HVOFusion: Incremental Mesh Reconstruction Using Hybrid Voxel Octree

Shaofan Liu[1] , **Junbo Chen**[2] , **Jianke Zhu**[1*]
[1]Zhejiang University
[2]Udeer.ai
{liushaofan, jkzhu}@zju.edu.cn, junbo@udeer.ai

Abstract

Incremental scene reconstruction is essential to the navigation in robotics. Most of the conventional methods typically make use of either TSDF (truncated signed distance functions) volume or neural networks to implicitly represent the surface. Due to the voxel representation or involving with time-consuming sampling, they have difficulty in balancing speed, memory storage, and surface quality. In this paper, we propose a novel hybrid voxel-octree approach to effectively fuse octree with voxel structures so that we can take advantage of both implicit surface and explicit triangular mesh representation. Such sparse structure preserves triangular faces in the leaf nodes and produces partial meshes sequentially for incremental reconstruction. This storage scheme allows us to naturally optimize the mesh in explicit 3D space to achieve higher surface quality. We iteratively deform the mesh towards the target and recovers vertex colors by optimizing a shading model. Experimental results on several datasets show that our proposed approach is capable of quickly and accurately reconstructing a scene with realistic colors. Code is available at https://github.com/Frankuzi/HVOFusion.

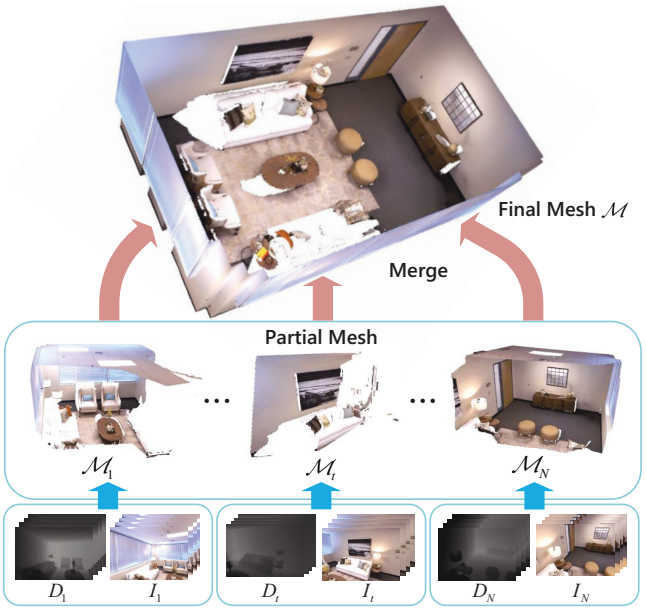


Figure 1: We reconstruct the scene in an incremental manner. The partial mesh $\mathcal{M}_t$ output at time t is constructed by the hybrid voxel-octree and deformed in the refinement branch. We merge all the partial meshes to get the final mesh $\mathcal{M}$.

1 Introduction

Incremental scene reconstruction has achieved great progress in the recent years, which enables a wide range of applications, including motion planning [Breitenmoser and Siegwart, 2012], dense mapping [Whelan *et al.*, 2015] and scene perception [Häne *et al.*, 2017]. In contrast to offline scheme, the online reconstruction provides real-time feedback and adapt to dynamic scenes. Since the success of KinectFusion [Izadi *et al.*, 2011] with Truncated Signed Distance Function (TSDF) integration on a set of voxels [Curless and Levoy, 1996], the underlying scene representation has seldom changed ever. It always relies on a large TSDF volume that usually costs a huge amount of memory, and the voxel resolution limitations restrict the geometric accuracy of complex scenes [Kim *et al.*, 2010].

On the other hand, recent research studies [Ortiz *et al.*, 2022; Zhu *et al.*, 2022; Yang *et al.*, 2022] have demonstrated the promising results of implicit representation parameterized by neural networks. By formulating the scene surface as a continuous implicit function, its geometry can be extracted at arbitrary resolution. However, most of these methods are trained by minimizing the reconstruction loss of function values at a set of sampled 3D locations, which requires expensive sampling strategies to locate the surface. Recently, 3D Gaussian Splatting [Kerbl *et al.*, 2023] attempts to represent the geometry by implicit ellipsoids in an explicit 3D space. The mesh and attributes encoded in 3D Gaussians cannot be easily extracted for various downstream tasks.

A possible solution to solve these problems is to make use of explicit surface representation [Shen *et al.*, 2021; Worchel *et al.*, 2022; Munkberg *et al.*, 2022; Hanocka *et al.*, 2020; Walker *et al.*, 2023], which directly optimize mesh vertices

*Corresponding authors

coordinates and topological structures without intensive sampling. Therefore, it is able to trade-off between the efficiency and performance. Unfortunately, most of these methods make assumption of a given and fixed mesh topology, which are typically suitable for recovering individual objects. It is usually hard to incrementally reconstruct the scene geometry that does not have the specific boundaries.

To address the above challenges, we introduce HVOFusion, incremental mesh reconstruction using hybrid voxel-octree. The hybrid voxel-octree fuses octree and voxel structures, where the leaf nodes consist of a hierarchical voxel. This hybrid structure can achieve higher storage precision without increasing the octree depth. The key feature of this structure is that it leverages both implicit and explicit triangular mesh representations. Specifically, the leaf node predicts a coarse triangular mesh through an implicit function, where the resolution of the triangular mesh depends on the level of the voxel. The triangular mesh is explicitly stored in the leaf nodes of the octree so that the scene mesh can be incrementally reconstructed while the octree is being built. As shown in Fig.1, this structure incrementally reconstructs the partial meshes of the scene with sequential input. This storage scheme allows us to naturally optimize the mesh in explicit 3D space to achieve higher surface quality. We refine the partial mesh extracted by the octree through image and point cloud supervision. Specifically, we iteratively deform the mesh towards the target, in which spherical harmonics are employed to model the vertex colors and ambient lighting. To the best of our knowledge, our method is the first to combine explicit surface optimization with online scene reconstruction. We conduct experiments on various datasets, whose promising results show the fast and accurate scene reconstruction with realistic colors.

