IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Dense Monocular SLAM in Real-time with Structured Gaussian Representation

Shaofan Liu[1], Xing Wei[1], Chong Zhao[1], Aoxiang Tian[1] and Bin Du[1]

***Abstract*— Monocular dense SLAM faces significant challenges in low-texture environments and under rapid camera motions. The recent development of 3D Gaussian Splatting (3DGS) offers a promising approach for real-time dense 3D reconstruction. However, existing 3DGS-based SLAM systems employ end-to-end optimization frameworks, which often struggle to achieve both efficient camera tracking and high-quality scene reconstruction simultaneously. To address this challenge, we propose a dense decoupled SLAM system that seamlessly integrates traditional visual odometry with 3DGS within a unified framework. Our system utilizes dense direct image alignment using pseudo-depth maps rendered from a global model, which is represented by an octree-managed structured Gaussian representation. This structured Gaussian supports fast rendering and efficient mesh extraction. Furthermore, we adopt a stereo 3D reconstruction model to generate dense depth maps from visual odometry for optimizing the 3D Gaussians. Experimental results demonstrate that our framework achieves state-of-the-art performance in both tracking robustness and reconstruction outperforming to existing monocular Gaussian-based SLAM systems, while maintaining real-time efficiency.**

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) [1] plays an important role in autonomous system, which aims to build an environment map and localize the agent within it. Traditional SLAM systems have been actively studied over the past fifteen years and have achieved promising results [2], [3], [4]. However, these systems are generally limited to constructing sparse point maps, which lack dense geometry and texture information.

Motivated by Neural Radiance Fields (NeRF) [5], several efficient SLAM approaches [6], [7], [8], [9] have been proposed that leverage the expressive capabilities of implicit neural representations to capture high-fidelity geometric and visual details from sparse observations. As pioneering works in this field, iMAP [6] and NICE-SLAM [7] utilize Coordinate-MLPs to jointly recover scene geometry and camera poses, achieving high-quality scene reconstruction. Nevertheless, these methods typically render only a limited subset of pixels to reduce optimization time, resulting in the reconstructed dense maps lacking the richness and intricacy of details.

Recently, 3D Gaussian Splatting (3DGS) [10] has shown remarkable superiority in the efficiency of high-resolution image rendering, making it especially suitable for applications such as dense SLAM [11], [12], [13], [14]. These methods not only reduce the computational overhead commonly associated with implicit neural representation methods but also maintain consistent rendering quality across varying scene complexities and resolutions. However, recent methods such as GS-SLAM [11] and MonoGS [12] rely exclusively on photometric loss backpropagation for camera pose optimization, which often results in tracking failures under rapid motion and poor initial localization. Unfortunately, 3DGS itself requires a well-initialized point cloud to begin reconstruction. This inherent contradiction has led to the development of decoupled SLAM systems [15], [16], of which MGSO [16] is most similar to our proposed method. However, the visual odometry (VO) of MGSO is entirely

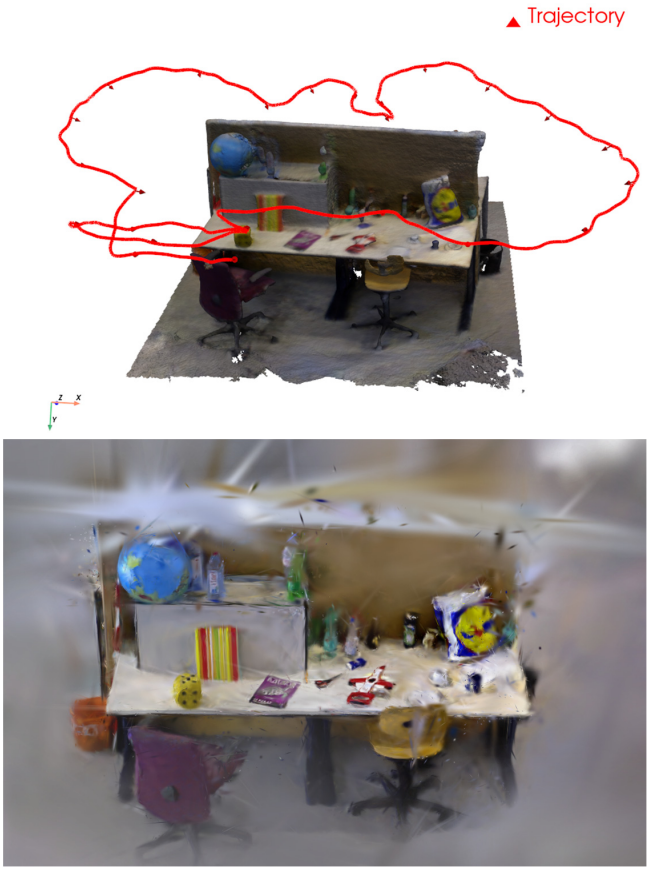

Fig. 1. Our monocular SLAM method is based on a structured Gaussian representation, enabling direct triangular mesh extraction from Gaussians. The top image shows the reconstructed mesh along with camera pose trajectory, while the bottom image shows the 3D Gaussian ellipsoids.

*This work was supported in part by Province Science and Technology Innovation Tackle Plan Project (202423k09020003)

[1]Shaofan Liu, Xing Wei, Chong Zhao, Aoxiang Tian and Bin Du are with the School of Computer and Information, Hefei University of Technology, Hefei, Anhui, China. shaofanliu,weixing,zhaochong@hfut.edu.cn; aoxiangtian,bindu@mail.hfut.edu.cn

Xing Wei is the corresponding author.

independent of the 3D Gaussian representation and thus cannot benefit from the dense mapping capabilities provided by 3DGS.

