SP-SLAM: Neural Real-Time Dense SLAM With Scene Priors

Zhen Hong, Bowen Wang, Haoran Duan, *Member IEEE*, Yawen Huang, Xiong Li, Zhenyu Wen, *Senior Member IEEE*, Xiang Wu, Wei Xiang, *Senior Member IEEE*, Yefeng Zheng, *Fellow IEEE*

***Abstract*—Neural implicit representations have recently shown promising progress in dense Simultaneous Localization And Mapping (SLAM). However, existing works have shortcomings in terms of reconstruction quality and real-time performance, mainly due to inflexible scene representation strategy without leveraging any prior information. In this paper, we introduce SP-SLAM, a novel neural RGB-D SLAM system that performs tracking and mapping in real-time. SP-SLAM computes depth images and establishes sparse voxel-encoded scene priors near the surfaces to achieve rapid convergence of the model. Subsequently, the encoding voxels computed from single-frame depth image are fused into a global volume, which facilitates high-fidelity surface reconstruction. Simultaneously, we employ tri-planes to store scene appearance information, striking a balance between achieving high-quality geometric texture mapping and minimizing memory consumption. Furthermore, in SP-SLAM, we introduce an effective optimization strategy for mapping, allowing the system to continuously optimize the poses of all historical input frames during runtime without increasing computational overhead. We conduct extensive evaluations on five benchmark datasets (Replica, ScanNet, TUM RGB-D, Synthetic RGB-D, 7-Scenes). The results demonstrate that, compared to existing methods, we achieve superior tracking accuracy and reconstruction quality, while running at a significantly faster speed.**

***Index Terms*—Dense Visual SLAM, Neural Implicit Representations, Sparse Voxel Encoding.**

I. Introduction

RECOVERING the camera motion trajectory and scene map from an image stream is a longstanding fundamental task in 3D computer vision, with widespread applications in many fields such as autonomous driving [1], robot navigation [2], virtual/augmented reality [3]. Visual Simultaneous Localization and Mapping (SLAM) systems are often used to solve this problem. Classical SLAM methods [4]–[6] extract feature points from consecutive image frames and perform accurate camera tracking based on the motion of these points, while constructing a scene map composed of sparse features. However, this SLAM methods are typically not ideal for domains that require high-fidelity representation of the scene surface, such as virtual/augmented reality and robotics applications. Some works [7]–[10] can create dense scene maps, but they often face a difficult trade-off between resolution and memory consumption, and are limited by a fixed resolution, always losing reconstruction details at smaller scales.

Recent advances in neural implicit representations—Neural Radiance Fields (NeRF) [11] have greatly inspired dense visual SLAM. Essentially, NeRF employs neural network architecture to directly encode the geometry and appearance information of 3D points in continuous scene space, thus enabling the extraction of geometry at any resolution without increasing memory consumption. In particular, iMAP [12] represents the entire scene as a multi-layer perceptron (MLP) and jointly optimizes scene representation and camera poses using the re-rendering losses. However, due to the limited expressive capacity of a single MLP, iMAP is only suitable for small-scale scenes and suffers from severe catastrophic forgetting. Subsequent works often substitute a single MLP with hybrid representation, which store trainable embeddings on explicit scene representation such as voxel grids [13], octrees [14], and tri-planes [15], and then model the scene geometry through implicit neural decoder. This hybrid representation improves the scalability of the system and mitigate catastrophic forgetting to some extent. However, these methods exhibit poor performance in terms of running speed. Existing SLAM systems based on NeRF typically follow the IMAP [12] framework, dividing the system into Tracking and Mapping processes. Under this paradigm, a significant amount of computational resources is consumed by the tracking iterative optimization run on each input frame and the mapping iterative optimization performed at regular intervals. Previous methods often set a large number of iterations for both tracking and mapping. On one hand, this is due to their lack of utilization of scene prior information. Specifically, they employ a fixed scene representation strategy, assuming that the optimizable scene embeddings are random values sampled from a normal distribution. This approach neglects the incorporation of scene prior knowledge, resulting in insufficient understanding of the scene by the system, which necessitates starting the optimization from scratch. Consequently, the system requires more mapping iterations to ensure accuracy (See Fig. 7). On the other hand, these methods adhere to the traditional SLAM paradigm, selecting a set of keyframes for continuous optimization during the mapping process, while the poses of non-keyframes are only iteratively optimized during their respective tracking processes. As a result, a greater number

Zhen Hong, Bowen Wang, Xiong Li, Zhenyu Wen, Xiang Wu are with the Institute of Cyberspace Security and College of Information Engineering, Zhejiang University of Technology, China. (E-mail: zhong1983@zjut.edu.cn, 221122120289@zjut.edu.cn, 492161180@qq.com, zhenyuwen@zjut.edu.cn, xiangwu@zjut.edu.cn)

Haoran Duan is with the Department of Computer Science, Durham University, UK. (E-mail: haoran.duan@ieee.org)

Yawen Huang and Yefeng Zheng are with Tencent Jarvis Lab, Shenzhen, China (E-mail: yawenhuang@tencent.com; yefengzheng@tencent.com).

Wei Xiang is with the School of Engineering and Mathematical Sciences, La Trobe University, Melbourne, VIC 3086, Australia (E-mail:w.xiang@latrobe.edu.au).

Zhenyu Wen and Haoran Duan are the corresponding authors.

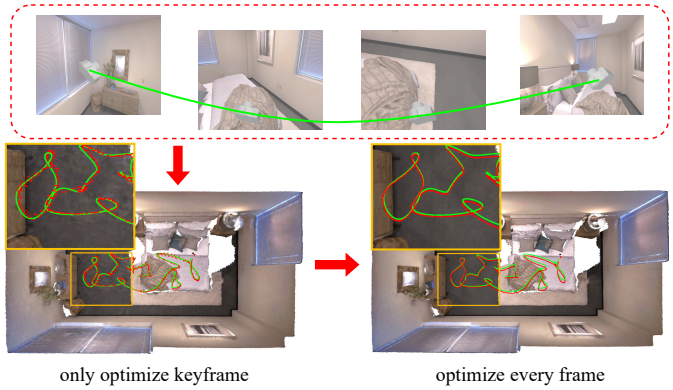


Fig. 1. The impact of mapping optimization strategies on our system. The green trajectory represents the ground truth camera motion, while the red trajectory represents the estimated camera motion. Compared to selecting a set of keyframes to maintain the scene map, optimizing each input frame using sparsely sampled pixels can achieve more robust camera tracking and more realistic scene reconstruction.

of tracking iterations for each frame are required to ensure reliable and accurate camera tracking (See Fig. 9), which not only significantly reduces the running speed of the system, but also limits tracking accuracy. For SLAM tasks, the real-time performance of the system is crucial. Therefore, the requirement for a large number of iterative optimizations is impractical from a time efficiency perspective.

Inspired by recent successful 3D reconstruction work [16], in this work, we propose SP-SLAM, which introduces scene prior information, i.e., signed distance field (SDF) priors for 3D points, within a framework of hybrid representation, aiming to achieve rapid model convergence while preserving the advantages of hybrid representation. Specifically, we back-project the depth map into a 3D point cloud. Utilizing the encoder pretrained on the ShapeNet dataset [17] from BNV-Fusion [16], we encode the SDF priors for each point as fixed-length embedding vectors, and aggregate them into local voxels. The core concept of this approach is to utilize existing depth information to initialize a sparse volume, which encodes scene priors. This sparse volume captures the fundamental structural features of the scene, providing the model with a preliminary understanding before the optimization process begins. As a result, the model is able to converge more rapidly with fewer mapping iterations. Meanwhile, to achieve texture mapping of geometry, we store scene appearance information on three axis-aligned planes, striking a good balance between texture quality and memory usage.

Furthermore, we introduce an efficient optimization strategy for mapping. We fully leverage the inherent capability of NeRF which enables tracking and mapping by calculating loss solely on a sparse set of pixels, no longer selecting keyframes, but sampling a small number of pixels from each input frame to maintain a pixel database in runtime. During the mapping process, we retrieve a set of pixels from the pixel database to optimize scene representation and the poses of corresponding frames. This optimization strategy enables SP-SLAM to continuously refine the pose of every input frame throughout all mapping processes. It not only improves tracking accuracy (See Fig. 1) but also allows the system to reduce the number of iterations per frame during tracking (See Fig. 9), thereby enhancing real-time performance. SP-SLAM obtains competitive performance comparing with representative works [13]–[15], [18], [19] on five benchmark datasets, Replica [20], Synthetic RGB-D [21], ScanNet [22], TUM RGB-D [23] and 7-Scenes [24]. In summary, we make the following contributions:

- We introduce scene priors into the dense SLAM task and design a novel neural RGB-D SLAM system, capable of real-time accurate camera tracking and high-fidelity surface reconstruction.
- We dispense with the concept of keyframes and introduce an effective optimization strategy for mapping that allows the system to perform ongoing pose refinement for each input frame throughout all mapping processes without adding additional computational load, improving real-time performance and achieving more accurate camera tracking.
- Our approach achieves superior tracking and mapping performance across various datasets, with significantly faster running speeds.

