Comput. Aided Geom. Des. 111 (2024) 102322

Contents lists available at ScienceDirect

Computer Aided Geometric Design

journal homepage: www.elsevier.com/locate/cagd

Real-time volume rendering with octree-based implicit surface representation

Jiaze Li [a,1], Luo Zhang [a,1], Jiangbei Hu [a,b], Zhebin Zhang [c], Hongyu Sun [c], Gaochao Song [d], Ying He [a,*]

[a] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[b] International School of Information Science & Engineering, Dalian University of Technology, Dalian, Liaoning, China
[c] InnoPeak Technology, Inc., Bellevue, WA, United States
[d] College of Intelligence and Computing, Tianjin University, Tianjin, China

ARTICLE INFO

Dataset link: https://github.com/LaoChui999/Octree-VolSDF

Keywords:
Neural rendering
Implicit surfaces
Octrees
Real-time volume rendering

ABSTRACT

Recent breakthroughs in neural radiance fields have significantly advanced the field of novel view synthesis and 3D reconstruction from multi-view images. However, the prevalent neural volume rendering techniques often suffer from long rendering time and require extensive network training. To address these limitations, recent initiatives have explored explicit voxel representations of scenes to expedite training. Yet, they often fall short in delivering accurate geometric reconstructions due to a lack of effective 3D representation. In this paper, we propose an octree-based approach for the reconstruction of implicit surfaces from multi-view images. Leveraging an explicit, network-free data structure, our method substantially increases rendering speed, achieving real-time performance. Moreover, our reconstruction technique yields surfaces with quality comparable to state-of-the-art network-based learning methods. The source code and data can be downloaded from https://github.com/LaoChui999/Octree-VolSDF.

1. Introduction

Reconstructing 3D representations from 2D images that involve geometric shapes and textures is one of the essential topics in graphics and 3D vision (Newcombe et al., 2011; Schönberger et al., 2016; Whelan et al., 2016), with the aim of achieving high-fidelity models that are indistinguishable from objects and scenes in the real world. Recently, methods based on neural radiance fields (NeRFs) (Mildenhall et al., 2021; Barron et al., 2021; Wang et al., 2021b; Verbin et al., 2022; Tancik et al., 2022; Niemeyer and Geiger, 2021; Weng et al., 2022) have shown remarkable success in the synthesizing of photorealistic images. These neural rendering frameworks optimize a neural network to map 3D spatial coordinates to different signal fields, such as radiance, color, occupancy (Reiser et al., 2021; Oechsle et al., 2021; Fridovich-Keil et al., 2023), signed distance values (Wang et al., 2021a; Yang et al., 2022; Fu et al., 2022; Johari et al., 2022), and others (Chen et al., 2022; Gao et al., 2023), to reconstruct the shape and appearance of scenes. However, the training process of NeRF is computationally expensive (Yariv et al., 2023; Zhang et al., 2020) and still needs to be more efficient for real-time rendering, which limits its application in interactive uses.

* Corresponding author.
E-mail addresses: liji0077@e.ntu.edu.sg (J. Li), LUO002@e.ntu.edu.sg (L. Zhang), hjb941220@dlut.edu.cn (J. Hu), zhebin.zhang@oppo.com (Z. Zhang), Dr.sunhongyu@gmail.com (H. Sun), gaochaosong_21@tju.edu.cn (G. Song), yhe@ntu.edu.sg (Y. He).
[1] The authors contribute equally.

https://doi.org/10.1016/j.cagd.2024.102322

Available online 8 May 2024

In order to enhance training speed while reducing rendering time, many methods have been proposed that use explicit voxel representation (Yu et al., 2021; Chen et al., 2022; Liu et al., 2020). Plenoxels (Fridovich-Keil et al., 2022) stores density in sparse grid nodes and characterize color dependent on the view without a neural network using spherical harmonic coefficients. By integrating explicit and implicit methods, DVGO (Sun et al., 2022a) and DVGOv2 (Sun et al., 2022b) incorporate a shallow multilayer perception on the voxel grid that represents features, achieving a better balance between the quality and efficiency of the novel view synthesis. Although these methods produce comparable rendering results faster than implicit methods, they often fail to reconstruct geometry accurately. This is due to the inherent limitations of density-based volume rendering. It is difficult to reconstruct accurate geometry through density, and surfaces are often discontinuous with noise and holes. To address this issue, Voxurf (Wu et al., 2023b) based on DVGO (Sun et al., 2022a) and NeuS (Wang et al., 2021a), representing the geometry through the SDF field, which allows a better reconstruction of geometry. Nevertheless, Voxurf still relies on the network to store and optimize the parameters, and there is no other means for accelerating rendering, so it cannot render in real-time. There are several existing works that focus on real-time rendering, such as BakedSDF (Yariv et al., 2023), BakingNeRF (Hedman et al., 2021). Although BakedSDF achieves a rendering speed of nearly 72 FPS on MacBook Pro, their method requires long training and sacrifices the view synthesis quality in real-time rendering. The recently popularized Gaussian Splatting (Kerbl et al., 2023) technique, employing Gaussian points combined with differentiable rasterization, enables the synthesis of photorealistic images at accelerated rendering speeds. 3DGS has greater advantages than neural rendering method such as NeRF (Mildenhall et al., 2021) in both reconstruction speed and rendering speed. For geometry reconstruction, although the positions of the optimized Gaussians roughly surround the object, it is still difficult to reconstruct high-quality geometry through state-of-the-art point based reconstruction methods such as iPSR (Hou et al., 2022). SuGaR (Guédon and Lepetit, 2023) promotes the alignment of 3D Gaussians with the potential surface, which can extract a mesh from the output of 3DGS.

In this work, we propose a novel octree-based approach to reconstruct implicit surfaces from multi-view images. We formulate the reconstruction as a two-stage optimization problem, which can be solved by gradients computed over compact octree grid points without any neural network. Specifically, we obtain a coarse geometric shape followed by a deeper subdivision based on an octree structure. We store property values that include the signed distance values and the spherical harmonic coefficients at the grid points of the octree structure to represent the geometry and the radiance field, respectively. For query points within the octree's leaf nodes, we determine their property values by utilizing trilinear interpolation, subsequently employing volume rendering techniques to synthesize 2D images. Our approach achieves competitive rendering performance in less time and allows us to reconstruct intricate geometric structures using higher-resolution grids. Moreover, our framework based on octree structures can be extended to enhance the rendering efficiency of methods based on neural networks such as VolSDF (Yariv et al., 2021). We validate the effectiveness and performance of our approach in various datasets such as the NeRF-Synthetic, Omni3dObjects, and DTU datasets.

We summarize our contributions as follows: We propose a novel octree-based method to reconstruct implicit surfaces from multi-view images. By eliminating the neural network, our method enables real-time rendering of novel views while preserving a rendering and geometric reconstruction accuracy comparable to existing techniques. Our algorithm is efficient and our training time is less than an hour. In addition, our approach can convert slow-rendering models (such as VolSDF) into our octree model to achieve real-time rendering.

2. Related work

2.1. Neural implicit surfaces

NeRF uses neural volume rendering to synthesize multiple points sampled on each camera ray into pixel colors. However, NeRF aims for multi-view reconstruction and learns the volume density field. Although geometry can be extracted from density, it is difficult to select an appropriate density threshold to extract high-quality surfaces. This can introduce unnecessary noise and potential issues with the reconstructed geometry. In contrast, methods grounded in neural implicit surface functions, such as occupancy fields and SDF (Oechsle et al., 2021; Wang et al., 2021a; Yariv et al., 2021), achieve superior surface modeling and geometric reconstruction without compromising the quality of synthetic views. IDR (Yariv et al., 2020) and DVR (Niemeyer et al., 2020) represent the scene as an occupancy field and employ differentiable rendering to recover high-frequency geometry and color. Nevertheless, it necessitates 2D mask supervision for training, which is often challenging. NeuS (Wang et al., 2021a) and VolSDF (Yariv et al., 2021) parameterize scenes as implicit SDF fields via MLPs, Neural Warp (Darmon et al., 2022) borrows from traditional MVS methods to obtain color values from different perspectives. HF-NeuS (Wang et al., 2022b) introduces a novel volume rendering method based on SDF via theoretical derivation. MonoSDF (Yu et al., 2022b) enhances reconstruction quality by incorporating geometric constraints but relies heavily on the quality of depth maps and normal maps obtained through pre-training. Regrettably, these methodologies frequently demand extended training periods and exhibit a slower rendering pace despite enhancing surface reconstruction quality.