The main contributions are summarized as follows:

- An incremental mesh reconstruction using hybrid voxel-octree.
- We propose the hybrid voxel-octree, which combines the advantages of octree and voxel structures, and integrates implicit and explicit triangular mesh representations.
- We adopt images and point clouds as supervisions which allow us to effectively match the colors and lighting of the scene and directly enhance surface recovery.

2 Related Work

2.1 TSDF-based Reconstruction

A wide range of methods, including stereo matching and SLAM-based techniques, have been investigated for traditional scene reconstruction [Engel *et al.*, 2013; Schonberger and Frahm, 2016]. KinectFusion [Izadi *et al.*, 2011] is the seminal work on this task. Many following studies focused on volume-based methods, including Kintinuous [Whelan *et al.*, 2012], BundleFusion [Dai *et al.*, 2017], InfiniTAM [Prisacariu *et al.*, 2017] and Voxgraph [Reijgwart *et al.*, 2019]. Usually, the resolution of each voxel is often fixed in practice due to efficiency, which limits the capability of such representation. Other methods attempt to address this problem by representing the scene by a set of points or surfels [Schöps *et al.*, 2019]. The main drawback of point-based method is their discrete nature, which does not have the important topology information. In this work, we employ the hybrid voxel-octree structure to store triangular faces so that the explicit meshes can be deformed to the target scene. Thus, our method are able to achieve higher reconstruction accuracy while consuming less memory.

2.2 Neural Scene Reconstruction

[Mildenhall *et al.*, 2021] represent the scene as a neural radiance field, which reconstruct indoor scenes through implicit neural representations. Later, iMAP [Sucar *et al.*, 2021] performs both tracking and mapping using neural implicit representations. To improve the scalability, NICE-SLAM [Zhu *et al.*, 2022] suggest a hierarchical multi-feature grids. Several follow-up works have improved upon these approaches from various perspectives. ESLAM [Johari *et al.*, 2023], iDF-SLAM [Ming *et al.*, 2022], and iSDF [Ortiz *et al.*, 2022] leverage surface rendering-based methods to improve the accuracy of reconstruction. Recently, 3D Gaussian splatting [Kerbl *et al.*, 2023] optimizes and renders scenes very efficiently with high quality. SuGaR [Guédon and Lepetit, 2023] extracts an accurate mesh from the Gaussians. SplaTAM [Keetha *et al.*, 2023] simultaneously estimates the camera poses and fit the underlying Gaussian splatting.

2.3 Explicit Surface Representation

Explicit surface reconstruction methods aim to estimate 3D triangular mesh from images. Meshes in a scene are firstly rasterized to create a screen-space map containing geometric information. The raster images are then processed by a shader to predict pixel-wise RGB values. These methods [Wang *et al.*, 2018; Liu *et al.*, 2019] have achieved impressive results in reconstructing and synthesizing simple shapes. DefTet [Gao *et al.*, 2020] represents a mesh with a deformable tetrahedral grid where the grid vertex coordinates and the occupancy values are learned from the neural network. DMTet [Shen *et al.*, 2021] optimizes the surface mesh directly through a differentiable marching tetrahedral layer. [Thies *et al.*, 2019] fully parameterize a deferred shader by a neural network, and hence synthesize novel views with arbitrary illumination and materials. In spite of the promising results, these methods are limited to smaller objects, which are hard to scale into larger scenes. To this end, our proposed approach incrementally reconstructs the large scene and refines each partial reconstruction individually.

3 Method

In contrast to the conventional offline methods, we aim to reconstruct the large scale scene in an online manner, as shown in Fig.1. Give a stream of input image I_t with the depth D_t or the point cloud P_t, we reconstruct a partial mesh $\mathcal{M}_t = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, which consists of vertex positions $\mathcal{V}$, a set of edges $\mathcal{E}$, and a set of faces $\mathcal{F}$. The final mesh $\mathcal{M}$ is composed of N partial meshes. The partial mesh is extracted and stored by hybrid voxel-octree (Sec. 3.1). We optimize partial mesh by point-based refinement (Sec. 3.2) and shading-based refinement (Sec. 3.3) to obtain refined partial mesh. Fig.2 illustrates the overall framework of our presented method.