II. RELATED WORK

A. Visual SLAM

Visual SLAM methods can be mainly categorized into two types based on the reconstructions of the scene map: sparse and dense. Sparse visual SLAM methods [4]–[6], [25], [26] mainly focus on recovering accurate camera motion trajectories. These methods typically use feature points or keypoints to represent the environment and estimate camera poses, generating coarse scene maps. However, in some application domains, such as robotics, virtual/augmented reality, there is often a need for globally consistent dense reconstruction. Existing dense visual SLAM methods can produce detailed geometric information for reconstructing scenes. They typically represent the scene as explicit surfels [7], [8], [27], [28] or volume [9], [10], [29] and store geometric information. However, these methods struggle to strike a balance between resolution and memory usage, and they are constrained by fixed resolutions, leading to a loss of reconstruction details at finer scales. Additionally, during the tracking process, they often estimate local poses only through motion estimation between adjacent frames, making them susceptible to cumulative estimation errors, leading to camera drift issues. Although BAD-SLAM [27] and BundleFusion [10] perform global optimization of camera trajectories through bundle adjustment to reduce error accumulation, due to computational complexity considerations, they can only optimize the poses of keyframes. As illustrated in Fig. 1, it demonstrates the limitations on tracking accuracy imposed by optimizing only keyframes during the bundle adjustment process. Recently, some methods [18], [30]–[35] have introduced deep learning into SLAM systems, eliminating the need for handcrafted feature extraction by optimizing end-to-end loss functions to learn the required features and representations from input data. Compared to traditional SLAM methods, learning-based SLAM systems

typically exhibit better accuracy and robustness. However, they still share similarities with traditional SLAM methods in terms of overall frameworks and global bundle adjustment strategies.

B. NeRF-based SLAM

Neural Radiance Field (NeRF) [11] is an innovative 3D representation method that employs neural network architecture to directly encode the geometry and appearance information of 3D points in continuous scene space, enabling scene modeling at arbitrary resolutions without increasing memory consumption. Recently, NeRF has shown promising results in tasks such as novel view synthesis [36]–[39], object-level reconstruction [40]–[42], and large-scale scene reconstruction [16], [21], [43]–[46]. These methods require pre-recovery of camera motion trajectories from input images, posing difficulties in their application in unknown environments. Some works [47]–[49] attempt view synthesis and scene reconstruction without the input of camera poses, demonstrating that camera poses can be optimized as learnable parameters through re-rendering losses. However, a common characteristic they share with NeRF is the substantial time required for optimization, making real-time applications challenging. Subsequent works have accelerated training by explicitly storing scene parameters as learnable parameters in voxel grids [50]–[53] or octrees [54], [55], and utilizing tinier MLP decoders. Based on these techniques, some NeRF-based SLAM methods [12]–[15]have been proposed, demonstrating advantages in generating high-precision maps. iMAP [12] combines NeRF for the first time in performing tracking and mapping, but is constrained by the limited expressive capacity of a single MLP, making it unsuitable for large-scale scenes. NICE-SLAM [13] employs a multi-level feature grid to encode the scene, improving system scalability. Vox-Fusion [14] and ESLAM [15] adopt octree and tri-planes, respectively, to represent the scene. They utilize SDF rather than occupancy for modeling scene geometry, thereby enhancing the mapping capabilities of the system. However, these SLAM methods employ a fixed scene representation strategy for all scenes without introducing any prior information. This results in slow model convergence, rendering them unsuitable for real-time applications. Moreover, they still adhere to the traditional SLAM paradigm of selecting a set of keyframes to maintain the scene map and perform global optimization of camera poses, facing the same limitations in tracking accuracy as traditional SLAM. In our proposed approach, we dynamically construct sparse voxels encoding scene priors based on depth image information to achieve rapid convergence of the model, enhancing the real-time performance of the system. Additionally, we leverage the property of the loss function of NeRF-based SLAM to only operate on sparse pixels in the image, no longer relying on keyframes. Specifically, we sample a few pixels on each input frame and add them to a global pixel database for bundle optimization. In this way, our method can optimize the camera poses of all historical input frames throughout all mapping processes, achieving more accurate camera tracking while avoiding information redundancy and an increase in computational load. Concurrent to our work, Co-SLAM [56] accelerates training by combining coordinate encoding with multi-resolution hash encoding. GO-SLAM [19] extends DROID-SLAM [18] for online loop closure detection and global bundle adjustment and integrates it with a map via Instant-NGP [53] for instant mapping.

III. METHOD

The overview of our system is illustrated in Fig. 2. Given an RGB-D image input stream, our system estimates the camera pose for each frame and generates a scene map. Specifically, a depth encoder extracts local geometric priors from the depth image and fuses them into a global sparse volume. The appearance features of the scene are stored on three axis-aligned planes. Any point x in the three-dimensional world coordinate system is mapped to the sparse volume and the tri-planes, where interpolated features are decoded into color c_x and truncated signed distance field (TSDF) s_x by two shallow MLPs. We sample a certain number of pixels from each input frame and add them to a pixel database. Tracking is performed on each frame, optimizing the camera pose through a small number of iterations. Mapping is carried out after tracking a fixed number of frames. We select optimized frames, and retrieve pixels from the pixel database to jointly optimize the camera pose of the corresponding frames and the hybrid scene representation.

A. Depth Encoding and Fusion

SP-SLAM extracts scene geometric priors from M input depth images. For a depth map I_m, $m \in \{1, ..., M\}$, we initially perform a back-projection, converting it into a 3D point cloud in the world coordinate system based on the corresponding estimated pose. Subsequently, our method utilizes an encoder to process the point cloud, encoding the geometric priors at the corner vertices of the local voxels where 3D points reside, thereby generating a collection of encoded voxels denoted as V_m. The encoder is a 3-layer fully connected (FC) network with 64 nodes in the hidden layer, and its pre-trained weights are derived from [16]. These geometric priors are represented as 8 dimensional, trainable embedding vectors. When the camera is in movement, we continuously fuse local encoding voxels into a global volume V_g, expressed as

$$V_g = \frac{V_g * W_g + V_m * W_m}{W_g + W_m}, \quad W_g = W_g + W_m, \tag{1}$$

where V represents the 8-dimensional trainable embedding vector for each voxel, and W represents the weight of the voxel, which depends on the voxel itself and the points contained within a neighborhood.

Our geometric scene representation consists of V_g and a shallow MLP decoder F_Θ^g with trainable parameters Θ. For any point $p \in \mathbf{R}^3$ within the volume V_g, we aggregate the information of the eight vertices of the voxel where it is located for trilinear interpolation to query the embedding vector of p. Afterwards, the embedding vector is interpreted by the decoder F_Θ^g as TSDF s, i.e.,

$$s = F_\Theta^g(p, TriLerp(p, V_g)), \tag{2}$$

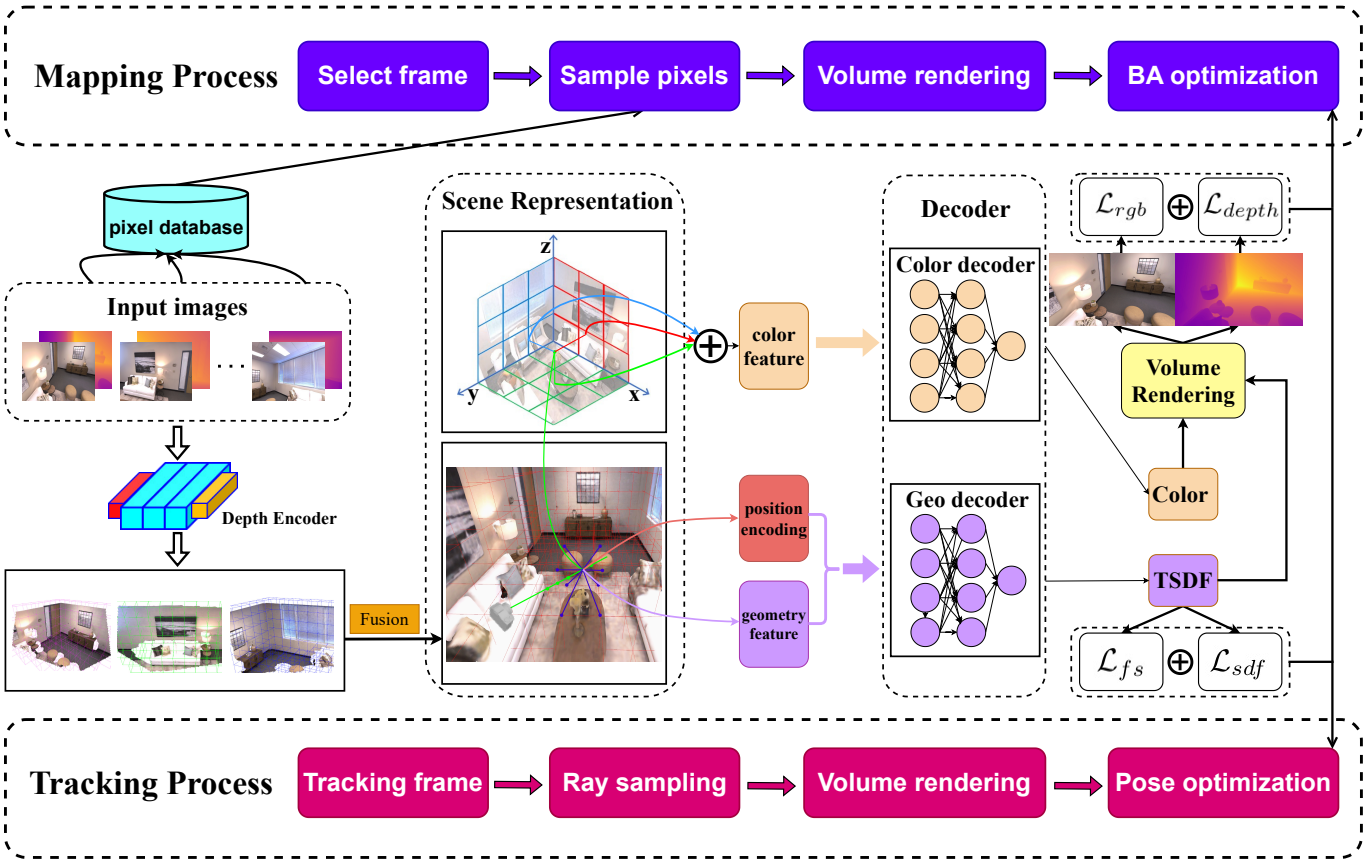


Fig. 2. Overview of SP-SLAM. The depth encoder extracts local geometric priors from the depth image and fuses them into a global sparse volume. Our hybrid scene representation consists of sparse volumes representing geometry, three planes representing appearance, and two shallow MLP decoders. We calculate the rays emitted from the camera and sample them layer by layer based on the estimated camera pose, and then predict the color and TSDF of each sampling point through our scene representation. Volume rendering predicts the color and depth of rays (Sec. III-B). The overall objective function consists of re-rendering losses and geometric losses (Sec. III-C). The tracking process optimizes the camera pose of the current frame by minimizing the overall objective function, while the mapping process jointly optimizes the camera pose and scene representation of the selected frame.

where $TriLerp(\cdot,\cdot)$ represents the trilinear interpolation function and F_Θ^g uses a tanh activation function at the output layer to map s to $[-1,1]$.