2.2. Accelerating on NeRF

To address the issues of slow training and rendering speeds in previous work, subsequent research has prioritized acceleration techniques. Notably, INGP (Müller et al., 2022) significantly reduces the number of training parameters through multi-resolution hash coding coupled with highly optimized CUDA kernels. NeuS2 (Wang et al., 2023a), PermutoSDF (Rosu and Behnke, 2023) Neuralangelo (Li et al., 2023) employ the multi-resolution hash encodings and Shallow MLP proposed in INGP to replace huge neural network for accelerated NeuS training. Plenoxels (Fridovich-Keil et al., 2022) utilizes sparse voxels and spherical harmonic

coefficients to represent scenes, substantially reducing training time without resorting to neural networks. DVGO (Sun et al., 2022a) and its subsequent work, Voxurf, achieve training acceleration by incorporating both voxel and smaller MLPs. PermutoSDF employed a novel triangular pyramid grid for faster training and reduced calculation overhead during interpolation. TensorRF (Chen et al., 2022), followed by Strivec (Gao et al., 2023), leverage tensor decomposition for tensor grid modeling. However, limited by the network structure, most methods cannot achieve high-speed rendering.

2.3. Real-time rendering

To eliminate the long waiting time and achieve three-dimensional real-time rendering, some work (Garbin et al., 2021; Wang et al., 2022a; Chen et al., 2023) has been explored on achieving real-time rendering. Currently, there are several main ideas for achieving high-speed rendering. For example, Plenoctree (Yu et al., 2021) and RT-Octree (Shu et al., 2023) accelerate by converting the NeRF model into an explicit octree for offline rendering, and Plenoxels optimizes the model in voxels for fast rendering. Additionally, BakedSDF precalculates and saves parameters through the baking method. During rendering, a smaller MLP reads the parameters for quick rendering. Another recent work (Wang et al., 2023b) also targets real-time rendering but cannot guarantee good geometric quality while accelerating rendering. Moreover, 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023), rooted in sparse point cloud reconstruction. The Gaussian points are sorted along depth and splatted onto 2D screen space. Then by using α-blending, these points are rasterized into pixel colors. 3DGS can render high quality images in real-time.

3. Preliminaries

Volume rendering Neural Radiance Fields represents 3D scenes by utilizing a trained network to map spatial coordinates $\mathbf{x}$ and viewing directions $\mathbf{d}$ to radiance $\mathbf{c}$ and density values σ, enabling photorealistic image synthesis from novel viewpoints. Following most volume rendering work (Yariv et al., 2021), we sample points $\{\mathbf{o}+t_i\mathbf{d}\}_{i=1}^{N}$ along the ray $\mathbf{r}$ and calculate the color as the following discrete integral form,

$$\mathbf{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - \exp(-\sigma_i \delta_i)\right) \mathbf{c}_i, \tag{1}$$

where $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_i \delta_i\right)$, δ_i is the distance to the next sample point. We calculate the volume density according to the signed distance field $S(\mathbf{x})$ as:

$$\sigma(\mathbf{x}) = \begin{cases} \frac{1}{2\beta} \exp\left(\frac{S(\mathbf{x})}{\beta}\right) & \text{if } S(\mathbf{x}) \le 0, \\ \frac{1}{\beta} - \frac{1}{2\beta} \exp\left(-\frac{S(\mathbf{x})}{\beta}\right) & \text{if } S(\mathbf{x}) > 0, \end{cases} \tag{2}$$

where β is a parameter that enables the adjustment of the smoothness of the density function σ near the object boundary by varying its value. Refer to Yariv et al. (2021) for more technical details about volume rendering for neural implicit surfaces.

Spherical harmonics Spherical Harmonics form a complete and orthogonal basis on the surface of a sphere (Atkinson and Han, 2012), and have been used to model Lambertian surfaces (Fridovich-Keil et al., 2022). PlenOctrees (Wang et al., 2022a) represents the RGB values at any given viewing direction $\mathbf{d} = (\theta, \phi)$ as a weighted sum of spherical harmonics,

$$\mathbf{C} = \sum_{l=0}^{L} \sum_{m=-l}^{l} a_l^m Y_l^m(\theta, \phi), \tag{3}$$

where $Y_l^m(\theta, \phi) : \mathbb{S}^2 \to \mathbb{R}$ is the spherical harmonic basis function, $\mathbf{a} = \{a_l^m\}$ are the spherical harmonics coefficients that are predicted by the network. The orthogonality of spherical harmonics on the sphere enables them to effectively capture lighting distributions, making them independent of the viewing direction.

4. Method

The core of our proposed method is that we reconstruct the 3D representations of one object from multi-view images based on octree structures without any neural networks. We introduce an octree-based representation (Sec. 4.1) using SDF values and SH coefficients to represent 3D shapes and radiance fields, respectively. Under the supervision of the Ground Truth (GT) images, we formulate the reconstruction task as an optimization problem (Sec. 4.4). This can be solved using a coarse-to-fine strategy (Sec. 4.2), driven by the gradient information defined at the points of the octree grid (Sec. 4.3). Fig. 1 illustrates the algorithmic pipeline.

4.1. Octree-based representation

An octree is a hierarchical data structure that is particularly well suited for managing spatial information in 3D representations (Wilhelms and Van Gelder, 1992). The process begins with a single root node that encompasses the entire space. This space is

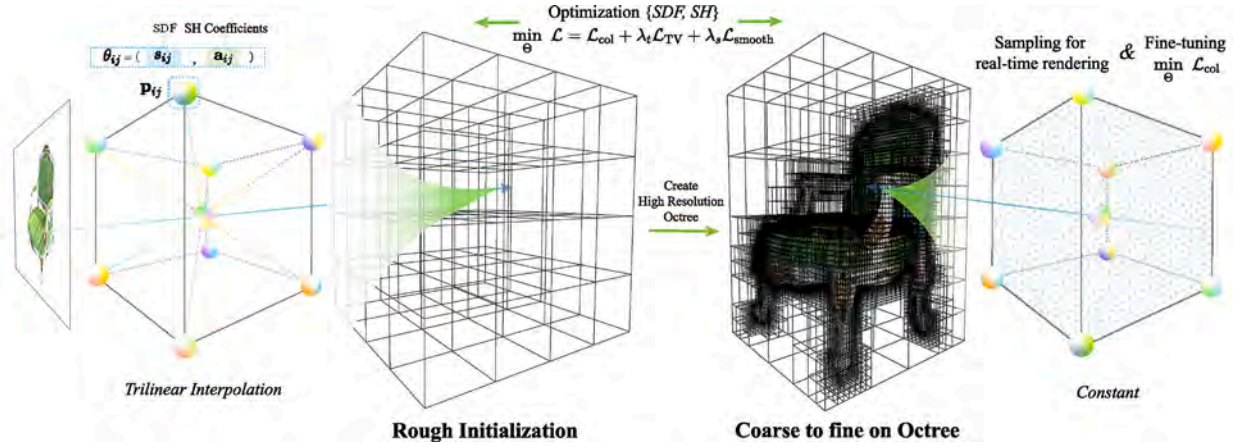

Fig. 1. Pipeline. We use an octree grid to reconstruct a 3D representation of an object from a set of images without the need for neural networks, allowing for real-time rendering. The grid points $\{\mathbf{p}_{ij}\}_{j=1}^{8}$ of each leaf node ℓ_i of the octree structure store property values θ_{ij} that include the SDF value s_{ij} and Spherical Harmonics coefficients $\mathbf{a}_{ij}$ for the geometry and radiance representation. For one rendering ray, we sample points inside ℓ_i and use trilinear interpolation of $\{\theta_{ij}\}_{j=1}^{8}$ to calculate the property values of these query points based on their distance. After generating the image with volume rendering techniques, we optimize the property values under the supervision of real images. We adopt a coarse-to-fine strategy to enhance stability and accuracy; i.e., we first reconstruct a rough shape from a low-resolution octree and then build a high-resolution octree to recover more details. Furthermore, as the resolution increases, we treat the property values within the leaf nodes as constants instead of using trilinear interpolation. This greatly reduces computational complexity, facilitating real-time rendering.

