MAGiC-SLAM: Multi-Agent Gaussian Globally Consistent SLAM

Vladimir Yugay Theo Gevers Martin R. Oswald
University of Amsterdam, Netherlands
{vladimir.yugay, th.gevers, m.r.oswald}@uva.nl
vladimiryugay.github.io/magic_slam

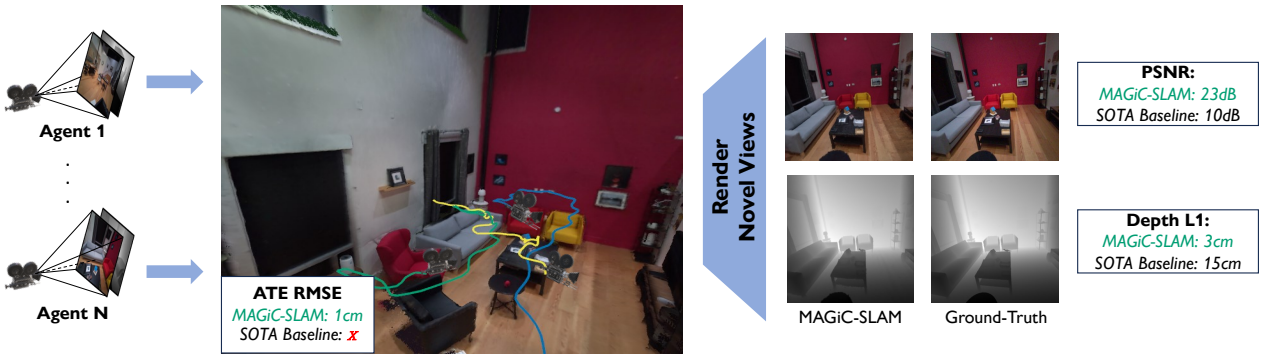

Figure 1. **MAGiC-SLAM** is a multi-agent SLAM method capable of novel view synthesis. Given single-camera RGBD input streams from multiple simultaneously operating agents MAGiC-SLAM estimates their trajectories and reconstructs a 3D Gaussian map that can be rendered from previously unseen viewpoints. We showcase the high-fidelity 3D Gaussian map of a real-world environment alongside multiple agent trajectories (depicted in green, yellow, and blue) within it. Our method effectively utilizes information from multiple agents to achieve centimeter-level tracking accuracy. Our mapping and map merging strategies allow for realistic rendering of color and depth, significantly improving the state of the art. Unlike previous methods, MAGiC-SLAM is flexible in the number of agents it can handle.

Abstract

Simultaneous localization and mapping (SLAM) systems with novel view synthesis capabilities are widely used in computer vision, with applications in augmented reality, robotics, and autonomous driving. However, existing approaches are limited to single-agent operation. Recent work has addressed this problem using a distributed neural scene representation. Unfortunately, existing methods are slow, cannot accurately render real-world data, are restricted to two agents, and have limited tracking accuracy. In contrast, we propose a rigidly deformable 3D Gaussian-based scene representation that dramatically speeds up the system. However, improving tracking accuracy and reconstructing a globally consistent map from multiple agents remains challenging due to trajectory drift and discrepancies across agents' observations. Therefore, we propose new tracking and map-merging mechanisms and integrate loop closure in the Gaussian-based SLAM pipeline. We evaluate MAGiC-SLAM on synthetic and real-world datasets and find it more accurate and faster than the state of the art.

1. Introduction

Visual Simultaneous Localization and Mapping (SLAM) enables a machine to build a 3D map of its surroundings while determining its location relying solely on its camera input. Imagine an autonomous car navigating a busy city or a robot exploring a cluttered room. For these systems to move safely and make decisions, they need a reliable understanding of their surroundings. Over the years, visual SLAM has advanced [8, 13, 36], learning to handle complex scenes with increasing accuracy. This capability forms the backbone for various systems from self-driving cars to virtual reality glasses, where spatial understanding is crucial for navigation, planning, and interactive experiences.

Recently, SLAM systems have been enhanced with novel view synthesis (NVS) capabilities [14, 22, 35], allowing them to generate realistic, immersive views of scenes. This allows more detailed scene exploration and supports the creation of high-quality virtual environments. However, these NVS-capable SLAM methods are slow, and tracking accuracy remains limited [14]. These limitations become even more pronounced in a multi-agent setup, with larger

scenes and numerous observations.

Effectively processing information from multiple agents operating simultaneously opens up new possibilities for SLAM systems. First, this naturally accelerates 3D reconstruction by enabling horizontal parallelization; each agent can map different parts of the environment concurrently, leading to faster, more efficient coverage of large areas [5]. Second, agents can collaboratively refine their location estimates by sharing their observations [17]. Finally, the global map constructed from the perspectives of multiple agents is more geometrically consistent and accurate [32].

The only solution for multi-agent NVS-capable SLAM hitherto has combined a distributed neural scene representation with a traditional loop closure mechanism [10]. However, this approach faces several issues. Firstly, it fails to reconstruct a coherent map capable of accurate NVS. Neural scene representations inherently lack support for rigid body transformations [19] making it impossible to update and merge the global map efficiently. This in turn leads to poor rendering quality of the novel views. Secondly, localization accuracy remains limited, as neural-based camera pose optimization struggles with the variability and imperfections inherent to real-world data. Finally, the system is prohibitively slow, demands extensive computational resources, and supports only two agents.

We present MAGiC-SLAM, a multi-agent NVS-capable SLAM pipeline designed to overcome these limitations. Firstly, MAGiC-SLAM utilizes 3D Gaussians as the scene representation supporting rigid body transformations. Every agent processes the input RGBD sequence in chunks (sub-maps) which can be effectively corrected and merged into a coherent global map. Secondly, we implement a loop closure mechanism to improve trajectory accuracy by leveraging information from all agents. It is further enhanced with a novel loop detection module, based on a foundational vision model, which enables better generalization to unseen environments. Finally, our flexible tracking and mapping modules allow our system to achieve superior speed and easily scale to varying numbers of agents.

