




Real-Time Dense Visual SLAM with Neural Factor Representation

Weifeng Wei ^{1,†} , Jie Wang ^{2,†}, Xiaolong Xie ³ , Jie Liu ² and Pengxiang Su ^{2,*} 

¹ School of Information Engineering, Nanchang University, Nanchang 330031, China; 416100220248@email.ncu.edu.cn

² School of Software, Nanchang University, Nanchang 330031, China; 8008121372@email.ncu.edu.cn (J.W.); ndliujie@ncu.edu.cn (J.L.)

³ School of Mathematics and Computer Science, Nanchang University, Nanchang 330031, China; 416100210092@email.ncu.edu.cn

* Correspondence: supengxiang@ncu.edu.cn

† These authors contributed equally to this work.

Abstract: Developing a high-quality, real-time, dense visual SLAM system poses a significant challenge in the field of computer vision. NeRF introduces neural implicit representation, marking a notable advancement in visual SLAM research. However, existing neural implicit SLAM methods suffer from long runtimes and face challenges when modeling complex structures in scenes. In this paper, we propose a neural implicit dense visual SLAM method that enables high-quality real-time reconstruction even on a desktop PC. Firstly, we propose a novel neural scene representation, encoding the geometry and appearance information of the scene as a combination of the basis and coefficient factors. This representation allows for efficient memory usage and the accurate modeling of high-frequency detail regions. Secondly, we introduce feature integration rendering to significantly improve rendering speed while maintaining the quality of color rendering. Extensive experiments on synthetic and real-world datasets demonstrate that our method achieves an average improvement of more than 60% for Depth L1 and ATE RMSE compared to existing state-of-the-art methods when running at 9.8 Hz on a desktop PC with a 3.20 GHz Intel Core i9-12900K CPU and a single NVIDIA RTX 3090 GPU. This remarkable advancement highlights the crucial importance of our approach in the field of dense visual SLAM.

Keywords: dense visual SLAM; computer vision; neural implicit representation; feature integration rendering



Citation: Wei, W.; Wang, J.; Xie, X.; Liu, J.; Su, P. Real-Time Dense Visual SLAM with Neural Factor Representation. *Electronics* **2024**, *13*, 3332. <https://doi.org/10.3390/electronics13163332>

Academic Editor: Janos Botzheim

Received: 4 July 2024

Revised: 9 August 2024

Accepted: 20 August 2024

Published: 22 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The primary goal of dense visual simultaneous localization and mapping (SLAM) is to estimate the pose of a visual sensor (such as an RGB-D camera) while constructing dense 3D maps of unknown environments. It has wide applications in the fields of autonomous driving, robotics, and virtual/augmented reality. Traditional visual SLAM methods such as [1–3] employ multi-view geometry to achieve robust camera tracking and represent scene maps using point clouds. However, this discrete form of scene representation typically requires substantial memory resources and exhibits heightened sensitivity to environmental conditions.

The emergence of neural radiance fields (NeRF) [4] demonstrates that neural implicit representation has powerful spatial continuous representation capabilities, giving full play to their advantages in dense visual SLAM. Unlike traditional SLAM methods, neural implicit SLAM does not require feature point-based multi-view stereo matching or hand-crafted loss terms. Instead, it uses neural radiance fields to represent a scene and uses multi-layer perceptrons (MLPs) to decode the scene's attributes. iMAP [5] utilizes a single MLP as scene representation, constructing a loss function via volume rendering to iteratively optimize the scene representation and estimate camera poses. NICE-SLAM [6]

introduces hierarchical feature grids and pre-trained tiny-MLPs for scene representation. This can locally update the constructed map, avoiding catastrophic forgetting. Afterward, ESLAM [7] and Co-SLAM [8] utilize axis-aligned feature planes and hash grids for scene representation, respectively. Compared to NICE-SLAM [6], they have a lower memory footprint growth rate.

However, the above methods have some issues. For scene representation, the model size of NICE-SLAM [6] grows cubically with the range of a scene map, and higher resolution grids are required to accurately represent the fine details in the scene, which further increases memory usage and training costs. Despite ESLAM [7] only having a quadratic memory footprint growth rate, it remains sensitive to the size of the map. In addition, axis-aligned feature planes decompose scene information along co-ordinate axes. If critical features or structures are not aligned with the main co-ordinate axes, axis-aligned transformations struggle to capture them effectively. Due to the hash grid being prone to hash collisions at fine scales, Co-SLAM [8] faces challenges in reconstructing detailed regions of a scene. Furthermore, Co-SLAM [8] uses a customized CUDA framework to accelerate the neural implicit SLAM system, which exhibits limited scalability when addressing downstream tasks in neural implicit dense visual SLAM. For color rendering, these methods use ray marching through query and volume rendering space points for scene color rendering. This process requires querying the RGB values of all points along the ray, resulting in huge MLP query costs and affecting the real-time performance of the neural implicit dense visual SLAM system.

In order to further balance accuracy and performance, we are inspired by [9,10] to utilize neural factors for neural implicit dense visual SLAM scene representation and propose employing a neural rendering method similar to [11], namely feature integration rendering to improve the real-time performance of neural implicit SLAM. This enables us to reconstruct detailed scene maps in real time without being affected by the memory growth associated with incremental reconstruction. In general, our contributions include the following:

- We propose a novel scene representation based on neural factors, demonstrating higher-quality scene reconstruction and a more compact model memory footprint in large-scale scenes.
- In order to address insufficient real-time performance due to high MLP query costs in previous neural implicit SLAM methods, we introduce an efficient rendering approach using feature integration. This improves real-time performance without relying on a customized CUDA framework.
- We conducted extensive experiments on both synthetic and real-world datasets to validate our design choices, achieving competitive performance against baselines in terms of 3D reconstruction, camera localization, runtime, and memory usage.

2. Related Work

This section provides a comprehensive overview of related works, categorizing them into traditional dense visual SLAM methods and highlighting the significant advancements in neural implicit representations that have recently garnered substantial attention. Section 2.1 reviews traditional dense visual SLAM methods, tracing their evolution from early tracking and mapping architectures to recent approaches that integrate deep neural networks for enhanced performance. Section 2.2 explores the advancements in neural implicit representations, with a particular focus on their applications in novel view synthesis and 3D reconstruction. Finally, Section 2.3 delves into the incorporation of neural implicit representations in dense visual SLAM, highlighting the latest systems and the challenges they address.

2.1. Traditional Dense Visual SLAM

Most of the existing dense visual SLAM methods follow the tracking and mapping architecture proposed by PTAM [12]. DTAM [13] introduces the first dense visual SLAM

system, utilizes cost volume for scene representation, and employs a dense per-pixel approach for tracking and mapping. KinectFusion [14] proposes a pioneering method for real-time scene reconstruction using an RGB-D camera as a sensor. They employ iterative closest point (ICP) for tracking and utilize TSDF-Fusion to update scene geometry for mapping. BAD-SLAM [15] utilizes dense surfels to represent a scene. These surface elements can be efficiently updated through direct bundle adjustment (BA). Some recent works [16–19] further improve the accuracy and robustness of dense visual SLAM systems compared to traditional methods by combining deep neural networks with a traditional visual SLAM framework.

2.2. Neural Implicit Representations

NeRF [4] has garnered extensive research interest in novel view synthesis and 3D reconstruction due to its continuous scene representation and memory efficiency. However, this MLP-based neural implicit representation requires a long training time. Subsequent works have adopted hybrid scene representations, encoding scene information into features and anchoring these features onto specific data structures such as octrees [20,21], voxel grids [22,23], tri-planes [24], and hash grids [25]. These methods significantly accelerate training speeds at the cost of increased memory usage. These methods provide support for the real-time performance of neural implicit dense visual SLAM methods.

2.3. Neural Implicit Dense Visual SLAM

More recently, neural implicit representations have become popular in dense visual SLAM. iMAP [5] first proposes a neural implicit dense visual SLAM system that uses a single MLP for tracking and mapping. While it exhibits efficient memory usage, it faces catastrophic forgetting in large-scale scenes. NICE-SLAM [6] introduces hierarchical feature grids for scene representation to avoid forgetting and achieve large-scale scene reconstruction, but it has a higher memory usage growth rate. While ESLAM [7] utilizes axis-aligned feature planes to reduce memory footprint growth rate, this representation will produce axis-aligned bias and make it difficult to handle complicated structures in the scene. Vox-fusion [26] employs a sparse octree for scene representation and does not require a predefined scene bounding box. However, vox-fusion [26] uses a larger voxel size and only includes surface points with valid depth measurements during ray sampling, resulting in poor scene reconstruction quality. Co-SLAM [8] employs a hash grid for scene representation and one-blob encoding for hole-filling, but it faces challenges with hash conflicts in detailed scene areas, resulting in artifacts in reconstructed scenes. Point-SLAM [27] adopts neural point clouds for better 3D reconstruction, yet it demands significant training time, meaning a lack of real-time performance. Therefore, how to better balance the accuracy, speed, and memory footprint of neural implicit dense visual SLAM is a challenging problem that needs to be solved.