then divided into eight bi-sectional grids. The division continues recursively for each grid that contains parts of objects until a certain level of detail is achieved or the grid is empty (i.e., contains no objects). We refer to the last undivided grid as leaf nodes $\{\ell_i\}, i \in I^l$, where l is the subdivision level, and each leaf node has eight grid points $\{\mathbf{p}_{ij}\}_{j=1}^{8}$. In contrast to methods based on uniform grid (Liu et al., 2020) or sparse grid (Fridovich-Keil et al., 2022; Wu et al., 2023b), octree structures offer a more memory-efficient representation by adaptive partitioning the 3D space into finer sub-grids only where the complexity of the scene requires it, which can lead to improvements in computational speed.

We employ the signed distance field (SDF) to implicitly represent the geometric shape and use the spherical harmonic (SH) interpolation function as in Eq. (3) to represent the radiance field. Specifically, we store the SDF value $s_{ij} = S(\mathbf{p}_j^i)$ and SH coefficients $\mathbf{a}_{ij} = \{a_l^m\}_{ij}$ for each grid point $\mathbf{p}_j^i$, which are the parameters $\Theta = \{\theta_{ij} = (s_{ij}, \mathbf{a}_{ij})\}$ to optimize in our framework. We must also determine the property values of the octree leaf node ℓ_i. We randomly select 256 points within the node grid and use trilinear interpolation to calculate the property values of these points. Then we calculate the minimum of the SDF values s_i^*, the mean of the SDF values $\hat{s}_i$, and the mean of the SH coefficients $\hat{\mathbf{a}}_{ij}$, which serve as the property values of ℓ_i. The value s_i^* will subsequently guide further subdivision of the octree, while $\hat{s}_i$ will be utilized to accelerate the rendering process (Sec. 4.5). For geometry reconstruction, we directly extract the zero-level set of the SDF stored in the octree. For radiance field reconstruction, we convert the SDF values into density using the Eq. (2) and obtain color information for different viewpoints through SH interpolation as shown in Eq. (3). Then, we can synthesize images from novel viewpoints using volume rendering techniques.

Subdivision criteria Given an octree with a subdivision level of l, we proceed with the subdivision to a depth $l+1$ based on the SDF values corresponding to the current octree leaf nodes. If the SDF value s_i^* of a node ℓ_i is no greater than a given threshold ϵ,[2] we subdivide it ℓ_i. This progress is repeated until the subdivision level limit $\overline{L}$ is reached, at which point the octree updates are terminated. The resulting octree structure features a higher concentration of grid points near the object's surface, ensuring detailed reconstruction of geometry and color information.

4.2. Training from coarse-to-fine

To accurately reconstruct the geometry and radiance field, we optimize the property values Θ stored at the octree grid points, driven by the gradient of the loss function (Sec. 4.4) between the synthetic and GT images. Simultaneously, we update the octree in accordance with the current values of the SDF. Optimizing on discrete grids without neural networks presents a challenging task, facing the risks of unstable convergence and the propensity to fall into local optima. Employing a coarse-to-fine strategy is an effective way to address this issue, which has been applied in various 3D reconstruction works (Wu et al., 2023b; Li et al., 2023). Inspired by this, we employ a two-stage strategy for constructing octrees from coarse to fine.

Coarse stage To rapidly capture the object's rough geometry, we initialize the octree with a subdivision level $l = 6$, which results in a uniform grid with a resolution of 64^3. Given that our method operates on a discrete grid without any neural networks and starts

[2] The default value of ϵ is $= 0.1$.

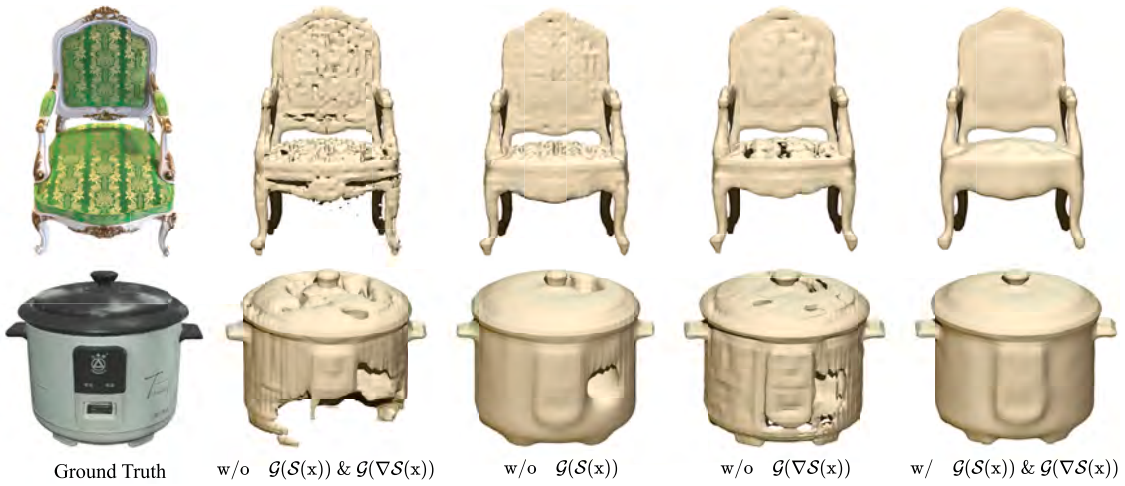

Fig. 2. Effects of the Gaussian kernels $\mathcal{G}$ on the SDF field $S(\mathbf{x})$ and its gradient field $\nabla S(\mathbf{x})$ in the initial stage. Both $\mathcal{G}(S(\mathbf{x}))$ and $\mathcal{G}(\nabla S(\mathbf{x}))$ enable the generation of a coarse shape roughly capturing the model's topology and overall geometry.

