2025 IEEE International Conference on Robotics and Automation (ICRA)
May 19-23, 2025. Atlanta, USA

VSS-SLAM: Voxelized Surfel Splatting for Geometally Accurate SLAM

Xuanhua Chen[1], Yunzhou Zhang[1]*, Zhiyao Zhang[1], Guoqing Wang[1], Bin Zhao[1], Xingshuo Wang[1]

***Abstract*— [1] Visual Simultaneous Localization and Mapping (SLAM) helps robots estimate their poses and perceive the environment in unknown settings. Recent work has demonstrated that implicit neural radiance fields and 3D Gaussian Splatting (3DGS) offer higher fidelity scene representation than traditional map representations. We propose VSS-SLAM, which utilizes voxelized surfels as the map representation for incremental mapping in unknown environments. This representation effectively addresses the issue of redundant and disordered primitives encountered in previous methods, thereby enhancing geometric accuracy during reconstruction. Specifically, our approach divides the scene using voxels and stores geometric and appearance information in feature vectors at the voxel vertices. Before rendering, these feature vectors are decoded to generate the corresponding surfels. Additionally, we align camera poses through image and depth rendering. Extensive experiments on the Replica and TUM-RGBD datasets demonstrate that VSS-SLAM delivers high-fidelity reconstruction and accurate pose estimation in both simulated and real-world environments. Source code will soon be available.**

I. Introduction

Visual Simultaneous Localization and Mapping (SLAM) is responsible for estimating camera poses and constructing maps in unknown environments. Over the last decade, researchers have extensively explored different types of map representations. Traditional methods, including point clouds, voxels, and surfels, have become well-established and have greatly facilitated accurate camera pose estimation and optimization. However, these conventional representations typically capture only partial geometric information about the environment, limiting their effectiveness for more complex downstream tasks.

To overcome the limitations of traditional map representations, many studies have started utilizing implicit neural radiance fields (NeRF) as the map representation in SLAM. Implicit NeRF [2]–[6] uses MLP networks to learn the properties of a scene, enabling high-fidelity novel view synthesis. Additionally, some methods that combine implicit NeRF with signed distance functions (SDF) have also achieved surface reconstruction [7]–[9] of the scene. However, since volumetric rendering is optimized based on ray sampling, these NeRF-based SLAM [10]–[13] systems face several challenges, such as low training efficiency and the issue of forgetting.

*The corresponding author of this paper
[1]Xuanhua Chen, Yunzhou Zhang, Zhiyao Zhang, Guoqing Wang, Bin Zhao and Xingshuo Wang are with College of Information Science and Engineering, Northeastern University, Shenyang 110819, China. zhangyunzhou@mail.neu.edu.cn

This work was supported by National Natural Science Foundation of China (No. 61973066) and Major Science and Technology Projects of Liaoning Province(No. 2021JH1/10400049).

Fig. 1: **Demonstration of voxelized surfels.** VSS-SLAM divides the underlying scene using voxels (left) and then generates surfel representations of the scene (right) based on the information within the voxel vertices.

Recently, with the development of 3D Gaussian Splatting [14], [15] (3DGS) technology, several methods [16]–[20] have begun to adopt 3DGS as a map representation, enabling faster novel view synthesis. As an explicit map representation, 3DGS effectively addresses the issue of forgetting. Additionally, its rasterization technique allows for rapid map optimization. However, existing SLAM methods [16]–[18] based on 3DGS often expand the map incrementally in an unstructured manner, resulting in the generation of a large number of redundant Gaussians. These redundant Gaussians cause the expansion to rely heavily on newly added Gaussians while neglecting the existing map. Specifically, when mapping new view data, the system tends to generate numerous new primitives to represent the freshly perceived scene rather than optimizing the existing map. These unstructured redundant Gaussians lead to geometric inaccuracies in the map, ultimately reducing the rendering accuracy of both depth and images.

The above raises a question: **How can we reduce redundant and unstructured primitives during incremental mapping and improve the geometric accuracy of the map?** We propose VSS-SLAM, which uses voxelized surfels to address the issue of geometric inaccuracies while also reducing memory usage. As shown in Fig.1, during incremental mapping, we partition the scene using voxels and store feature vectors at voxel vertices, which are used to decode neural surfels. These voxel vertices generate neural surfels in real time, guiding their structured distribution. Additionally, we achieve accurate pose estimation by directly aligning the camera pose through image and depth rendering.

Our main contributions are as follows:

- We propose VSS-SLAM, a system composed of two

2025 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3315-4139-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICRA55743.2025.11128492

threads: tracking and mapping. The tracking thread directly optimizes the camera pose by rendering images and depth, while the mapping thread leverages voxelized surfels to achieve high-precision reconstruction of the environment.