3. Method

Figure 1 shows an overview of our work. We utilize two sets of factor grids with respective decoders to represent the geometry and appearance of the scene. We use the estimated camera pose to cast a ray for each pixel, sampling 3D points by ray marching and querying the network to render the depth and color values for each pixel. By minimizing our proposed loss functions, we are able to optimize both the camera poses and scene representation simultaneously. Section 3.1 describes our scene representation in detail. Section 3.2 introduces feature integration color rendering, and Section 3.3 introduces the details of tracking, mapping, and loss functions.

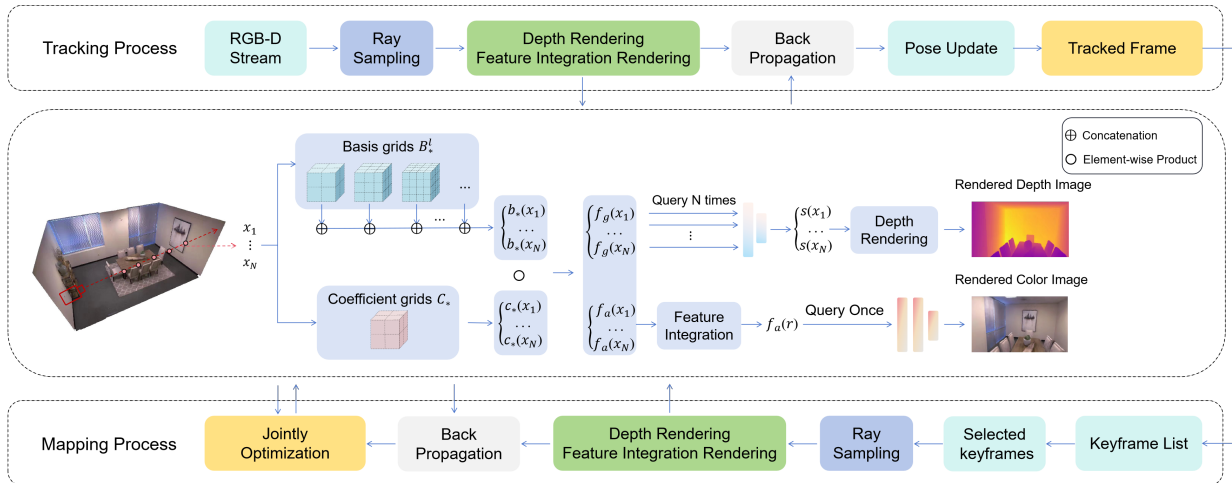


Figure 1. Overview. (1) Scene representation: We use two different sets of factor grids to represent the scene geometry and appearance, respectively. In order to simplify our overview, we use the symbol $*$ to denote both geometry, g , and appearance, a , e.g., $b_*(x_i)$ can be either $b_g(x_i)$ or $b_a(x_i)$. For sample points along the ray, we query the basis and coefficient factors for depth and feature integration rendering. (2) Mapping process: We uniformly sample pixels from a selected set of keyframes and jointly optimize scene representation and camera poses of these keyframes. (3) Tracking process: Factor grids and MLPs remain fixed, and only the camera pose of each input frame in the RGB-D stream is estimated.

3.1. Neural Factors Representation

In contrast to [6–8,26], which directly encodes scene information into features, we encode scene information into basis and coefficient factors, utilizing the combination of these factors as the features. Since the geometry of a scene generally converges faster than its appearance, utilizing a unified factors model to jointly represent both can readily lead to the forgetting of geometry learning in the scene. This reduces pose estimation robustness. Therefore, we adopt different basis and coefficient factors to represent the geometry and appearance of the scene, respectively. The basis and coefficient factors are implemented via learnable tensor grids. For the basis factors, we implement them using the multi-level tensor grids B_g^l and B_a^l , where each level of resolution increases linearly; this enables us to model the different frequencies of information in the scene more accurately. For the coefficient factors, we use the single-level tensor grids C_g and C_a . Specifically, For N -sampled points $x_i = \mathbf{o} + z_i \mathbf{d}$, $i \in \{1, \dots, N\}$ along the ray, where \mathbf{o} denotes camera origin, and z_i corresponds to the depth value of sampled point x_i , we query the geometry basis factor, $b_g(x_i)$, and geometry coefficient factor, $c_g(x_i)$, from the geometry basis grids and geometry coefficient grid via trilinear interpolation. By element-wise multiplying the geometry basis factor with the geometry coefficient factor of the sampled point, we obtain the geometry feature $f_g(x_i)$ of x_i . We obtain the appearance feature $f_a(x_i)$ of sampled point x_i similarly. Then, we input the geometry feature of the sampled point into the geometry decoder \mathbf{MLP}_g to decode the truncated signed distance (TSDF) value $s(x_i)$ of x_i :

$$s(x_i) = \mathbf{MLP}_g(f_g(x_i)) \tag{1}$$

In order to allocate a sampled point weight to rendering depth and color, we adopt a method similar to [7] to convert the TSDF value into weight for rendering along the ray:

$$\sigma(x_i) = 1 - \exp(-\beta \cdot \text{sigmoid}(-s(x_i) \cdot \beta)) \tag{2}$$

$$w_i = \sigma(x_i) \prod_{j=1}^{i-1} (1 - \sigma(x_j)) \tag{3}$$

where $\sigma(x_i)$ is the volume density of x_i and β is a hyperparameter. Therefore, the rendered depth of the ray is

$$\hat{D}(r) = \sum_{i=1}^N w_i z_i \quad (4)$$

3.2. Feature Integration Rendering

Previous neural implicit dense visual SLAM methods [5–8,26,27] obtain rendered pixel color by querying the raw color of sampled points and relying on volume rendering by using a marching camera ray. However, this rendering approach is relatively inefficient for implementing a real-time neural implicit dense visual SLAM system. In our work, instead of decoding the appearance features of sampled points to raw colors, we input the appearance feature of the entire ray into the color decoder to obtain the final rendered pixel color. Compared to [11], which uses a large MLP to ensure color rendering quality, our method requires only a shallow MLP to achieve high-quality color reconstruction, thereby reducing the number of parameters that require optimization during backpropagation. This further balances the quality and efficiency of the neural implicit SLAM system. Specifically, after obtaining the appearance features and weights of all sampled points along the ray, we approximate the overall appearance feature of the ray by performing a weighted summation of the appearance features of all sampled points along the ray:

$$f_a(r) = \sum_{i=1}^N w_i f_a(x_i) \quad (5)$$

the ray feature $f_a(r)$ is decoded into rendered pixel color by using the color decoder MLP_a .

$$\hat{C}(r) = \text{MLP}_a(f_a(r)) \quad (6)$$

3.3. Training

This section details the training process of our method. The tracking and mapping processes are performed synchronously during training. The tracking process estimates the camera pose for each input frame, and the mapping process jointly optimizes the scene representation and the camera poses of selected keyframes. Both processes require iterative optimization by minimizing losses. Section 3.3.1 introduces the loss functions used in the optimization process, while Sections 3.3.2 and 3.3.3 describe the tracking and mapping processes in detail, respectively.

3.3.1. Loss Functions

In order to optimize scene representation and camera poses, we apply color loss, depth loss, free-space loss, and SDF loss. The color loss is the \mathcal{L}_2 loss between the rendered pixel color $\hat{C}(r)$ and the ground truth pixel color $C(r)$:

$$\mathcal{L}_c = \frac{1}{|R|} \sum_{r \in R} (C(r) - \hat{C}(r))^2 \quad (7)$$

Similarly, the depth loss is the \mathcal{L}_2 loss between the ground truth depth $D(r)$ and the rendered depth $\hat{D}(r)$ of the pixel.

$$\mathcal{L}_d = \frac{1}{|R_{vd}|} \sum_{r \in R_{vd}} (D(r) - \hat{D}(r))^2 \quad (8)$$

where R is a set of pixels, and R_{vd} denotes the camera rays with a valid depth in R . In order to achieve more accurate surface reconstruction, we employed a depth sensor measurement to approximate the SDF loss. When the sampled point is far from the truncation region, we apply the free-space loss, L_{fs} , to enforce consistency between the TSDF value and

the truncation distance. When the sampled point is within the truncation region, tr , i.e., $|z_i - D(r)| < tr$, we apply SDF loss \mathcal{L}_{sdf} to learn the surface geometry within the truncated region.

$$\mathcal{L}_{fs} = \frac{1}{|R_{vd}|} \sum_{r \in R_{vd}} \frac{1}{|P_r^{fs}|} \sum_{x_i \in P_r^{fs}} (s(x_i) - tr)^2 \quad (9)$$

$$\mathcal{L}_{sdf} = \frac{1}{|R_{vd}|} \sum_{r \in R_{vd}} \frac{1}{|P_r^t|} \sum_{x_i \in P_r^t} (s(x_i) + z_i - D(r))^2 \quad (10)$$

where P_r^{fs} denotes the sampled points along the ray located between the camera origin and the surface truncation region of the depth sensor measurement, while P_r^t denotes the sampled points along the ray within the truncation region.