To address these challenges, we propose a monocular dense decoupled SLAM method, which integrates a structured Gaussian representation into a traditional optimization-based VO pipeline. Our key idea is an octree-based organization of structured 3D Gaussians, which not only enables surface mesh extraction through sparse voxel traversal but also enhances the VO through differentiable rendering. Specifically, we leverage the differentiable rendering capability of 3DGS to generate pseudo-depth maps from the input viewpoint at runtime. These pseudo-depth maps are integrated into the VO pipeline as additional geometric constraints, significantly enhancing tracking robustness. To further improve the accuracy of the Gaussian representation, we adopt DUSt3R [17] to predict confidence-weighted depth maps, which guide the iterative optimization process of the Gaussians. Additionally, we leverage the sparse point cloud generated by VO to provide initial spatial distributions for the Gaussian primitives. As shown in Fig.1, our method requires only a single monocular camera to achieve both robust camera tracking and dense 3D reconstruction.

The key contributions of this paper are summarized as follows:

- We propose a dense decoupled SLAM system that seamlessly integrates traditional visual odometry with 3DGS within a unified framework.
- We introduce the structured 3DGS representation, which employs an octree-based structure to efficient organization and surface extraction.
- We propose the depth-guided tracking method based on differentiable rendering, which integrates rendered pseudo-depth maps into visual odometry, improving tracking accuracy and robustness.

II. RELATED WORK

A. Visual SLAM

Visual SLAM systems can be categorized into traditional SLAM systems [1], [2], [3], [4] and learning-based SLAM systems [18], [19], [20]. Traditional systems typically follow feature-based [1], [2] or direct [3], [4] approaches. Feature-based methods leverage sparse keypoint matching for efficient pose estimation but struggle with dense mapping due to their inherent sparse representation. Direct methods, such as LSD-SLAM [4], optimize photometric error across entire images, enabling semi-dense reconstruction at the cost of increased computational complexity and sensitivity to lighting variations. Recent learning-based approaches attempt to overcome these limitations through data-driven priors. Deep neural networks have been employed for depth prediction [18], optical flow estimation [19], and end-to-end pose regression [20]. These SLAM systems exhibit robustness to noise, allowing operation in diverse environments, scalability to large-scale settings, and the handling of complex maps. However, they are limited to constructing sparse point-based maps while achieving accurate camera trajectory estimation.

B. Implicit Neural SLAM

Recent advances in implicit neural representations [5] have revolutionized 3D scene modeling by encoding both geometry and appearance into compact neural networks. These continuous parameterizations have been increasingly adopted in robotic systems for simultaneous mapping and localization. Pioneering work by iMAP [6] demonstrated real-time performance with a single MLP-based representation. However, it suffered from catastrophic forgetting in large-scale environments. Subsequent efforts have addressed scalability through various architectural innovations. For instance, NICE-SLAM [7] introduces hierarchical multi-scale feature grids decoded by pre-trained MLPs, while Vox-Fusion [8] implements dynamic octree expansion for adaptive resolution control. Similarly, Point-SLAM [21] employ tri-plane factorization and neural point clouds, respectively, to enhance surface detail preservation. However, as they are based on time-consuming volume rendering, they require expensive sampling strategies to locate the surface, and their implicit scene representations can be difficult to integrate with other systems.

C. Gaussian-based SLAM

3D Gaussian Splatting (3DGS) [10] is a scene representation technique that models the environment as a large set of 3D Gaussians, resembling blurry overlapping clouds. Similar to NeRF, 3DGS allows for gradient-based optimization of camera poses by minimizing the discrepancy between rendered and input images. To extend 3DGS into SLAM systems, methods such as SplaTAM [14] and GS-SLAM [11] adopt a one-stage framework in which tracking and mapping are tightly coupled. However, these methods heavily rely on depth data for 3D reconstruction, making them dependent on RGB-D sensors. To overcome this limitation, recent methods like MonoGS [12] attempt to achieve tracking and reconstruction using only monocular RGB cameras. Meanwhile, RTG-SLAM [13] aim to improve efficiency by optimizing depth models or modifying the Gaussian representation. Among existing methods, MGSO [16] is the most similar to our method. However, the visual odometry of MGSO is entirely independent of the 3D Gaussian representation, it cannot benefit from the dense mapping capabilities provided by 3DGS. In contrast, our method leverages the differentiable rendering of 3DGS to enhance visual odometry and to extract surface meshes through a structured 3DGS representation.

III. METHOD

A. System Overview

Fig.2 shows the overview of our proposed monocular dense SLAM method. We aim to estimate the camera poses $\{\mathbf{P}_i\}_{i=1}^{N}$ for every frame and simultaneously reconstruct a dense scene map by giving an input sequential RGB stream $\{I_i\}_{i=1}^{M}$ with known camera intrinsics $\mathbf{K} \in \mathbb{R}^{3\times3}$. Our system consists of an odometry module based on Direct Sparse Odometry (DSO) [3] in the tracking branch and a dense mapping module in the mapping branch. In the tracking branch, each input image I_i is first processed by