B. Color and Depth Rendering

Feature Tri-plane. In our system, we construct sparse encoded voxels exclusively in the vicinity of scene surfaces to reconstruct geometry, eschewing the allocation of voxels in free space, which is typically devoid of meaningful geometric information. Nevertheless, this approach falls short for texture mapping, as color information is continuous. The color of each pixel in an RGB image is influenced by the cumulative colors of all 3D points intercepted by the rays emanating from the camera. Consequently, to precisely represent color information, voxel grids must extend across the full scene space traversed by these rays, resulting in a cubic memory consumption growth. To overcome this issue, we deposit trainable color features onto three axis-aligned feature planes [57], denoted as $\Omega : \{\Omega_{xy}, \Omega_{xz}, \Omega_{yz}\}$, which serve to simulate the functionality of three-dimensional voxels. This technique curtails memory from cubic to quadratic growth, allowing the resolution of the feature planes to match that of the sparse volume representing geometry, which in turn improves the fidelity of the textures. In practice, to retrieve the color of a point p in the scene, we first project it onto three feature planes and subsequently apply bi-linear interpolation to get the corresponding features $f_{xy}(p), f_{xz}(p)$ and $f_{yz}(p)$. The color feature $f(p)$ of point p is computed by a straightforward summation of them as

$$f(p) = f_{xy}(p) + f_{xz}(p) + f_{yz}(p). \tag{3}$$

Finally, we interpret the color feature of point p as raw color via a color decoder F_θ^c with trainable parameters θ as

$$c_p = F_\theta^c(f(p)). \tag{4}$$

Rendering. For any pixel in the current input frame, the direction $\mathbf{d}$ of its corresponding emitted ray r can be computed using the estimated camera pose of that frame. To render color and depth of the ray/pixel, we need to sample along the ray. We pre-filter out pixels without ground truth depths to ensure that the ray has a vaild depth measurement D, allowing us to use depth value to guide ray sampling. Specifically, we sample a total of N points along the ray as $p_i = \mathbf{o} + z_i\mathbf{d}$, $i \in \{1, ..., N\}$, where $\mathbf{o}$ represents the camera center, and z_i is the depth of point along the ray. These N points include N_c points uniformly sampled from the interval $[near, far]$ and

N_f points sampled near the depth within a truncation distance Tr, where $near = n_1 * D$ and $far = n_2 * D$. n_1, n_2 are hyper-parameters that control the distance between the start and end points of light rays and the surface. For all sampling points $\{p_1, ..., p_N\}$, we map them to the volume V_g and the tri-plane Ω to predict their TSDF $\{s_1, ..., s_N\}$ and color $\{c_1, ..., c_N\}$. We use volume rendering technique to calculate the color and depth of the ray/pixel by performing a weighted summation of the color and depth values from all the sample points along the ray, as

$$\hat{C} = \sum_{i=1}^{N} w_i c_i, \quad \hat{D} = \sum_{i=1}^{N} w_i z_i, \tag{5}$$

where w_i is the weight, representing the termination probability of the ray at the sampling point. We use the bell shaped function proposed by [31] to calculate w_i as

$$w_i = \sigma(\frac{s_i}{tr}) \cdot \sigma(-\frac{s_i}{tr}), \tag{6}$$

where tr is truncation distance and σ is the sigmoid function.

C. Optimization

In this subsection, we aim to optimize the hybrid scene representation $\{\Theta, \theta, V_g, \Omega\}$ and camera parameters γ through minimizing the overall objective function:

$$\min_{\{\Theta,\theta,V_g,\Omega,\gamma\}} \lambda_{rgb}\mathcal{L}_{rgb} + \lambda_{depth}\mathcal{L}_{depth} + \lambda_{fs}\mathcal{L}_{fs} + \lambda_{sdf}\mathcal{L}_{sdf}, \tag{7}$$

where $\mathcal{L}_{rgb}$ and $\mathcal{L}_{depth}$ are the re-rendering losses for optimizing appearance representation. $\mathcal{L}_{fs}$ and $\mathcal{L}_{sdf}$ are the free-space loss and SDF loss for optimizing geometric representation. $\{\lambda_{rgb}, \lambda_{depth}, \lambda_{fs}, \lambda_{sdf}\}$ are their weight coefficients.

We sample M pixels with ground-truth depths and calculate their corresponding rays. As described in Sec. III-B, we sample a set of N points on each ray, denoted as $P_m = \{p_1, ..., p_N\}$, $m \in \{1, ..., M\}$. Then we calculate their rendered color $\{\hat{C}_1, ..., \hat{C}_M\}$ and depth $\{\hat{D}_1, ..., \hat{D}_M\}$ through Eq. 5. The re-rendering losses are composed of color loss and depth loss, which are defined as the mean squared error between the rendered values and the observed values:

$$\mathcal{L}_{rgb} = \frac{1}{|M|} \sum_{m=1}^{M} (\hat{C}_m - C_m)^2, \tag{8}$$

$$\mathcal{L}_{depth} = \frac{1}{|M|} \sum_{m=1}^{M} (\hat{D}_m - D_m)^2, \tag{9}$$

where C_m and D_m are the corresponding observed color and depth values, respectively.

For sampled points which located within the truncation region near the surface on the ray, denoted as P_m^{tr}, we use depth observations to calculate approximate SDF values for supervision to learn scene surface shapes as:

$$\mathcal{L}_{sdf} = \frac{1}{|M|} \sum_{m=1}^{M} \frac{1}{|P_m^{tr}|} \sum_{p \in P_m^{tr}} (s_p - (D_m - z_p))^2, \tag{10}$$

where z_p represents the depth of point p along the ray.

For sampled points which located between the camera center and the truncation region, denoted as P_m^{fs}, we use free-space loss to force SDF prediction of these points to approach the pre-defined truncation distance tr:

$$\mathcal{L}_{fs} = \frac{1}{|M|} \sum_{m=1}^{M} \frac{1}{|P_m^{fs}|} \sum_{p \in P_m^{fs}} (s_p - tr)^2. \tag{11}$$

D. End-to-End Tracking and Mapping

We follow the framework proposed by iMAP [12], dividing the system into tracking and mapping processes. The tracking is performed on each frame, while mapping is performed at fixed frame intervals.

Tracking. In the absence of ground truth camera pose information, we initialize the initial camera pose with the identity matrix. For the subsequent input frame k, we use a constant-speed motion model to initialize its pose T_k. The 4×4 transformation matrix T_k for the camera-to-world transformation includes a 3x3 rotation matrix R_k and a 3-dimensional translation vector $\mathbf{t}_k$, describing the camera's orientation and position in space, respectively, i.e.,

$$T_k = \begin{bmatrix} R_k & \mathbf{t}_k \\ \mathbf{0} & 1 \end{bmatrix}, \; R_k = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \; \mathbf{t}_k = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}. \tag{12}$$

We convert R_k to a quaternion and combine it with $\mathbf{t}_k$ to form a with seven degrees of freedom (DoF) vector γ_k. During the I_t iterations of the optimization process, the parameters of geometric embeddings, appearance embeddings, and their corresponding decoders are kept fixed. In each iteration, we randomly sample M_t pixels from tracked frame k and perform ray sampling and volume rendering as described in Sec. III-B. The optimization of camera parameters γ_k is performed by minimizing the overall objective function. We record the minimum value of the overall objective function and its corresponding optimized camera pose over these I_t iterations, which serves as the final optimization result. After each tracking is completed, we sample M_k pixels from the current frame to maintain a runtime pixel database.

Mapping. During the mapping process, we select a total of 200 frames for optimization, including the most recent 20 frames, 90 frames randomly chosen from those with a co-visibility area greater than 10% with the current frame, and an additional 90 frames randomly selected from all past frames to avoid catastrophic forgetting. For each frame, we randomly retrieve $M_p/200$ pixels from the pixel database and calculate the corresponding rays. In other words, we retrieve a total of M_p rays, and during the I_m iterations, we jointly optimize the scene representation and the camera poses of these 200 frames.

IV. EXPERIMENTS

In this section, we firstly describe the experimental setup, and then report the comparative evaluation results of SP-SLAM and baselines in terms of tracking accuracy, reconstruction quality, and running time. In addition, we report

TABLE I
QUANTITATIVE COMPARISON IN RECONSTRUCTION, RUNTIME, AND MODEL SIZE ON REPLICA DATASET [20] AND SYNTHETIC RGB-D DATASET [21]. THE TIME FOR TRACKING AND MAPPING IS REPORTED AS THE TIME OF EACH ITERATION × NUMBER OF ITERATIONS. VOX-FUSION RUNS MAPPING ON EACH FRAME, ESLAM RUNS MAPPING EVERY FOUR FRAMES, WHILE OTHER METHODS RUN MAPPING EVERY FIVE FRAMES. THE AVG. FPS IS CALCULATED BY DIVIDING THE TOTAL RUNTIME OF THE SYSTEM BY THE NUMBER OF FRAMES. THE MODEL SIZE IS THE AVERAGE OF EIGHT SCENES.