The global loss function we use is as follows:

$$\mathcal{L} = \lambda_c \mathcal{L}_c + \lambda_d \mathcal{L}_d + \lambda_{fs} \mathcal{L}_{fs} + \lambda_{sdf} \mathcal{L}_{sdf} \quad (11)$$

where λ_c , λ_d , λ_{fs} , and λ_{sdf} are weighting coefficients. We employ the same global loss function in both tracking and mapping but set different weighting coefficients. Furthermore, following [7], we use different SDF loss coefficients for the sampling points at the center and the tail of the truncation region.

3.3.2. Tracking

During tracking, we first initialize the camera pose of the current frame, i , and then obtain a copy of the factor grids and MLPs from the mapping process and keep the parameters in them fixed, only optimizing the transformation matrix $\mathbf{T}_i \in SE(3)$ in the world co-ordinate for frame i . Since the first frame is assigned the ground truth camera pose, the tracking process starts from the second frame. We apply a constant speed assumption to initialize the camera pose for frame i :

$$\mathbf{T}_i = \mathbf{T}_{i-1} \mathbf{T}_{i-2}^{-1} \mathbf{T}_{i-1}, i \geq 2 \quad (12)$$

we assume that $i = 0$ represents the first frame. If $i = 1$, it is initialized directly using the camera pose of the first frame. For the current frame, $|R_t|$ pixels are randomly selected to participate in iterative optimization. During this process, we only update the camera pose, and the factor grids and decoders remain fixed.

3.3.3. Mapping

When the SLAM system starts, we first perform the mapping process. For the first input frame, similar to previous neural implicit SLAM methods [5–8,26,27], we read the ground truth camera pose to fix the origin of the world co-ordinate system. Next, we perform ray sampling on the first frame for iterative optimization to initialize the learnable parameters in the scene representation. For subsequent input frames, we perform the mapping process to jointly optimize the scene representation and camera poses for every four input frames, which is similar to [6–8]. During joint optimization, we define a keyframe window as having a size of W . If the keyframe list has fewer than or equal to W keyframes, we select all keyframes for joint optimization; otherwise, we select W keyframes from the keyframe list that are co-viewed with the current frame. If the last keyframe is not included, we add it. After selecting the keyframes, we also add the current frame to the set. For these selected keyframes, we evenly distribute $|R_m|$ pixels across these frames for joint optimization. After the joint optimization is completed, we add the current frame to the keyframe list.

4. Experiments

This section presents the experimental validation of our method. Comprehensive evaluations on both synthetic and real-world datasets are conducted to demonstrate the effectiveness of our method. The experiments are designed to test the performance of our method in terms of localization accuracy, reconstruction quality, runtime, and memory usage. Section 4.1 outlines the experimental setup, including the datasets used, the baselines, the evaluation metrics, the hyperparameters, and post-processing. Section 4.2 presents the reconstruction evaluation results, and Section 4.3 discusses the camera localization evaluation. Section 4.4 analyzes runtime and memory usage. Finally, Section 4.5 conducts an ablation study to validate our design choices.

4.1. Experimental Setup

In Section 4.1, we describe the experimental setup in detail. Section 4.1.1 discusses the datasets used for evaluation. Section 4.1.2 introduces the baselines against which we compare our method. Section 4.1.3 outlines the evaluation metrics employed to assess performance. Section 4.1.4 details the hyperparameters used in our experiments. Finally, Section 4.1.5 explains the post-processing steps applied to meshes.

4.1.1. Datasets

We evaluated our method using the Replica [28], ScanNet [29], and TUM-RGBD [30] datasets. In this study, we selected eight synthetic scenes from Replica [28] and six real-world scenes from ScanNet [29] to evaluate localization and reconstruction; we selected three real-world scenes from TUM-RGBD [30] to evaluate localization. For tracking evaluation, the ground-truth camera poses of ScanNet [29] were obtained using BundleFusion [31].

4.1.2. Baselines

We compare our method to existing state-of-the-art neural implicit dense visual SLAM methods: iMAP [5], NICE-SLAM [6], Vox-Fusion [26], ESLAM [7], Co-SLAM [8], and Point-SLAM [27]. Note that iMAP* and Vox-Fusion* denote the results we reproduced using the corresponding open-source codes.

4.1.3. Evaluation Metrics

We evaluated the reconstructed mesh using 3D metric accuracy (cm), completion (cm), and completion ratio (%) and a 2D metric depth, L1 (cm). Accuracy (cm) is then defined as the average distance between points on the reconstructed mesh and their nearest points on the ground truth mesh. Completion (cm) is similarly defined as the average distance between the sampled points on the ground-truth mesh and their nearest sampled points on the reconstructed mesh. The completion ratio (%) denotes the percentage of points in the reconstructed mesh with a completion distance under 5 cm. For Depth L1 (cm), following [7], we render depth in both the reconstructed and ground truth meshes for 1000 synthetic views. These views are uniformly sampled within the mesh, and views not observed by the input frames are rejected. For the evaluation of camera localization, we adopt ATE RMSE (cm) [30]. The quantitative results for tracking and mapping are the averages of five runs.

4.1.4. Hyperparameters

Default settings. For scene representation, we uniformly set the truncation distance tr to 6 cm and β to 10. We employed a total of six levels of grids with linearly increasing resolution as our basis grids. We set resolutions similar to [10]: $[32, 128]^T \cdot \frac{\min(a-b)}{1024}$, where a and b are scene bounding boxes. The channels for each level of basis grids are $[4, 4, 4, 2, 2, 2]^T$, respectively. The coefficient grid has a single level with a resolution of 32. Both geometry and appearance feature vectors have 18 channels. The geometry decoder is a single-layer MLP with 64 channels in the hidden layer; the color decoder uses a two-layer MLP with 128 channels. We used $R_t = 2000$ pixels as the default for tracking and $R_m = 4000$ pixels for

mapping. For the mapping process, we set the following loss coefficients: $\lambda_c = 5$, $\lambda_d = 0.1$, $\lambda_{fs} = 5$, $\lambda_{center} = 2000$, and $\lambda_{tail} = 10$. For the tracking process, we set $\lambda_c = 5$, $\lambda_d = 0.1$, $\lambda_{fs} = 10$, $\lambda_{center} = 5000$, and $\lambda_{tail} = 50$.

For the learning rates, all tensor grids have a unified rate of 0.02, while the decoders have a learning rate of 0.005. For the keyframe selection strategy, we follow [6] and provide two different approaches: (1) Overlap: we selected keyframes that have a visual overlap with the current frame. (2) Global: we randomly drew a subset of keyframes from the keyframe list. The keyframe window size, W , was consistently set at 20. We used the first keyframe selection strategy as our default setting.

Replica settings. For the Replica dataset [28], we performed eight iterations for tracking and 15 iterations for mapping. We used regular sampling $N_s = 32$ points along each ray and $N_i = 8$ points near the surface. The learning rates for the rotation matrix and translation vector were set at 0.001 and 0.002, respectively. We employed the default loss coefficients and keyframe selection strategy.

ScanNet settings. For the ScanNet dataset [29], we performed 15 iterations for tracking and 30 iterations for mapping. We used regular sampling $N_s = 48$ points along each ray and $N_i = 8$ points near the surface. The learning rates for the rotation matrix and translation vector were set at 0.0025 and 0.0005, respectively. For the loss coefficients, we set $\lambda_{center} = 500$ for the tracking and mapping processes and used the default settings for the rest. In addition, we used the global keyframe selection strategy.

TUM-RGBD settings. For the TUM-RGBD dataset [30], we set $N_s = 48$ and $N_i = 8$. We performed 60 iterations for mapping and 200 iterations for tracking, and we randomly sampled 5000 rays for each scene. The learning rates of both the rotation matrix and the translation vector were set at 0.0025. In addition, we set $\lambda_{center} = 6000$ for tracking and mapping. The global keyframe selection strategy was also employed.