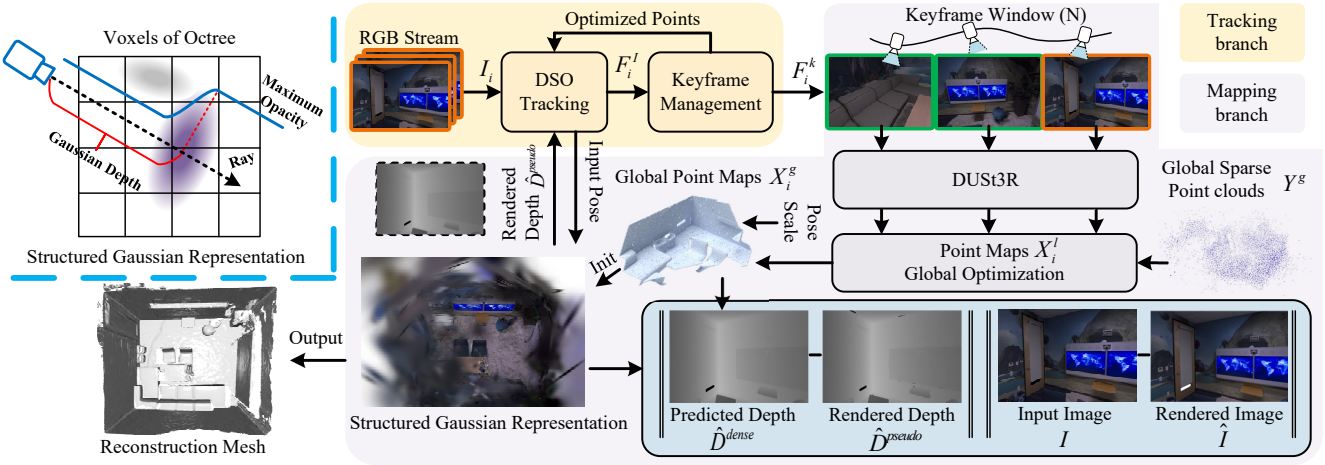

Fig. 2. Overview of our monocular SLAM method. In the tracking branch, we employ DSO [3] to achieve frame-rate camera pose estimation. In the mapping branch, we first estimate predicted depth maps for keyframes by jointly optimizing keyframe images and sparse point clouds. Then, we optimize the structured Gaussian representation using both keyframe images and depth maps. Finally, we render pseudo-depth maps from the structured Gaussian and integrate them into the visual odometry pipeline.

DSO to estimate the camera pose $\mathbf{P}_i$, and then keyframes F_i^k are selected based on its strategy (Sec.III-B). In the mapping branch, we first select the top N keyframes with the highest similarity to the input image and feed them into DUSt3R [17] to generate the global point map X_i^g (Sec.III-C). The global point map is then projected to obtain dense depth maps $\hat{D}^{dense}$, which are further optimized by minimizing the loss between the rendered images and the input images (Sec.III-D). Finally, we extract a surface mesh from the structured Gaussian representation (Sec.III-E).

B. Monocular Odometry

The tracking backbone of our system is built upon a lineage of visual SLAM approaches that originate from DSO [3]. In DSO, the camera motion is modeled as a series of poses, and the goal is to jointly optimize the camera poses and the inverse depths by minimizing the photometric error between the observed and projected pixel intensities. The photometric error for a pixel $\mathbf{p}$ observed in a reference frame i and a target frame j is defined as follows:

$$E_{\mathbf{p}} = \left| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{p}] - b_i) \right|_{\delta} \tag{1}$$

where t_i and t_j denote the exposure times of the images I_i and I_j, respectively. $|\cdot|_{\delta}$ indicates the Huber norm for robust outlier suppression. The pixel $\mathbf{p}'$ is the projection of the sparse point $(\mathbf{p}, d_{\mathbf{p}})$ corresponding to $\mathbf{p}$ in the reference frame i onto the target frame j:

$$\mathbf{p}' = \Pi_c \left(\mathbf{R} \Pi_c^{-1}(\mathbf{p}, d_{\mathbf{p}}) + \mathbf{t} \right) \tag{2}$$

where Π_c and Π_c^{-1} represent the camera projection and back-projection functions, respectively, while $\mathbf{R}$ and $\mathbf{t}$ denote the relative rotation and translation between frames i and j. For the point $(\mathbf{p}, d_{\mathbf{p}})$, $d_{\mathbf{p}}$ is the distance from the camera to the point in the reference frame i.

In the original DSO, a new frame is tracked against the last keyframe n by direct image alignment, with a sparse depth map D^{sparse} projected from the sparse point cloud Y^g. As a result, the accuracy of the sparse point cloud directly influences the localization accuracy of DSO. However, the sparse point cloud in DSO is highly vulnerable to low-texture and high-noise environments. To overcome this limitation, we employ 3DGS to efficiently render a dense pseudo-depth map $\hat{D}^{pseudo}$ from the viewpoint of the reference frame i. For each pixel $\mathbf{p}$ in the current keyframe n, we assign a depth value either based on the sparse VO points D^{sparse}, if available, or based on the rendered dense pseudo-depth map $\hat{D}^{pseudo}$, otherwise. Since the depth map rendered by 3DGS is dense, DSO can utilize this dense combined depth map for two-frame direct image alignment.