		Reconstruction (%)			Runtime (ms)				Memory Usage	
Datasets	Method	Comp.↑	Accu.↑	F1.↑	Track.↓	Map.↓	Track. FPS↑	Avg. FPS↑	#param (MB)↓	GPU (GB)↓
Replica [20]	NICE-SLAM [13]	94.47	97.33	95.90	7.6×10	71.4×60	13.15	1.12	48.5 MB	8 GB
	Vox-Fusion [14]	97.73	88.62	93.17	15.8×30	46.0×15	2.11	1.67	**0.15 MB**	5 GB
	ESLAM [15]	98.83	**99.13**	**98.98**	7.2×8	17.8×15	17.36	1.58	27.2 MB	9 GB
	DROID-SLAM [18]	47.58	23.16	35.37	-	-	-	**17.85**	15.3 MB	12 GB
	GO-SLAM [19]	84.58	89.98	87.28	-	-	-	8.64	63.4 MB	15 GB
	Ours	**99.06**	98.14	98.60	**6.7×4**	**10.5×20**	**37.31**	11.05	26.8 MB	**4 GB**
Synthetic [21]	NICE-SLAM [13]	81.12	80.04	80.58	12.6×10	77.5×60	8.47	<1	13.8 MB	8 GB
	Vox-Fusion [14]	**86.92**	80.83	83.88	16.6×30	46.2×15	2.01	1.48	**0.06 MB**	6 GB
	ESLAM [15]	84.74	71.81	78.28	6.7×8	25.3×15	18.66	2.21	21.2 MB	6 GB
	DROID-SLAM [18]	80.21	55.84	68.03	-	-	-	**20.12**	15.3 MB	12 GB
	GO-SLAM [19]	46.27	60.76	53.52	-	-	-	10.51	63.4 MB	15 GB
	Ours	83.65	**95.62**	**89.64**	**6.1×4**	**9.8×20**	**40.98**	11.58	15.1 MB	**3 GB**

a detailed analysis of the system's performance. Finally, we conduct extensive ablation experiments to demonstrate the effectiveness of our proposed strategy for mapping and system components.

A. Experimental Setup

a) Baselines: We select three representative neural RGB-D SLAM methods as our baselines, namely NICE-SLAM [20], Vox-Fusion [14] and ESLAM [15]. For Vox-Fusion, in addition to the results reported in the original paper [14], we also run its officially released code and reproduce results as Vox-Fusion*. Additionally, we select two deep learning-based SLAM methods, DROID-SLAM [18] and GO-SLAM [19], as baselines. Both DROID-SLAM and GO-SLAM are run in RGB-D mode during the experiment. The reconstruction results of DROID-SLAM are obtained with TSDF-Fusion [58].

b) Datasets: We evaluate on five datasets. The Replica dataset [20] contains several highly realistic synthetic 3D indoor scenes and provides motion trajectories for RGB-D sensors. We follow the previous work and select eight scene sequences for evaluation. The Synthetic RGB-D dataset [21] includes several synthetic scenes with simulated noisy depth maps and camera motion trajectories. We select six scene sequences for evaluation. In addition, we also benchmark on three real-world datasets with low-resolution images, ScanNet [22], TUM RGB-D [23] and 7-Scenes [24]. All of these datasets exhibit significant depth noise and severe motion blur. We select six scenes from the ScanNet dataset, three scenes from the TUM-RGBD dataset, and all scenes from the 7-Scenes dataset for evaluation. The ground truth camera trajectories for ScanNet were recovered using BundleFusion [10], while the ground truth camera trajectories for TUM RGB-D were captured using an external motion capture system.

c) Evaluation Metrics: For evaluation of reconstruction quality, we consider $Accuracy$ (Accu.), $Completeness$ (Comp.) and $F1\ score$ (F1). We sample N_p and N_q points from both the reconstructed mesh and the ground truth mesh (in our experimental setting, $N_p = N_q = 100,000$). $Accuracy$ measures the percentage of points among N_q that have a distance less than 5 cm to their nearest point among N_p. $Completeness$ measures the percentage of points among N_p that have a distance less than 5 cm to their nearest point among N_q. $F1\ score$ reflects the overall reconstruction quality, which is defined as the harmonic mean of $Accuracy$ and $Completeness$. For evaluation of camera tracking accuracy, we consider ATE proposed in [23], which reflects the translation error between the estimated pose and the ground truth camera pose.

d) Implementation Details: We run our system on a PC with a 3.8GHz AMD Ryzen 7 5800X CPU and an NVIDIA RTX 3090ti GPU. Our system are performed with the following default settings: The voxel has a size of $0.04m$, and the side length of the feature tri-plane is $0.04m$. The truncation distance $tr = 0.08m$. Along each ray, $N_c = 32$ and $N_f = 11$ sampling points are taken, while the hyperparameters for the start and end values of ray sampling are set as $n_1 = 0.2$ and $n_2 = 1.02$, respectively. During the tracking process, we randomly sample $M_t = 1024$ pixels, and set the learning rates of rotation quaternion and translation vector to 0.001 and 0.002, respectively. After that, we sample $M_k = 15000$ pixels from the current frame and add them to the runtime pixel database. Mapping is performed every five frames, with $M_p = 2048$ sampled pixels. The weight coefficients of the overall objective function are set as $\lambda_{depth} = 0.1$, $\lambda_{rgb} = 10$, $\lambda_{fs} = 20$, $\lambda_{sdf} = 1000$. The geometry decoder and color encoder both have a 2-layer MLP with 32 nodes in the hidden layer. In the output layer, the tanh activation function is used for the geometry decoder, and the sigmod activation function is used for the color decoder. The geometry decoder and color encoder both have a 2-layer MLP with 32 nodes in the hidden layer. In the output layer, the tanh activation function is used for the geometry decoder, and the sigmod activation function is used for the color decoder. The learning rates for learnable embeddings of sparse voxels, feature tri-planes, decoder parameters, and camera parameters are set to 0.004, 0.004, 0.001, and 0.001, respectively. For the Replica dataset,

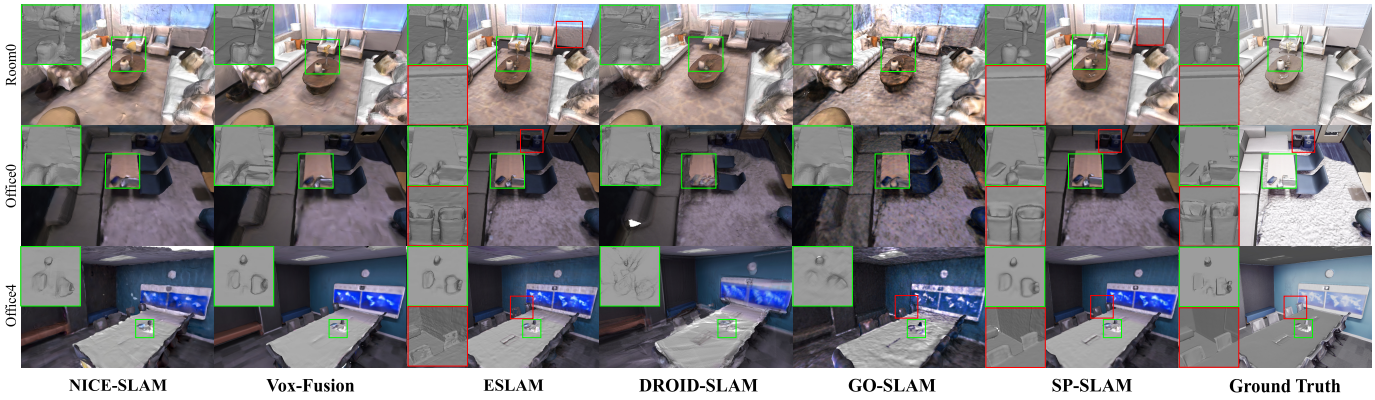


Fig. 3. Qualitative comparison in reconstruction on Replica dataset [20]. The region highlighted by the green rectangle showcases the higher fidelity of our geometry, and the region highlighted by the red rectangle demonstrates that our method is capable of generating smoother surfaces.

we set $I_t = 4$ tracking iterations and $I_m = 20$ mapping iterations. For the ScanNet dataset, the tracking iterations $I_t = 6$ and the mapping iterations $I_m = 20$. Regarding the TUM RGBD dataset, the tracking iterations $I_t = 6$, and the mapping iterations $I_m = 30$.

B. Evaluation Results on Replica dataset

We evaluate reconstruction quality and camera tracking on the Replica dataset [20]. Due to both NICE-SLAM [13] and ESLAM [15] assuming dense scene representation at full space resolution, they generate surfaces in unobserved areas. While they can build continuous scene maps, these approaches face two issues. Firstly, the generated surfaces often deviate significantly from reality in large unobserved regions, providing reasonable surface predictions only in the presence of smaller gaps. Secondly, the dense representation will produce numerous artefacts outside the scene, requiring significant time for mesh culling. In contrast, we only establish sparse voxels near the observed surface and do not consider geometric representation in unobserved areas. As mentioned in Vox-Fusion [14], for many real-world tasks, understanding which areas remain unobserved is often more important than generating predictions that diverge from reality.