4.1.5. Post-Processing

After processing all input frames, we generated a 3D mesh based on the scene representation using the marching cubes algorithm [32]. Since the neural representation can predict scene information beyond the frustum range, which is unfair in evaluation, we needed to perform an additional mesh culling step before evaluating the 3D mesh. Specifically, we culled all regions that are not within any camera frustum. For a fair comparison, other baseline methods also employed this culling strategy for the quantitative results.

4.2. Reconstruction Evaluation

We show qualitative comparisons of the Replica [28] and ScanNet [29] datasets in Figures 2–5, and we report quantitative comparisons of the Replica dataset [28] in Table 1. We used meshlab [33] to visualize all meshes. Due to the incomplete ground truth mesh of the ScanNet dataset [29], we only provide a qualitative analysis of geometric reconstructions for this dataset, similar to previous work [5–8,27]. Our method outperforms the baselines on average, and compared to other baseline methods, it exhibits superior performance in recovering the fine details of the scenes. Our method achieves higher-quality scene reconstruction, even in real-world scenes with noise, and is able to complete unobserved areas of the input frame.

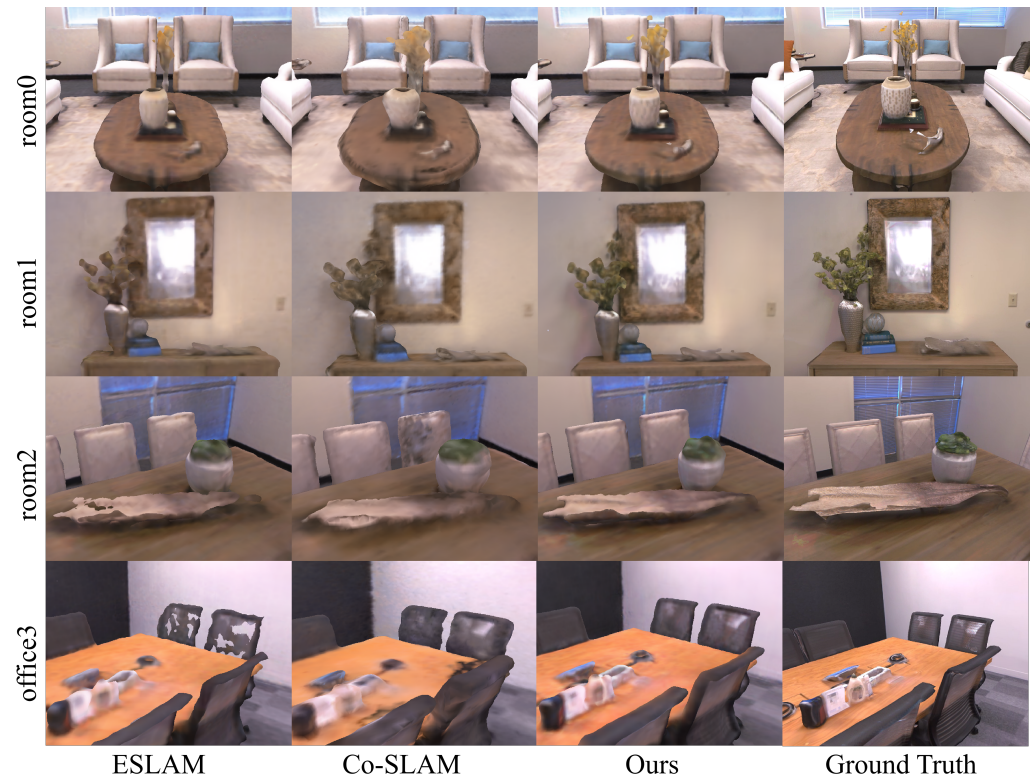


Figure 2. Comparison of qualitative results for reconstruction using the Replica dataset [28].

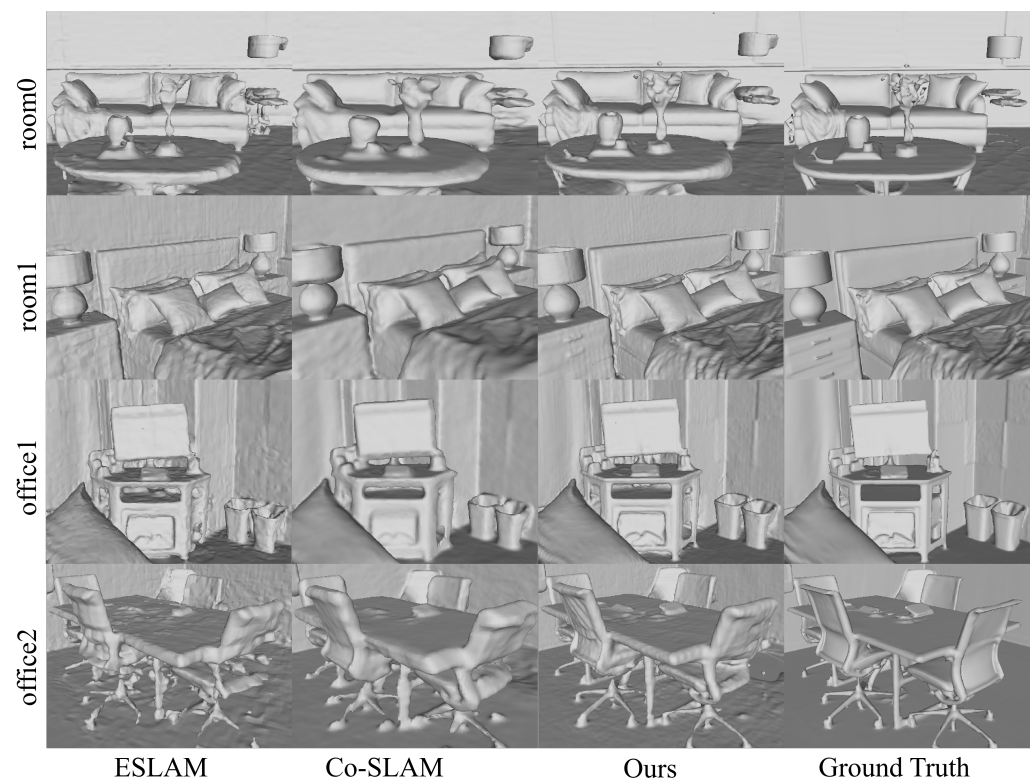


Figure 3. Comparison of qualitative results for reconstruction using the Replica dataset [28]. We visualize the untextured meshes.

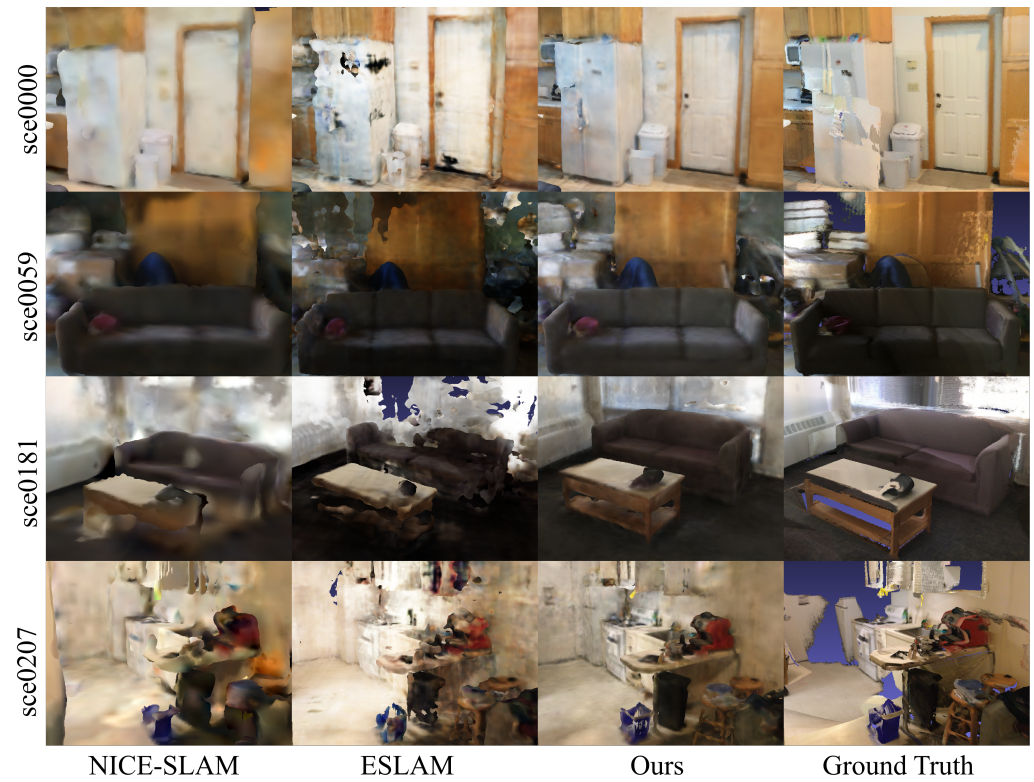


Figure 4. Comparison of qualitative results for reconstruction using ScanNet [29].