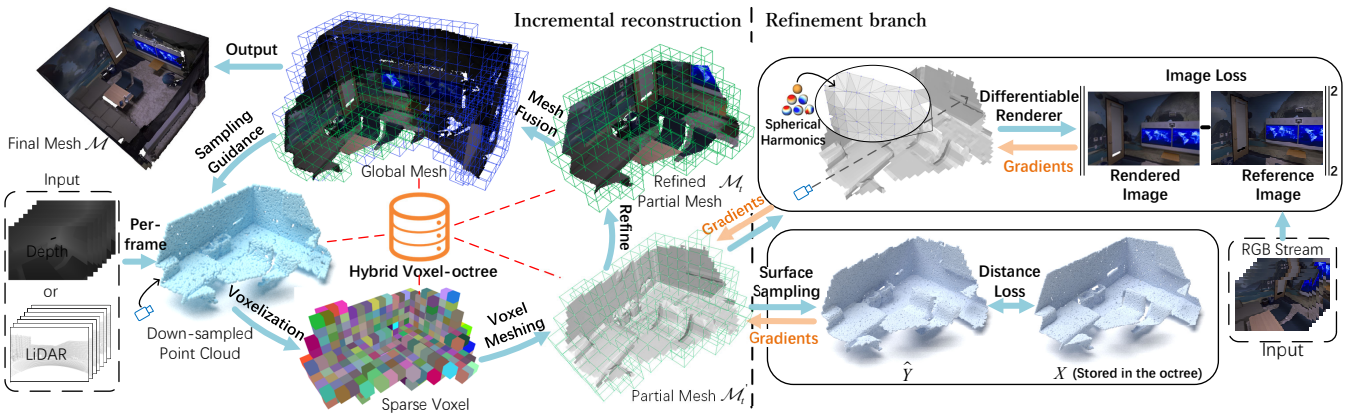


Figure 2: The architecture of our proposed incremental mesh reconstruction approach. In the incremental reconstruction pipeline, each frame of the point cloud is downsampled and inserted into the hybrid voxel-octree, resulting in a sparse voxel structure composed of leaf nodes. The hybrid voxel-octree is then used to extract partial mesh $\mathcal{M}'_t$. The refined partial mesh $\mathcal{M}_t$ is finally fused into the global mesh.

3.1 Hybrid Voxel-octree

Hybrid Voxel-octree Surface Representation

Generally, it is easy to implement the voxel-based surface representation, which enables the parallel data access. However, it often consumes large amount of memory. Instead, the octree-based surface representation offers high spatial efficiency while their complexity increases along with the depth.

To tackle these issues, we propose a novel hybrid voxel-octree structure that is able to take advantages of these two kinds of representations. Instead of using the single list like the typical octree, the leaf nodes of our voxel-octree are composed of the hierarchical voxel, which can achieve higher storage precision without increasing the octree depth. As shown in Fig.3, we insert the points into the voxel at the leaf node of voxel-octree. Then, we compute the normal vector for each point to determine the average curvature of the voxel. When the average curvature is above the threshold $\mathcal{T}_{cur}$, we subdivide it into L^3 multi-layered voxels. L represents the level of the voxel. Such scheme enables to generate the more detailed triangular faces without increasing the octree depth or reinserting points. During the construction of the octree, we extract the triangulated mesh on the leaf nodes to obtain the partial surface reconstruction. It is worthy of noting that we only store the vertices and topology of the triangular faces, which greatly reduces the memory consumption.

Voxelization

The sparse point cloud is obtained from either LiDAR scanner or the back-projection of depth map. We insert each point into the leaf nodes of the octree, which is further stored in the hierarchical voxel. To reduce the computational complexity, the point cloud is downsampled at a certain ratio in the insertion process. The minimum $\mathcal{L}_1$ distance from the current point to the other points within the voxel is employed to determine whether the point could be inserted into the voxel. In our implemntation, we choose the points whose $\mathcal{L}_1$ distances fall below the threshold $\mathcal{T}_{min}$. As shown in Fig. 3, we encode voxel coordinates as Morton codes for fast voxel indexing, which is able to locate voxel of leaf node and its 26 adjacent nodes in the time complexity of $O(1)$.

Voxel Meshing

For each leaf node in the octree, the goal of voxel meshing is to find a zero level-set $\mathcal{S}$ within the voxel, which is equivalent to the implicit surface to be recovered. Unlike TSDF-based methods that predict only one TSDF value for each voxel, as shown in Fig.4, we treat each voxel as a cube and compute the signed distance function (SDF) by measuring the distance from its cube corners to the zero level-set

$$SDF(p_i) = \mathbf{n}_q^T(q - p_i), i \in [0, (L+1)^3) \quad (1)$$

where p_i represents the corner of the cube. q denotes the nearest neighbor of p_i, and $\mathbf{n}_q$ represents the normal vector of the point q. The number of corner point p_i is determined by the voxel's level L.

The TSDF-based methods, such as voxel hashing [Nießner *et al.*, 2013], typically employ the Marching Cubes algorithm [Lorensen and Cline, 1987] to extract triangulated mesh from a large TSDF volume stored in hashing. However, these methods usually adopt voxels of the same size to ensure that the triangular faces can be correctly extracted from the adjacent 8 voxels. To improve the surface accuracy, they use smaller voxel size to increase the number of triangular faces, which causes inefficient computations for regions with simple geometry. In contrast, we suggest a sparse voxel Marching Cubes method, which supports extracting triangular faces from voxels with different levels. This method is able to achieve more accurate surface reconstruction without changing the octree structure.

Specifically, when extracting triangular faces within the voxel, the vertices of triangular faces are located on the boundaries of the voxels. As shown in Fig.4, the positions of the vertices p_0 to p_6 and q_0 to q_6 are obviously consistent when the adjacent voxels are at the same level. This is because the adjacent voxels have the same corner division at the connection. When adjacent voxels are at different levels, we use the vertices p_0 to p_6 of the high-level voxel to linearly approximate the vertices q_0 to q_6 of the low-level voxel, resulting in vertices p'_0 to p'_6. This ensures the consistency of the vertex positions between any voxels.

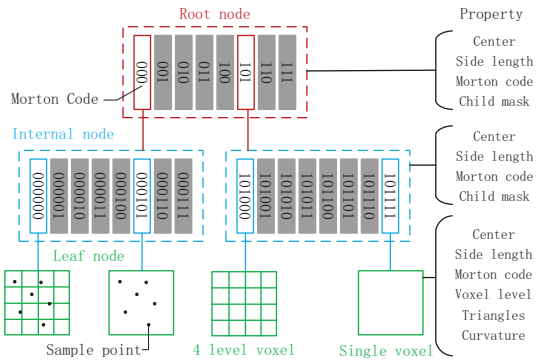


Figure 3: Example of the structure and node properties of the hybrid voxel-octree. Each node is indicated by a binary Morton code representing its position.