C. Dense Local Mapping

DSO represents scenes through sparse point clouds, focusing primarily on high-gradient regions. We adopt a method similar to MGSO to initialize Gaussians based on these sparse point clouds. However, MGSO optimizes the Gaussians only through a photorealistic image loss, making it difficult to accurately reconstruct scenes without depth supervision. To overcome these limitations, we employ DUSt3R [17] to generate dense local point maps for obtaining depth maps. DUSt3R is a novel approach for dense and unconstrained stereo point map generation that operates without prior information about camera calibration or viewpoint poses. By inputting two images, its outputs include point maps, confidence maps of the two input images:

$$X^l, C = Head(Decoder(Encode_1(I_1), Encode_2(I_2))) \tag{3}$$

where I_1 and I_2 represent the input image pair to be matched. $X^l \in \mathbb{R}^3$ and C denote the local dense point map and the confidence map, respectively. In practice, we select the top N keyframes with the highest co-visibility relative to the input frame within a sliding window. These keyframes are paired with the input frame to generate N local point maps

$\{X_i^l\}_{i=1}^N$. We then solve for a scale factor s that aligns the dense local point maps $\{X_i^l\}_{i=1}^N$ with the sparse point cloud Y^g estimated by DSO:

$$\mathcal{M}_i = \left\{ (X_i^l(u_j), Y^g(u_j)) \mid j = 1, \ldots, M_i \right\}$$
$$\arg\min_{s,\{X_i^l\}} \sum_{i=1}^{N} \sum_{j=1}^{M_i} C_i(u_j) \left\| s \cdot X_i^l(u_j) - (\mathbf{R}_i Y^g(u_j) + \mathbf{t}_i) \right\|_2^2 \tag{4}$$

where $\mathcal{M}_i$ denotes the set of matched point pairs between $\{X_i^l\}_{i=1}^N$ and Y^g, obtained by projecting both sets onto the same keyframe and matching points with identical pixel coordinates u_j. $\{\mathbf{R}_i, \mathbf{t}_i\}$ represents the relative pose between the two sets, estimated by DSO. We derive the global dense point map X_i^g by applying the scale factor s and the pose transformation $\{\mathbf{R}_i, \mathbf{t}_i\}$ as follows:

$$X_i^g = s^* \mathbf{R}_i X_i^{l*} + \mathbf{t}_i \tag{5}$$

Simultaneously, we extract the predicted depth map $\hat{D}^{dense}$ as the z-coordinate of X_i^g.

D. Structured Gaussian Representation

1) Scene Representation: We represent the scene using a collection of Gaussians $\{G_i\}$. Similar to [10], the geometry of each Gaussian G_i is parameterized by an opacity $\alpha_i \in [0, 1]$, center $\mathbf{p}_i \in \mathbb{R}^{3\times1}$, scale $\mathbf{s} \in \mathbb{R}^{3\times1}$, and rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$. According to GaussianShader [22], the Gaussian sphere gradually becomes flattened and approaches a plane during the optimization process. Therefore, the direction of its shortest axis can approximate the normal direction $\mathbf{n}_i$ of the Gaussian. The semi-major axis radius r_i can be determined from the Gaussian's half-width at half-maximum (HWHM):

$$\mathbf{n}_i = \mathbf{R}_i[l, :], \quad l = \arg\min([s_1, s_2, s_3]) \tag{6}$$
$$r_i = \sqrt{2\lambda_i \ln 2} \tag{7}$$

where $diag(s_1, s_2, s_3) = \mathbf{s}_i$, and λ_i are the eigenvalues of Σ_i, which represent the variances along the principal directions.

The vanilla 3D Gaussian splatting allows Gaussians to freely drift and split but generally neglects the underlying scene structure. In contrast, our method employs an octree structure for Gaussian management, which can efficiently extract a mesh from the Gaussians. The octree divides the entire scene into mutually exclusive axis-aligned voxels where each voxel serves as both the storage unit for Gaussians and the leaf nodes of the scene octree. Specifically, we insert optimized Gaussians into the octree based on their semi-major axis radius r_i and the center position of their ellipsoidal disc. If the radius of a Gaussian exceeds the edge length of the leaf voxel, neighboring voxels are generated to collectively represent the Gaussian. Notably, optimized Gaussians refer to those whose optimization iterations exceed a threshold τ, after which their size and position remain fixed and no longer change. Additionally, to enhance the efficiency of insertion and search operations within the octree, we employ Morton codes to encoding voxel coordinates. Given the 3D coordinates (x, y, z) of a voxel, we can quickly locate its position in the octree by traversing its Morton code.

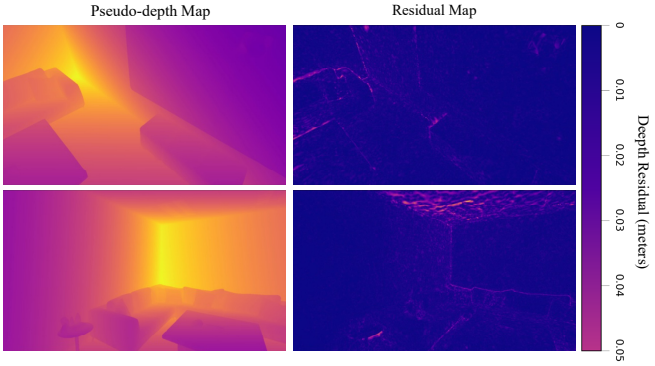

Fig. 3. Visualization of pseudo-depth maps and their residuals compared to ground truth depth maps. The pseudo-depth maps rendered by our method more accurately represent the scene geometry.

2) Fast Rasterization-based Rendering: The core of optimizing the 3D Gaussian scene lies in estimating residuals with the input images. In terms of color rendering, the color image $\hat{I}$ can be rendered by alpha-blending proposed in [10]:

$$\hat{I} = \sum_{i \in N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{8}$$

where $\mathbf{c}_i$ represents the Gaussian color based on the view direction and the SH coefficients. α_i is the density computed by multiplying 2D covariance Σ' with opacity.