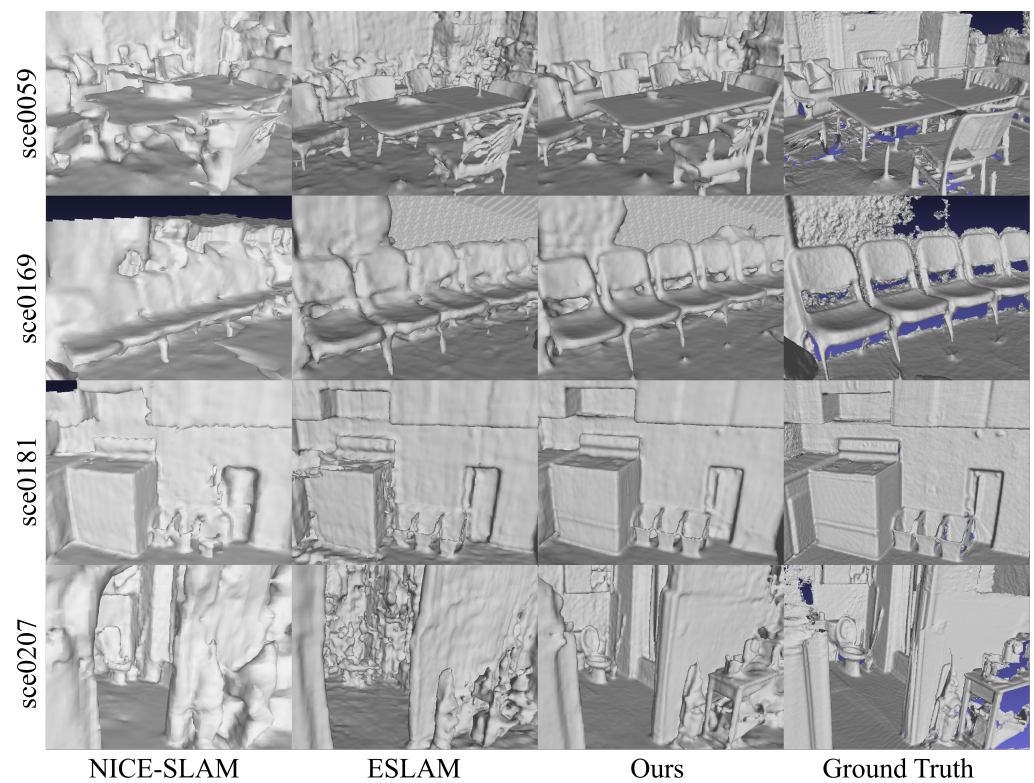


Figure 5. Qualitative reconstruction using the ScanNet dataset [29]. We visualize the untextured meshes.

Table 1. Reconstruction results for eight synthetic scenes using the Replica dataset [28]. Bold formatting is used to emphasize the data in the table. The up/down arrow (↑/↓) indicates that a larger/smaller indicator is better. The symbol (*) denote the results we reproduced using the corresponding open-source codes.

| Method | Metric | room0 | room1 | room2 | office0 | office1 | office2 | office3 | office4 | Avg. |
|-------------------|-------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| iMAP * [5] | Acc. ↓ | 5.75 | 5.44 | 6.32 | 7.58 | 10.25 | 8.91 | 6.89 | 5.34 | 7.06 |
| | Comp. ↓ | 5.96 | 5.38 | 5.21 | 5.16 | 5.49 | 6.04 | 5.75 | 6.57 | 5.69 |
| | Comp. Ratio (%) ↑ | 67.13 | 68.91 | 71.69 | 70.14 | 73.47 | 63.94 | 67.68 | 62.30 | 68.16 |
| | Depth L1 ↓ | 5.55 | 5.47 | 6.93 | 7.63 | 8.13 | 10.61 | 9.66 | 8.44 | 7.8 |
| NICE-SLAM [6] | Acc. ↓ | 1.47 | 1.21 | 1.66 | 1.28 | 1.02 | 1.54 | 1.95 | 1.60 | 1.47 |
| | Comp. ↓ | 1.51 | 1.27 | 1.65 | 1.74 | 1.04 | 1.62 | 2.57 | 1.67 | 1.63 |
| | Comp. Ratio (%) ↑ | 98.16 | 98.71 | 96.42 | 95.98 | 98.83 | 97.19 | 92.05 | 97.34 | 96.83 |
| | Depth L1 ↓ | 3.38 | 3.03 | 3.76 | 2.62 | 2.31 | 4.12 | 8.19 | 2.73 | 3.77 |
| Vox-Fusion * [26] | Acc. ↓ | 1.08 | 1.05 | 1.21 | 1.41 | 0.82 | 1.31 | 1.34 | 1.31 | 1.19 |
| | Comp. ↓ | 1.07 | 1.97 | 1.62 | 1.58 | 0.84 | 1.37 | 1.37 | 1.44 | 1.41 |
| | Comp. Ratio (%) ↑ | 99.46 | 94.76 | 96.37 | 95.80 | 99.12 | 98.20 | 97.55 | 97.32 | 97.32 |
| | Depth L1 ↓ | 1.62 | 10.43 | 3.06 | 4.12 | 2.05 | 2.85 | 3.11 | 4.22 | 3.93 |
| Co-SLAM [8] | Acc. ↓ | 1.11 | 1.14 | 1.17 | 0.99 | 0.76 | 1.36 | 1.44 | 1.24 | 1.15 |
| | Comp. ↓ | 1.06 | 1.37 | 1.14 | 0.92 | 0.78 | 1.33 | 1.36 | 1.16 | 1.14 |
| | Comp. Ratio (%) ↑ | 99.62 | 97.49 | 97.86 | 99.07 | 99.25 | 98.81 | 98.48 | 98.96 | 98.69 |
| | Depth L1 ↓ | 1.54 | 6.41 | 3.05 | 1.66 | 1.68 | 2.71 | 2.55 | 1.82 | 2.68 |
| ESLAM [7] | Acc. ↓ | 1.07 | 0.85 | 0.93 | 0.85 | 0.83 | 1.02 | 1.21 | 0.97 | 0.97 |
| | Comp. ↓ | 1.12 | 0.88 | 1.05 | 0.96 | 0.81 | 1.09 | 1.42 | 1.05 | 1.05 |
| | Comp. Ratio (%) ↑ | 99.06 | 99.64 | 98.84 | 98.34 | 98.85 | 98.60 | 96.80 | 98.60 | 98.60 |
| | Depth L1 ↓ | 0.97 | 1.07 | 1.28 | 0.86 | 1.26 | 1.71 | 1.43 | 1.18 | 1.18 |
| Ours | Acc. ↓ | 0.98 | 0.76 | 0.84 | 0.76 | 0.59 | 0.90 | 1.06 | 1.0 | 0.86 |
| | Comp. ↓ | 0.99 | 0.78 | 0.94 | 0.79 | 0.66 | 0.94 | 1.11 | 1.07 | 0.91 |
| | Comp. Ratio (%) ↑ | 99.65 | 99.76 | 99.13 | 99.50 | 99.41 | 99.21 | 98.95 | 99.23 | 99.36 |
| | Depth L1 ↓ | 0.84 | 0.76 | 1.04 | 0.68 | 0.90 | 1.24 | 1.07 | 0.61 | 0.89 |

4.3. Camera Localization Evaluation

We report the quantitative comparisons of the three datasets in Tables 2–4. In addition, Figure 6 shows a qualitative comparison of camera trajectories for the scene0207 scene of the ScanNet dataset [29]. Our method has more robust tracking performance in both synthetic and real-world datasets and significantly reduces the impact of trajectory drift. This is due to the fact that our method has a more accurate scene representation and is able to guide the tracking process to achieve higher camera localization accuracy.