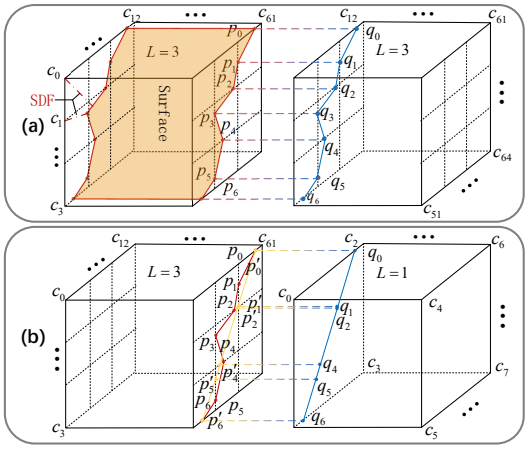


Figure 4: Example of voxel meshing. c_n is the voxel's corner related to voxel level. p_n and q_n are vertices of triangular face within the voxel. a). The positions of p_n and q_n are consistent when adjacent voxels are at the same level b). To ensure consistency, p_n is modified to p'_n when adjacent voxels are at different levels.

3.2 Point-based Refinement

The TSDF-based reconstruction methods [Izadi *et al.*, 2011; Nießner *et al.*, 2013] may have difficulty accurately representing complex and intricate regions due to the restrictions on voxel resolution. Instead, by directly preserve the triangular faces, we can refine the mesh in explicit 3D space to overcome this limitation. We iteratively deform the partial mesh generated by the hybrid voxel-octree towards the input point cloud X. The point cloud X is stored in the octree, which effectively preserves the detailed scene geometry. Specifically, we optimize the differential vertex position Δv_t of the mesh rather than its absolute coordinates. The deformed vertex position can be represented as $v_t = v_{t-1} + \Delta v_t$. In contrast the conventional methods [Sharf *et al.*, 2006; Hanocka *et al.*, 2020] learning the priors through neural networks, we directly treat the partial mesh as an explicit geometric prior, which enables the rapid convergence of vertices towards their optimal positions and without any neural networks. The vertices of refined partial mesh M_t are driven by the distance of the partial mesh M'_t to the input point cloud X. We sample the deformed mesh M'_t to obtain the sampled points $\hat{Y}$. Then, we minimize the Chamfer distance between point cloud X and $\hat{Y}$ as follows

$$\mathcal{L}_{poins} = \sum_{x \in X} \min_{\hat{y} \in \hat{Y}} \|x - \hat{y}\|_2 + \sum_{\hat{y} \in \hat{Y}} \min_{x \in X} \|x - \hat{y}\|_2 \quad (2)$$

In this work, we adopt the same differential sampling mechanism like [Hanocka *et al.*, 2020]. The distance between the sampled points and the corresponding point cloud contributes to a gradient update, which guides the update of the three vertices coordinates of each triangle.

3.3 Shading-based Refinement

Lighting Estimation

As in [Wu *et al.*, 2011], we learn the vertex colors of the triangular mesh and refine the surface through optimizing the lighting model. Our method is able to recover more realistic colors compared to implicit methods that calculate vertex colors through integration of cumulative color. In general, the irradiance equation for each vertex can be defined as below

$$B(i, \mathbf{n}) = \rho(i) S(\mathbf{n}) + \beta(i) \quad (3)$$

where $B(i, \mathbf{n})$ represents the image lighting at each vertex. $S(\mathbf{n})$ denotes the shading, and $\rho(i)$ is the vertex albedo to adjust the local shadow intensity. $\beta(i)$ accounts for local lighting variations. For simplicity, $\beta(i)$ is set to zero during the training. We describe the local lighting and shadows as low order spherical harmonic. Therefore, the shading can be formulated as a linear polynomial of the normal vector and spherical harmonic coefficients

$$S(\mathbf{n}) = \sum_{k=1}^{4} l_k \mathbf{n} \quad (4)$$

where l represents a vector of the four first order spherical harmonics coefficients. $\mathbf{n}$ denotes vertex normal.

Albedo Recovery

The shading estimation in Eq.4 is a rough assessment of the lighting, which only holds for diffuse surfaces with uniform albedo. It has difficulties in handling the specularities, shadows, and nearby light sources that may exist in the real-world scenarios. Therefore, we simultaneously optimize both the albedo and the spherical harmonic in Eq.3. Similar to [Worchel *et al.*, 2022], we adopt a differentiable deferred shading pipeline to render into images, which is inspired by real-time graphics. Given the camera parameter, the mesh is rasterized to yield the per pixel triangle index and barycentric coordinates, which are further employed to interpolate both the vertex albedo and vertex spherical harmonics. Our goal is to minimize the differences between the rendered image and the input frame as follows

$$\mathcal{L}_{shading} = \|B(i, \mathbf{n}) - I\|_2^2 \quad (5)$$

Since the gradient of $\mathcal{L}_{shading}$ can be backpropagated to the mesh vertices, we are able to recover the details of the surface geometry and the corresponding vertex colors.

3.4 Objective Function

Directly deforming the mesh may lead to the undesirable meshes with degenerated triangles and self-intersections. To