In terms of depth rendering, we integrate along the ray for each depth of the Gaussian:

$$\hat{D}^{pseudo} = \sum_{i \in N} \mathbf{p}_i^z \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{9}$$

where $\mathbf{p}_i^z$ is the z-component of the position of a Gaussian. As shown in Fig.3, our rendered depth $\hat{D}^{pseudo}$ is very close to the ground truth depth. Finally, we perform Gaussian optimization based on the color loss and depth loss between the predicted images and the input images for each keyframe. We use the $\mathcal{L}_1$ loss for optimization:

$$\mathcal{L}_{\text{color}} = \sum_{i}^{K} |I_i - \hat{I}_i|, \quad \mathcal{L}_{\text{depth}} = \sum_{i}^{K} C_i |\hat{D}_i^{dense} - \hat{D}_i^{pseudo}| \tag{10}$$

E. Mesh Extraction

After inserting 3D Gaussians into the octree, we compute the opacity of each octree voxel's vertices to extract the mesh. Specifically, the opacity of a vertex is determined by projecting the opacity of the Gaussians into image space. As shown on the left side of Fig.2, we follow the construction of the Gaussian opacity field proposed in [23], which allows the evaluation of the opacity value or transmittance at any

TABLE I

CAMERA TRAJECTORY EVALUATION ON THE REPLICA DATASET, RMSE[CM].

Method	R0	R1	R2	O0	O1	O2	O3	O4	Avg.
DROID-SLAM [25]	0.77	0.42	**2.70**	0.28	0.52	0.33	0.62	0.37	0.75
Photo-SLAM [9]	0.58	**0.32**	5.03	0.47	0.58	0.35	1.18	0.23	1.09
MonoGS [12]	1.38	2.15	7.19	1.46	1.68	2.05	3.16	1.77	2.61
MGSO [16]	0.35	1.02	5.93	0.22	0.54	0.28	0.34	0.20	1.11
Ours	**0.33**	0.45	3.61	**0.21**	**0.47**	**0.24**	**0.31**	**0.20**	**0.73**

TABLE II

CAMERA TRAJECTORY EVALUATION ON THE TUM RGB-D DATASET, RMSE[CM].

Method	fr1/desk	fr2/xyz	fr3/office	Avg.
DSO [3]	2.45	1.10	5.50	3.02
DSO+DUSt3R	2.10	1.10	3.98	2.39
DROID-SLAM [25]	1.80	1.23	4.21	2.41
Photo-SLAM [9]	1.54	0.98	**1.26**	**1.26**
MonoGS [12]	3.78	4.60	3.50	3.96
Ours	**1.48**	**0.85**	2.02	1.45

point along a ray:

$$O(\mathbf{o},\mathbf{r},t) = \sum_{k=1}^{K} \alpha_k O_k(G_k,\mathbf{o},\mathbf{r},t) \cdot \prod_{j=1}^{k-1} (1-\alpha_j O_k(G_k,\mathbf{o},\mathbf{r},t)) \tag{11}$$

where $\mathbf{o} \in \mathbb{R}^{3\times1}$ represents the camera center, $\mathbf{r} \in \mathbb{R}^{3\times1}$ represents the ray direction, and t represents the position along the ray. We define the opacity of a 3D point x as the minimal opacity value among all viewing directions:

$$O(\mathbf{x}) = \min_{(\mathbf{r},t)} O(\mathbf{o},\mathbf{r},t) \tag{12}$$

the opacity of the corner points of the leaf node voxels is computed by the tile-based evaluation algorithm from [23]. Finally, we employ the sparse voxel-based marching cubes algorithm from [24] to extract the mesh from the octree.

IV. EXPERIMENTS

A. Implementation and Setup

1) Datasets: To comprehensively validate our approach, we evaluate it on three datasets, including both synthetic and real-world sequences: Replica [26], which contains photorealistic 3D indoor scenes; EuRoC MAV [27], which consists of multiple indoor sequences with varying motion dynamics and illumination conditions; and TUM RGB-D [28], a real-world dataset featuring RGB-D sequences recorded from a handheld camera in indoor environments.

2) Implementation Details: Our main program is implemented in Python, with tracking-related functions from DSO encapsulated as interfaces using Pybind11. For both 3DGS and DSO, we adopt their default configuration settings. For DUSt3R, we use weights pre-trained on indoor scenes and select the top four keyframes with the highest co-visibility relative to the input frame as input pairs and optimize the point maps for 300 iterations. To distinguish between optimized and non-optimized Gaussians, we set τ to 400. For experiments on the Replica dataset, we set the octree voxel size to 0.02 m and perform new Gaussian insertions every 20 frames. For the EuRoC MAV and TUM RGB-D datasets, we set the octree voxel size to 0.05 m and perform new Gaussian insertions every 20 frames. During evaluation, we use Sim(3) to align the output trajectory and reconstructed mesh with the ground-truth reference. All experiments are conducted on a desktop PC equipped with an Intel Xeon Gold 6132 CPU and an NVIDIA RTX 3090 GPU.

B. Evaluation on Camera Tracking

We evaluate the camera tracking performance on the TUM RGB-D and Replica datasets and adopt the mean RMSE error as the evaluation metric. To show the effectiveness of our approach, we compare it against DROID-SLAM [25], Photo-SLAM [9], MonoGS [12], and MGSO [16]. For MGSO, we directly use the numbers reported in their paper. As shown in Tab.I, our method achieves the lowest mean RMSE error on the synthetic Replica dataset. For the real scenes TUM RGB-D dataset, we use only RGB images as input. The comparative results in Tab.II reveal that our method significantly outperforms the classical SLAM method DSO in pose estimation accuracy. This improvement validates that the dense depth maps rendered through our structured Gaussian representation effectively enhance the precision of visual odometry by providing reliable geometric constraints.