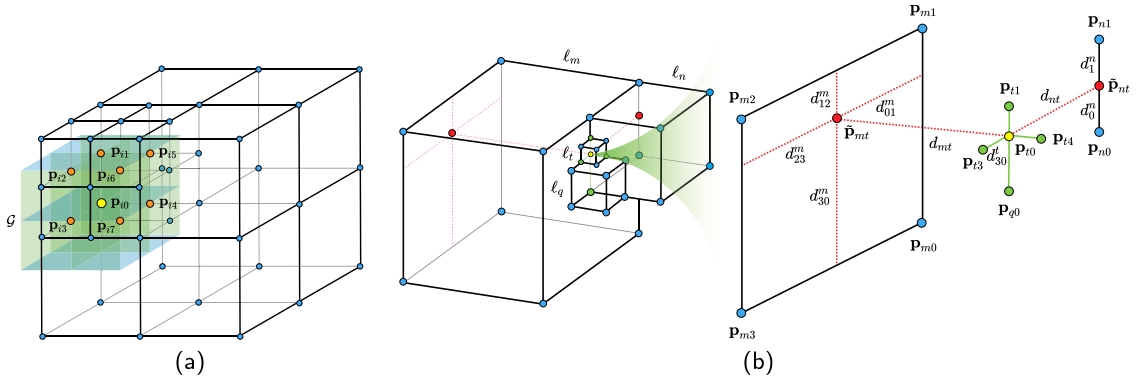

Fig. 3. Examples of discrete operators on octree. (a) An example of a Gaussian kernel on the corner $\mathbf{p}_{i0}$. We take $\mathbf{p}_{i0}$ as the center of the Gaussian kernel with $k_g = 3$. Since there are only eight grid points $(\mathbf{p}_{i0}, \mathbf{p}_{i1}, \cdots, \mathbf{p}_{i7})$ in the kernel, the corresponding coefficient of the kernel is normalized by these eight grid points. (b) The most general octree neighborhood configuration around a corner $\mathbf{p}_{t0}$. In this example, $\tilde{\mathbf{p}}_{nt}$ is a ghost node lying on an edge, while $\tilde{\mathbf{p}}_{mt}$ is a ghost node lying on a face, $\mathbf{p}_{t1}$, $\mathbf{p}_{t3}$, $\mathbf{p}_{t4}$, and $\mathbf{p}_{q0}$ are existing adjacent neighbors of $\mathbf{p}_{t0}$.

with a relatively low grid resolution, directly optimizing the parameters at the grid points often leads to holes and noise on the model's surface, as shown in the second column of Fig. 2. Our findings showed that the quality of geometric initialization does not improve with higher subdivision levels; thus, it is essential to have a smooth initial geometry in order to obtain an accurate final model. To address this, we apply the Gaussian kernels $\mathcal{G}(N, k_g, \sigma_g)$ to SDF $\mathcal{S}$ and its gradient $\nabla\mathcal{S}$ similar to Voxurf (Wu et al., 2023b). We compute the weight matrix as follows,

$$\omega_{i,j,k} = \frac{1}{N}\exp\left(-\frac{((i-\lfloor\frac{k_g}{2}\rfloor)^2 + (j-\lfloor\frac{k_g}{2}\rfloor)^2 + (k-\lfloor\frac{k_g}{2}\rfloor)^2)}{2\sigma_g^2}\right) \tag{4}$$

where $i, j, k \in 0, 1, \cdots, k_g - 1$, N represents the normalization term, k_g is the kernel size, and σ_g is the standard deviation to adjust the smoothness (default 0.8 in our experiments). This can enhance the continuity and smoothness of the reconstructed SDF, making the optimization process more stable. In our experiments, we set $k_g = 5$ and $k_g = 3$ for $\mathcal{S}$ and $\nabla\mathcal{S}$, respectively. We demonstrate the necessity of applying Gaussian kernels to both fields, as shown in Fig. 2. This technique assists in achieving a smooth initial structure, which is essential for facilitating detailed reconstruction in the subsequent fine stage. Due to the non-uniformity of the octree structure, not all positions corresponding to the Gaussian kernel will have associated grid points as the resolution increases. Consequently, our calculations are restricted to positions where grid points are present, as shown in Fig. 3(a).

Fine stage In the fine stage, we first reconstruct a new octree structure based on the rough shape obtained in the coarse stage, ensuring the compactness between the grid and the model to optimize memory storage, which further enables a higher resolution. Specifically, we construct a new octree with a subdivision level of $l = 7$ (yielding a uniform grid with a resolution of 128^3) and

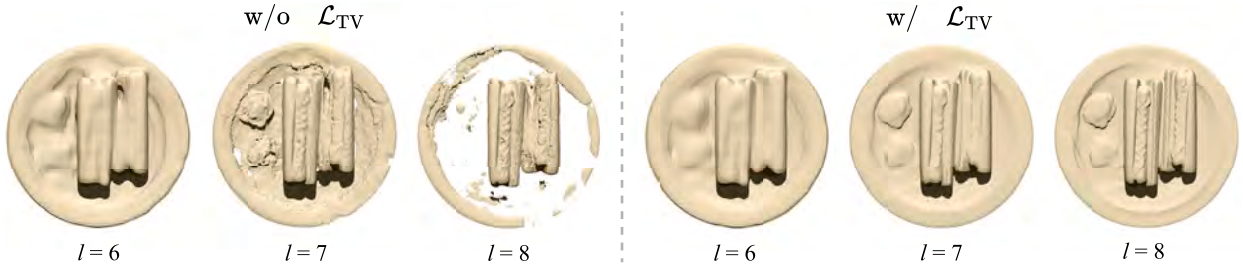

Fig. 4. The analysis demonstrates that incorporating TV regularization $\mathcal{L}_{\text{TV}}$ on the SH coefficients facilitates preserving a coherent shape across various subdivision level l.

calculate the property values of the new grid points by applying trilinear interpolation of data from the coarse octree. Subsequently, we perform deeper octree subdivisions on this more compact octree structure during optimization iterations, as described in Sec. 4.1.

4.3. Discretization of gradient operators

We utilize gradient-based methods to solve optimization problems. Unlike neural network-based approaches, we directly compute the gradient information of the parameters on a discretized octree grid. In our experiments, we apply the central difference operator to calculate the gradient information. Compared to uniform voxels, octree structures present a challenge in the discretization of differential operators due to the presence of T-junctions. In other words, there may be no adjacent neighbor grid points when using the central difference method to calculate the gradient for a given grid point. To address this issue (Min and Gibou, 2008), we project points of T-junctions onto the faces or edges on the side without neighbors to obtain *ghost points*. We then determine the property values of these ghost points by interpolation and calculate the gradient information. Note that ghost points are utilized only during computation and do not actually exist in the octree structure. We present an example in Fig. 3(b) to illustrate specifically, where $\mathbf{p}_{t0}$ has two sides without neighbors available for computing the central difference. We project $\mathbf{p}_{t0}$ onto one face of the leaf node ℓ_m and one edge of the leaf node ℓ_n, resulting in two ghost points $\tilde{\mathbf{p}}_{mt}$ and $\tilde{\mathbf{p}}_{nt}$. We interpolate the SDF values of these two ghost points as follows,

$$\mathcal{S}(\tilde{\mathbf{p}}_{nt}) = \frac{d_1^n s_{n0} + d_0^n s_{n1}}{d_1^n + d_0^n}, \quad \mathcal{S}(\tilde{\mathbf{p}}_{mt}) = \frac{d_{30}^m d_{23}^m s_{m1} + d_{30}^m d_{01}^m s_{m2} + d_{12}^m d_{23}^m s_{m0} + d_{12}^m d_{01}^m s_{m3}}{(d_{12}^m + d_{30}^m)(d_{01}^m + d_{23}^m)}, \tag{5}$$

where d indicates the distance. Subsequently, we calculate the gradient using the following formula, which takes the gradient D_0^x of v_0 on the x-axis as an example:

$$D_0^x(\mathcal{S}) = \frac{s_{t3} - s_{t0}}{d_{30}^t} \cdot \frac{d_{nt}}{d_{30}^t + d_{nt}} + \frac{s_{t0} - \mathcal{S}(\tilde{\mathbf{p}}_{nt})}{d_{nt}} \cdot \frac{d_{30}^t}{d_{30}^t + d_{nt}}. \tag{6}$$

We calculate the gradient of the SH coefficients in the same way as described above.