- We utilize voxels to segment the unknown scene and guide the structured distribution of primitives, effectively reducing memory requirements and improving the geometric accuracy of the map.
- We utilize three MLP networks to decode the information stored in voxel vertices to generate neural surfels. Using neural surfels as the map representation significantly enhances the accuracy of rendering. Additionally, we introduce a learnable scale during decoding to constrain the shape of the neural surfels.
- Our system was evaluated on the Replica [21] and TUM-RGBD [22] datasets, achieving higher rendering accuracy while reducing memory usage by 78%.

II. Related Works

A. NeRF-SLAM

NeRF can train continuous volumetric scene functions from sparse views, enabling high-fidelity scene reconstruction. In recent years, this technique has been widely applied in dense SLAM systems. iMAP [23] was the first SLAM system to use a multilayer perceptron (MLP) as a scene representation, enabling real-time incremental training of implicit scene networks. Nice-SLAM [10] employs a coarse-to-fine hierarchical voxel grid to encode the scene, allowing for local map updates and effectively mitigating the issue of forgetting. Orbeez-SLAM [13] uses ORB-SLAM2 [24] as the visual odometry frontend, combined with a fast NeRF framework to achieve real-time dense map generation. Co-SLAM [12], ESLAM [25], and Point-SLAM [11] use multi-resolution hash grids, multi-scale axis-aligned feature planes, and neural point clouds, respectively, to encode scenes, achieving high-precision surface reconstruction through feature interpolation.

B. 3D Gaussian Splatting

3D Gaussian Splatting [14] uses explicit primitives for scene modeling and achieves fast rendering through rasterization. This approach has been extended to surface reconstruction [26]–[28] tasks, enabling high-precision reconstruction of surfaces. Additionally, some methods manage primitives by introducing data structures like voxels [27] or octrees [29], which enable more structured Gaussian distributions and thus achieve higher rendering accuracy. Other approaches focus on using 3DGS as a SLAM map representation to achieve high-fidelity online reconstruction of scenes. SplaTAM [16] uses isotropic Gaussians to represent scenes, combining them with silhouette masks for fast mapping and pose optimization. MonoGS [17] employs Lie algebra for joint optimization of camera and Gaussian parameters. GS-SLAM [18] introduces a coarse-to-fine strategy for selecting reliable 3D Gaussians to improve camera pose estimation. Photo-SLAM [20] integrates ORB-SLAM3 as the visual odometry frontend and uses a Gaussian pyramid to learn multi-level scene features. GS-ICP-SLAM [19] utilizes GICP [30] as the frontend odometry and achieves fast mapping through scale alignment.

III. Method

A. Overview

As shown in Fig.2, VSS-SLAM estimates camera poses and performs incremental mapping using input RGB-D images. During the incremental mapping process, we employ voxelized surfels as the scene representation. Specifically, we use a voxel grid to segment the unknown scene, storing geometric and appearance information as feature vectors at the voxel vertices. This information is decoded into surfels using three MLP networks, which are subsequently employed for rendering and optimization. In the tracking thread, we directly optimize the camera pose by constructing photometric and depth errors between the rendered data and the real data.

B. Voxelized Surfel

Scene representation. To address the issue of redundant primitives generated during incremental mapping, we propose a scene representation better suited for incremental expansion: voxelized surfels. The voxelized surfel module is divided into the lower layer and the upper layer. The lower layers voxel vertices are used to generate neural surfels and guide their structured distribution. This structured distribution effectively prevents the creation of redundant primitives during incremental mapping, thereby improving mapping quality. The upper layers neural surfels are decoded from the feature vectors stored in the voxel vertices along with a learnable scale, forming the final scene representation used for rendering images and depth.

In the lower layer, we use voxel vertices to represent the structure of the scene. Each voxel vertex is characterized by the following parameters: voxel vertex position $\mathbf{p}_v \in \mathbb{R}^3$, a 32-dimensional feature vector $\mathbf{f}_v \in \mathbb{R}^{32}$, an offset $\delta_v \in \mathbb{R}^{k\times 3}$, and a scale $\mathbf{s}_v \in \mathbb{R}^3$. Here, the feature vector represents the properties of the neural surfels generated by the voxel vertex, while k is a hyperparameter that indicates the number of surfels that can be generated by each voxel vertex.