Overall, our contributions can be summarized as follows:

- A multi-agent NVS-capable SLAM system for consistent 3D reconstruction supporting an arbitrary number of simultaneously operating agents
- A loop closure mechanism for Gaussian maps leveraging a foundational vision model for loop detection
- Efficient map optimization and fusion strategies that reduce required disk storage and processing time
- A robust Gaussian-based tracking module

2. Related Work

Neural SLAM. Neural radiance fields (NeRF) [23] have achieved remarkable results in novel view synthesis [2, 24, 41]. [35] started a whole new line of research [19, 20, 31, 37, 40, 44, 46] by proposing a SLAM system using neural fields to represent the map and optimize the camera poses. Various neural field approaches have provided insights for neural SLAM developments. For instance, NICE-SLAM [46] uses a voxel grid to store neural features, while Vox-Fusion [40] improves the grid with adaptive sizing. Point-SLAM [31] attaches feature embeddings to point clouds on object surfaces, offering more flexibility and the ability to encode concentrated volume density. Co-SLAM [37] employs a hybrid representation combining coordinate encoding and hash grids to achieve smoother reconstruction and faster convergence. A group of methods [6, 29] use neural fields solely for mapping while relying on traditional feature point-based visual odometry for tracking. Loopy-SLAM [19] explores the usage of pose-graph optimization to handle trajectory drift. While these methods have shown success in NVS, they are computationally intensive, slow, and struggle to render real-world environments accurately [14, 42]. Additionally, neural maps inherently lack support for rigid body transformations [19], which greatly limits the efficiency of map correction. In contrast, our scene representation is fast to render and optimize, significantly accelerating our SLAM pipeline. With native support for rigid body transformations, our approach enables faster map updates and seamless map merging.

Gaussian SLAM. Recently, 3D Gaussian Splatting(3DGS) [15] revolutionized novel view synthesis, achieving photorealistic real-time rendering at over 100 FPS without relying on neural networks. Compared to NeRFs, 3DGS is more efficient in terms of memory, and faster to optimize. These factors inspired the line of SLAM systems [11, 12, 14, 22, 39, 42] using 3D Gaussians instead of neural fields for tracking and mapping. [11, 22, 39] estimated camera pose by computing camera gradients from the 3DGS field analytically. Others [14, 42] design a warp-based camera pose optimization. Finally, [12, 28] use existing sparse SLAM systems to speed up tracking. While addressing many limitations of neural SLAM, the proposed methods overlook global consistency, leading to trajectory error accumulation [33]. In contrast, MAGiC-SLAM integrates a loop closure mechanism within the 3DGS SLAM pipeline to correct accumulated trajectory errors. It effectively generalizes to unknown environments leveraging a foundational vision model for loop detection. Concurrently, Zhu et al. [45] proposed a way to do loop closure in a 3DGS-based SLAM pipeline. However, the method cannot handle multiple agents, uses a standard loop closure detection mechanism, and is less efficient in tracking and mapping.

Multi-agent Visual SLAM. Despite significant progress in single-agent systems, multi-agent SLAM remains less developed due to the complexity of designing a multi-agent pipeline. Multi-agent SLAM can be categorized into two

types: centralized and distributed. In a distributed system [5, 18] agents communicate sporadically. While being more robust in environments with stringent networking constraints, their performance is limited by the computing power of a separate agent. Centralized systems [17, 32] use a centralized server to manage agents' maps and globally optimize their trajectories. Despite the success of these methods, they do not provide a map with novel view synthesis capabilities. Hu et al. [10] proposed a centralized neural-based multi-agent NVS-capable SLAM to address this. However, it inherited all the problems associated with neural representations such as low speed, compute requirements, and ability to render real-world data [14]. In addition, CP-SLAM [10], can only support two agents operating simultaneously. In contrast, MAGiC-SLAM employs an efficient scene representation that enables seamless global map reconstruction with NVS capabilities suited for real-world environments. Additionally, it delivers improved tracking accuracy through robust tracking and loop closure modules. Finally, our system is flexible in handling multiple agents, constrained only by the capacity of the centralized server.

3. Method

We introduce MAGiC-SLAM, which architecture is shown in Figure 2. Every agent processes an RGB-D stream performing local mapping and tracking. 3D Gaussians are used to represent the agents' sub-maps and to improve tracking accuracy. The foundation vision model-based loop detection module extracts features from RGB images and sends them to the centralized server with the local sub-map data. The server detects the loops based on the image encodings, performs pose graph optimization, and sends the optimized poses back to the agents. At the end of the run, the server fuses the agents' sub-maps into a global Gaussian map. This section introduces per-agent mapping and tracking mechanisms and describes global loop closure and map construction processes.

3.1. Mapping

Every agent processes a single sub-map of a limited size, represented as a collection of 3D Gaussians. Sub-maps are initialized from the first frame. θ_{sample} points sampled from the lifted to 3D RGBD frame serve as the means for new 3D Gaussians. New Gaussians are added to regions of the active sub-map with low Gaussian density, based on rendered opacity, and are optimized using the loss:

$$L_{\text{mapping}} = \lambda_{\text{color}} \cdot L_{\text{color}} + \lambda_{\text{depth}} \cdot L_{\text{depth}} + \lambda_{\text{reg}} \cdot L_{\text{reg}} \ , \quad (1)$$

where are $\lambda_* \in R$ are hyperparameters. The color loss L_{color} is is defined as:

$$L_{\text{color}} = (1-\lambda) \cdot |\hat{I} - I|_1 + \lambda\big(1 - \text{SSIM}(\hat{I}, I)\big) \ , \quad (2)$$

where $\hat{I}, I$ are the rendered and input images respectively, and $\lambda \in R$ is the weighting factor. Depth loss L_{depth} is formulated as:

$$L_{\text{depth}} = |\hat{D} - D|_1 \ , \quad (3)$$

where $\hat{D}, D$ are the rendered and input depth maps. The regularization loss L_{reg} is represented as:

$$L_{\text{reg}} = |K|^{-1} \sum_{k \in K} |s_k - \overline{s}_k|_1, \quad (4)$$

where $s_k \in \mathbb{R}^3$ is the scale of a 3D Gaussian, $\overline{s}_k$ is the mean sub-map scale, and $|K|$ is the number of Gaussians in the sub-map. We do not optimize the spherical harmonics of the Gaussians to reduce their memory footprint and improve tracking accuracy [22].