To ensure fairness in the evaluation, we remove unobserved surfaces and only evaluate the reconstruction quality of the observed area. We provide quantitative reconstruction evaluation and system runtime of the Replica dataset in Tab. I, and the results are the average of five runs on eight scene sequences. It is worth noting that ESLAM spends a significant amount of time extracting scene meshes, which greatly reduces its average FPS. Our method surpasses NICE-SLAM and Vox-Fusion in reconstruction accuracy, performs on par with ESLAM, and is significantly faster than all of them. DROID-SLAM has the fastest runtime, and the running speed of GO-SLAM is comparable to ours, but both have low reconstruction accuracy. Fig. 3 shows the qualitative reconstruction results of three scenes. DROID-SLAM and GO-SLAM exhibit significant scene detail loss and overly rough surfaces. Our method recovers more detailed and high-fidelity geometric structures (note the position marked by the green rectangle in the image) while also generating smoother surfaces (note the position marked by the red rectangle in the image). Furthermore, we evaluate camera tracking on Replica. As shown in Tab. II, despite fewer tracking iterations, our tracking performance still surpasses several existing neural SLAM methods, thanks to our mapping optimization strategy, which continuously refines the pose of each input frame.

TABLE II
QUANTITATIVE COMPARISON IN TRACKING PERFORMANCE ON REPLICA DATASET [20]. THE NUMBERS FOR NICE-SLAM ARE TAKEN FROM [15]. ALL RESULTS ARE THE AVERAGES OF FIVE RUNS PER SCENE.

Method	Rm0	Rm1	Rm2	Of0	Of1	Of2	Of3	Of4	Avg.
NICE-SLAM [13]	1.69	2.13	1.87	1.26	0.84	1.71	3.98	2.82	2.05
Vox-Fusion [14]	0.40	0.54	0.54	0.50	0.46	0.75	0.50	0.60	0.54
Vox-Fusion* [14]	0.58	1.11	0.81	17.79	0.91	0.88	0.70	0.86	2.95
ESLAM [15]	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63	0.63
Droid-SLAM [18]	**0.45**	**0.31**	**0.38**	**0.32**	**0.34**	**0.41**	0.54	0.51	**0.41**
GO-SLAM [19]	0.85	0.49	0.46	0.39	0.50	0.48	0.63	0.70	0.56
Ours	0.50	0.66	0.45	0.48	0.45	0.56	**0.53**	**0.49**	0.52

TABLE III
QUANTITATIVE COMPARISON IN TRACKING PERFORMANCE ON SYNTHETIC RGB-D DATASET [21](ATE RMSE ↓ [CM]). THE RESULTS ARE THE AVERAGE OF FIVE RUNS ON EACH SCENE.

Scene ID	b.r.	c.k.	g.r.	g.w.r	m.a.	w.r.	Avg.
NICE-SLAM [13]	1.61	2.38	1.59	1.72	1.26	6.63	2.49
Vox-Fusion [14]	1.72	2.79	2.69	2.48	1.99	2.06	2.29
ESLAM [15]	2.92	4.17	1.72	2.02	2.91	3.68	2.90
DROID-SLAM [18]	1.12	**1.58**	**0.43**	**1.23**	**0.47**	**0.94**	**0.96**
GO-SLAM [19]	2.06	2.36	1.24	2.61	1.12	1.41	1.80
Ours	**1.09**	1.95	1.18	1.53	0.79	1.89	1.41

C. Evaluation Results on Synthetic RGB-D dataset

In Tab. I, we present a quantitative analysis of reconstruction and runtime on the Synthetic RGB-D dataset [21]. The experimental parameters for all methods on the Synthetic RGB-D dataset are consistent with those used on the Replica dataset [20]. Our method achieves the best reconstruction accuracy at real-time speed (12fps). Fig. 4 shows the qualitative

reconstruction results for three scene sequences. Compared to the baseline methods, SP-SLAM produces more detailed and precise geometric structures with fewer artifacts. Tab. III presents the quantitative results of camera tracking. Overall, SP-SLAM demonstrates competitive tracking accuracy, outperforming previous neural SLAM methods and slightly trailing behind the learning-based method DROID-SLAM. Notably, DROID-SLAM focuses on camera tracking but is unable to perform dense map reconstruction.

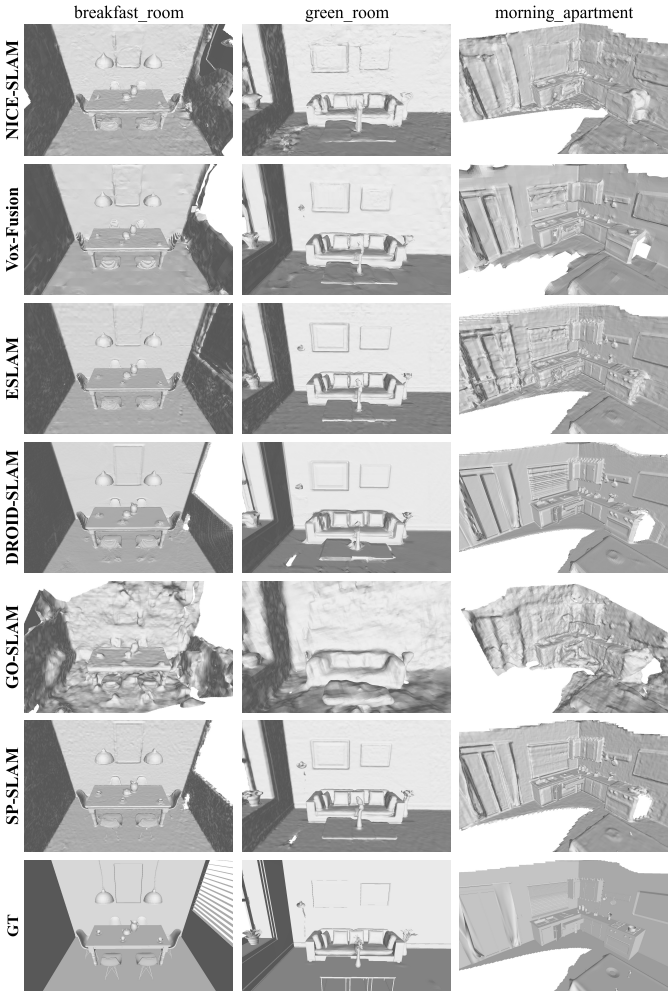


Fig. 4. Qualitative comparison in reconstruction on Synthetic RGB-D dataset [21]. Our method can produce clearer and more detailed geometric structures.

D. Evaluation Results on ScanNet dataset

To validate the robustness of the system, we also benchmark our method and baselines on real-world dataset ScanNet [22], which contains low-resolution images and severe motion blur. Tab. IV and Tab. V present the runtime on ScanNet and quantitative camera tracking results for six selected scenes, respectively. We achieve an improvement of more than 10 times in total runtime compared to NICE-SLAM, Vox-Fusion, and ESLAM. At the same time, on average, we achieve the second best camera tracking performance with fewer tracking and mapping iterations, which mainly benefits from the fusion of multi-frame observation information and the optimization

TABLE IV
QUANTITATIVE COMPARISON IN RUNTIME ON SCANNET DATASET [22], TUM-RGBD DATASET [23] AND 7-SCENES DATASET [24]. THE TIME FOR TRACKING AND MAPPING IS REPORTED AS THE TIME OF EACH ITERATION × NUMBER OF ITERATIONS. ON THE SCANNET DATASET AND 7-SCENES DATASET, VOX-FUSION RUNS MAPPING ON EVERY FRAME, ESLAM RUNS MAPPING EVERY FOUR FRAMES, WHILE OTHER METHODS RUN MAPPING EVERY FIVE FRAMES. ON THE TUM RGB-D DATASET, BOTH VOX-FUSION AND ESLAM RUN MAPPING ON EACH FRAME, WHEREAS OTHER METHODS RUN MAPPING EVERY FIVE FRAMES.

	Method	Track. (ms)↓	Map. (ms)↓	Track. FPS↑	Avg. FPS↑
ScanNet	NICE-SLAM [13]	11.7×50	114.5×60	1.71	<1
	Vox-Fusion [14]	28.6×30	85.2×15	1.17	<1
	ESLAM [15]	7.7×30	23.2×30	4.33	1.06
	DROID-SLAM [18]	-	-	-	**19.01**
	GO-SLAM [19]	-	-	-	8.32
	Ours	**6.8×6**	**10.9×20**	**24.51**	9.69
TUM	NICE-SLAM [13]	44.6×200	180.4×60	0.11	<1
	Vox-Fusion [14]	29.3×30	87.5×30	1.14	<1
	ESLAM [15]	11.7×200	28.9×60	0.43	<1
	DROID-SLAM [18]	-	-	-	**12.31**
	GO-SLAM [19]	-	-	-	8.33
	Ours	**6.8×6**	**10.9×30**	**24.51**	8.14
7-Scene	NICE-SLAM [13]	11.3×50	106.9×60	1.77	<1
	Vox-Fusion [14]	28.3×30	84.9×30	1.18	<1
	ESLAM [15]	7.3×30	25.9×30	4.56	1.18
	DROID-SLAM [18]	-	-	-	**21.64**
	GO-SLAM [19]	-	-	-	12.05
	Ours	**6.5×6**	**10.4×20**	**25.64**	9.92

TABLE V
QUANTITATIVE COMPARISON IN TRACKING PERFORMANCE ON SCANNET DATASET [22](ATE RMSE ↓ [CM]). THE RESULTS ARE THE AVERAGE OF FIVE RUNS ON EACH SCENE.

Scene ID	0000	0059	0106	0169	0181	0207	Avg.
NICE-SLAM [13]	8.64	12.25	8.09	10.28	12.93	5.59	9.63
Vox-Fusion [14]	8.39	9.18	7.44	6.53	12.20	5.57	8.65
Vox-Fusion* [14]	15.99	9.16	8.21	9.84	16.29	7.73	11.21
ESLAM [15]	7.31	8.51	7.51	6.51	9.01	5.71	7.43
DROID-SLAM [18]	7.63	7.49	7.86	8.08	10.92	6.28	8.04
GO-SLAM [19]	5.35	7.52	**7.03**	7.74	**6.84**	**5.29**	**6.63**
Ours	**5.27**	**7.11**	8.02	**5.99**	11.04	6.35	7.30

strategy for mapping (see Sec. IV-G for more details). While GO-SLAM demonstrates impressive tracking results, it pays less attention to scene reconstruction. Despite using neural implicit methods to construct the scene map, GO-SLAM perform only minimal iterative optimization due to runtime considerations, which leads to extremely coarse and less smooth surfaces. As shown in the geometric reconstruction results in Fig. 5, we observe that SP-SLAM can produce smoother and higher-fidelity scene surfaces with more detailed geometric structures.