In the upper layer, we use neural surfels to represent the scene, with each surfel parameterized as a specific 3D Gaussian distribution. Therefore, each surfel is characterized by the following parameters: 3D mean $\mu \in \mathbb{R}^3$, covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{3\times 3}$, color $\mathbf{c} \in \mathbb{R}^3$, and opacity $o \in \mathbb{R}$. During the optimization process, we decouple the covariance matrix into a rotation matrix $\mathbf{R}$ and a scale matrix $\mathbf{S}$:

$$\mathbf{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T \tag{1}$$

The matrix $\mathbf{S}$ is a diagonal matrix where the first two diagonal elements represent the lengths of the two axes of the surfel, and the third element is zero. Therefore, we can further represent the rotation as a quaternion $\mathbf{q} \in \mathbb{R}^4$ and the scale as a 2D vector $\mathbf{s} \in \mathbb{R}^2$.

Decoding. As shown in Fig.2(b), we use three MLP networks to generate neural surfels by decoding the feature

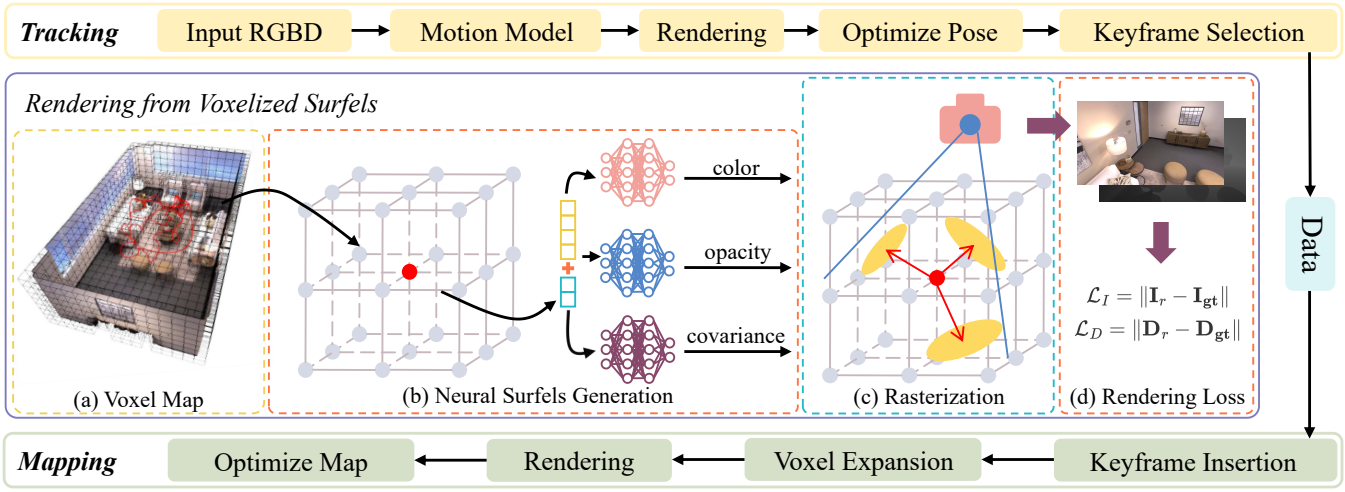


Fig. 2: **An overview of VSS-SLAM.** VSS-SLAM operates with two parallel threads: tracking and mapping. The tracking thread leverages a constant velocity motion model to initialize the camera pose, which is then refined using the voxelized surfel map. Upon determining that the current frame qualifies as a keyframe, the tracking thread transmits the associated data to the mapping thread. The mapping thread subsequently expands the voxel grid based on the camera pose and RGB-D data, followed by rendering and map optimization. The rendering process is structured into four key steps: (a) retrieving all voxel vertices within the camera frustum; (b) decoding the feature vector $\mathbf{f}_v$, offset δ_v, and scale $\mathbf{s}_v$ from the voxel vertices to generate k surfels; (c) rasterizing the surfels to produce the rendered image and depth; (d) constructing an error term from the rendered image and depth to optimize both the map and pose.

vector of the voxel vertex to obtain color, opacity, and covariance (represented by networks F_c, F_o, and F_Σ, respectively). The decoding of color and opacity follows a similar process:

$$\mathbf{c} = F_c(\mathbf{f}_v, \mathbf{r}) \quad o = F_o(\mathbf{f}_v, \mathbf{r}) \tag{2}$$

where $\mathbf{r}$ represents the observation direction from the camera to the voxel vertex. For the estimation of covariance, we directly estimate the quaternion $\mathbf{q} \in \mathbb{R}^4$, which represents the rotation, and a normalized scale $\mathbf{s}_n$:

$$\mathbf{q}, \mathbf{s}_n = F_\Sigma(\mathbf{f}_v, \mathbf{r}) \tag{3}$$

Learnable scale. The decoded color, opacity, and rotation can be directly used as the parameters for the neural surfel. However, for the 3D mean μ and the 2D scale $\mathbf{s}$, these need to be computed using the learnable scale $\mathbf{s}_v$ of the voxel vertex:

$$\mathbf{s} = \mathbf{s}_n \cdot \mathbf{s}_v \quad \mu = \mathbf{p}_v + sigmoid(\mathbf{s}_v \cdot \delta_v) \cdot size \tag{4}$$

where $size$ represents the size of the voxel, and $sigmoid()$ is applied to $\mathbf{s}_v \cdot \delta_v$ to constrain it. This helps ensure that the neural surfels generated by the voxel vertices do not shift too far, thereby preventing damage to the scene's geometric structure.