A new sub-map is created, and the previous one is sent to the server after every θ_{submap} frame. Creating new sub-maps in already mapped areas introduces some computational redundancy, but it limits overall computational cost and maintains tracking and mapping speed as the scene expands. Moreover, unlike in [42], only Gaussians with zero rendered opacity in the current camera frustum are dispatched to the server. This approach significantly reduces the disk space required to store the sub-maps and speeds up the map merging process. The supplementary material provides a quantitative evaluation of the strategy.

3.2. Tracking

Typically GS SLAM systems optimize camera pose either explicitly or implicitly. In the explicit case, the camera pose gradient is derived [21] or approximated [11, 39] analytically. In the implicit case [14, 42], the relative pose between two frames is optimized by warping the GS point clouds and minimizing re-rendering depth and color errors. In both cases, the camera pose is estimated based on the existing map, following a frame-to-model paradigm [13].

Considering the advantages and disadvantages of frame-to-frame and frame-to-model tracking paradigms [8, 13, 36] and the convenience of using sub-maps for loop closure, we propose a hybrid implicit tracking approach that combines the strengths of both. Specifically, we initialize the relative pose using a deterministic frame-to-frame dense registration and then refine it through frame-to-model optimization. This approach differs from previous NVS SLAM methods, initializing the relative pose based on a constant speed assumption.

Moreover, we found implicit tracking to be more accurate than the explicit approach given robust pose initialization. At the same time, explicit tracking methods do not benefit from pose initialization since camera poses are optimized across the whole co-visibility window at every mapping step. We refer to supplementary material for experiments highlighting this phenomenon.

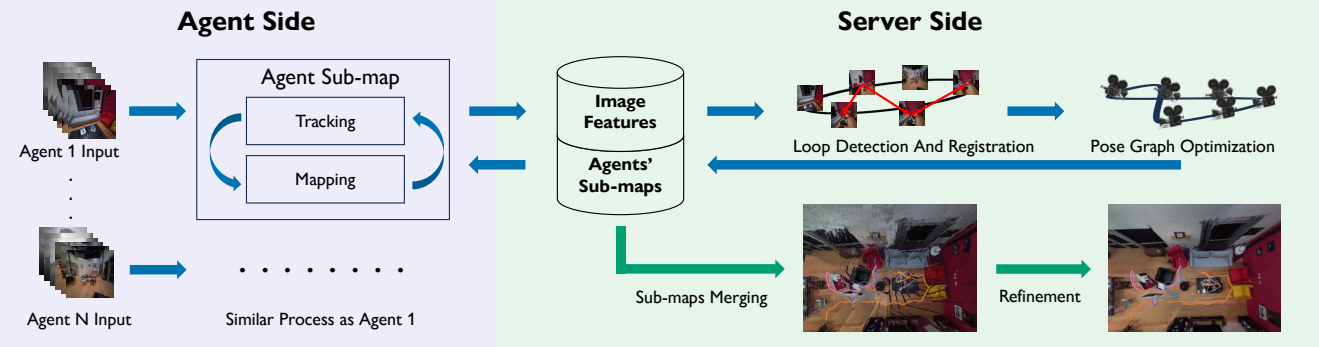

Figure 2. **MAGiC-SLAM Architecture.** *Agent Side:* Each agent processes a separate RGBD stream, maintaining a local sub-map and estimating its trajectory. When an agent starts a new sub-map, it sends the previous sub-map and image features to the centralized server. *Server Side:* The server stores the image features and sub-maps from all agents and performs loop closure detection, loop constraint estimation, and pose graph optimization. It then updates the stored sub-maps and returns the optimized poses to the agents. Once the algorithm completes (denoted by green arrows), the server merges the accumulated sub-maps into a single unified map and refines it.

Pose Initialization. At time t, given the input colored point cloud P_t, the goal is to estimate its pose $T_{t-1,t} \in SE(3)$ relative to the previous input point cloud P_{t-1}. The registration process is performed iteratively at several scales, starting from the coarsest, $l = 0$, down to the finest, $l = L$. At every scale, both point clouds are voxelized, and the set of correspondences $K = \{(p, q)\}$ between P_t, P_{t-1} is computed using ICP [3]. For every scale, a loss function:

$$E(\mathbf{T_{t-1,t}}) = (1-\sigma) \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} \left(r_C^{(\mathbf{p},\mathbf{q})}(\mathbf{T_{t-1,t}})\right)^2 + \sigma \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} \left(r_G^{(\mathbf{p},\mathbf{q})}(\mathbf{T_{t-1,t}})\right)^2, \quad (5)$$

is optimized where $\sigma \in [0, 1]$ is a scalar weight. $r_G^{(p,q)}$ is a geometric residual defined as:

$$r_G^{(p,q)} = \Big(\big(T_{t-1,t}(q) - p\big)n_p\Big)^2. \quad (6)$$

where n_p is a normal vector of p. The color residual $r_C^{(p,q)}$ is computed as:

$$r_C^{(p,q)} = C_p\Big(f_p\big(T_{t,t-1}(q)\big) - p\Big) - C(q), \quad (7)$$

where f_p is a function projecting a point on the tangent plane of p, $C(q)$ is the intensity of point q, and C_p is a continuous intensity function defined over point cloud P_t. The loss is optimized until convergence with the Gauss-Newton method. For more details about the registration mechanism please refer to [27].