E. Evaluation Results on TUM RGB-D dataset and 7-Scenes dataset

We compare the runtime and tracking performance of SP-SLAM with existing methods on the TUM RGB-D dataset [23]. Due to the absence of configuration files for running the official release code of Vox-Fusion on the TUM RGBD

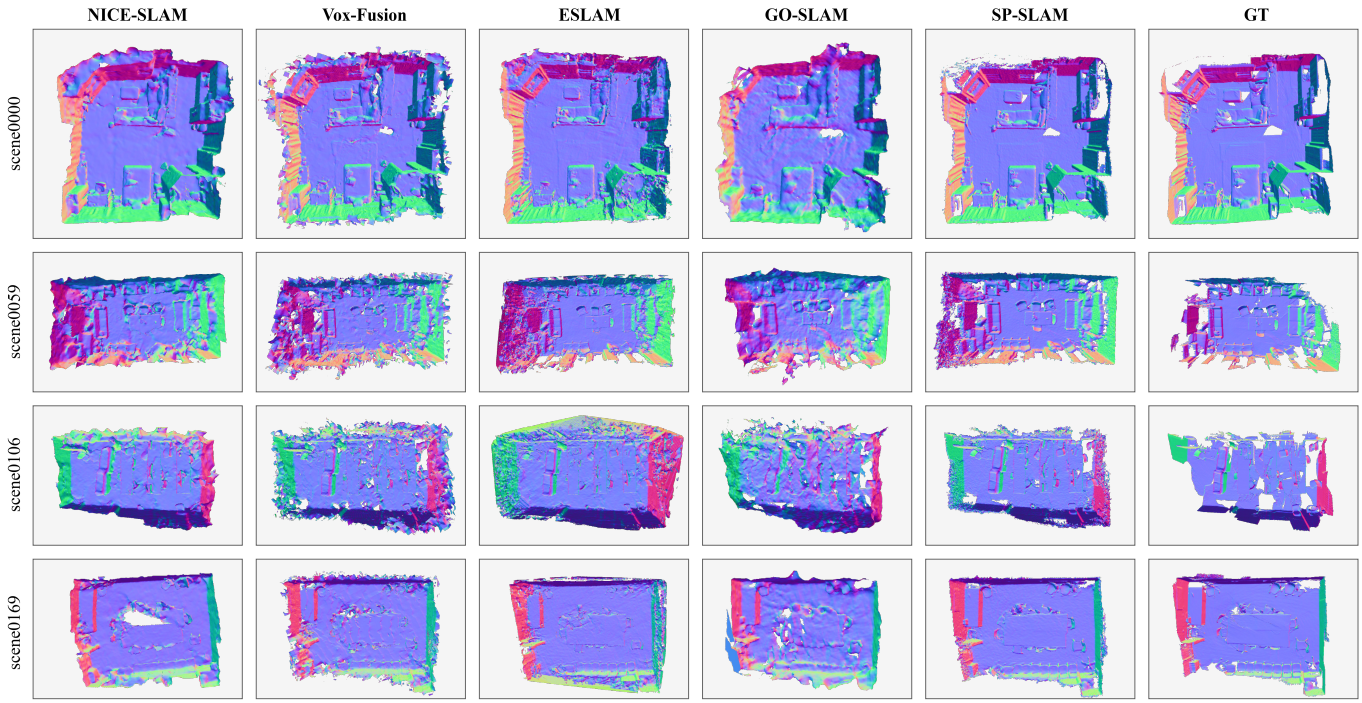


Fig. 5. Qualitative comparison in reconstruction quality on ScanNet dataset [22]. The ground truth mesh for ScanNet is obtained through BundleFusion [10]. Compared to existing methods, our method generates smoother scene surfaces and more detailed geometric structures.

TABLE VI
QUANTITATIVE COMPARISON IN TRACKING PERFORMANCE ON TUM RGB-D DATASET [23].

	fr1/desk (cm)	fr2/xyz (cm)	fr3/office (cm)
NICE-SLAM [13]	2.85	2.39	3.02
Vox-Fusion* [14]	2.75	1.42	6.08
ESLAM [15]	2.47	1.11	2.42
DROID-SLAM [18]	1.64	1.61	2.19
GO-SLAM [19]	**1.49**	**0.68**	**1.42**
Ours	2.41	1.31	2.39

TABLE VII
QUANTITATIVE COMPARISON IN TRACKING PERFORMANCE ON 7-SCENES DATASET [24](ATE RMSE ↓ [CM]). THE RESULTS ARE THE AVERAGE OF FIVE RUNS ON EACH SCENE. COMPARED TO EXISTING METHODS, WE ACHIEVE THE BEST AVERAGE PERFORMANCE.

Scene ID	chess	fire	heads	office	pumpkin	kitchen	stairs	Avg.
NICE-SLAM [13]	2.73	1.89	23.39	8.83	22.50	3.63	3.76	9.46
Vox-Fusion [14]	2.72	2.63	3.13	8.54	16.29	3.51	4.15	5.85
ESLAM [15]	3.29	1.96	4.71	6.01	16.54	**3.35**	7.43	7.22
DROID-SLAM [18]	6.39	3.83	4.08	9.62	15.71	5.01	13.34	8.28
GO-SLAM [19]	5.15	3.26	**2.06**	9.57	**15.43**	3.81	17.99	8.18
Ours	**2.03**	**1.37**	2.26	**4.27**	16.22	3.48	**3.46**	**4.76**

dataset, we conduct our experiments with the default settings of Vox-Fusion, setting both the tracking and mapping iterations to 30 and truncation distance to 0.05. As illustrate in Tab. IV and Tab. VI, on the challenging TUM RGB-D dataset, we achieve competitive tracking performance at an average speed of 9 FPS. Although there is still a gap compared to learning-based SLAM methods, we have narrowed this gap and achieved high-fidelity surface reconstruction (See Fig. 6).

We also conduct benchmark tests on seven real-world scene sequences from 7-Scenes dataset [24], which similarly contain low-resolution, high-noise depth images and severe motion blur. The experimental parameters for all methods on the 7-Scenes dataset are consistent with those used on the ScanNet dataset. We evaluate runtime and camera tracking performance, with the results presented in Tab. IV and Tab. VII. Our method achieves the best tracking performance at real-time speed (10 FPS).

F. Performance Analysis

Tab. I provides the average model size for eight scenes from the Replica dataset [20] and six scenes from the Synthetic RGB-D dataset [21]. According to the results, our model occupies less storage space than NICE-SLAM [13] and ESLAM [15], but more than Vox-Fusion [14]. The reason for minimal storage space used by Vox-Fusion is its use of large-sized (0.2m) voxels for constructing the octree representation of the scene. However, this approach results in the loss of scene details during reconstruction, leading to overly smooth surfaces. In contrast, our sparse volume with 0.04m voxels allows for higher-quality scene reconstruction. Although it consumes more memory than Vox-Fusion, we believe that a storage size of 15.1 MB to 26.8 MB is reasonable in practical computing environments. Furthermore, Tab. I compares GPU usage of our method with existing approaches on the Replica and Synthetic datasets. The results indicate that our system requires lower GPU memory, making it more suitable for

Fig. 6. Qualitative comparison in reconstruction on TUM RGB-D dataset [23]. Our method is capable of generating higher-fidelity surfaces.

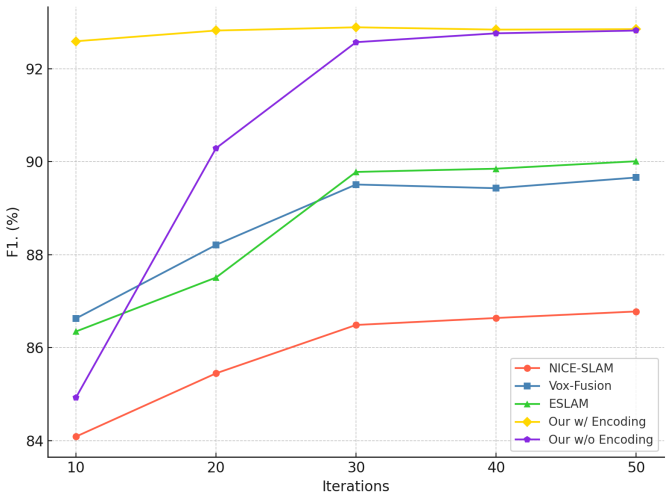


Fig. 7. Ablation study of depth encoding on *morning_apartment* in the Synthetic RGB-D dataset. Iterations here refer to mapping iterations (with all methods having tracking iterations set to 10). With scene prior information, our model converges significantly faster than other methods.

TABLE VIII
ABLATION STUDY OF DEPTH FUSION ON *Scene0000* IN THE SCANNET DATASET AND *Room0* IN THE REPLICA DATASET.

	Scene0000	Room0		
	RMSE (cm)↓	Comp. (%)↑	Accu. (%)↑	RMSE (cm)↓
w/o fusion	10.26	99.07	98.86	0.59
w/ fusion	**5.27**	**99.53**	**99.04**	**0.50**

running on resource-constrained portable devices.

G. Ablation Study

We conduct ablation studies in this section to validate the effectiveness of depth encoding and fusion, and optimization strategy for mapping.