C. Mapping

The goal of the mapping thread is to initialize the parameters of the voxel vertices and perform training given the camera intrinsics $\mathbf{K}$, pose $\mathbf{T}$, image $\mathbf{I}_{gt}$, and depth $\mathbf{D}_{gt}$ of the i-th frame. The mapping process is divided into three stages: voxel expansion, rendering, and optimization.

Voxel Expansion. First, the depth $\mathbf{D}$ is projected into the camera coordinate system using the intrinsics $\mathbf{K}$ to obtain a point cloud. This point cloud is then transformed into the world coordinate system using the pose $\mathbf{T}$, resulting in the point cloud $\mathcal{P}_w$. For each point $\mathbf{p}_w$ in the point cloud $\mathcal{P}_w$, we compute the corresponding voxel coordinate $\mathbf{v}_w \in \mathbb{R}^3$.

$$\mathbf{v}_w = round(\frac{\mathbf{p}_w}{\mathbf{s}_v}) \cdot size \tag{5}$$

The $round()$ function is used to round the coordinates. Then, for each coordinate, we check: if the coordinate has not been created yet, we create the corresponding voxel vertex; if it has already been created, we skip that coordinate.

Rendering. After creating the new voxel vertices, we generate neural surfels for rendering. For a given set of voxel vertices $\mathcal{V}_w = \{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{m-1}\}$, we obtain the set of neural surfels $\mathcal{S}_w = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{m\times k-1}\}$ according to the decoding process described in III-B. Next, we perform rendering by rasterizing the neural surfels.

Following 2DGS [26], we define the surfels on a local plane in the world coordinate system:

$$Surf(u,v) = \mathbf{H}(u,v,1,1)^T, where \quad \mathbf{H} = \begin{bmatrix} \mathbf{RS} & \mu \\ \mathbf{0} & 1 \end{bmatrix} \tag{6}$$

This local coordinate system can be transformed into Normalized Device Coordinates (NDC) using the perspective projection matrix $\mathbf{P}$ and the camera extrinsic matrix $\mathbf{T}$:

$$(xz, yz, z, 1) = \mathbf{PT}Surf(u,v) = \mathbf{PTH}(u,v,1,1)^T \tag{7}$$

Define the pixel coordinates on the screen as (x, y). The ray originating from this pixel can be parameterized as the intersection of the plane $\mathbf{h}_x = (-1, 0, 0, x)^T$ and the plane $\mathbf{h}_y = (0, -1, 0, y)^T$. In the local coordinates of the surfel, the x and y planes can be expressed as $\mathbf{h}_u = (\mathbf{PTH})^T\mathbf{h}_x$ and $\mathbf{h}_v = (\mathbf{PTH})^T\mathbf{h}_y$. Using (x, y), the local coordinates $\mathbf{u}$ of the intersection point on the surfel can be computed:

$$\mathbf{u}(x,y) = \left(\frac{\mathbf{h}_u^2\mathbf{h}_v^4 - \mathbf{h}_u^4\mathbf{h}_v^2}{\mathbf{h}_u^1\mathbf{h}_v^2 - \mathbf{h}_u^2\mathbf{h}_v^1}, \frac{\mathbf{h}_u^4\mathbf{h}_v^1 - \mathbf{h}_u^1\mathbf{h}_v^4}{\mathbf{h}_u^1\mathbf{h}_v^2 - \mathbf{h}_u^2\mathbf{h}_v^1}\right) \tag{8}$$

Furthermore, we can compute the weight of the intersection point relative to the surfel based on the Gaussian distribution:

$$\alpha(\mathbf{u}) = \exp\left(-\frac{u^2+v^2}{2}\right) \tag{9}$$

Finally, we can compute the color and depth of the pixel:

$$\begin{aligned} \mathbf{C}(x,y) &= \sum_{i=1} \mathbf{c}_i w_i \prod_{j=1}^{i-1}(1-w_j) \\ Z(x,y) &= \sum_i T_i w_i z_i / (\sum_i T_i w_i + \epsilon) \end{aligned} \tag{10}$$

where $w_k = o_k \alpha_k(\mathbf{u}(x,y))$, $T_i = \prod_{j=1}^{i-1}(1 - o_j \alpha_j(\mathbf{u}(\mathbf{x})))$.