Pose Refinement. The pose obtained in the initialization step is further refined using re-rendering losses [42] of the scene. To refine the initial pose estimate, we freeze all Gaussian parameters and minimize the loss:

$$\arg\min_{T_{t-1,t}} L_{\text{tracking}}\Big(\hat{I}(T_{t-1,t}), \hat{D}(T_{t-1,t}), I_t, D_t, \alpha_t\Big), \quad (8)$$

where $\hat{I}(T_{t-1,t})$ and $\hat{D}(T_{t-1,t})$ are the rendered color and depth from the sub-map warped with the relative transformation $T_{t-1,t}$, I_t and D_t are the input color and depth map at frame t.

We use soft alpha and color rendering error masking to avoid contaminating the tracking loss with pixels from previously unobserved or poorly reconstructed areas [14, 42]. Soft alpha mask M_{alpha} is a polynomial of the alpha map rendered from the 3D Gaussians. Error boolean mask M_{inlier} discards all the pixels where the color and depth errors are larger than a frame-relative error threshold:

$$L_{\text{tracking}} = \sum M_{\text{inlier}} \cdot M_{\text{alpha}} \cdot \big(\lambda_c |\hat{I} - I|_1 + (1-\lambda_c)|\hat{D} - D|_1\big). \quad (9)$$

The weighting ensures the optimization is guided by well-reconstructed regions where the accumulated alpha values are close to 1 and rendering quality is high.

3.3. Loop Closure

Loop closure is the process that detects when a system revisits a previously mapped area and adjusts the map and camera poses to reduce accumulated drift, ensuring global consistency. It includes four key steps: loop detection, loop constraint estimation, pose graph optimization, and integration of optimized poses. Loop detection identifies when a previously mapped location is revisited. Loop constraint estimation calculates the relative pose between frames in the loop. Pose graph optimization then refines all camera poses, minimizing discrepancies between odometry and loop constraints to maintain global consistency. Finally, the optimized poses are integrated into the reconstructed map.

Loop Detection. Throughout the run, each agent extracts features from the first frame in each sub-map and sends them to a centralized server. The features are stored in a GPU database [7] optimized for similarity search. The

server queries the database for potential loop candidates for every new sub-map frame. A pair of frames is considered a loop if the distance between them is less than a threshold θ_{feature} in the image feature space. Loops belonging to the same agent are additionally filtered based on time threshold θ_{time} to avoid too many uninformative loops.

Loop detection heavily influences the accuracy of loop edge constraint estimation since large frame overlap is crucial for registration [19]. Common approaches for loop closure detection are based on ORB [9] or NetVLAD [1] image descriptors. Hu et al. [10] argue that NetVLAD global image descriptors are better suited for loop closure detection. However, NetVLAD is trained on a relatively small dataset which leads to a lack of generalization to unknown environments. To overcome this limitation we propose to use a foundational vision model as a feature extractor. We use a small variation of DinoV2 [25] because of the large amount of data it was trained on, its compactness, and the quality of the features it produces for the downstream tasks.

Loop Constraints. We use a coarse-to-fine registration approach to estimate loop edge constraints. For the coarse alignment, we apply the global registration method of Rusu et al. [30], which extracts Fast Point Feature Histograms (FPFH) from downsampled versions of the source (P_s) and target (P_t) point clouds. Correspondence search is then performed in the FPFH feature space rather than in Euclidean space. Optimization is embedded in a RANSAC framework to reject outlier correspondences, producing a rigid transformation of the source point cloud S_s to align with the target S_t. Finally, ICP [3] is applied to the full-resolution point clouds to refine the coarse alignment estimate.

We found that directly registering Gaussian means from different agents is unreliable, as Gaussians representing overlapping regions can have widely varying distributions across agents. To address this, we anchor an input point cloud at the start of each sub-map and use it for registration.

Pose Graph Optimization. Each node in a pose graph, $T_i \in SE(3)$, represents a distinct sub-map. Neighboring sub-maps are connected by odometry edges, derived from the tracker, representing the relative transformations between them. Loop edge constraints $T_{st} \in SE(3)$ are added between non-adjacent sub-maps and computed using a registration method different from that of the tracker. The error term between two nodes i and j is defined as:

$$e_{ij} = \log\left(T_{ij}^{-1}(T_i^{-1}T_j)\right), \tag{10}$$

where T_{ij} is the relative transformation between the nodes i and j, T_i, T_j are the node poses, and $\log$ is the logarithmic map from $SE(3)$ to $se(3)$. To get the optimized camera poses we minimize the error term:

$$F(T) = \sum_{\langle i,j\rangle \in \mathcal{C}} \left(e(T_i, T_j)^\top \Omega_{ij} e(T_i, T_j)\right) \tag{11}$$

where $\Omega_{ij} \in R^{6\times 6}$ is a positive semi-definite information matrix reflecting the uncertainty of the constraint estimate - the higher the confidence of an edge, the larger the weight is applied to the residual. The error terms are linearized using a first-order Taylor expansion, and the loss is minimized with the Gauss-Newton method. For further details, please see [16].