4.4. Optimization problem

With the target of reconstructing the geometry and radiance fields accurately, we directly optimize the property values $\Theta = \{\theta_{ij} = (s_{ij}, \mathbf{a}_{ij})\}$ stored at the octree's grid points, guided by the gradient of the following loss functions. To ensure the photorealism of the synthesized images, we calculate the mean squared error (MSE) of the pixel colors compared to the real images:

$$\mathcal{L}_{\text{col}} = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{C}_n - \hat{\mathbf{C}}_n \right\|_2^2, \tag{7}$$

where N is the total number of pixels. Moreover, as the optimization of color and geometry mutually enhances each other, we incorporate a total variation (TV) term to constrain the property values, thereby maintaining stability in the optimization process and ensuring the precision of the geometric reconstruction.

$$\mathcal{L}_{\text{TV}} = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{p} \in \mathcal{V}} \sqrt{(D_{\mathbf{p}}^x)^2 + (D_{\mathbf{p}}^y)^2 + (D_{\mathbf{p}}^z)^2}, \tag{8}$$

where $\mathcal{V}$ is the set of grid points. Observations from Fig. 4 indicate that the TV regularization of SH plays a crucial role in preserving geometric continuity in our method, particularly at low octree resolution. We also introduce the TV regularization for the SDF. In our experiments, we only randomly select 1% points from $\mathcal{V}$ to calculate $\mathcal{L}_{\text{TV}}$ to speed up the training process. As introduced in Sec. 4.2, we regularize the gradient of SDF $\nabla \mathcal{S}$ by $\mathcal{L}_{\text{smooth}}$ to encourage a smooth surface:

$$\mathcal{L}_{\text{smooth}} = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{p} \in \mathcal{V}} \| \mathcal{G}(\nabla \mathcal{S}(\mathbf{p})) - \nabla \mathcal{S}(\mathbf{p}) \|_2^2. \tag{9}$$

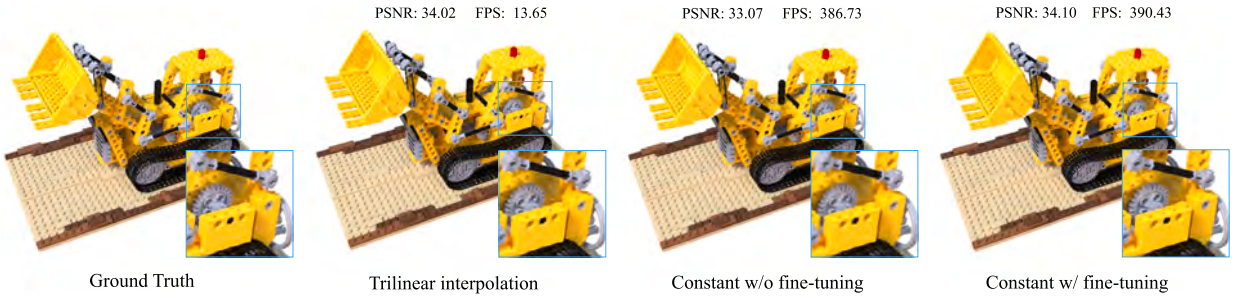

Fig. 5. A comparison of the visualization results rendered by property values θ_{ij} and $\hat{\theta}_i$ which are stored at the grid points and leaf nodes, respectively. The rendering results based on leaf nodes and fine-tuning can achieve visual effects very close to those based on grid points, but with a speed increase of over 30 times.

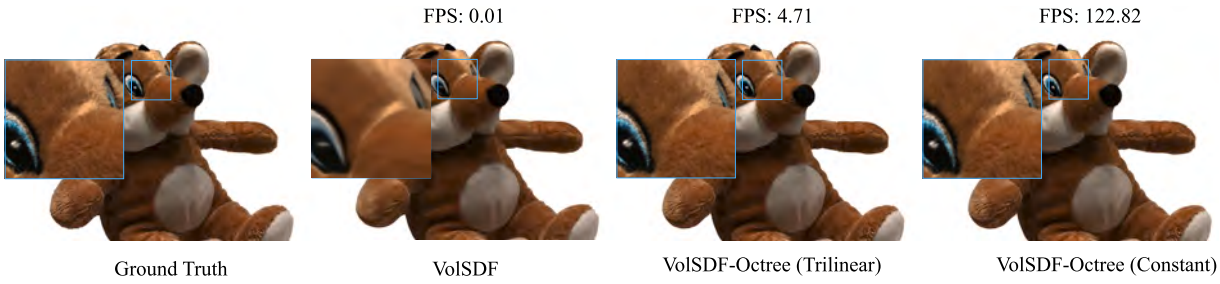

Fig. 6. We can integrate the network-based method VolSDF with our octree structure to boost rendering speed while ensuring rendering quality, achieving real-time rendering effects.

In our experiments, we set $\mathcal{V} = \{\mathbf{p} \mid -0.1 \leq \mathcal{S}(\mathbf{p}) \leq 0.1\}$ when the subdivision level is $l \leq 8$; when $l > 8$, we only select 10% points from $\mathcal{V}$ to calculate $\mathcal{L}_{\text{smooth}}$ for fast training. In terms of β in Eq. (2), we adopt the strategy proposed in (Wu et al., 2023b) to make β gradually decrease as the iterations progress.

In summary, we formulate our optimization problem as follows:

$$\min_{\Theta} \ \mathcal{L} = \mathcal{L}_{\text{col}} + \lambda_t \mathcal{L}_{\text{TV}} + \lambda_s \mathcal{L}_{\text{smooth}}, \tag{10}$$

where λ_t and λ_s represent the weights for the loss terms.

4.5. Real-time rendering

Trilinear interpolation can ensure the continuity of property fields, thus utilizing $\theta_{ij} = (s_{ij}, \mathbf{a}_{ij})$ of grid points can enhance rendering quality. However, the computational complexity of trilinear interpolation is relatively high. To further accelerate the rendering speed, when the resolution of the octree is sufficiently high, we can opt to directly use the property values $\hat{\theta}_i = (\hat{s}_i, \hat{\mathbf{a}}_i)$ at the leaf node ℓ_i for rendering. Although the rendering speed has improved significantly, as shown in Fig. 5, this downsampling method will decrease the rendering quality. To mitigate it, we continue fine-tuning the property values of leaf nodes, which helps the rendering quality be comparable to trilinear interpolation.

4.6. Integration with network-based methods

Neural network-based methods (Yariv et al., 2021; Wang et al., 2021a; Yariv et al., 2020) are proficient at reconstructing geometry from multi-view images. However, the forward inference of these methods is computationally intensive, leading to slow rendering. Our method exhibits strong extensibility and can be integrated with pre-trained network-based methods to achieve real-time rendering effects. We demonstrate the extension process using VolSDF (Yariv et al., 2021) as an example. We modify the output of the color network in VolSDF from colors to SH coefficients, which we refer to as VolSDF-SH in our paper. Instead of adding the view directions to the input of the color network in VolSDF-SH, we input the given view directions after obtaining the SH coefficients to compute the color of sampling points. For the conversion to our octree, we also adopt a fine-to-coarse strategy. We input the coordinates of the grid points into the VolSDF-SH network to obtain the corresponding SDF values and SH coefficients. Similarly, we subdivide the leaf nodes according to the SDF values. After achieving the target resolution, we perform fine-tuning on the grid points' property values through training images. For the post-conversion to fast real-time rendering by property values at leaf nodes, please refer to Sec. 4.5. The qualitative results can be seen in Fig. 6.

Table 1
Quantitative evaluation on NeRF-synthetic dataset.

		Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
PSNR(↑)	VolSDF	30.75	20.07	20.64	35.72	29.01	28.90	30.76	21.86
	Plenoxels	33.87	25.40	31.86	36.56	34.17	29.02	33.17	29.78
	Voxurf	34.86	25.63	34.52	37.39	33.84	29.81	34.20	28.73
	BakedSDF (*Offline*)	**36.09**	25.18	29.42	**38.52**	34.59	**31.86**	35.24	30.88
	3DGS	35.75	**26.32**	**35.53**	38.17	**36.28**	30.53	**36.70**	**31.78**
	Ours (*Trilinear*, $l = 9$)	34.24	24.43	30.70	34.92	31.74	25.01	31.41	27.83
	Ours (*Trilinear*, $l = 10$)	34.62	24.58	31.13	35.01	32.43	25.30	31.14	28.22
	Ours (*Constant*, $l = 9$)	33.91	24.38	30.49	34.75	31.68	25.47	31.54	27.65
	Ours (*Constant*, $l = 10$)	34.21	24.57	30.74	34.91	32.02	25.15	31.49	27.81
CD(↓)	VolSDF	1.57	5.46	8.59	6.56	3.31	1.14	1.06	20.89
	Voxurf	**1.27**	0.52	0.90	3.95	1.31	1.08	0.82	**7.68**
	BakedSDF (*Offline*)	1.59	1.65	**0.31**	**1.01**	**0.37**	**0.77**	**0.70**	18.91
	Ours (*Trilinear*, $l = 9$)	1.44	0.39	1.01	4.10	2.68	1.43	0.89	17.51
	Ours (*Trilinear*, $l = 10$)	1.39	**0.37**	0.95	3.94	2.55	1.44	0.99	16.49
FPS(↑)	VolSDF	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01
	Plenoxels	17.86	14.18	18.66	13.88	13.47	13.11	19.33	8.45
	Voxurf	1.27	1.10	1.31	1.13	1.08	1.16	1.32	0.95
	3DGS	119.51	169.03	**216.33**	233.57	173.08	**299.82**	186.49	156.63
	Ours (*Trilinear*, $l = 9$)	18.51	14.04	15.42	15.30	13.65	15.72	18.35	9.93
	Ours (*Trilinear*, $l = 10$)	17.23	12.50	14.78	14.38	13.88	16.09	17.05	10.22
	Ours (*Constant*, $l = 9$)	**288.62**	**292.87**	168.47	**395.68**	**390.43**	249.17	**692.03**	**255.36**
	Ours (*Constant*, $l = 10$)	185.90	144.15	140.37	275.86	**313.67**	211.80	378.85	208.33

5. Results

5.1. Experimental setup

Implementation We realize the octree by creating a specialized PyTorch CUDA extension library, which enables efficient and differentiable volume rendering, contributing to a notable reduction in training time. We employed SH of degree 2. We experimentally set $\lambda_t^{SH} = 10^{-5}$ and turn off the SH TV term after $l > 8$ in order to prevent excessive smoothness and ensure the recovery of details. For models with brighter scenes (i.e. Drums with metallic and transparent materials and Ship with water reflections in NeRF-Synthetic (Mildenhall et al., 2021) dataset; Omni-IntenseLight dataset with a large area of white noise in the images (see Fig. 8), which affected by the intense light of the real shooting environment), we recommend increasing λ_t^{SH} in the range of $(10^{-5}, 10^{-3}]$. A small λ_t^{SH} will result in coarse shape with holes when the octree resolution is low, and it is difficult to patch the holes up as the octree resolution increases.

For smooth regularization, we set $\lambda_s = 2 \times 10^{-4}$ and $\lambda_t^{SDF} = 10^{-6}$ for the SDF TV term. We conducted our experiments on an NVIDIA Tesla V100 GPU with 32 GB memory. In most cases, we can obtain an optimized octree with $l = 9$ after 75k iterations in 40 minutes and an octree with $l = 10$ after 100k iterations in 60 minutes.

Datasets We validated our proposed method on three benchmark datasets: NeRF-Synthetic (Mildenhall et al., 2021), DTU (Jensen et al., 2014), and OmniObject3D (Wu et al., 2023a). The NeRF-Synthetic dataset comprises 8 scenes, each featuring a central object surrounded by 100 inward-facing cameras randomly distributed on the upper hemisphere. The images are 800×800 and come with corresponding ground truth camera poses. OmniObject 3D provides a variety of scanned objects, each with 100 images per scene, all at a resolution of 800×800. To showcase the robustness of our method, we specifically select six scenes characterized by intense lighting and strong specular highlights. We refer to this subset as the *Omni-IntenseLight* sub-dataset. For the comparison method, we evaluated against VolSDF (Yariv et al., 2021) and Plenoxels (Fridovich-Keil et al., 2022). Voxurf concentrates mainly on the accuracy of the geometric reconstruction, whereas PermutoSDF and BakedSDF are more geared toward achieving real-time rendering. Since the official source code of BakedSDF is not available, we opted to use the implementation from SDFStudio (Yu et al., 2022a) and only evaluated the performance of offline rendering. We report quantitative results of these two datasets in Table 1 and Table 3, respectively, which include PSNR (rendering quality), CD (geometric quality), and FPS (rendering efficiency).

5.2. Training and real-time rendering

The results demonstrate that our method achieves real-time rendering performance while maintaining competitive rendering quality. Our rendering speed for some objects is the fastest of all comparison methods, even surpassing 3DGS. In the Omni-IntenseLight dataset, the PSNRs of our results are lower than those of Voxurf, BakedSDF, and 3DGS, as these methods are able to partially recover the colors in regions affected by intensive lighting and/or specular highlights. However, the surfaces generated by Voxurf and BakedSDF are highly sensitive to lighting conditions, frequently exhibiting holes in areas with high specular highlights or intense lighting. Our method, although cannot recover specular highlights, consistently produces satisfactory geometry across all test models (see Fig. 8). In the NeRF-Synthetic dataset, we can also obtain competitive geometry compared to Voxurf and BakedSDF. Our method

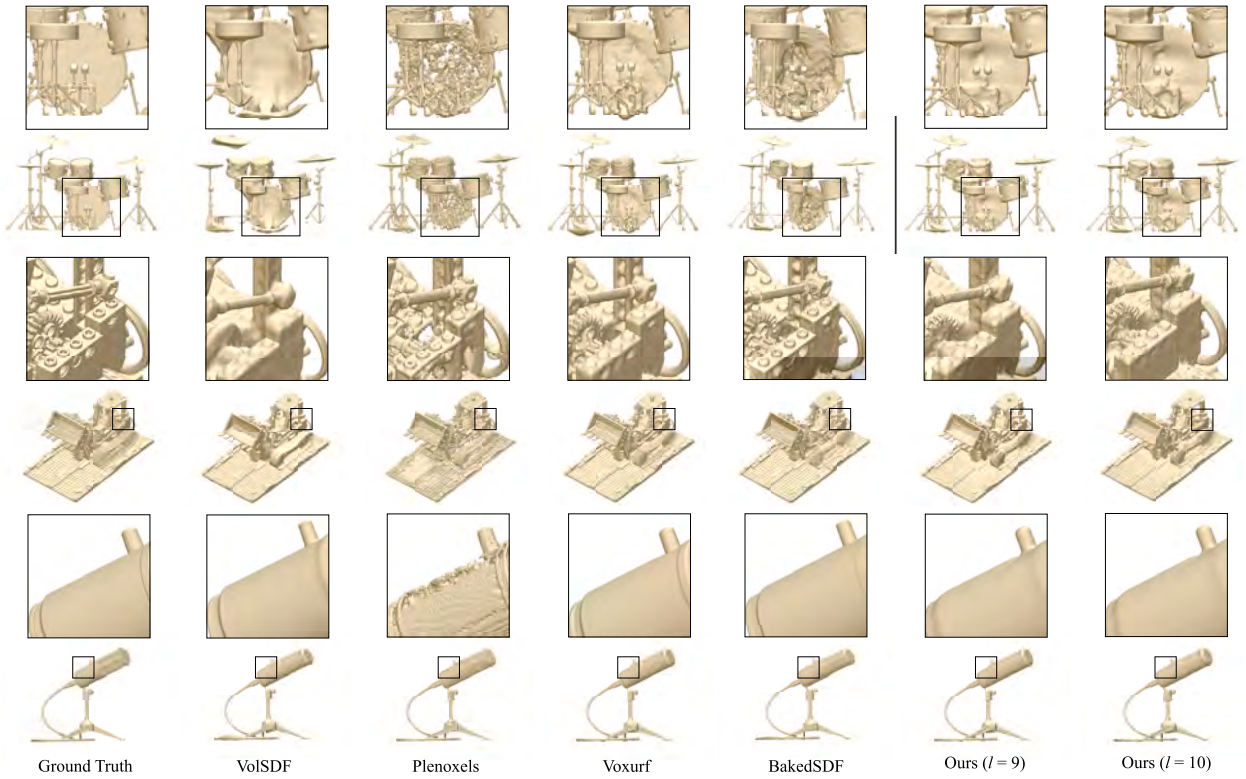

Fig. 7. Comparison of geometric reconstruction results on the NeRF-synthetic dataset. Close-up views reveal the quality of the reconstructed geometry.