Optimization. For each pixel, we render the color and depth to obtain the image $\mathbf{I}_r$ and depth $\mathbf{D}_r$. The input image $\mathbf{I}_{gt}$ and depth $\mathbf{D}_{gt}$ are used as supervisory signals to construct the residual function:

$$\mathcal{L} = a_m \left\|\mathbf{I}_r - \mathbf{I}_{gt}\right\| + (1-a_m)\left\|\mathbf{D}_r - \mathbf{D}_{gt}\right\| \tag{11}$$

where a_m is a hyperparameter with a typical value of 0.8. We optimize the map by minimizing the above loss function. Specifically, we update the information in the voxel vertices and the weights of the decoder network.

D. Tracking

The tracking thread primarily performs the task: pose estimation using the input RGBD data. In our system, the current camera pose $\mathbf{T}_i$ is defined as the transformation from the world coordinate system to the camera coordinate system, with the pose of the previous frame denoted as $\mathbf{T}_{i-1}$. Based on the constant velocity model, the initial value of the current camera pose is set as $\mathbf{T}_i^* = \mathbf{T}_{i-1}\mathbf{T}_{i-2}^{-1}\mathbf{T}_{i-1}$. Using this initial pose, we perform image and depth rendering as in III-C, and then construct the residual:

$$\mathcal{L}_t = (a_t \left\|\mathbf{I}_r - \mathbf{I}\right\| + (1-a_t)\left\|\mathbf{D}_r - \mathbf{D}\right\|) \odot mask \tag{12}$$

where a_t is a hyperparameter with a typical value of 0.65. The $mask$ represents a valid depth mask, which retains only the data with valid depth values. Unlike the mapping process, we keep the information within the voxels and the decoder network fixed. The pose is optimized via gradient descent to minimize the residual mentioned above.

IV. Experiments

A. Experimental Setup

Datasets. We evaluated VSS-SLAM on two well-known datasets: Replica [21] and TUM-RGBD [22]. Replica is a synthetic dataset widely used for evaluating NeRF-base SLAM and 3DGS-base SLAM. TUM-RGBD, on the other hand, is a more challenging real-world dataset with lower image quality and significant motion blur.

Metrics. For the mapping component, we evaluated image rendering quality using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). Depth rendering quality was assessed using L1 loss. Tracking accuracy was evaluated using the Root Mean Square Error (RMSE) of Absolute Trajectory Error (ATE) for each frame.

Baselines. For NeRF-based SLAM, we selected NICE-SLAM [10], Orbeez-SLAM [13] and Point-SLAM [11] as baselines. For 3DGS-based SLAM, we chose SplaTAM [16], GS-SLAM [18], MonoGS [17],GS-ICP-SLAM [19] and Photo-SLAM [20] as baselines.

Implementation details. All evaluations were conducted on a PC with an Intel Core i7-11700 (16 cores @ 2.50GHz), 32GB RAM, and a single NVIDIA GeForce RTX 3090 GPU. The metrics for all methods are averaged over five runs.

B. Rendering Evaluation

As shown in Tab.I, our method demonstrates significant advantages in depth and image rendering quality compared to other baselines. In particular, we achieve an average 50% improvement in depth rendering over the state-of-the-art (SOTA). This indicates that our system performs highly accurate reconstructions and enables high-fidelity image rendering. We attribute this success to using voxelized surfels for incremental mapping, which ensures a structured distribution of primitives. Additionally, the neural surfels decoded through the network enable more precise image and depth rendering, leading to more efficient and accurate scene representation.

As shown in Fig.3, we rendered images and depth from novel viewpoints. In these challenging perspectives, both SplaTAM [16] and GS-ICP-SLAM [19] exhibit significant distortions in the rendered images, while the depth suffer from considerable holes and edge noise. In contrast, our method produces higher-fidelity images and more accurate depth. We attribute this improvement to the voxelized Gaussians, which enable more precise and comprehensive geometric reconstructions. Additionally, we introduced a learnable scale and offset to decode the geometric parameters of the surfels, ensuring that the surfels are properly distributed around the voxel vertices and effectively preventing them from degenerating into pathological, elongated ellipses.

Tab.II shows the rendering performance of our method on the TUM-RGBD [22] dataset. Our approach achieved the best results in both image and depth rendering. The TUM-RGBD [22] dataset is collected in real-world scenarios, where the sensor suffers from significant noise, and the images exhibit noticeable motion blur. These challenges make scene reconstruction particularly difficult, especially regarding the geometric structure of the scene. VSS-SLAM overcame these issues by using voxelized surfels to guide the uniform distribution of primitives, effectively addressing geometric distortions.