Pose Update Integration. The pose graph optimization module provides pose corrections $\{T_c^i \in SE(3)\}_{i=1}^{N_s}$ for every sub-map of every agent. All camera poses $\{T_j^i\}_{j=1}^{N_p}$ belonging to a sub-map i are corrected as:

$$T_j^i \leftarrow T_i^c T_j^i. \tag{12}$$

All Gaussians $\{G_j^i(\mu_j^i, \Sigma_j^i, o_j^i, c_j^i)\}_{j=1}^{N_g}$ belonging to sub-map i are updated as well:

$$\mu_j^i \leftarrow T_i^c \mu_j^i, \quad \Sigma_j^i \leftarrow T_{i,R}^c \Sigma_j^i, \tag{13}$$

where $T_{i,R}^c$ is a rotation component of $T_i^c \in SE(3)$. We do not correct Gaussian colors since we do not optimize spherical harmonics (Subsection 3.1).

3.4. Global Map Construction

Once the agents complete processing their data, the server merges the sub-maps from all agents into a unified global map. The map is merged in two stages: coarse and fine. During the coarse stage, the server loads the cached sub-maps and appends them into a single global map. Caching Gaussians that are not visible from the first keyframe of the next sub-map allows for appending the Gaussians without the need for costly intersection check [42]. However, this results in visual artifacts at the edges of the renderings. This happens because some Gaussians with zero opacity for a given view still influence the rendering through the projected 2D densities at the edges. Additionally, the coarse merging of sub-maps might introduce geometric artifacts at their intersections. The fine merging stage addresses these issues by optimizing the Gaussian parameters using color and depth rendering losses for a small number of iterations and pruning Gaussians with zero opacity. The visual effects of the refining step are shown in Fig. 3.

4. Experiments

We describe our experimental setup and compare MAGiC-SLAM with state-of-the-art baselines. We assess tracking and rendering performance on synthetic and real-world multi-agent datasets and provide ablation studies on key components of our pipeline. Please refer to the supplementary material for the implementation details and hyperparameters for tracking, mapping, and loop closure modules.

Baselines. To evaluate tracking, following [10] we compare our method to the state-of-the-art multi-agent SLAM

(a) Visual artifacts (b) Refined view

(c) Geometric artifacts (d) Refined view

Figure 3. **Map Merging.** Our coarse-to-fine strategy effectively removes (a) visual artifacts caused by the GS mechanism and (c) geometric artifacts resulting from Gaussian sub-map intersections.

systems like SWARM-SLAM [17], CCM-SLAM [32], and CP-SLAM [10]. To make the comparison more comprehensive, we include several single-agent systems like Gaussian-SLAM [42], MonoGS [22], and Orb-SLAM3 [4]. We include Orb-SLAM3 since it is one of the most popular and reliable single-agent SLAM methods, Gaussian-SLAM since our approach uses sub-maps, and MonoGS as the most accurate single-agent NVS SLAM system [36]. We evaluate rendering against CP-SLAM [10] since it is the only existing multi-agent NVS-capable SLAM system.

Datasets. We test our method on the MultiagentReplica [10] dataset which contains four two-agent RGB-D sequences in a synthetic environment. Each sequence consists of 2500 frames, except for the Office-0 scene, which has 1500 frames. MultiagentReplica is a synthetic dataset and does not have a test set for novel view synthesis evaluation. Therefore, we extend our evaluation to real-world scenes using the ego-centric Aria [26] dataset. The Aria dataset provides ground-truth depth and camera information from recordings of two rooms within a real-world environment. We selected this dataset for its high-quality ground-truth data and the growing prevalence of egocentric wearable devices. However, the original dataset includes many dynamic objects, and since our method is not designed for dynamic environments, this restricted the amount of usable data. We selected three sequences from every room to simulate multi-agent operations, choosing sequences with sufficient consecutive frames without dynamic objects. We then extracted 500 consecutive frames from each sequence for training. We sampled 100 unseen frames from each room to test our method's NVS capabilities. We refer to this dataset as AriaMultiagent in all tables and figures.

Evaluation Metrics. To assess tracking accuracy, we use ATE RMSE [34], and for rendering we compute PSNR, SSIM [38] and LPIPS [43]. Rendering metrics on ReplicaMultiagent are evaluated by rendering full-resolution images along the estimated trajectory with mapping intervals similar to [31]. The same metrics over training and hold-out test frames are used for the AriaMultiagent dataset. We evaluate the depth error using the L1 norm in centimeters.

4.1. Tracking Performance

In Table 1 we compare our method against state-of-the-art multi-agent SLAM systems on the ReplicaMultiagent dataset processing two agents simultaneously. In Table 2 we evaluate MAGiC-SLAM on the AriaMultiagent dataset processing three agents simultaneously. Since CP-SLAM does not support the operation of more than two agents, we report its results on the first two out of three agents. To make our evaluation more comprehensive, we compare our method against single-agent SLAM systems in Table 3. We run single-agent systems on each agent separately and report their average performance. MAGiC-SLAM achieves superior tracking accuracy thanks to our novel two-stage tracking mechanism. The loop closure mechanism effectively utilizes multi-agent information to enhance accuracy even further.

Method	Agent	Off-0	Apt-0	Apt-1	Apt-2
CCM-SLAM [32]	**Agent 1**	9.84	✗	2.12	0.51
Swarm-SLAM [17]		1.07	1.61	4.62	2.69
CP-SLAM [10]		0.50	0.62	1.11	1.41
MAGiC-SLAM (w.o. Loop Closure)		0.44	0.30	0.48	0.91
MAGiC-SLAM		0.31	0.13	0.21	0.42
CCM-SLAM [32]	**Agent 2**	0.76	✗	9.31	0.48
Swarm-SLAM [17]		1.76	1.98	6.50	8.53
CP-SLAM [10]		0.79	1.28	1.72	2.41
MAGiC-SLAM (w.o. Loop Closure)		0.41	0.46	0.61	0.41
MAGiC-SLAM		0.24	0.21	0.30	0.22
CCM-SLAM [32]	**Average**	5.30	✗	5.71	0.49
Swarm-SLAM [17]		1.42	1.80	5.56	5.61
CP-SLAM [10]		0.65	0.95	1.42	1.91
MAGiC-SLAM (w.o. Loop Closure)		0.42	0.38	0.54	0.66
MAGiC-SLAM		**0.27**	**0.16**	**0.26**	**0.32**

Table 1. **Tracking performance on ReplicaMultiagent [10] dataset** (ATE RMSE [cm]↓). Comparison between MAGiC-SLAM (w.o. Loop Closure) and MAGiC-SLAM reveals the importance of loop closure for trajectory consistency. ✗ indicates invalid results due to the failure of CCM-SLAM.