Table 2. Camera localization results (ATE RMSE [cm] ↓) in eight synthetic scenes using the Replica dataset [28]. Bold formatting is used to emphasize the data in the table. The down arrow (↓) indicates that a smaller indicator is better. The symbol (*) denote the results we reproduced using the corresponding open-source codes.

| Method | room0 | room1 | room2 | office0 | office1 | office2 | office3 | office4 | Avg. |
|------------------|-------------|-------------|-------|-------------|-------------|-------------|-------------|-------------|-------------|
| iMAP * [5] | 3.88 | 3.01 | 2.43 | 2.67 | 1.07 | 4.68 | 4.83 | 2.48 | 3.13 |
| NICE-SLAM [6] | 1.76 | 1.97 | 2.2 | 1.44 | 0.92 | 1.43 | 2.56 | 1.55 | 1.73 |
| VoxFusion * [26] | 0.73 | 1.1 | 1.1 | 7.4 | 1.26 | 1.87 | 0.93 | 1.49 | 1.98 |
| CoSLAM [8] | 0.82 | 2.03 | 1.34 | 0.6 | 0.65 | 2.02 | 1.37 | 0.88 | 1.21 |
| ESLAM [7] | 0.71 | 0.7 | 0.52 | 0.57 | 0.55 | 0.58 | 0.72 | 0.63 | 0.63 |
| Point-SLAM [27] | 0.61 | 0.41 | 0.37 | 0.38 | 0.48 | 0.54 | 0.69 | 0.72 | 0.52 |
| Ours | 0.48 | 0.37 | 0.35 | 0.32 | 0.24 | 0.43 | 0.38 | 0.35 | 0.37 |

Table 3. Camera localization results (ATE RMSE [cm] ↓) in six real-world scenes using the ScanNet dataset [29]. Bold formatting is used to emphasize the data in the table. The down arrow (↓) indicates that a smaller indicator is better. The symbol (*) denote the results we reproduced using the corresponding open-source codes.

| Scene ID | 0000 | 0059 | 0106 | 0169 | 0181 | 0207 | Avg. |
|------------------|------------|------------|------------|------------|------------|------------|-------------|
| iMAP * [5] | 32.2 | 17.3 | 12.0 | 17.4 | 27.9 | 12.7 | 19.42 |
| NICE-SLAM [6] | 13.3 | 12.8 | 7.8 | 13.2 | 13.9 | 6.2 | 11.2 |
| VoxFusion * [26] | 11.6 | 26.3 | 9.1 | 32.3 | 22.1 | 7.4 | 18.13 |
| CoSLAM [8] | 7.9 | 12.6 | 9.5 | 6.6 | 12.9 | 7.1 | 9.43 |
| ESLAM [7] | 7.3 | 8.5 | 7.5 | 6.5 | 9.0 | 5.7 | 7.4 |
| Point-SLAM [27] | 10.24 | 7.81 | 8.65 | 22.16 | 14.77 | 9.54 | 12.19 |
| Ours | 6.8 | 7.8 | 7.2 | 5.6 | 10.3 | 4.2 | 6.98 |

Table 4. Camera localization results (ATE RMSE [cm] ↓) in three real-world scenes using TUM-RGBD [30]. Bold formatting is used to emphasize the data in the table. The down arrow (↓) indicates that a smaller indicator is better. The symbol (*) denote the results we reproduced using the corresponding open-source codes.

| Method | fr1/desk | fr2/xyz | fr3/office | Avg. |
|------------------|-------------|-------------|-------------|-------------|
| iMAP * [5] | 5.9 | 2.2 | 7.6 | 5.23 |
| NICE-SLAM [6] | 2.72 | 31 | 15.2 | 16.31 |
| VoxFusion * [26] | 3.2 | 1.6 | 25.4 | 10.06 |
| CoSLAM [8] | 2.88 | 1.85 | 2.91 | 2.55 |
| ESLAM [7] | 2.47 | 1.11 | 2.42 | 2.00 |
| Point-SLAM [27] | 4.34 | 1.31 | 3.48 | 3.04 |
| Ours | 2.11 | 1.32 | 2.48 | 1.97 |

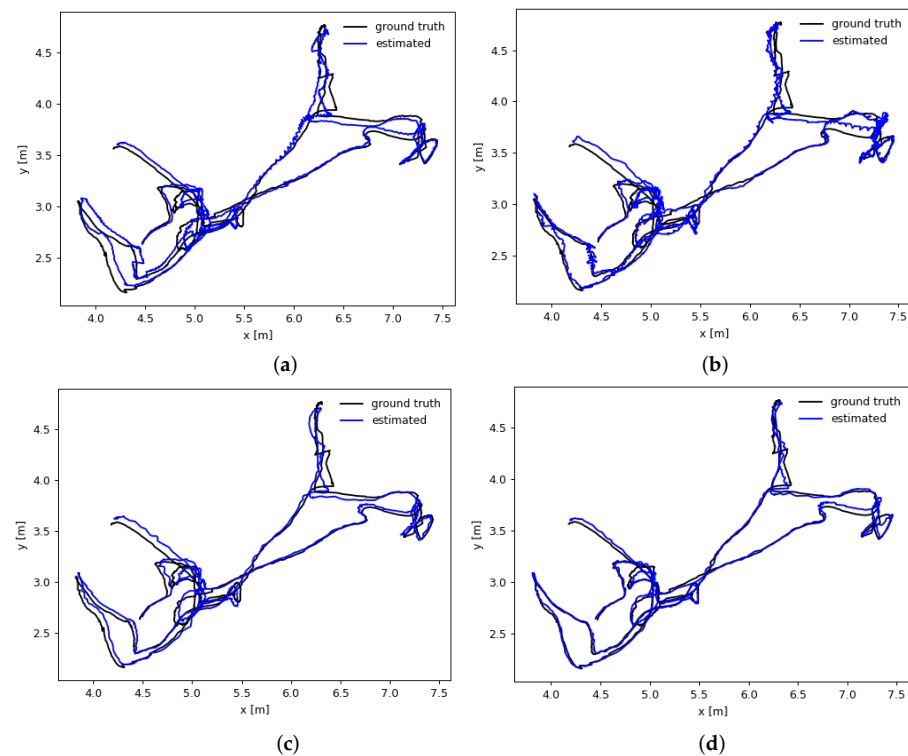


Figure 6. Camera trajectories of (a) NICE-SLAM [6], (b) Co-SLAM [8], (c) E-SLAM [7] and (d) our method for scene0207 from the ScanNet dataset [29].

4.4. Runtime and Memory Usage Analysis

We report the runtime and memory usage for the room0 scene from the Replica dataset [28] in Table 5. The runtimes were analyzed on a desktop PC with a 3.20 GHz Intel Core i9-12900K CPU and a single NVIDIA RTX 3090 GPU. FPS (Hz) is calculated based on the total runtime of the system, and the number of parameters is the sum of the model size of the scene representation and decoders. Our method has the fastest iteration time during the mapping process. iMAP [5] uses a single MLP to represent the entire scene, so it has constant spatial complexity. Vox-Fusion [26] reduces memory usage by lowering voxel resolution, but it results in missed details in scene modeling. Both ESLAM [7] and Co-SLAM [8] utilize feature compression. ESLAM [7] projects features onto axis-aligned feature planes, leading to potential axis-aligned bias. In contrast, Co-SLAM [8] employs hash grids to store the explicit features of sampling points. Hash storage offers certain advantages in reducing memory usage, but it also suffers from hash conflicts, which can lower the reconstruction quality of complex scenes. Although our method is consistent with NICE-SLAM [6] when using tensor grids to store features, it demonstrates significantly more advantages in terms of memory usage compared to NICE-SLAM [6]. Furthermore, we discuss the effect of scene map size on memory usage in Figure 7. For memory usage, both NICE-SLAM [6] and ESLAM [7] are sensitive to map size, whereas our method is not affected by map size, making it more suitable for large-scale scenes.

Table 5. Runtime and memory usage comparison using Replica [28] room0. Bold formatting is used to emphasize the data in the table. The up/down arrow (\uparrow/\downarrow) indicates that a larger/smaller indicator is better. The symbol (*) denote the results we reproduced using the corresponding open-source codes.

| Method | Tracking Time (ms/it) \downarrow | Mapping Time (ms/it) \downarrow | FPS (Hz) \uparrow | Param. (MB) \downarrow |
|-------------------|---------------------------------------|--------------------------------------|------------------------|-----------------------------|
| iMAP * [5] | 34.63 | 20.15 | 0.18 | 0.22 |
| NICE-SLAM [6] | 7.48 | 30.59 | 0.71 | 11.56 |
| Vox-Fusion * [26] | 11.2 | 52.7 | 1.67 | 1.19 |
| Co-SLAM [8] | 6.15 | 14.33 | 10.5 | 0.26 |
| ESLAM [7] | 7.11 | 20.32 | 5.6 | 6.79 |
| Point-SLAM [27] | 12.23 | 35.21 | 0.29 | 27.23 |
| Ours | 6.47 | 10.14 | 9.8 | 10.15 |

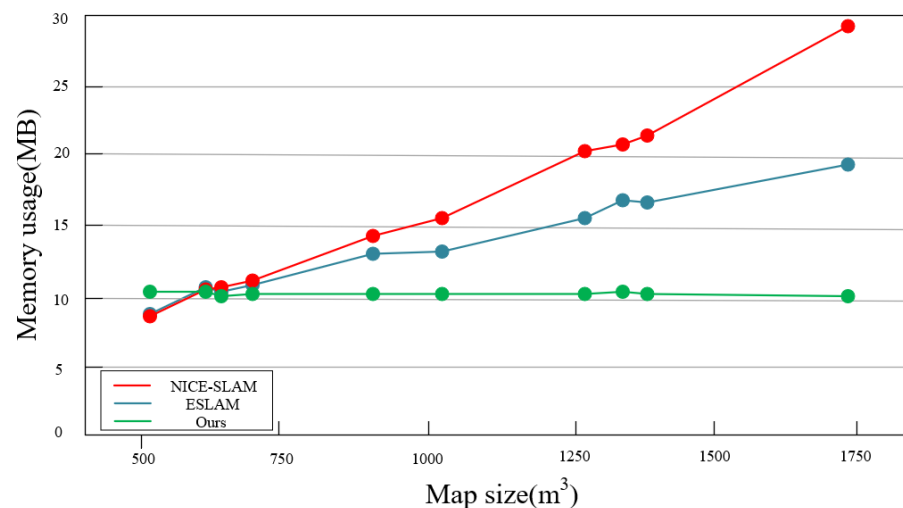


Figure 7. Comparison of the relationship between map size and memory usage changes. We select some scenes from the Replica [28] and ScanNet [29] datasets for evaluation.