C. Tracking Evaluation

As shown in Tab.III, we evaluated the camera ATE RMSE during tracking on the TUM-RGBD [22] dataset. We categorize the methods into classic-based and learning-based approaches. Classic-based methods use established algorithms such as ORB-SLAM2 [24] and GICP [30] as the front-end visual odometry, decoupling tracking from the Gaussian

TABLE I: **Comparison of rendering results on Replica [21].** We report four mapping metrics: PSNR[dB]↑, SSIM↑, LPIPS↓ and Depth L1[cm]↓ .* denotes results obtained by running the official code. Our image rendering metrics are comparable to those of GS-ICP-SLAM [19] and significantly outperform other baselines. In terms of depth rendering metrics, our method outperforms all other approaches. The top three results are highlighted as follows: **first** , second and third .

Method	Metric	Office0	Office1	Office2	Office3	Office4	Room0	Room1	Room2	Avg.
NICE-SLAM* [10] (CVPR2022)	PSNR↑	29.27	30.12	18.96	22.28	24.71	22.23	22.87	24.32	24.35
	SSIM↑	0.876	0.889	0.789	0.799	0.859	0.685	0.767	0.832	0.812
	LPIPS↓	0.229	0.181	0.235	0.209	0.198	0.330	0.271	0.208	0.233
	Depth L1↓	1.41	1.59	2.72	2.15	2.06	1.82	1.39	2.28	1.93
Point-SLAM* [11] (ICCV2023)	PSNR↑	38.72	39.65	34.02	33.29	33.52	32.49	34.02	35.49	35.15
	SSIM↑	0.982	0.987	0.968	0.962	0.972	0.976	**0.979**	0.977	0.975
	LPIPS↓	0.101	0.119	0.149	0.142	0.151	0.109	0.119	0.112	0.125
	Depth L1↓	2.15	3.51	5.10	2.93	4.66	3.16	4.96	5.06	3.94
SplaTAM* [16] (CVPR2024)	PSNR↑	38.44	39.17	31.97	29.70	31.81	32.86	33.89	35.25	34.14
	SSIM↑	0.982	0.981	0.967	0.949	0.945	**0.979**	0.971	**0.984**	0.970
	LPIPS↓	0.084	0.097	0.097	0.119	0.157	0.069	0.097	0.072	0.099
	Depth L1↓	0.38	0.27	0.66	1.35	1.24	0.49	0.44	0.57	0.68
GS-SLAM [18] (CVPR2024)	PSNR↑	38.70	41.17	32.36	32.03	32.92	31.56	32.86	32.59	34.27
	SSIM↑	**0.986**	**0.993**	**0.978**	**0.970**	0.968	0.968	0.973	0.971	**0.976**
	LPIPS↓	0.05	**0.033**	0.094	0.110	0.112	0.094	0.075	0.093	0.083
	Depth L1↓	0.81	0.96	1.41	1.53	1.08	1.31	0.82	1.26	1.15
GS-ICP-SLAM* [19] (ECCV2024)	PSNR↑	41.54	42.57	36.06	36.23	38.32	34.70	37.16	37.74	38.04
	SSIM↑	0.979	0.982	0.971	0.965	0.970	0.957	0.967	0.971	0.970
	LPIPS↓	0.038	0.038	0.050	0.047	0.050	0.055	0.052	0.056	0.048
	Depth L1↓	2.94	2.28	4.18	3.73	3.56	7.03	4.81	4.03	4.07
Ours	PSNR↑	**42.64**	**42.90**	**37.42**	**37.25**	**39.09**	**35.98**	**38.89**	**37.91**	**39.01**
	SSIM↑	0.983	0.981	0.975	0.965	**0.973**	0.978	0.972	0.972	0.975
	LPIPS↓	**0.032**	**0.033**	**0.046**	**0.042**	**0.043**	**0.043**	**0.038**	**0.062**	**0.042**
	Depth L1↓	**0.23**	**0.13**	**0.29**	**0.50**	**0.35**	**0.45**	**0.26**	**0.51**	**0.34**

Fig. 3: **Novel View Rendering on Replica.** In these challenging viewpoints, GS-ICP-SLAM [19] and SplaTAM [16] exhibited significant holes and distortions in both image and depth rendering. VSS-SLAM achieved superior performance in both image and depth rendering, attributed to its precise geometric reconstruction.

TABLE II: **Comparison of rendering results on TUM-RGBD [22].** We report three mapping metrics: PSNR[dB], SSIM, LPIPS.