Additionally, we implement DSO+DUSt3R, which utilizes the dense depth maps estimated by DUSt3R for direct image alignment by incorporating depth information from all pixels. However, it is worth noting that DUSt3R has a slow matching speed, making it unsuitable for real-time SLAM applications. Instead, we render depth maps using 3DGS and apply DUSt3R for depth estimation only on keyframes. Thanks to the global scene modeling capability and fast rendering speed of 3DGS, our method can generate more efficient and accurate pseudo-depth maps, significantly improving both tracking performance and computational efficiency.

C. Evaluation on Mapping

1) Rendering Performance: To ensure a fair comparison, we evaluate our rendering quality using standard image quality metrics: PSNR, SSIM, and LPIPS. Since DROID-SLAM is unable to synthesize novel view images, it is excluded from this comparison. Results for other systems were obtained from their respective publications, except for MonoGS. We re-implemented MonoGS and report the experimental results under the monocular camera setting. As shown in Tab.III, our method achieves higher rendering quality compared to other baseline methods. This improvement is attributed to the dense and reliable depth estimation provided by DUSt3R, which guides the 3D Gaussians to better capture scene geometry and texture details. Additionally, Tab.IV presents rendering performance on the challenging EuRoC MAV dataset, which contains sequences with significant motion blur and noise caused by high-speed camera movements.

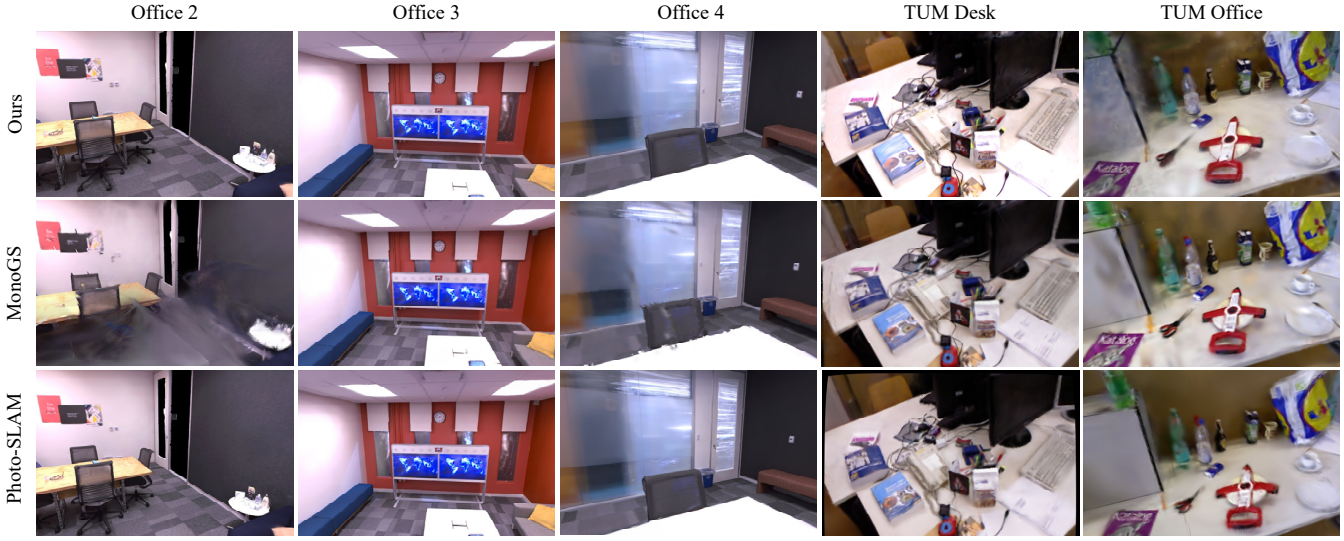

Fig. 4. Evaluation of rendering quality on Replica and TUM RGB-D datasets. Qualitative results demonstrate that our method produces higher-quality rendering results compared to the baseline methods.

TABLE III
QUANTITATIVE EVALUATION OF RENDERING QUALITY ON THE REPLICA DATASET.

Method	Metric	Room0	Room1	Room2	Office0	Office1	Office2	Office3	Office4	Avg.
Photo-SLAM [9]	PSNR [dB]↑	26.75	27.78	29.43	35.22	34.35	29.58	28.55	32.05	30.46
	SSIM↑	**0.94**	**0.93**	0.91	0.89	0.92	0.79	0.84	0.89	0.89
	LPIPS↓	0.31	0.28	0.24	0.21	0.23	**0.26**	0.26	**0.22**	0.25
MonoGS [12]	PSNR [dB]↑	24.48	25.47	23.69	31.87	33.00	23.73	27.03	27.61	27.11
	SSIM↑	0.77	0.80	0.84	0.90	0.91	0.83	0.86	0.90	0.85
	LPIPS↓	0.28	0.32	0.35	0.24	**0.17**	0.30	**0.21**	0.23	0.26
MGSO [16]	PSNR [dB]↑	28.11	**30.04**	31.89	36.34	38.20	28.90	30.27	31.41	31.90
	SSIM↑	0.82	0.87	0.92	0.95	0.96	0.90	0.91	0.93	0.91
	LPIPS↓	0.33	0.27	0.24	0.22	0.25	0.29	0.26	0.26	0.27
Ours	PSNR [dB]↑	**29.42**	29.94	**32.16**	**37.29**	**38.31**	**30.30**	**30.55**	**33.67**	**32.71**
	SSIM↑	0.83	0.90	**0.95**	**0.97**	**0.96**	**0.92**	**0.93**	**0.95**	**0.93**
	LPIPS↓	**0.27**	**0.25**	**0.22**	**0.21**	0.22	0.27	0.25	0.25	**0.24**

TABLE IV
QUANTITATIVE EVALUATION OF RENDERING QUALITY ON THE EUROC MAV DATASET.