Table 2
Quantitative evaluation of FPS (↑) on the DTU dataset.

ID	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122
VolSDF	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
PermutoSDF	17.24	22.05	15.18	19.15	26.65	20.57	15.41	30.86	23.72	22.51	22.78	23.13	24.34	24.20	26.66
Ours (*Trilinear*, $l = 10$)	5.59	4.22	6.20	6.04	5.74	5.56	4.67	5.41	5.18	4.71	5.28	5.08	5.02	5.17	5.98
Ours (*Constant*, $l = 10$)	**146.27**	**130.07**	**155.00**	**174.22**	**119.07**	**117.16**	**100.87**	**143.77**	**116.20**	**122.82**	**152.41**	**139.51**	**134.56**	**128.45**	**186.57**

is not as good as these two methods in terms of geometric details, but in the case of Drum, which contains transparent materials, the quality of our geometry is better (see Fig. 7).

We assess the performance of our proposed VolSDF-Octree framework on the DTU dataset. The results in Table 2 demonstrate that the rendering speed has been significantly increased and can now be achieved in real-time. PermutoSDF (Rosu and Behnke, 2023) suffers from artifacts at surfaces with a grazing angle when rendering in real time (Rosu and Behnke, 2023). Both rendering quality and rendering speed of our method is better than PermutoSDF under the DTU dataset. Fig. 11 and Fig. 9 respectively show the results of synthesizing novel view rendering images on the Omni-intenseLight and DTU dataset, with our method being able to achieve comparatively realistic images in real-time.

To better illustrate the differences between the methods, we qualitatively compared each method as shown in Table 4. For Plenoxels, RT-Octree, INGP and 3DGS, these four methods are density-based and cannot reconstruct high-fidelity geometry. Compared to RT-Octree, Plenoxels is faster on training, which can reduce the training time from hours to less than 15 min, but its rendering speed is slower than RT-Octree (15 FPS VS 200+ FPS on NeRF-Synthetic dataset). The training time of INGP is less than 15 min and its rendering spend on NeRF-Synthetic dataset is about 30 FPS, which is not outstanding. 3DGS is superior in both training speed (less than 15 min) and rendering speed (around 200 FPS), but it lacks a good geometric representation. To obtain the geometry from 3DGS, we applied iPSR to the optimized Gaussian points. Since some of the Gaussian points are inside the object, the geometric quality reconstructed by iPSR is relatively low. SuGaR first trains on 3DGS for 7k steps, and then jointly optimizes Gaussian points and meshes, leading to surface with better quality. However, its results still contain holes, as shown in Fig. 12. (Since SuGaR does not generate surface in the invisible region, for the fairness of comparison, the CD in Fig. 12 is obtained after cutting off the object by invisible region in images of dataset.). VolSDF, BakedSDF, Voxurf, PermutoSDF and our method are SDF-based, which can reconstruct good geometry. VolSDF and BakedSDF suffer from long training time (10+ h), while Voxurf, PermutoSDF and our method take less than one hour. Our method achieves geometric reconstruction quality comparable to the other neural SDF methods and significantly faster rendering speed than them, while also maintaining a fast training speed (Fig. 14).

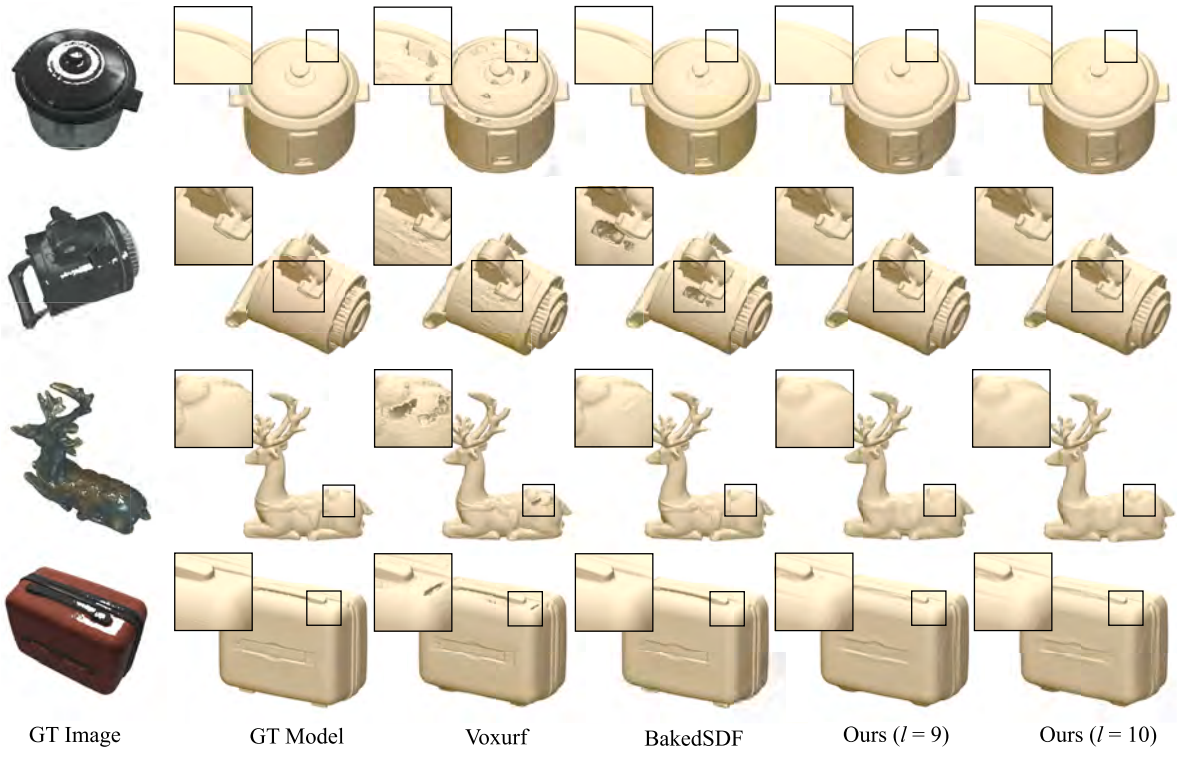

Fig. 8. Comparison of geometric reconstruction results on the Omni-IntenseLight dataset.

Table 3
Quantitative evaluation on Omni-intenseLight dataset.