4.2. Rendering Performance

We evaluate the rendering performance on the merged scene obtained from all the agents on ReplicaMultiagent in Ta-

Method	Agent	Room0	Room1
CCM-SLAM [32]	**Agent 1**	✗	✗
Swarm-SLAM [17]		6.11	4.29
CP-SLAM [10]		0.68	5.06
MAGiC-SLAM (w.o. Loop Closure)		0.92	1.15
MAGiC-SLAM		0.67	0.96
CCM-SLAM [32]	**Agent 2**	✗	✗
Swarm-SLAM [17]		8.43	4.95
CP-SLAM [10]		5.39	0.68
MAGiC-SLAM (w.o. Loop Closure)		1.72	1.33
MAGiC-SLAM		1.13	0.53
CCM-SLAM [32]	**Agent 3**	✗	✗
Swarm-SLAM [17]		4.82	5.12
CP-SLAM [10]		–	–
MAGiC-SLAM (w.o. Loop Closure)		3.76	1.25
MAGiC-SLAM		1.67	0.46
CCM-SLAM [32]	**Average**	✗	✗
Swarm-SLAM [17]		6.45	4.78
CP-SLAM [10]		3.03	2.87
MAGiC-SLAM (w.o. Loop Closure)		2.13	1.24
MAGiC-SLAM		**1.15**	**0.65**

Table 2. **Tracking performance on AriaMultiagent dataset** (ATE RMSE [cm]↓). Our method shows strong performance on real-world data. ✗ indicates invalid results due to the failure of CCM-SLAM. – indicates that CP-SLAM does not support the operation of more than two agents.

Methods	AriaMultiagent			ReplicaMultiagent				
	Room0	Room1	Avg.	Off0	Apt0	Apt1	Apt2	Avg.
ORB-SLAM3 [4]	3.18	2.85	3.01	0.60	1.07	4.94	1.36	1.99
Gaussian-SLAM [42]	✗	✗	✗	0.33	0.41	30.13	121.96	38.21
MonoGS [22]	1.90	2.71	2.30	0.38	0.21	3.33	0.54	1.15
MAGiC-SLAM w.o. LC	2.13	1.24	1.69	0.42	0.38	0.54	0.66	0.50
MAGiC-SLAM	**1.15**	**0.65**	**0.90**	**0.27**	**0.16**	**0.26**	**0.32**	**0.25**

Table 3. **Tracking performance compared to single-agent methods**(ATE RMSE [cm]↓). We compare our method with traditional and state-of-the-art Gaussian-based single-agent SLAM systems. Our method outperforms all the baselines even without loop closure (LC).

ble 4 and AriaMultiagent datasets in Table 5. Since CP-SLAM does not support the operation of more than two agents, we report its results on the first two out of three agents in the AriaMultiagent dataset. Thanks to 3D Gaussian splatting, agents' sub-maps can accurately render real-world data. Moreover, they are efficiently corrected using optimized poses from the loop closure module. Combined with an effective map-merging strategy, this approach allows our method to significantly outperform previous work in rendering both real-world training data and novel views. We also provide qualitative results in Fig. 4.

4.3. Ablation Studies

Effect of Pose Initialization. In Table 6 we numerically evaluate our two-stage tracking mechanism on the Aria-

Methods	Metrics	Off-0	Apt-0	Apt-1	Apt-2	Avg.
CP-SLAM [10]	PSNR [dB] ↑	28.56	26.12	12.16	23.98	22.71
	SSIM ↑	0.87	0.79	0.31	0.81	0.69
	LPIPS ↓	0.29	0.41	0.97	0.39	0.51
	Depth L1 [cm] ↓	2.74	19.93	66.77	2.47	22.98
MAGiC-SLAM	PSNR [dB] ↑	39.32	36.96	30.01	30.73	**34.26**
	SSIM ↑	0.99	0.98	0.95	0.96	**0.97**
	LPIPS ↓	0.05	0.09	0.18	0.17	**0.12**
	Depth L1 [cm] ↓	0.41	0.64	3.16	0.99	**1.30**

Table 4. **Training view synthesis performance on Replica-Multiagent dataset.** The global map built by merging the maps from two agents is evaluated by synthesizing training views. Our method significantly outperforms previous state of the art.

Methods	Metrics	Novel Views			Training Views		
		S1	S2	Avg.	S1	S2	Avg.
CP-SLAM [10]	PSNR [dB] ↑	8.96	9.17	9.06	10.01	10.45	10.23
	SSIM ↑	0.32	0.24	0.28	0.24	0.30	0.27
	LPIPS ↓	0.91	0.95	0.93	0.93	0.98	0.95
	Depth L1 [cm] ↓	17.14	13.23	15.18	18.42	12.01	15.12
MAGiC-SLAM	PSNR [dB] ↑	23.45	21.78	**22.61**	24.11	26.17	**25.14**
	SSIM ↑	0.89	0.85	**0.87**	0.91	0.93	**0.92**
	LPIPS ↓	0.22	0.21	**0.22**	0.20	0.14	**0.17**
	Depth L1 [cm] ↓	1.33	4.96	**3.15**	1.87	1.30	**1.59**

Table 5. **Novel and training view synthesis performance on AriaMultiagent dataset.** The global map built by merging the maps from two agents is evaluated by synthesizing novel and training views. Previous state-of-the-art methods struggle to render real-world data. *Note.* CP-SLAM is evaluated only on two agents since it does not support more than two.