Method	Metric	MH	V1	V2	Avg.
Photo-SLAM [9]	PSNR [dB]↑	18.60	18.30	17.94	18.28
	SSIM↑	0.65	0.73	0.65	0.68
	LPIPS↓	0.39	0.44	0.53	0.45
MonoGS [12]	PSNR [dB]↑	17.32	17.71	16.03	17.02
	SSIM↑	0.69	0.70	0.61	0.67
	LPIPS↓	0.39	0.47	0.49	0.45
MGSO [16]	PSNR [dB]↑	20.75	20.26	20.31	20.44
	SSIM↑	0.72	**0.79**	0.75	0.75
	LPIPS↓	0.36	0.39	0.39	0.38
Ours	PSNR [dB]↑	**21.74**	**20.21**	**21.39**	**21.11**
	SSIM↑	**0.80**	0.77	**0.80**	**0.79**
	LPIPS↓	**0.35**	**0.39**	**0.36**	**0.37**

To enhance robustness, our method employs confidence-weighted depth filtering, as described in Eq.10, to mitigate the impact of noise in the input data. As shown in Fig.4, our rendered images exhibit fewer artifacts and preserve fine-grained details compared to baseline methods.

2) Mesh Evaluation: As shown in Fig.5, we present visualizations of meshes extracted from structured 3D Gaussians. It is important to note that previous Gaussian-based SLAM methods cannot directly extract meshes from Gaussian representations online, which limits their applicability in downstream tasks. In contrast, we adopt the Gaussian opacity field proposed in [23] to extract meshes directly from Gaussians. Unlike the offline mesh extraction approach in [23], our method organizes Gaussians within an octree-based hierarchical structure, which allows for efficient mesh extraction by leveraging the opacity values at the voxel vertices of leaf nodes in the octree.

D. Evaluation on Efficiency

The SLAM system requires real-time inference of the network. We evaluate our system on the Replica dataset, measuring the operating frame rate, tracking frame rate, rendering speed, GPU memory usage, and runtime. As shown

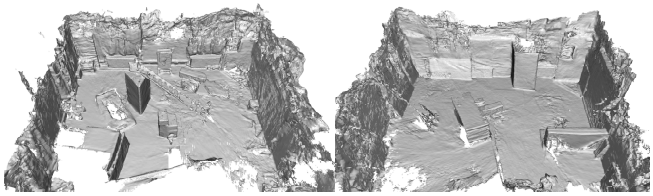

Fig. 5. Visualizations of meshes extracted from structured 3D Gaussians on the EuRoC MAV Dataset.

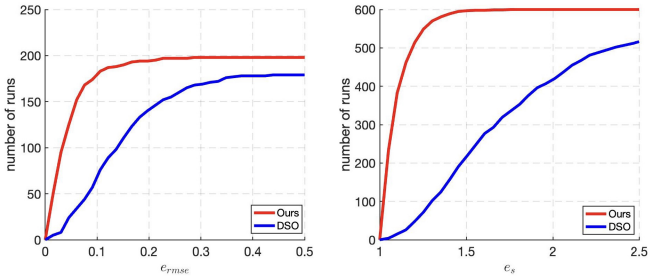

Fig. 6. Trajectory and scale drift evaluation on the EuRoC MAV data.

TABLE V

PERFORMANCE COMPARISON ON REPLICA/OFFICE 0.

Method	Operating FPS↑	Tracking FPS↑	Rendering FPS↑	GPU Memory Usage↓	Run Time↓
DROID-SLAM [25]	29.54	36.02	-	11 GB	<2 min
Photo-SLAM [9]	**34.88**	40.24	**896.62**	**6 GB**	<2 min
MonoGS [12]	1.32	1.32	759.14	14 GB	>10 min
Ours	34.67	**43.83**	802.53	11 GB	**<2 min**

in Tab.V, thanks to our decoupled tracking and mapping SLAM framework, our method achieves a tracking frame rate of 43.83 FPS and a rendering frame rate of 802.53 FPS, making it sufficiently fast to meet the real-time requirements of visual odometry alignment. While our method exhibits slightly higher GPU memory usage and a slightly reduced operating frame rate compared to baseline methods, this discrepancy stems from the computational demands of DUSt3R for dense point cloud estimation, coupled with inter-component communication and integration overhead within the different system components.

E. Evaluation on Scale Drift

Both DSO and our method inevitably experience some degree of drift over time due to the inherent limitations of monocular visual odometry, particularly the inability to directly observe absolute scale. Following the setup in DSO, we performed 200 runs on the EuRoC MAV dataset to assess the trajectory RMSE error e_{rmse}, and 600 runs to measure scale drift using the metric e_s. As shown in the Fig. 6, our method consistently outperforms DSO on both metrics, exhibiting significantly lower scale drift and trajectory error. This improvement can be attributed to the dense pseudo-depth maps $\hat{D}^{\text{pseudo}}$ generated by the 3DGS, which provide more stable and geometrically consistent depth supervision for the pose tracking process.

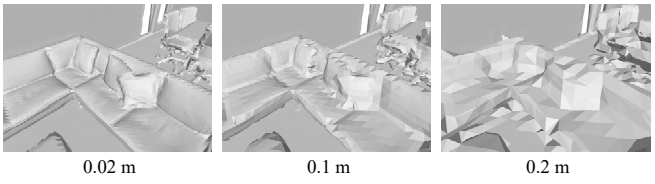

Fig. 7. Reconstruction results with different voxel sizes of the octree.