Effect of depth encoding and fusion. Fig. 7 illustrates the impact of depth encoding on the convergence speed of the model. We set the tracking iterations for all methods to 10 to investigate the changes in reconstruction accuracy at different mapping iterations, thereby analyzing the model's convergence speed. The results indicate that, with fixed tracking iterations, our reconstruction accuracy converges after 10 mapping iterations, while other methods require a higher number of mapping iterations to achieve convergence in accuracy. This suggests that encoding scene priors using depth information can significantly enhance the model's convergence speed. Our system achieves high-quality surface reconstruction with

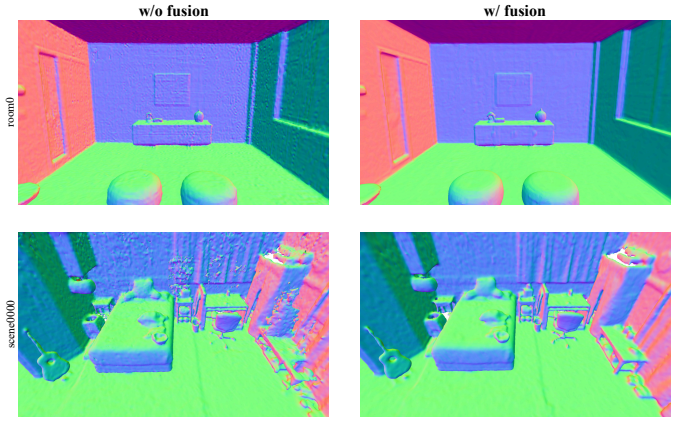


Fig. 8. Visual results of depth fusion ablation for geometric reconstruction on *Room0* in the Replica dataset and *Scene0000* in the ScanNet dataset.

TABLE IX
AVERAGE CAMERA TRACKING RESULTS OF DIFFERENT MAPPING OPTIMIZATION STRATEGIES ON THE REPLICA DATASET AND SCANNET DATASET.

	Select KF	Replica [20]	ScanNet [22]
		RMSE (cm)↓	RMSE (cm)↑
KF strategy	✔	1.24	11.41
Our strategy	✗	**0.52**	**7.30**

fewer mapping iterations. Tab. VIII shows the ablation study results for depth fusion. *w/o fusion* means that we only insert newly observed voxels from the local encoded voxels V_m into the global volume V_g without performing fusion updates. In contrast, fusing the geometric prior encoded information from multiple viewpoints improves tracking and reconstruction performance. The mesh visualization results shown in Fig. 8 demonstrate that depth fusion leads to smoother and more faithful reconstructions. Additionally, the more precise camera tracking eliminates artifacts and noise in the reconstruction results.

Effect of optimization strategy for mapping. Tab. IX shows the average camera tracking results of our method on the Replica dataset [20] and ScanNet dataset [22] using different mapping optimization strategies. The detailed settings and implementation of the keyframe (KF) strategy are as follows: We follow the keyframe addition strategy from previous work [13]–[15], i.e., adding a frame to a global keyframe list every fixed frame interval T. Here, we adhere to the ESLAM setting with $T = 4$. During each mapping process, we select 20 keyframes for bundle optimization from the global keyframe list, including 18 frames randomly chosen from those with

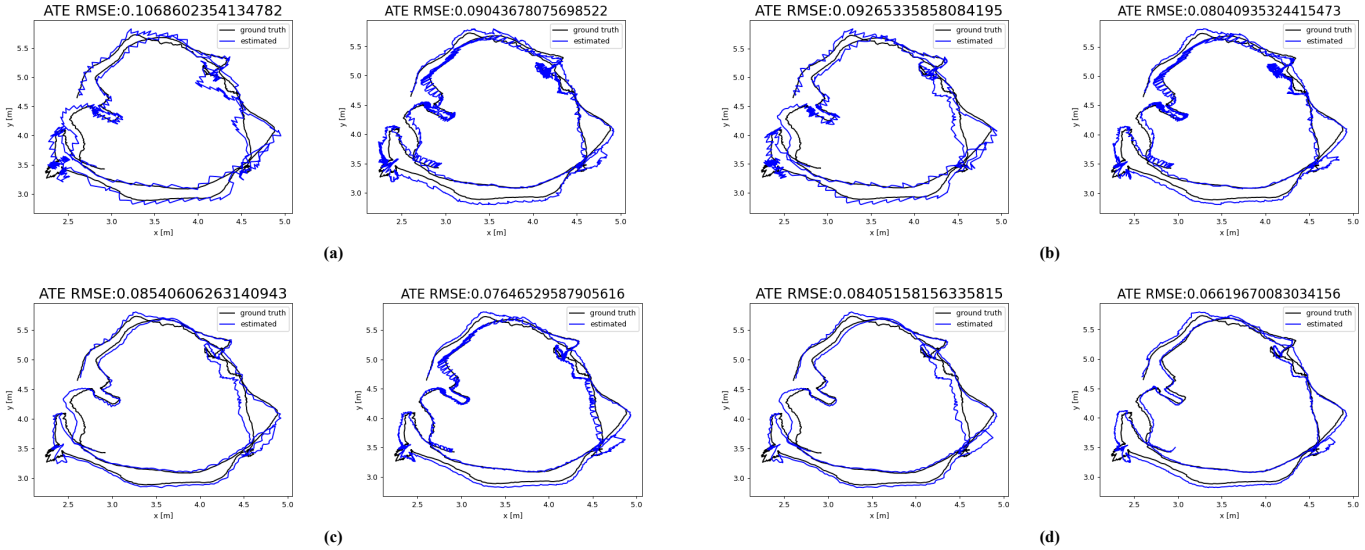


Fig. 9. Visualization of camera tracking results on the ScanNet dataset $scene0059$, comparing our method (using the KF strategy with different tracking iterations or under default settings) to ESLAM with varying tracking iterations. (a) ESLAM with 6 tracking iterations vs. our method with 6 iterations using the KF strategy. (b) ESLAM with 10 tracking iterations vs. our method with 10 iterations using the KF strategy. (c) ESLAM with 20 tracking iterations vs. our method with 20 iterations using the KF strategy. (d) ESLAM with 30 tracking iterations vs. our method under default settings (6 tracking iterations with the proposed mapping optimization strategy).

co-visibility relationships with the current frame and the two most recently added keyframes. For a detailed description and implementation of our mapping optimization strategy, please refer to Sec. III-D. Compared to the KF strategy, we significantly reduce ATE errors. Fig. 9 illustrates the camera tracking results of our method under default settings, as well as using the keyframe strategy with different tracking iterations. By continuously optimizing the pose of each input frame during all mapping processes, we only set a low number of tracking iterations, enabling the system to achieve real-time accurate camera tracking. Additionally, Fig. 9 shows the unstable tracking results of ESLAM when fewer iterations are set per frame, due to their focus on optimizing only keyframes. Neural SLAM methods that rely on keyframe selection need to increase the number of iterations per frame to achieve stable tracking. In contrast, our mapping optimization strategy achieves more stable and accurate camera tracking with fewer tracking iterations per frame.

V. Conclusion

We proposed SP-SLAM, a real-time dense RGB-D SLAM system, incorporates scene geometry priors into the neural implicit SLAM framework, aiming to boost both real-time performance and accuracy. We demonstrate that the neural implicit SLAM is capable of optimizing the pose of each input frame without increasing computational load, without adhering to the traditional SLAM paradigm of selecting a set of key frames for optimization. Extensive experiments confirm that SP-SLAM surpasses existing methods in terms of tracking and reconstruction, while achieve a marked increase in running speed.

References

[1] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: A survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, pp. 1–11, 2017.

[2] I. A. Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A Survey of State-of-the-Art on Visual SLAM," *Expert Systems with Applications*, p. 117734, 2022.

[3] H. Liu, L. Zhao, Z. Peng, W. Xie, M. Jiang, H. Zha, H. Bao, and G. Zhang, "A Low-cost and Scalable Framework to Build Large-Scale Localization Benchmark for Augmented Reality," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[5] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, IEEE, 2007.

[6] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[7] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM Without A Pose Graph," in *Robotics: Science and Systems*, 2015.

[8] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-Time Dense SLAM and Light Source Estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[9] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, *et al.*, "KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, pp. 559–568, 2011.

[10] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration," *ACM Transactions on Graphics*, vol. 36, no. 3, pp. 1–18, 2017.

[11] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[12] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6229–6238, 2021.

[13] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12786–12796, 2022.

[14] X. Yang, H. Li, H. Zhai, Y. Ming, Y. Liu, and G. Zhang, "Vox-Fusion: Dense Tracking and Mapping with Voxel-based Neural Implicit Representation," in *IEEE International Symposium on Mixed and Augmented Reality*, pp. 499–507, IEEE, 2022.

[15] M. M. Johari, C. Carta, and F. Fleuret, "ESLAM: Efficient Dense Slam System Based on Hybrid Representation of Signed Distance Fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17408–17419, 2023.

[16] K. Li, Y. Tang, V. A. Prisacariu, and P. H. Torr, "BNV-Fusion: Dense 3D Reconstruction using Bi-level Neural Nolume Fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6166–6175, 2022.

[17] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," *arXiv preprint arXiv:1512.03012*, 2015.

[18] Z. Teed and J. Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16558–16569, 2021.

[19] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, "GO-SLAM: Global Optimization For Consistent 3d Instant Reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3727–3737, 2023.

[20] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, *et al.*, "The Replica dataset: A Digital Replica of Indoor Spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[21] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, "Neural RGB-D Surface Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6290–6301, 2022.

[22] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5828–5839, 2017.

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

[24] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2930–2937, 2013.

[25] X. Shao, L. Zhang, T. Zhang, Y. Shen, and Y. Zhou, "MOFIS SLAM: A Multi-Object Semantic SLAM System With Front-View, Inertial, and Surround-View Sensors for Indoor Parking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 7, pp. 4788–4803, 2021.