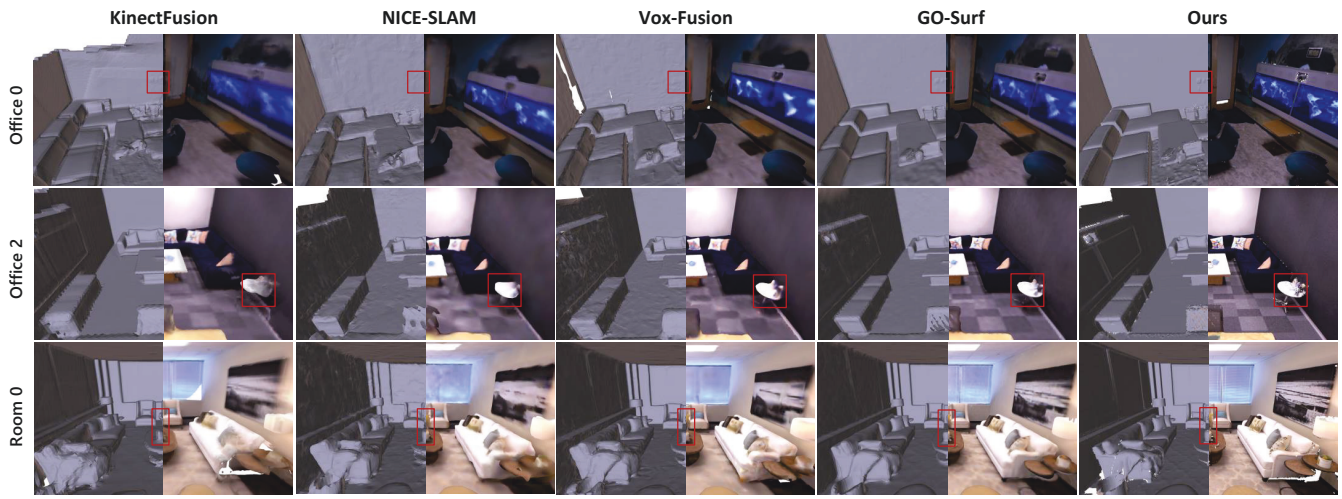


Figure 5: Reconstruction Result for Replica Dataset. The right half shows the result after vertex coloring. Our method recovers more accurate and smoother geometric shapes than implicit methods, particularly in flat and detailed regions.

tackle this issue, the geometric regularization [Luan *et al.*, 2021] is typically employed.

Let V denote an $n \times 3$ matrix with vertex positions as rows, and $L \in \mathbb{R}^{n \times n}$ is the Laplacian matrix. Thus, the Laplacian regularization term is defined as follows

$$\mathcal{L}_{lap} = \lambda_{lap} \|LV\|^2 \tag{6}$$

The normal consistency term enforces the surface smoothness by calculating the cosine similarity between the neighboring face normals as follows

$$\mathcal{L}_{normal} = \lambda_{normal} \Sigma_{i,j} [1 - (\mathbf{n}_i \cdot \mathbf{n}_j)]^2 \tag{7}$$

where the pair(i, j) represents the i-th and the j-th triangles sharing a common edge.

The edge length loss penalizes the long edge in the mesh as follows

$$\mathcal{L}_{edge} = \lambda_{edge} (\Sigma_i e_i^2)^{1/2} \tag{8}$$

where e_i denotes the length of the i-th face edge.

Finally, the overall objective function can be formulated as follows

$$\mathcal{L} = \mathcal{L}_{poins} + \mathcal{L}_{shading} + \mathcal{L}_{lap} + \mathcal{L}_{normal} + \mathcal{L}_{edge} \tag{9}$$

4 Experiments

In this section, we conduct a series of experiments to evaluate the reconstruction and rendering quality of our proposed approach. Please refer to our supplementary materials for more ablation experiments and visualization results.

4.1 Experimental Setup

Datasets and Baselines

We conduct experiments on three different kinds of datasets, including Replica [Straub *et al.*, 2019], ScanNet++ [Yeshwanth *et al.*, 2023], and Newer College Dataset [Ramezani *et al.*, 2020]. They represent, an indoor synthetic dataset, a real indoor scene dataset, and an outdoor 3D LiDAR dataset, respectively.

Methods	Metrics	R0	R1	R2	O0	O1	O2	O3	O4	Avg.
KinectFusion	Acc [cm]↓	12.85	22.44	21.46	11.67	10.44	14.77	13.87	15.23	15.34
	NC↑	0.83	0.76	0.78	0.77	0.77	0.83	0.82	0.78	0.79
	F-score↑	0.28	0.25	0.20	0.37	0.42	0.37	0.33	0.40	0.33
NICE-SLAM	Acc [cm]↓	2.73	2.58	2.65	2.26	2.50	3.82	3.50	2.77	2.85
	NC↑	0.92	0.79	0.91	0.90	0.88	0.88	0.86	0.91	0.88
	F-score↑	0.90	0.52	0.90	0.88	0.84	0.83	0.69	0.90	0.81
Vox-Fusion	Acc [cm]↓	2.41	1.62	3.11	1.74	1.69	2.23	2.84	3.31	2.37
	NC↑	0.94	0.91	0.91	0.91	0.89	0.92	0.91	0.90	0.91
	F-score↑	0.93	0.81	0.83	0.88	0.87	0.88	0.87	0.91	0.87
GO-Surf	Acc [cm]↓	5.16	2.49	7.13	4.40	3.60	3.48	5.32	5.52	4.64
	NC↑	0.93	0.91	0.93	0.94	0.93	0.93	0.89	0.92	0.92
	F-score↑	0.89	0.81	0.85	0.85	0.84	0.86	0.85	0.87	0.85
Ours	Acc [cm]↓	**0.57**	**0.56**	**0.57**	**0.53**	**0.55**	**0.58**	**0.59**	**0.57**	**0.56**
	NC↑	**0.97**	**0.96**	**0.97**	**0.94**	**0.94**	**0.95**	**0.94**	**0.93**	**0.94**
	F-score↑	**0.93**	**0.94**	**0.93**	**0.89**	**0.88**	**0.91**	**0.93**	**0.93**	**0.92**

Table 1: Quantitative evaluation of the reconstruction quality on the Replica dataset of 8 synthetic scenes.