Multiagent and ReplicaMultiagent [10] datasets. This result highlights the importance of pose initialization for tracking accuracy. In addition, we show that pose initialization does not add significant computational overhead.

Methods	Dataset	Tracking Frame [s] ↓	Avg. ATE RMSE [cm] ↓
MAGiC-SLAM (w.o. Pose Initialization)	AriaMultiagent	**0.66**	0.874
	ReplicaMultiagent	**0.68**	0.825
MAGiC-SLAM	AriaMultiagent	0.67	**0.670**
	ReplicaMultiagent	0.69	**0.365**

Table 6. **Ablation on pose initialization on AriaMultiagent and ReplicaMultiagent datasets**. Our method benefits from robust pose initialization without much computational overhead.

Loop Closure Detection. We evaluate the performance of our loop closure detection against the NetVLAD-based loop closure detection mechanism proposed in [10]. For this, we integrate a NetVLAD-based loop closure detection module into our pipeline keeping all the thresholds from [10]. Thanks to the huge amount of data DinoV2 [25] was trained, its features can accurately encode a large variety of data. This leads to more accurate loop closure detection as shown in Table 7.

Memory and Runtime Analysis. In Table 8, we compare

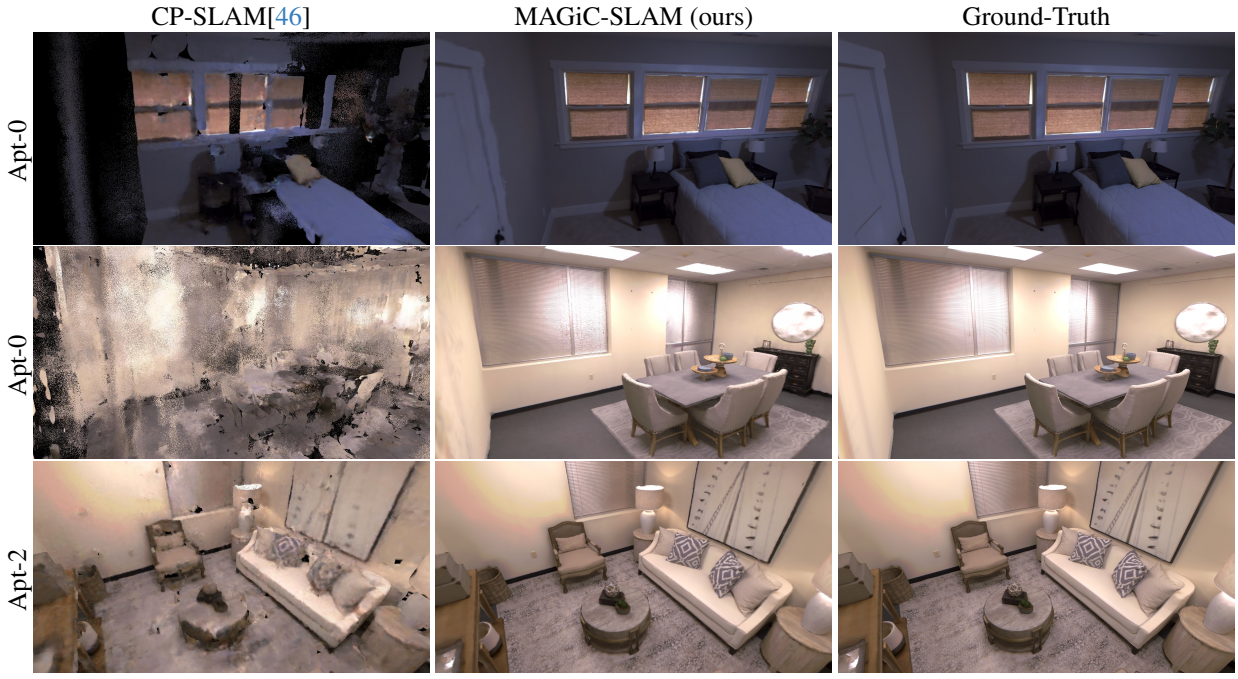

Figure 4. **Rendering performance on ReplicaMultiagent [10].** Thanks to GS scene representation and effective merging strategy, MAGiC-SLAM encodes more high-frequency details and substantially increases the quality of the renderings.

Methods	Datasets	Avg.
MAGiC-SLAM (w. NetVLAD [1])	AriaMultiagent	1.363
MAGiC-SLAM (w. NetVLAD [1])	ReplicaMultiagent	0.351
MAGiC-SLAM	AriaMultiagent	**0.900**
MAGiC-SLAM	ReplicaMultiagent	**0.252**

Table 7. **Ablation on loop detection mechanism on ReplicaMultiagent and AriaMultiagent datasets** (ATE RMSE [cm]↓). Our foundation vision model-based loop detection mechanism generalizes better than Net-VLAD descriptors to unseen data.

runtime and memory usage against the most recent multi-agent neural SLAM system [10]. Specifically, we evaluate the time required for tracking, mapping, map merging, and peak VRAM consumption. Using Gaussian sub-maps for agent mapping and tracking significantly accelerates the pipeline and limits the VRAM required per agent. Additionally, our sub-map caching and merging strategies reduce the merging time.

Limitations. As shown in Table 8, MAGiC-SLAM is significantly faster than the previous state-of-the-art in tracking, mapping, and global map reconstruction. However, the system operates slightly faster than 1 FPS. The implicit tracking mechanism is accurate but requires many iterations to converge. Future work could investigate solutions that enable faster convergence without compromising the accuracy of the current pipeline.