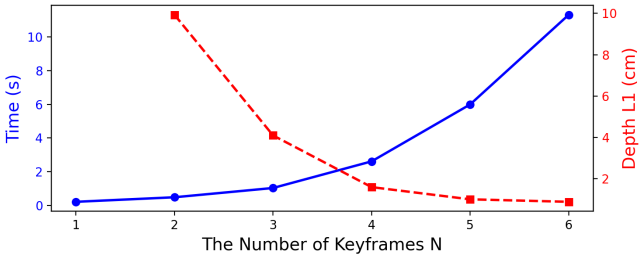

Fig. 8. Ablation Study on the Number of Keyframes N. We evaluate the difference between the depth maps predicted by DUSt3R and the ground truth depth maps using the Depth L_1 metric.

TABLE VI

ABLATION STUDY OF DUST3R AND PSEUDO-DEPTH MAPS ON REPLICA/OFFICE 1.

DUSt3R	Pseudo Depth	PSNR [dB]↑	RMSE [cm]↓
		36.84	0.60
✓		37.88	0.60
	✓	37.10	0.51
✓	✓	**38.31**	**0.47**

F. Ablation Study

In this section, we conduct ablation analysis on hyperparameters and network structures.

1) Octree Voxel Size: Since our meshes are directly extracted from the leaf node voxels of the octree via the sparse Marching Cubes algorithm, the voxel size of the leaf node voxels significantly impacts the resolution of the reconstructed mesh. Fig.7 visualizes the reconstructed meshes under three voxel sizes. The large voxel size fails to capture fine-grained details, resulting in oversimplified and coarse reconstructions. Conversely, the small voxel size recovers detailed geometry but comes at the cost of significant memory and computational overhead. To balance reconstruction accuracy and efficiency, we select voxel sizes of 0.02 m and 0.05 m based on the scale of the scene.

2) Impact of DUSt3R and Pseudo-Depth Maps: As shown in Tab. VI, we analyze the impact of DUSt3R and pseudo-depth maps on system performance. The pseudo-depth maps rendered by 3DGS significantly improve pose estimation accuracy by providing dense depth priors for tracking, while also indirectly improving the scene reconstruction geometry. Without pseudo-depth constraints, the system reverts to the tracking strategy in original DSO and thus cannot benefit from additional depth supervision to refine pose estimates. Simultaneously, DUSt3R ensures geometric consistency during 3DGS optimization, thereby boosting rendering fidelity.

When both modules are enabled, the system benefits from synergistic interaction, achieving the best results in pose estimation and rendering quality.

3) Keyframe Selection for DUSt3R: We perform an ablation study to determine the optimal number of keyframes N required for DUSt3R to estimate the point map. As shown in Fig.8, increasing the number of keyframes significantly increases evaluation time of DUSt3R, which in turn affects the mapping speed. To balance rendering accuracy with computational efficiency, we select $N = 4$ keyframes as the optimal configuration.

V. CONCLUSION

We propose a monocular dense SLAM method that integrates a structured Gaussian representation into a traditional optimization-based visual odometry pipeline. Specifically, we introduce a structured Gaussian representation organized via octree-based hierarchical management. This structure allows the rendered pseudo-depth maps to be seamlessly integrated into the visual odometry pipeline as dense geometric constraints, which significantly enhances tracking robustness under challenging conditions. To further refine the Gaussian representation, we utilize DUSt3R to estimate dense depth maps for selected keyframes. Given the slow inference speed of DUSt3R, future work may explore potential acceleration strategies, such as incorporating lightweight stereo networks or developing more efficient training schemes for 3D Gaussian representation.

REFERENCES

[1] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

[2] Raúl Mur-Artal and Juan D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[3] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[4] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[5] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99–106, December 2021.

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

[7] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12786–12796, June 2022.

[8] Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 499–507, 2022.

[9] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular stereo and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21584–21593, June 2024.

[10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[11] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, June 2024.

[12] Hidenobu Matsuki, Riku Murai, Paul H.J. Kelly, and Andrew J. Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18039–18048, June 2024.

[13] Zhexi Peng, Tianjia Shao, Yong Liu, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[14] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21357–21366, 2024.

[15] Chi-Ming Chung, Yang-Che Tseng, Ya-Ching Hsu, Xiang-Qian Shi, Yun-Hung Hua, Jia-Fong Yeh, Wen-Chin Chen, Yi-Ting Chen, and Winston H Hsu. Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9400–9406. IEEE, 2023.

[16] Yan Song Hu, Nicolas Abboud, Muhammad Qasim Ali, Adam Srebrnjak Yang, Imad H. Elhajj, Daniel Asmar, Yuhao Chen, and John S. Zelek. Mgso: Monocular real-time photometric slam with efficient 3d gaussian splatting. *ArXiv*, abs/2409.13055, 2024.

[17] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024.

[18] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[19] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[20] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2043–2050. IEEE, 2017.

[21] Erik Sandström, Yue Li, Luc Van Gool, and Martin R Oswald. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18433–18444, 2023.

[22] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5322–5332, 2024.

[23] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes. *arXiv preprint arXiv:2404.10772*, 2024.

[24] Shaofan Liu, Junbo Chen, and Jianke Zhu. Hvofusion: Incremental mesh reconstruction using hybrid voxel octree. *arXiv preprint arXiv:2404.17974*, 2024.

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

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

[27] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[28] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012.