To demonstrate the efficacy of our proposed approach, the following methods are employed as baseline: 1) KinectFusion [Izadi *et al.*, 2011] incrementally fuses the consecutive frames of depth data into a 3D volumetric representation of an implicit surface; 2) NICE-SLAM [Zhu *et al.*, 2022] incorporates multi-level local information via a hierarchical scene representation; 3) Vox-Fusion [Yang *et al.*, 2022] combines both neural and volumetric methods; 4) SplaTAM [Keetha *et al.*, 2023] represents a scene by 3D Gaussian Splatting using a single unposed monocular RGB-D camera; 5) GO-Surf [Wang *et al.*, 2022] combines a multi-resolution feature grid with a hybrid volume rendering scheme; 6) Puma [Vizzo *et al.*, 2021] employs the Poisson surface reconstruction to obtain triangular mesh from the LiDAR scan. To facilitate fair evaluation on surface reconstruction, we directly make use of ground truth poses for all baselines in our experiments.

Implementation Details

Our presented hybrid voxel-octree method is implemented by C++. For indoor scenes, the edge length of the voxel-octree leaf nodes is set to $0.05\ m$. The threshold $\mathcal{T}_{min}$ is $0.02\ m$,

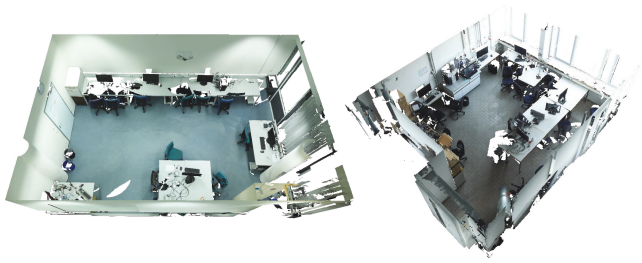


Figure 6: Other qualitative textured results on ScanNet++ reconstructed online with our method.

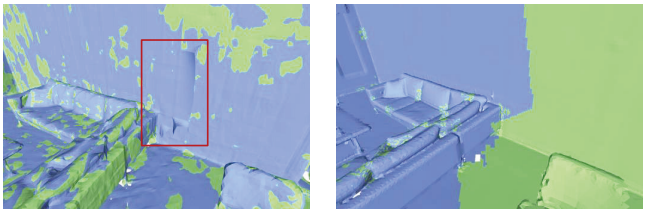


Figure 7: Visualization results of mesh edges during incremental reconstruction. The blue and green regions represent two distinct partial meshes, and the red box indicates the artifacts produced by Vox-Fusion during mesh fusion. Our meshes have minimal overlapping triangles and accurate edge alignment.

and the threshold $\mathcal{T}_{cur}$ is set to 0.01. The refinement process is implemented by PyTorch with the ADAM optimizer. The learning rates for vertex positions, vertex colors, and spherical harmonics are set to 0.0001, 0.01, and 0.001, respectively. The weights for the loss terms are set to 50 for λ_{lap}, 1 for λ_{normal} and 1 for λ_{edge}. We run 300 iterations to optimize each partial mesh. The entire pipeline ran in an end-to-end fashion without requiring any network or pre-training. All experiments are conducted on a desktop PC with an NVIDIA RTX3090 GPU.

4.2 Evaluation on Surface Reconstruction

We investigate the reconstruction results on the Replica dataset. We measure accuracy, normal consistency and F-score to evaluate the reconstruction quality. The metrics are computed between point clouds sampled at a density of 1 point per cm^2. F-score is computed using a threshold of 5 cm. Fig. 5 shows that our method recovers more detailed geometry, especially retains the smoothness of the wall and floor surfaces. This is because we directly optimize the triangular face coordinates to enable a more faithful representation of the geometric intricacies in the scene. Our shading model also enhances the accuracy and richness of the surface texture and color. Tab.1 shows that our presented method surpasses all the other baselines in terms of accuracy, normal consistency, and F-score metrics.

We further compare our incremental reconstruction results with Vox-Fusion. We export and fuse partial meshes every 200 frames on the Replica dataset to simulate the runtime mesh output. As shown in Fig.7, our meshes have minimal overlapping triangles and accurate edge alignment. In contrast, Vox-Fusion suffers from the overlapped triangles and surface intersections.

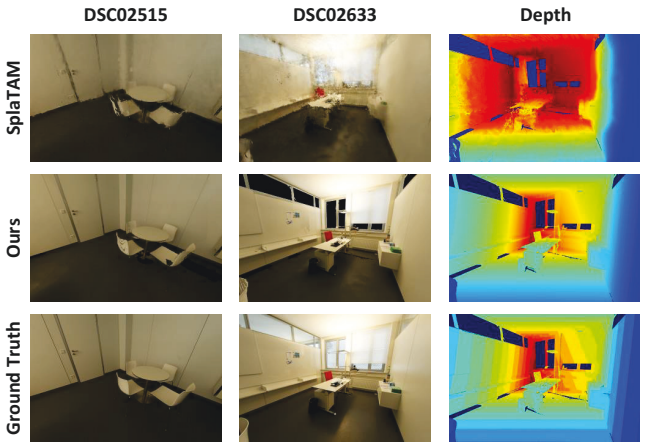


Figure 8: Evaluation of Rendering Quality on the ScanNet++ Dataset. Qualitative results demonstrate that our method produces higher-quality rendering results compared to the baselines.

Methods	Metric	Novel View			Training View		
		S1	S2	Avg.	S1	S2	Avg.
SplaTAM	PSNR↑	23.99	24.48	24.41	27.82	**28.14**	27.98
	SSIM↑	0.88	0.87	0.88	**0.94**	**0.94**	**0.94**
	LPIPS↓	0.21	0.26	0.24	**0.12**	0.13	**0.12**
	Depth L1 [cm]↓	1.91	2.23	2.07	**0.93**	1.64	**1.28**
Ours	PSNR↑	**26.64**	**25.43**	**26.04**	**28.06**	28.05	**28.05**
	SSIM↑	**0.89**	**0.87**	**0.88**	0.93	0.93	0.93
	LPIPS↓	**0.18**	**0.24**	**0.21**	0.13	**0.13**	0.13
	Depth L1 [cm]↓	**1.84**	**1.99**	**1.92**	1.53	**1.48**	1.51

Table 2: Novel & Train View Rendering Performance on ScanNet++. Our method is on par in terms of performance with SplaTAM but exhibits better results in novel view rendering.