4.5. Ablation Study

In this section, we report a series of ablation experiments to validate our design choices. The experimental details are as follows: (1) We did not use separate factor grids; instead, we employed a set of factor grids to represent both scene geometry and appearance. (2) We

eliminated the multi-level basis grids setup, setting only a single level for basis grids. (3) We did not utilize feature integration rendering, instead continuing to employ traditional volume rendering.

Tables 6–8 show a quantitative comparison of the aforementioned three settings with our complete model in terms of localization and reconstruction for the Replica [28] and ScanNet [29] datasets. Our full model has higher accuracy and better completion than other setups. Figure 8 shows the effect of using feature integration rendering on color rendering. When compared to our method without feature integration rendering, our full model achieves a 60% improvement in rendering speed while maintaining rendering quality and is even competitive for peak signal-to-noise ratio (PSNR).

Table 6. Ablation results for reconstruction in eight synthetic scenes using the Replica dataset [28]. Bold formatting is used to emphasize the data in the table. The up/down arrow (↑/↓) indicates that a larger/smaller indicator is better.

| Method | Metric | room0 | room1 | room2 | office0 | office1 | office2 | office3 | office4 | Avg. |
|-----------------------------------|-------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| w/o separate factor grids | Acc. ↓ | 1 | 0.82 | 0.86 | 0.76 | 0.66 | 0.98 | 1.13 | 1.08 | 0.91 |
| | Comp. ↓ | 1.04 | 0.85 | 0.96 | 0.8 | 0.72 | 1 | 1.17 | 1.12 | 0.94 |
| | Comp. Ratio (%) ↑ | 99.47 | 99.62 | 99.14 | 99.54 | 99.31 | 99.35 | 98.90 | 99.09 | 99.3 |
| | Depth L1 ↓ | 0.92 | 1.13 | 1.22 | 0.75 | 1.20 | 1.17 | 1.53 | 1.31 | 1.22 |
| w/o multi-level basis grids | Acc. ↓ | 1.11 | 0.97 | 1.03 | 1.05 | 1.06 | 1.0 | 1.18 | 1.12 | 1.07 |
| | Comp. ↓ | 1.04 | 1 | 1.24 | 0.93 | 0.98 | 0.98 | 1.19 | 1.16 | 1.06 |
| | Comp. Ratio (%) ↑ | 99.63 | 99.49 | 97.46 | 99.58 | 98.62 | 99.52 | 98.99 | 98.92 | 99.03 |
| | Depth L1 ↓ | 0.91 | 1.65 | 2.75 | 1.12 | 1.76 | 1.66 | 1.67 | 1.29 | 1.60 |
| w/o feature integration rendering | Acc. ↓ | 0.98 | 0.79 | 0.86 | 0.79 | 0.73 | 0.93 | 1.08 | 1.03 | 0.90 |
| | Comp. ↓ | 1.01 | 0.81 | 0.98 | 0.82 | 0.71 | 0.96 | 1.14 | 1.09 | 0.94 |
| | Comp. Ratio (%) ↑ | 99.51 | 99.73 | 98.94 | 99.55 | 99.32 | 99.41 | 98.84 | 99.28 | 99.32 |
| | Depth L1 ↓ | 0.82 | 0.89 | 1.22 | 0.7 | 1.01 | 1.57 | 1.23 | 0.77 | 1.03 |
| Ours (Complete model) | Acc. ↓ | 0.98 | 0.76 | 0.84 | 0.76 | 0.59 | 0.90 | 1.06 | 1.0 | 0.86 |
| | Comp. ↓ | 0.99 | 0.78 | 0.94 | 0.79 | 0.66 | 0.94 | 1.11 | 1.07 | 0.91 |
| | Comp. Ratio (%) ↑ | 99.65 | 99.76 | 99.13 | 99.50 | 99.41 | 99.21 | 98.95 | 99.23 | 99.36 |
| | Depth L1 ↓ | 0.84 | 0.76 | 1.04 | 0.68 | 0.90 | 1.24 | 1.07 | 0.61 | 0.89 |

Table 7. Ablation results (ATE RMSE [cm] ↓) for camera localization in eight synthetic scenes using the Replica dataset [28]. Bold formatting is used to emphasize the data in the table. The down arrow (↓) indicates that a smaller indicator is better.

| Method | room0 | room1 | room2 | office0 | office1 | office2 | office3 | office4 | Avg. |
|-----------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| w/o separate factor grids | 0.62 | 0.92 | 0.59 | 0.41 | 0.54 | 0.56 | 0.56 | 0.57 | 0.59 |
| w/o multi-level basis grids | 0.63 | 1.67 | 0.67 | 0.92 | 1.48 | 0.56 | 0.60 | 0.86 | 0.92 |
| w/o feature integration rendering | 0.54 | 0.6 | 0.42 | 0.35 | 0.24 | 0.41 | 0.47 | 0.36 | 0.42 |
| Ours (Complete model) | 0.48 | 0.37 | 0.35 | 0.32 | 0.24 | 0.43 | 0.38 | 0.35 | 0.37 |

Table 8. Ablation results (ATE RMSE [cm] ↓) for camera localization in six real-world scenes using the ScanNet [29] dataset. Bold formatting is used to emphasize the data in the table. The down arrow (↓) indicates that a smaller indicator is better.

| Scene ID | 0000 | 0059 | 0106 | 0169 | 0181 | 0207 | Avg. |
|-----------------------------------|------------|------------|------------|------------|-------------|------------|-------------|
| w/o separate factor grids | 7.4 | 8.5 | 8.1 | 6.2 | 10.8 | 4.4 | 7.57 |
| w/o multi-level basis grids | 7.3 | 9.9 | 8.0 | 6.0 | 11.9 | 5.9 | 8.17 |
| w/o feature integration rendering | 7.3 | 8.5 | 7.6 | 6.4 | 10.8 | 4.5 | 7.51 |
| Ours (Complete model) | 6.8 | 7.8 | 7.2 | 5.6 | 10.3 | 4.2 | 6.98 |



Figure 8. Ablation results for feature integration rendering for Replica [28] room0.

5. Conclusions

In this paper, we present a novel neural implicit dense visual SLAM method. Extensive experiments demonstrate that our approach is capable of reconstructing high-fidelity scenes in real time. The following are the notable contributions of our method:

- (1) We propose a novel scene representation method for neural implicit SLAM that encodes both the geometric and appearance information of a scene as a combination of the basis and coefficient factors. Compared to existing state-of-the-art methods, this representation not only enables higher-quality scene reconstruction but also exhibits a more stable memory growth rate when dealing with incremental reconstruction tasks for large-scale scenes. Furthermore, by employing multi-level tensor grids for the basis factors and a single-level tensor grid for the coefficient factors, our method can more accurately model high-frequency detail regions within the scene. Additionally, this representation significantly enhances camera localization accuracy, thereby demonstrating greater robustness in large-scale scenes.
- (2) In order to enhance the rendering efficiency of neural implicit SLAM, we introduce an efficient rendering approach based on feature integration. Feature integration rendering calculates the approximate features of rays by using a weighted summation of the features at sampling points and then feeds these into a shallow MLP to decode the final pixel colors. Compared to traditional volumetric rendering methods, this rendering approach significantly reduces the number of MLP queries during the color rendering stage, thereby improving real-time performance. Additionally, our method does not rely on customized CUDA frameworks, offering better scalability.
- (3) We conducted extensive experiments on both synthetic and real-world datasets to validate our design choices. Our method shows competitive performance compared to existing state-of-the-art methods in terms of 3D reconstruction, camera localization, runtime, and memory usage. Specifically, for 3D reconstruction, we demonstrate both qualitative and quantitative results for the Replica dataset [28] and qualitative results for the ScanNet dataset [29]. The qualitative results indicate that our method performs better in regions with complex details. In terms of quantitative results, our method's Depth L1 average outperforms other methods by 61%. For camera localization, we conduct comprehensive quantitative evaluations across three datasets, and our method's ATE RMSE average improves by 75%, 46%, and 68%, respectively, compared to existing state-of-the-art methods. We validate the effectiveness of feature integration rendering through a series of ablation studies. In terms of color rendering, our method achieves a 60% speedup compared to traditional volume rendering while maintaining nearly the same PSNR. Additionally, we conduct quantitative experiments to analyze the memory footprint of our method. As the scene map grows, the memory usage of our method remains stable. Even though ESLAM [7] employs feature compression, our method remains competitive in large-scale scenes.