[26] Z. Wang, L. Zhang, S. Zhao, and Y. Zhou, "Ct-LVI: A Framework Towards Continuous-time Laser-Visual-Inertial Odometry and Mapping," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[27] T. Schops, T. Sattler, and M. Pollefeys, "BAD SLAM: BundleAdjusted Direct RGB-D SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 134–144, 2019.

[28] J. Chen, D. Bautembach, and S. Izadi, "Scalable Real-time Volumetric Surface Reconstruction," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 1–16, 2013.

[29] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D Reconstruction at Scale using Voxel Hashing," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 1–11, 2013.

[30] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "CodeSLAM—Learning a Compact, Optimisable Representation for Dense Visual SLAM," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2560–2568, 2018.

[31] J. Zhang, L. Tai, M. Liu, J. Boedecker, and W. Burgard, "Neural SLAM: Learning to Explore with External Memory," *arXiv preprint arXiv:1706.09520*, 2017.

[32] R. Li, S. Wang, and D. Gu, "DeepSLAM: A Robust Monocular SLAM System With Unsupervised Deep Learning," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3577–3587, 2020.

[33] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1281–1292, 2020.

[34] Z. Zhang, J. Sun, Y. Dai, B. Fan, and M. He, "VRNet: Learning the Rectified Virtual Corresponding Points for 3D Point Cloud Registration," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 8, pp. 4997–5010, 2022.

[35] Y. Wang, Y. Qiu, P. Cheng, and J. Zhang, "Hybrid CNN-Transformer Features for Visual Place Recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 3, pp. 1109–1122, 2022.

[36] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "Nerf++: Analyzing and Improving Neural Radiance Fields," *arXiv preprint arXiv:2010.07492*, 2020.

[37] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7210–7219, 2021.

[38] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-NeRF: A Multiscale Representation for Anti-aliasing Neural Radiance Fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5855–5864, 2021.

[39] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron, "NeRF in the dark: High Dynamic Range View Synthesis from Noisy Raw Images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16190–16199, 2022.

[40] M. Oechsle, S. Peng, and A. Geiger, "UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5589–5599, 2021.

[41] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction," *arXiv preprint arXiv:2106.10689*, 2021.

[42] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "MVSNeRF: Fast Generalizable Radiance Field Reconstruction From Multi-View Stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14124–14133, 2021.

[43] Z.-X. Zou, S.-S. Huang, Y.-P. Cao, T.-J. Mu, Y. Shan, and H. Fu, "MonoNeuralFusion: Online Monocular Neural 3D Reconstruction with Geometric Priors," *arXiv preprint arXiv:2209.15153*, 2022.

[44] X. Zhang, S. Bi, K. Sunkavalli, H. Su, and Z. Xu, "NeRFusion: Fusing Radiance Fields for Large-Scale Scene Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5449–5458, 2022.

[45] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger, "MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25018–25032, 2022.

[46] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao, "NeuralRecon: Real-Time Coherent 3D Reconstruction From Monocular Video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15598–15607, 2021.

[47] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "NeRF–: Neural Radiance Fields Without Known Camera Parameters," *arXiv preprint arXiv:2102.07064*, 2021.

[48] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "BARF: Bundle-Adjusting Neural Radiance Fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5741–5751, 2021.

[49] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "iNeRF: Inverting Neural Radiance Fields for Pose Estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1323–1330, IEEE, 2021.

[50] H. Li, X. Yang, H. Zhai, Y. Liu, H. Bao, and G. Zhang, "Vox-Surf: Voxel-Based Implicit Surface Representation," *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[51] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural Sparse Voxel Fields," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15651–15663, 2020.

[52] C. Sun, M. Sun, and H.-T. Chen, "Direct Voxel Grid Optimization: Super-Fast Convergence for Radiance Fields Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5459–5469, 2022.

[53] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–15, 2022.

[54] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "PlenOctrees for Real-time Rendering of Neural Radiance Fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5752–5761, 2021.

[55] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural Geometric Level of Detail: Real-Time Rendering With Implicit 3D Shapes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11358–11367, 2021.
[56] H. Wang, J. Wang, and L. 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*, pp. 13293–13302, 2023.
[57] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, *et al.*, "Efficient Geometry-Aware 3D Generative Adversarial Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16123–16133, 2022.
[58] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303–312, 1996.

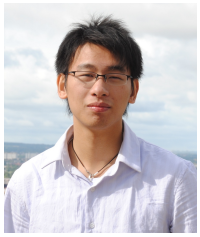
Zhen Hong received the B.S. degree from the Zhejiang University of Technology (ZJUT), Hangzhou, China, and the University of Tasmania, Australia, in 2006, and the Ph.D. degree from ZJUT in 2012. He has visited at the Sensorweb Laboratory, Department of Computer Science, Georgia State University, Atlanta, GA, USA, in 2011. He was at the CAP Research Group, School of Electrical and Computer Engineering, Georgia Institute of Technology, as a Research Scholar, from 2016 to 2018. He is currently a Full Professor with the Institute of Cyberspace Security and the College of Information Engineering, ZJUT. Before joining ZJUT, he was an Associate Professor with the Faculty of Mechanical Engineering and Automation, Zhejiang Sci-Tech University, China. His research interests include the Internet of Things, wireless sensor networks, cyberspace security, and data analytics. He received the first Zhejiang Provincial Young Scientists Title in 2013 and the Zhejiang Provincial New Century 151 Talent Project in 2014. He is a Senior Member of CCF and CAA. He serves on the Youth Committee for the Chinese Association of Automation and Blockchain Committee and CCF YOCSEF, respectively.

Bowen Wang received a B.S. degree from Shenyang University of Technology, Shenyang, China, in 2022. He is currently pursuing a M.S. degree in computer science and technology with the School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. His research interests include neural SLAM systems and 3D reconstruction.

Haoran Duan (Graduate Student Member, IEEE) received a Distinction M.S. degree in Data Science from Newcastle University UK in 2019. He was also a research student at OpenLab, Newcastle University. He obtained his PhD degree from Durham University, UK, and now he is a postdoc research associate at Networked and Ubiquitous Systems Engineering Group, School of Computing, Newcastle University, working on deep learning applications. His current research interests focus on the applications/theories of deep learning.

Yawen Huang received the M.Sc. and Ph.D. degrees from the Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield, U.K., in 2015 and 2018, respectively. She is currently a Senior Scientist of Tencent Jarvis Laboratory, Shenzhen, China. Her research interests include computer vision, machine learning, medical imaging, deep learning, and practical AI for computer aided diagnosis.

Xiong Li received the B.S. degree from Jiangxi University of Science and Technology, Ganzhou, China, in 2020. He is currently pursuing the Ph.D. degree in control theory and control engineering with the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. His current research interests include unmanned system security and machine learning.



Zhenyu Wen (Senior Member, IEEE) is currently a Tenure-Tracked Professor with the Institute of Cyberspace Security and the College of Information Engineering, Zhejiang University of Technology. His current research interests include the IoT, crowd sources, AI systems, and cloud computing. For his contributions to the area of scalable data management for the Internet of Things, he was awarded the IEEE TCSC Award for Excellence in Scalable Computing (Early Career Researchers) in 2020.

Xiang Wu received the B.E. degree in automation, the M.E. degree in control engineering, and the Ph.D. degree in control theory and control engineering from the Zhejiang University of Technology, Hangzhou, China, in 2012, 2015, and 2020, respectively. He is currently a Postdoctor with the College of Information Engineering, Zhejiang University of Technology. His research interests include disturbance rejection, networked motion control systems, and cloud control systems.

Wei Xiang (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 1997 and 2000, respectively, and the Ph.D. degree in telecommunications engineering from the University of South Australia, Adelaide, Australia, in 2004. From 2004 to 2015, he was with the School of Mechanical and Electrical Engineering, University of Southern Queensland, Toowoomba, Australia. He is currently the Founding Professor and the Head of the Discipline of Internet of Things Engineering with the College of Science and Engineering, James Cook University, Cairns, Australia. He has authored or co-authored over 200 peer-reviewed journal and conference papers. His research interests are in the broad areas of communications and information theory, particularly the Internet of Things, and coding and signal processing for multimedia communications systems. He is an Elected Fellow of the IET and Engineers Australia. He received the TNQ Innovation Award in 2016, and was a finalist for 2016 Pearcey Queensland Award. He was a co-recipient of three best paper awards at 2015 WCSP, 2011 IEEE WCNC, and 2009 ICWMC. He has been awarded several prestigious fellowship titles. He was named a Queensland International Fellow (2010–2011) by the Queensland Government of Australia, an Endeavour Research Fellow (2012–2013) by the Commonwealth Government of Australia, a Smart Futures Fellow (2012–2015) by the Queensland Govern ment of Australia, and a JSPS Invitational Fellow jointly by the Australian Academy of Science and Japanese Society for Promotion of Science (2014–2015). He is the Vice Chair of the IEEE Northern Australia Section. He was an Editor of the IEEE COMMUNICATIONSLETTERS (2015–2017), and is an Associate Editor of Telecommunications Systems (Springer). He has severed in a large number of international conferences in the capacity of General Co-Chair, TPC Co-Chair, Symposium Chair, and so on.

Yefeng Zheng (Fellow, IEEE) received the B.E. and M.E. degrees from Tsinghua University, Beijing, in 1998 and 2001, respectively, and the Ph.D. degree from the University of Maryland, College Park, MD, USA, in 2005. After graduation, he joined Siemens Corporate Research, Princeton, NJ, USA. He is currently the Director and the Distinguished Scientist of Tencent Jarvis Laboratory, Shenzhen, China, leading the company's initiative on Medical AI. His research interests include medical image analysis, graph data mining, and deep learning. Dr. Zheng is a fellow of the American Institute for Medical and Biological Engineering (AIMBE).