4.3 Evaluation on Rendering Quality

We evaluate the rendering quality on the input views from the ScanNet++ dataset in Tab.2. We employ thee same evaluation metrics as in [Keetha *et al.*, 2023], including PSNR, SSIM and LPIPS. We evaluate the presented method on two scenes (8b5caf3398 (S1) and b20a261fdf (S2)). It can be seen that our approach performs comparably against the state-of-the-art method SplaTAM while achieving better results in rendering novel views. As shown in Fig.8, our rendered images of novel views have fewer artifacts and occlusions. This is because we directly shade the triangular faces in the mesh without optimizing the blank spaces. Moreover, we evaluate the geometric reconstruction of the scene by comparing the rendered depth against the ground-truth. Our method achieves better accuracy in both novel view and training view synthesis. Fig.6 also shows the visual results of our reconstruction on ScanNet++ dataset.

4.4 Discussion

Mesh Reconstruction from LiDAR Scans

In addition to depth map, our presented method can directly deal with LiDAR point clouds. We conduct experiments on the Newer College Dataset, where the point clouds

$a = 0.1m$ w/o $a = 0.05m$ w/o $a = 0.05m$

Figure 9: Reconstruction results with different voxel sizes α. w/o means the results without the refinement process.

Methods	Params↓	Iteration FPS↑	Runtime↓
NICE-SLAM	58.95 K	0.57	61.5 min
Vox-Fusion	54.28 K	0.79	42.5 min
GO-Surf	3.11 K	11.11	7.0 min
Ours	**0**	**16.67**	**5.0 min**

Table 3: Performance comparison on Replica/Office 0.

and their corresponding images are treated as input. As shown in Fig.10, we compared the reconstruction results with Puma. Our method achieved similar reconstruction accuracy to Puma but with more complete results. The promising results indicate that our presented method is able to handle the LiDAR scan as well. Note that our result may contain invalid triangles in blank regions due to the noisy observations.

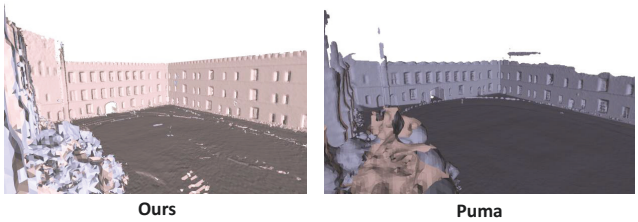
Efficiency

Differently from implicit surface reconstruction, our approach does not require time-consuming sampling, which leads to faster runtime. Tab.3 compares the model parameters, iteration FPS, and runtime. The runtime refers to the total time of running the model, including data loading and processing, while the iteration FPS refers to the processing time of a single frame on the GPU. Our method do not require any model parameters and outperforms the baseline methods on all metrics. NICE-SLAM consists of three hierarchical feature grids, which require costly query operations for reconstruction. Vox-Fusion adopts a standard octree structure, which results in longer query time compared to our hybrid voxel-octree. Although GO-Surf has a fast iteration speed, it requires the traditional Marching Cubes algorithm to extract the mesh. This requires more computational power. Our method only adjusts the surface triangles, avoiding expensive and time-consuming sampling. Thus, it is quite efficient in practice.

Ablation Study

We conduct ablation studies to demonstrate the impact of different branches and parameters of the hybrid voxel-octree.

We compare the impact of different voxel sizes and levels on the hybrid voxel-octree. Tab.4 shows the experimental results on Replica dataset. It can be clearly seen that the reconstruction accuracy increases as the voxel size decreases. However, smaller voxels lead to an exponential growth in the number of leaf nodes in the octree, which significantly increases the processing time and memory usage. Moreover, the hierarchical voxels have similar reconstruction accuracy as the traditional octree with deeper levels, which have shorter



Ours Puma

Figure 10: Reconstruction results on the Newer College Dataset.

Voxel (size [m] / level)	Acc [cm]	Insertion time	Memory usage
0.1 / 3	0.92	0.18 s	13.35 MiB
0.05 / 3	0.59	0.32 s	21.40 MiB
0.02 / 3	0.44	1.03 s	59.13 MiB
0.02 / 1	0.58	1.03 s	19.07 MiB

Table 4: Performance analysis on different voxel sizes and levels. Insertion time means the time required to insert a single poind cloud (819,200 points) into the leaf nodes. Memory usage means the amount of CPU memory utilized to store a frame of point cloud.

Point-based Refinement	Shading-based Refinement	Acc [cm]↓	NC↑	F-score↑
×	×	0.96	0.93	0.87
✓	×	0.61	0.95	0.92
×	✓	0.95	0.93	0.88
✓	✓	**0.59**	**0.96**	**0.93**

Table 5: Refinement branch Ablation on Replica/Office 3.

insertion time. This indicates that our underlying structure can better trade-off between efficiency and accuracy.

Finally, we examine the effectiveness of different refinement branches. As shown in Tab.5, the point-based refinement plays a crucial role in significantly improving the accuracy. This is because we directly use the target point cloud extracted from the octree to supervise the position of triangle vertices. The shading-based refinement is supervised by images, which may not provide accurate geometric information. The shading-based refinement primarily focuses on learning the shading model and can be jointly optimized with the point refine branch. Fig.9 shows the reconstruction results with various settings.