Method	PSNR[dB]↑	SSIM↑	LILPS↓
NICE-SLAM [10]	14.10	0.574	0.395
Point-SLAM [11]	21.40	0.738	0.447
SplaTAM [16]	23.46	0.906	0.156
Photo-SLAM [20]	21.90	0.763	0.187
GS-ICP-SLAM [19]	20.39	0.762	0.227
Ours	**24.57**	**0.919**	**0.124**

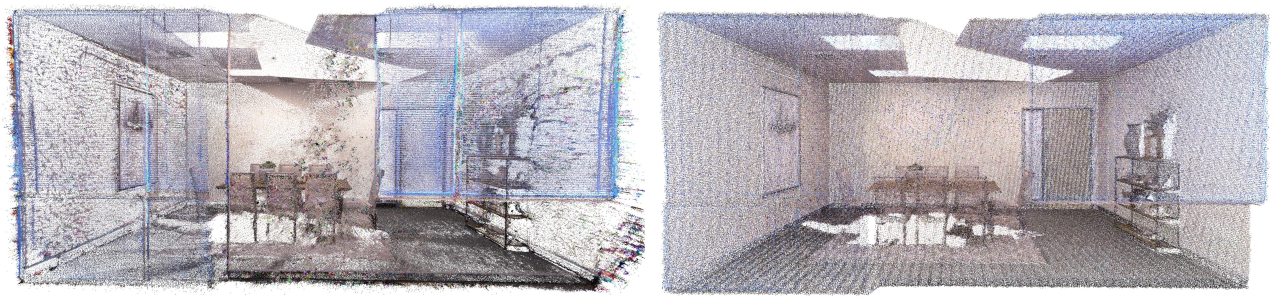
map or NeRF map. Learning-based methods leverage the constructed scene map to directly align camera poses through rendered images and depth. Orbeez-SLAM [13] achieved the best tracking performance, indicating that the robustness of classic methods still holds a strong advantage. Among the learning-based methods, our approach demonstrated the best average performance.

As shown in Tab.IV, we evaluated eight sequences from the Replica dataset and reported the ATE RMSE of the

TABLE III: **Tracking performance comparison on the TUM-RGBD [22].** We report the camera ATE RMSE[cm]↓ during tracking. The methods are categorized into classic-based and learning-based approaches. Among the learning-based methods, our approach achieved the best average performance.

	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
classic	Orbeez-SLAM [13]	**1.90**	**0.30**	**1.00**	**1.07**
	GS-ICP-SLAM [19]	2.70	1.80	2.70	2.40
	Photo-SLAM [20]	2.60	0.35	**1.00**	1.32
learning	NICE-SLAM [10]	2.80	2.10	7.20	4.03
	Point-SLAM [11]	2.70	1.30	3.90	2.63
	GS-SLAM [18]	3.30	1.32	6.60	3.74
	SplaTAM [16]	3.35	1.24	5.16	3.25
	Mono-GS [17]	**1.49**	1.60	1.55	1.55
	Ours	1.56	**1.39**	**1.51**	**1.49**

camera during tracking. Among the learning-based methods, we achieved the best performance. Unlike classical methods, learning-based approaches align the camera pose by rendering images and depth. Consequently, the tracking accuracy



(a) Non-voxelized (b) Voxelized (Ours)

Fig. 4: **Effectiveness of Voxelization.** We demonstrate the mapping results as a colored point cloud to compare two different primitive expansion strategies during the incremental mapping process: non-voxelized (SplaTAM [16]/Surfel-SLAM) and voxelized (VSS-SLAM/V3-SLAM). It is evident that the non-voxelized strategy results in a large number of redundant primitives that are incorrectly distributed, where as the voxelized strategy employs fewer primitives to represent the scene more accurately.

TABLE IV: **Tracking performance comparison on the Replica [21].** We report the ATE RMSE[cm]↓ of the camera during tracking across 8 sequences.

	Method	o0	o1	o2	o3	o4	r0	r1	r2	Avg.
classic	Photo-SLAM [20]	-	-	-	-	-	-	-	-	0.60
	GS-ICP-SLAM [19]	**0.18**	**0.12**	**0.17**	**0.16**	**0.21**	**0.16**	**0.15**	**0.16**	**0.11**
learning	NICE-SLAM [10]	0.88	1.00	1.06	1.10	1.13	0.97	1.31	1.07	1.07
	Point-SLAM [11]	0.38	0.48	0.54	0.69	0.72	0.61	0.41	0.37	0.53
	GS-SLAM [18]	0.52	0.41	0.59	0.46	0.70	0.48	0.53	0.33	0.50
	SplaTAM [16]	0.44	0.27	**0.26**	0.36	0.56	0.30	0.45	0.31	0.37
	Ours	**0.32**	**0.22**	0.28	**0.29**	**0.39**	**0.25**	**0.30**	**0.21**	**0.28**

TABLE V: **Voxelized surfels ablation on Replica [21].** We report the average mapping performance and memory usage [MB] across 8 sequences.