Methods	Mapping Frame [s]	Tracking Frame [s]	Map Merging [s]	Peak GPU [GiB]
CP-SLAM [10]	16.95	3.36	1448	7.70
MAGiC-SLAM	**0.71**	**0.69**	**167**	**1.12**

Table 8. **Runtime and Memory Analysis on ReplicaMultiagent** `office0`. Mapping, tracking, and peak GPU consumption are computed per agent. Map merging is the time required to construct the global map from the agents' sub-maps. All metrics are computed using an NVIDIA RTX A6000 GPU.

5. Conclusion

We present MAGiC-SLAM, a multi-agent SLAM with novel view synthesis capabilities. Thanks to our tracking and loop closure modules, MAGiC-SLAM demonstrates superior tracking accuracy on both synthetic and real-world datasets. Our efficient map-merging strategy allows high-quality rendering in various scenarios. The Gaussian-based scene representation significantly reduces processing time, disk, and VRAM consumption compared to the previous state of the art. Finally, our method can handle a varied number of simultaneously operating agents.

Acknowledgements. This work was supported by TomTom, the University of Amsterdam, and the allowance of Top Consortia for Knowledge and Innovation (TKIs) from the Netherlands Ministry of Economic Affairs and Climate Policy.

References

[1] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition, 2016. 5, 8

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021. 2

[3] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 4, 5

[4] Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez, Jose M. M. Montiel, and Juan D. Tardos. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6): 1874–1890, 2021. 6, 7

[5] Yun Chang, Yulun Tian, Jonathan P. How, and Luca Carlone. Kimera-multi: a system for distributed multi-robot metric-semantic simultaneous localization and mapping. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11210–11218, 2021. 2, 3

[6] 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. *arXiv preprint arXiv:2209.13274*, 2022. 2

[7] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2024. 4

[8] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial intelligence review*, 43:55–81, 2015. 1, 3

[9] Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012. 5

[10] Jiarui Hu, Mao Mao, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cp-slam: Collaborative neural point-based slam system, 2023. 2, 3, 5, 6, 7, 8

[11] Jiarui Hu, Xianhao Chen, Boyin Feng, Guanglin Li, Liangjing Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cg-slam: Efficient dense rgb-d slam in a consistent uncertainty-aware 3d gaussian field, 2024. 2, 3

[12] Huajian Huang, Longwei Li, Cheng Hui, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photo-realistic mapping for monocular, stereo, and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2

[13] Iman Abaspur Kazerouni, Luke Fitzgerald, Gerard Dooly, and Daniel Toal. A survey of state-of-the-art on visual slam. *Expert Systems with Applications*, 205:117734, 2022. 1, 3

[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. *arXiv preprint*, 2023. 1, 2, 3, 4

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2

[16] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. 5

[17] Pierre-Yves Lajoie and Giovanni Beltrame. Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. *IEEE Robotics and Automation Letters*, 9(1):475–482, 2024. 2, 3, 6, 7

[18] Pierre-Yves Lajoie, Benjamin Ramtoula, Yun Chang, Luca Carlone, and Giovanni Beltrame. Door-slam: Distributed, online, and outlier resilient slam for robotic teams. *IEEE Robotics and Automation Letters*, 5(2):1656–1663, 2020. 3

[19] Lorenzo Liso, Erik Sandström, Vladimir Yugay, Luc Van Gool, and Martin R Oswald. Loopy-slam: Dense neural slam with loop closures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20363–20373, 2024. 2, 5

[20] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. *arXiv e-prints*, pages arXiv–2211, 2022. 2

[21] Hidenobu Matsuki, Keisuke Tateno, Michael Niemeyer, and Federic Tombari. Newton: Neural view-centric mapping for on-the-fly large-scale slam. *arXiv preprint arXiv:2303.13654*, 2023. 3

[22] 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*, 2024. 1, 2, 3, 6, 7

[23] 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. In *European Conference on Computer Vision (ECCV)*. CVF, 2020. 2

[24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2

[25] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. 5, 7

[26] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Carl Yuheng Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception, 2023. 6

[27] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 143–152, 2017. 4

[28] Zhexi Peng, Tianjia Shao, Liu Yong, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. In *ACM SIGGRAPH Conference Proceedings, Denver, CO, United States, July 28 - August 1, 2024*, 2024. 2

[29] Antoni Rosinol, John J. Leonard, and Luca Carlone. NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields. *arXiv*, 2022. 2

[30] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009. 5

[31] Erik Sandström, Yue Li, Luc Van Gool, and Martin R Oswald. Point-slam: Dense neural point cloud-based slam. In *International Conference on Computer Vision (ICCV)*. IEEE/CVF, 2023. 2, 6

[32] Patrik Schmuck and Margarita Chli. CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. In *Journal of Field Robotics (JFR)*, 2018. 2, 3, 6, 7

[33] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986. 2

[34] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2012. 6

[35] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. iMAP: Implicit Mapping and Positioning in Real-Time. In *International Conference on Computer Vision (ICCV)*. IEEE/CVF, 2021. 1, 2

[36] Fabio Tosi, Youmin Zhang, Ziren Gong, Erik Sandström, Stefano Mattoccia, Martin R. Oswald, and Matteo Poggi. How nerfs and 3d gaussian splatting are reshaping slam: a survey, 2024. 1, 3, 6

[37] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13293–13302, 2023. 2

[38] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[39] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting, 2024. 2, 3

[40] 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 *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 499–507. IEEE, 2022. 2

[41] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks, 2021. 2

[42] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting, 2024. 2, 3, 4, 5, 6, 7

[43] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[44] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. *arXiv preprint arXiv:2309.02436*, 2023. 2

[45] Liyuan Zhu, Yue Li, Erik Sandström, Shengyu Huang, Konrad Schindler, and Iro Armeni. Loopsplat: Loop closure by registering 3d gaussian splats, 2024. 2

[46] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022. 2, 8