5 Conclusion

We introduce HVOFusion, incremental mesh reconstruction using hybrid voxel-octree. The hybrid voxel-octree effectively fuse octree with voxel structures so that we can take advantage of both implicit surface and explicit triangular mesh representation. This structure allows the partial reconstruction of the scene geometry during the octree construction. To enhance the reconstruction accuracy, we designed point-based and shading-based refinement branch to recover surface geometry and vertex colors. Experiments show that our method can quickly and accurately reconstruct a scene with realistic colors. One limitation of our method is that it is sensitive to the quality of the input point cloud and has difficulty reconstructing unobserved regions. We intend to enhance the completeness and robustness of the model in future work.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grants (62376244). It is also supported by Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

[Breitenmoser and Siegwart, 2012] Andreas Breitenmoser and Roland Siegwart. Surface reconstruction and path planning for industrial inspection with a climbing robot. In *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, pages 22–27. IEEE, 2012.

[Curless and Levoy, 1996] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[Dai *et al.*, 2017] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1, 2017.

[Engel *et al.*, 2013] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013.

[Gao *et al.*, 2020] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. *Advances In Neural Information Processing Systems*, 33:9936–9947, 2020.

[Guédon and Lepetit, 2023] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023.

[Häne *et al.*, 2017] Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Paul Furgale, Torsten Sattler, and Marc Pollefeys. 3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing*, 68:14–27, 2017.

[Hanocka *et al.*, 2020] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *ACM Trans. Graph.*, 39(4), 2020.

[Izadi *et al.*, 2011] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[Johari *et al.*, 2023] 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*, pages 17408–17419, 2023.

[Keetha *et al.*, 2023] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. *arXiv preprint arXiv:2312.02126*, 2023.

[Kerbl *et al.*, 2023] 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.

[Kim *et al.*, 2010] Hansung Kim, Muhammad Sarim, Takeshi Takai, Jean-Yves Guillemaut, and Adrian Hilton. Dynamic 3d scene reconstruction in outdoor environments. In *In Proc. IEEE Symp. on 3D Data Processing and Visualization*, 2010.

[Liu *et al.*, 2019] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2019.

[Lorensen and Cline, 1987] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery.

[Luan *et al.*, 2021] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified shape and svbrdf recovery using differentiable monte carlo rendering. In *Computer Graphics Forum*, volume 40, pages 101–113. Wiley Online Library, 2021.

[Mildenhall *et al.*, 2021] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[Ming *et al.*, 2022] Yuhang Ming, Weicai Ye, and Andrew Calway. idf-slam: End-to-end rgb-d slam with neural implicit mapping and deep feature tracking. *arXiv preprint arXiv:2209.07919*, 2022.

[Munkberg *et al.*, 2022] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8280–8290, 2022.

[Nießner *et al.*, 2013] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013.

[Ortiz *et al.*, 2022] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer,

and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. In *Robotics: Science and Systems*, 2022.

[Prisacariu *et al.*, 2017] V A Prisacariu, O Kähler, S Golodetz, M Sapienza, T Cavallari, P H S Torr, and D W Murray. InfiniTAM v3: A Framework for Large-Scale 3D Reconstruction with Loop Closure. *arXiv pre-print arXiv:1708.00783v1*, 2017.

[Ramezani *et al.*, 2020] Milad Ramezani, Yiduo Wang, Marco Camurri, David Wisth, Matias Mattamala, and Maurice Fallon. The newer college dataset: Handheld lidar, inertial and vision with ground truth. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4353–4360, 2020.

[Reijgwart *et al.*, 2019] Victor Reijgwart, Alexander Millane, Helen Oleynikova, Roland Siegwart, Cesar Cadena, and Juan Nieto. Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps. *IEEE Robotics and Automation Letters*, 5(1):227–234, 2019.

[Schonberger and Frahm, 2016] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[Schöps *et al.*, 2019] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. Surfelmeshing: Online surfel-based mesh reconstruction. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2494–2507, 2019.

[Sharf *et al.*, 2006] Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. Competing fronts for coarse–to–fine surface reconstruction. In *Computer Graphics Forum*, volume 25, pages 389–398. Wiley Online Library, 2006.

[Shen *et al.*, 2021] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.

[Straub *et al.*, 2019] 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.

[Sucar *et al.*, 2021] 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.

[Thies *et al.*, 2019] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *Acm Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[Vizzo *et al.*, 2021] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss. Poisson Surface Reconstruction for LiDAR Odometry and Mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[Walker *et al.*, 2023] Thomas Walker, Octave Mariotti, Amir Vaxman, and Hakan Bilen. Explicit neural surfaces: Learning continuous geometry with deformation fields. *NeurIPS*, 2023.

[Wang *et al.*, 2018] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–67, 2018.

[Wang *et al.*, 2022] 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)*. IEEE, 2022.

[Whelan *et al.*, 2012] Thomas Whelan, John B McDonald, Michael Kaess, Maurice Fallon, Hordur Johannsson, and John J Leonard. Kintinuous: Spatially extended kinectfusion. In *RSS '12 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, pages 1, 6, 2012.

[Whelan *et al.*, 2015] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015.

[Worchel *et al.*, 2022] Markus Worchel, Rodrigo Diaz, Weiwen Hu, Oliver Schreer, Ingo Feldmann, and Peter Eisert. Multi-view mesh reconstruction with neural deferred shading. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6187–6197, 2022.

[Wu *et al.*, 2011] Chenglei Wu, Bennett Wilburn, Yasuyuki Matsushita, and Christian Theobalt. High-quality shape from multi-view stereo and shading under general illumination. In *CVPR 2011*, pages 969–976. IEEE, 2011.

[Yang *et al.*, 2022] 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.

[Yeshwanth *et al.*, 2023] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision*, 2023.

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