Nonetheless, it is crucial to recognize the following limitations of our method:

- (1) Our approach does not use compression storage for factor grids, which maintains high memory efficiency for large-scale scenes but still consumes memory resources for small-scale or object-centric scenes. Future work will explore more memory-

efficient scene representation methods to enhance memory usage efficiency without compromising localization accuracy and reconstruction quality.

- (2) Due to the utilization of feature integration rendering, our method is prone to artifacts in color rendering when dealing with input frames that have extreme motion blur. In order to address this challenge, future work will focus on designing a deblurring process for the neural implicit SLAM framework. This process will perform blur detection and processing operations on each input frame, ensuring consistency of scene content across multi-view inputs. Consequently, this will enhance the robustness of our neural implicit SLAM method in the presence of extreme motion blur situations.

Author Contributions: Conceptualization, W.W. and J.W.; Methodology, W.W. and J.W.; Software, W.W. and J.W.; Validation, W.W. and J.W.; Formal Analysis, W.W., J.W. and P.S.; Investigation, W.W. and J.W.; Resources, X.X. and J.L.; Data Curation, J.W.; Writing—Original Draft Preparation, W.W. and J.W.; Writing—Review and Editing, J.W., X.X. and P.S.; Visualization, W.W.; Supervision, J.L. and P.S.; Project Administration, P.S.; Funding Acquisition, P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Natural Science Foundation of China, grant number 62262040.

Data Availability Statement: The data that support the findings of this study are openly available at [Replica](#), [ScanNet](#) and [TUM-RGBD](#) were accessed on 9 November 2023.

Acknowledgments: I am deeply grateful to my supervisor for their indispensable guidance and support during this research. I also appreciate everyone who contributed to this study in any capacity. The authors would like to thank the anonymous reviewers for their insightful comments.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [[CrossRef](#)]
2. Mur-Artal, R.; Tardós, J.D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [[CrossRef](#)]
3. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.; Tardós, J.D. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [[CrossRef](#)]
4. Mildenhall, B.; Srinivasan, P.P.; Tancik, M.; Barron, J.T.; Ramamoorthi, R.; Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* **2021**, *65*, 99–106. [[CrossRef](#)]
5. Sucar, E.; Liu, S.; Ortiz, J.; Davison, A.J. iMAP: Implicit mapping and positioning in real-time. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 6229–6238.
6. Zhu, Z.; Peng, S.; Larsson, V.; Xu, W.; Bao, H.; Cui, Z.; Oswald, M.R.; Pollefeys, M. Nice-slam: Neural implicit scalable encoding for slam. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12786–12796.
7. Johari, M.M.; Carta, C.; Fleuret, F. 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, Vancouver, BC, Canada, 17–24 June 2023; pp. 17408–17419.
8. Wang, H.; Wang, J.; Agapito, L. 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, Vancouver, BC, Canada, 17–24 June 2023; pp. 13293–13302.
9. Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; Su, H. Tensorf: Tensorial radiance fields. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 June 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 333–350.
10. Chen, A.; Xu, Z.; Wei, X.; Tang, S.; Su, H.; Geiger, A. Factor fields: A unified framework for neural fields and beyond. *arXiv* **2023**, arXiv:2302.01226.
11. Han, K.; Xiang, W.; Yu, L. Volume Feature Rendering for Fast Neural Radiance Field Reconstruction. *arXiv* **2023**, arXiv:2305.17916.
12. Klein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 225–234.
13. Newcombe, R.A.; Lovegrove, S.J.; Davison, A.J. DTAM: Dense tracking and mapping in real-time. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 2320–2327.

14. Newcombe, R.A.; Izadi, S.; Hilliges, O.; Molyneaux, D.; Kim, D.; Davison, A.J.; Kohi, P.; Shotton, J.; Hodges, S.; Fitzgibbon, A. Kinectfusion: Real-time dense surface mapping and tracking. In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, Basel, Switzerland, 26–29 October 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 127–136.
15. Schops, T.; Sattler, T.; Pollefeys, M. Bad slam: Bundle adjusted direct rgb-d slam. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 134–144.
16. Bloesch, M.; Czarnowski, J.; Clark, R.; Leutenegger, S.; Davison, A.J. Codeslam—Learning a compact, optimisable representation for dense visual slam. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2560–2568.
17. Teed, Z.; Deng, J. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 16558–16569.
18. Sucar, E.; Wada, K.; Davison, A. NodeSLAM: Neural object descriptors for multi-view shape reconstruction. In Proceedings of the 2020 International Conference on 3D Vision (3DV), Fukuoka, Japan, 25–28 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 949–958.
19. Li, R.; Wang, S.; Gu, D. DeepSLAM: A robust monocular SLAM system with unsupervised deep learning. *IEEE Trans. Ind. Electron.* **2020**, *68*, 3577–3587. [[CrossRef](#)]
20. Takikawa, T.; Litalien, J.; Yin, K.; Kreis, K.; Loop, C.; Nowrouzezahrai, D.; Jacobson, A.; McGuire, M.; Fidler, S. 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, Nashville, TN, USA, 20–25 June 2021; pp. 11358–11367.
21. Yu, A.; Li, R.; Tancik, M.; Li, H.; Ng, R.; Kanazawa, A. Plenotrees for real-time rendering of neural radiance fields. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 5752–5761.
22. Sun, C.; Sun, M.; Chen, H.T. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 5459–5469.
23. Li, H.; Yang, X.; Zhai, H.; Liu, Y.; Bao, H.; Zhang, G. Vox-surf: Voxel-based implicit surface representation. *IEEE Trans. Vis. Comput. Graph.* **2022**, *30*, 1743–1755. [[CrossRef](#)] [[PubMed](#)]
24. Chan, E.R.; Lin, C.Z.; Chan, M.A.; Nagano, K.; Pan, B.; De Mello, S.; Gallo, O.; Guibas, L.J.; Tremblay, J.; Khamis, S.; et al. Efficient geometry-aware 3d generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 16123–16133.
25. Müller, T.; Evans, A.; Schied, C.; Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph. (TOG)* **2022**, *41*, 1–15. [[CrossRef](#)]
26. Yang, X.; Li, H.; Zhai, H.; Ming, Y.; Liu, Y.; Zhang, G. Vox-Fusion: Dense tracking and mapping with voxel-based neural implicit representation. In Proceedings of the 2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Singapore, 17–21 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 499–507.
27. Sandström, E.; Li, Y.; Van Gool, L.; Oswald, M.R. Point-slam: Dense neural point cloud-based slam. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–3 October 2023; pp. 18433–18444.
28. Straub, J.; Whelan, T.; Ma, L.; Chen, Y.; Wijmans, E.; Green, S.; Engel, J.J.; Mur-Artal, R.; Ren, C.; Verma, S.; et al. The Replica dataset: A digital replica of indoor spaces. *arXiv* **2019**, arXiv:1906.05797.
29. Dai, A.; Chang, A.X.; Savva, M.; Halber, M.; Funkhouser, T.; Nießner, M. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5828–5839.
30. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 573–580.
31. Dai, A.; Nießner, M.; Zollhöfer, M.; Izadi, S.; Theobalt, C. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph. (ToG)* **2017**, *36*, 1. [[CrossRef](#)]
32. Lorensen, W.E.; Cline, H.E. Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal Graphics: Pioneering Efforts that Shaped the Field*; Association for Computing Machinery: New York, NY, USA, 1998; pp. 347–353.
33. Cignoni, P.; Callieri, M.; Corsini, M.; Dellepiane, M.; Ganovelli, F.; Ranzuglia, G. Meshlab: An open-source mesh processing tool. In Proceedings of the Eurographics Italian Chapter Conference, Salerno, Italy, 2–4 July 2008; Volume 2008, pp. 129–136.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.