	Method	PSNR↑	SSIM↑	LILPS↓	Depth L1↓	Memory↓
	SplaTAM [16]	34.14	0.970	0.099	0.68	289.8
Ours	Surfel-SLAM	36.55	0.971	0.072	0.42	278.2
	V3-SLAM	38.52	0.974	0.050	0.62	65.5
	VSS-SLAM	**39.01**	**0.975**	**0.042**	**0.34**	**59.6**

largely depends on the quality of the mapping. The voxelized surfel representation constructs a higher-fidelity map, which reduces tracking errors.

D. Ablation

VSS-SLAM primarily uses voxelized surfels as the foundational representation for incremental mapping. The voxelized surfel is divided into two components: the voxel vertex at the lower layer and the neural surfels at the upper layer. To separately evaluate their effectiveness, we designed two additional systems for experimentation. We evaluated their performance in image rendering, depth rendering, and model memory usage on the Replica dataset. Additionally, we used SplaTAM [16] as a representative of mainstream 3DGS-based SLAM.

Effectiveness of surfel. We designed the first experimental system, Surfel-SLAM, where the voxel vertex was removed, and surfels were directly used as the map's primitives. Notably, Surfel-SLAM was implemented based on SplaTAM. As shown in Tab.V, Surfel-SLAM significantly improves both image and depth rendering performance when using surfels instead of 3DGS as the primitive. This improvement is attributed to the fact that surfels can project more accurately onto the camera plane.

TABLE VI: **Learnable scale ablation on Replica [21].** We report the four mapping metrics of our method without (**w/o**) and with (**w**) learnable scale.

Method	PSNR[dB]↑	SSIM↑	LILPS↓	Depth L1[cm]↓
w/o learnable scale	36.24	0.968	0.102	0.56
w learnable scale	**39.01**	**0.975**	**0.042**	**0.34**

Effectiveness of voxelization. We designed the second experimental system, Voxelized-3DGS-SLAM (V3-SLAM), to demonstrate the effectiveness of voxel-based scene expansion. Compared to VSS-SLAM, V3-SLAM retains the voxel vertices at the lower layer but replaces the surfels with 3D Gaussians. As shown in Fig.4, we illustrate the impact of using voxels for expanding unknown scenes on the distribution of primitives. The results indicate that voxel-based scene partitioning effectively guides the distribution of primitives to the correct locations and reduces the generation of redundant Gaussians. Our method delivers better rendering performance while using only 22% of the memory compared to the SplaTAM model.

Effectiveness of the learnable scale. As shown in Tab.VI, we verified the effectiveness of the learnable scale in mapping on the Replica dataset. After introducing the learnable scale, the system achieved higher fidelity rendering and more accurate geometric reconstruction. This improvement is due to the learnable scale effectively constraining the geometric properties of the neural surfels during decoding, preventing them from degenerating into ill-shaped, elongated ellipses.

V. Conclusion

In this paper, we propose a novel SLAM, VSS-SLAM, which employs voxelized surfels for incremental mapping in unknown environments. Utilizing voxelized surfels as a map representation effectively reduces redundant primitives and improves the accuracy of geometric reconstruction. Additionally, we directly align the camera pose through image and depth rendering. Extensive experiments on benchmark datasets demonstrate that VSS-SLAM excels in image rendering, geometric reconstruction, and pose estimation.

REFERENCES

[1] Ethan Elms, Yasir Latif, Tae Ha Park, and Tat-Jun Chin. Event-based structure-from-orbit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19541–19550, June 2024.

[2] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99106, dec 2021.

[3] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[4] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[5] Wenpu Li, Pian Wan, Peng Wang, Jinghang Li, Yi Zhou, and Peidong Liu. Benerf: Neural radiance fields from a single blurry image and event stream. In *European Conference on Computer Vision (ECCV)*, 2024.

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

[7] Jingwen Wang, Tymoteusz Bleja, and Lourdes Agapito. Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. In *2022 International Conference on 3D Vision (3DV)*, pages 433–442. IEEE, 2022.

[8] Radu Alexandru Rosu and Sven Behnke. Permutosdf: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8466–8475, 2023.

[9] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021.

[10] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12776–12786, 2022.

[11] Erik Sandstrm, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-slam: Dense neural point cloud-based slam. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18387–18398, 2023.

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

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

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

[15] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19447–19456, June 2024.

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

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

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

[19] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *European Conference on Computer Vision (ECCV)*, 2024.

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

[21] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[22] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012.

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

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

[25] Mohammad Mahdi Johari, Camilla Carta, and François 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 (CVPR)*, pages 17408–17419, June 2023.

[26] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024.

[27] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5354–5363, June 2024.

[28] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024.

[29] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898*, 2024.

[30] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.