video technology 17.11 (2007): 1587-1605.

[5] Pagliari, Diana, et al. "Kinect fusion improvement using depth camera calibration." The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 40 (2014): 479-485.

[6] Dai, Angela, et al. "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration." ACM Transactions on Graphics (ToG) 36.4 (2017): 1.

[7] Prisacariu, Victor Adrian, et al. "Infinitam v3: A framework for large-scale 3d reconstruction with loop closure." arXiv preprint arXiv:1708.00783 (2017).

[8] Sucar, Edgar, et al. "imap: Implicit mapping and positioning in real-time." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

[9] Zhu, Zihan, et al. "Nice-slam: Neural implicit scalable encoding for slam." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.

[10] Zou, Danping, and Ping Tan. "Coslam: Collaborative visual slam in dynamic environments." IEEE transactions on pattern analysis and machine intelligence 35.2 (2012): 354-366.

[11] Johari, Mohammad Mahdi, Camilla Carta, and François Fleuret. "Eslam: Efficient dense slam system based on hybrid representation of signed distance fields." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

[12] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 27171–27183.

[13] Thomas Müller, Lutz Kobbelt, and Andreas Nüchter. Neural Sparse Voxel Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 10234–10244.

[14] Nicolai Wolpert, Luca Penate-Sanchez, Nico Engel, and C. S. Andreas Hahne. ESLAM: Efficient Dense SLAM System Based on Hybrid Representation of Signed Distance Fields and Multi-Level Surfel Maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[15] Engel, Jakob, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry." IEEE transactions on pattern analysis and machine intelligence 40.3 (2017): 611-625.

[16] Teed, Zachary, and Jia Deng. "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras." Advances in neural information processing systems 34 (2021): 16558-16569.

[17] Matsuki, Hidenobu, et al. "Gaussian splatting slam." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.

[18] Johannes L. Schönberger, Chenglei Wu, Siyu Tang, and Marc Pollefeys. SplaTAM: Splat Track & Map 3D Gaussians for Dense RGB-D SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[19] 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, Apr. 2004. doi: 10.1109/TIP.2003.819861.

[20] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571.

[22] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[23] 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*, vol. 35, no. 10, pp. 1157–1163, 2016. [Online]. Available

[24] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580. [Online]. Available