		Ricecook*	Suitcase	Light	Woman	Deer	Dinosaur
PSNR (↑)	Voxurf	33.88	34.81	35.79	32.86	36.56	39.22
	BakedSDF (*Offline*)	**36.16**	**36.03**	**37.16**	**33.36**	37.36	**40.26**
	3DGS	34.90	33.91	37.02	32.61	**37.42**	40.10
	Ours (*Trilinear*, $l = 9$)	30.63	32.13	33.04	31.99	35.65	37.88
	Ours (*Trilinear*, $l = 10$)	30.48	31.77	32.59	32.04	35.58	37.87
	Ours (*Constant*, $l = 9$)	30.67	32.50	33.68	31.88	35.73	38.01
	Ours (*Constant*, $l = 10$)	30.50	32.46	33.39	31.88	35.67	37.93
CD(↓)	Voxurf	3.17	0.98	1.96	4.92	0.63	**0.52**
	BakedSDF (*Offline*)	**0.01**	**0.22**	22.04	0.98	-	0.82
	Ours (*Trilinear*, $l = 9$)	0.07	0.54	**0.66**	**0.88**	0.23	0.85
	Ours (*Trilinear*, $l = 10$)	0.08	0.53	0.67	**0.88**	**0.22**	0.82
FPS(↑)	Voxurf	1.08	1.05	1.48	1.53	1.68	1.50
	3DGS	**303.12**	**306.25**	308.08	**307.66**	303.38	**308.14**
	Ours (*Trilinear*, $l = 9$)	15.77	26.47	19.28	18.22	25.46	20.58
	Ours (*Trilinear*, $l = 10$)	11.29	25.33	17.62	17.70	23.08	21.63
	Ours (*Constant*, $l = 9$)	295.36	278.82	**345.67**	138.49	**394.94**	284.51
	Ours (*Constant*, $l = 10$)	176.53	199.04	220.10	124.21	287.39	275.04

* There is no visible view for the bottom of Ricecooker and Deer. To ensure a fair comparison, we cropped the bottom of all reconstructed models for Ricecooker. In the case of Deer, only BakedSDF is significantly affected.

Table 4
Qualitative comparison of various methods.

	VolSDF	Plenoxels	RT-Octree	BakedSDF	Voxurf	PermutoSDF	INGP	3DGS	SuGaR	Ours
Geometry Quality	High	Low	Low	High	High	High	Low	Low	Medium	High
Rendering Quality	Medium	High	High	High	High	High	High	High	High	Medium
Training Efficiency	Low	High	Low	Low	High	High	High	High	High	High
Real-time Rendering	Low	Medium	High	High	Low	Medium	Medium	High	High	High

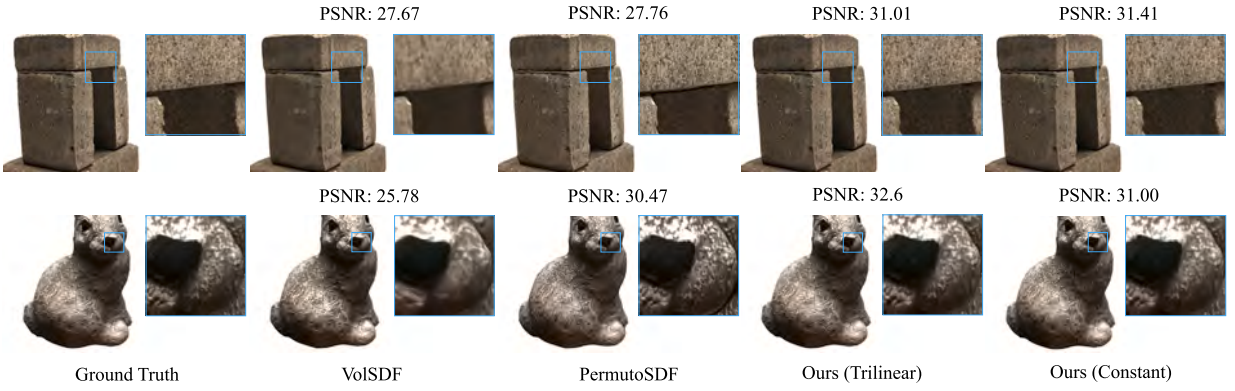

Fig. 9. Comparison of rendering results under novel views on the DTU dataset.

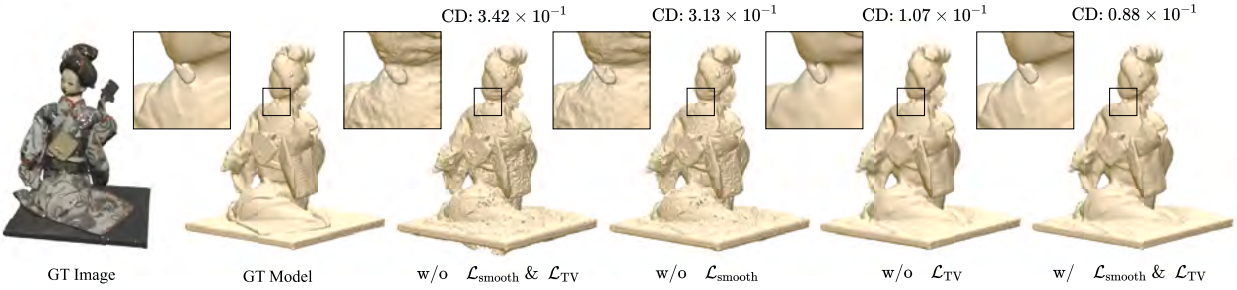

Fig. 10. Ablation study of the smoothness regularization.

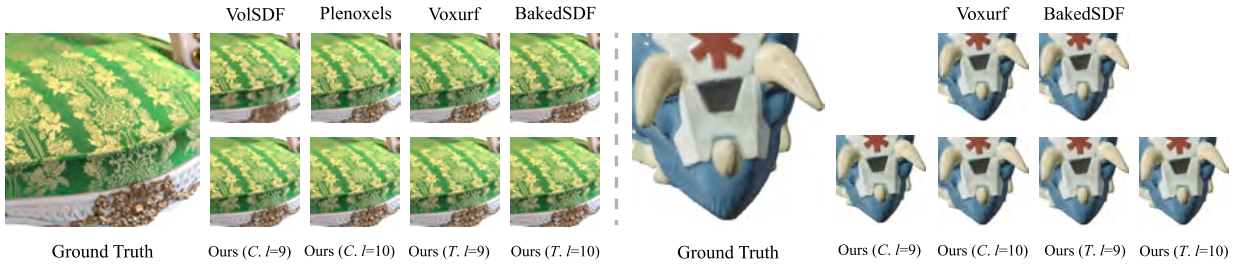

Fig. 11. Comparison of rendering results under novel views on the NeRF-synthetic and Omni-IntenseLight dataset. *C.* and *T.* indicate constant and trilinear scheme, respectively.

5.3. Ablation study

We ablate the necessity of smooth regularization and show the quantitative results in Fig. 10. It can be seen that the reconstructed surface tends to be rough with holes without smooth regularization. The introduction of $\mathcal{L}_{\text{smooth}}$ can ameliorate this problem, and $\mathcal{L}_{\text{TV}}$ on SDF can further improve the geometry.

6. Conclusions

In this study, we propose a network-free framework based on octree structures to reconstruct implicit surface and radiance fields from multi-view images. By explicitly representing signed distance fields, our approach facilitates real-time rendering while maintaining high geometric quality. We adopt a coarse-to-fine strategy which quickly obtains good geometric initialization in the rough stage by applying Gaussian smoothing terms. The gradually refined subdivision minimizes space waste to ensure the reconstruction of geometric details in the subsequent fine stage. We provide a conversion system that can combine other network-based techniques with our octree representations, allowing for real-time rendering. Our method can handle images with resolutions up to 2K, such as the one shown in Fig. 13, and render them in real-time.

Our method struggles to represent sharp highlights accurately. This is due to the fact that Spherical Harmonics are effective for simulating low-frequency lighting information, like indirect lighting, but are not well suited for expressing high-frequency lighting information, such as highlights or reflections. Besides, there is still room for improving the reconstruction of fine geometric details,

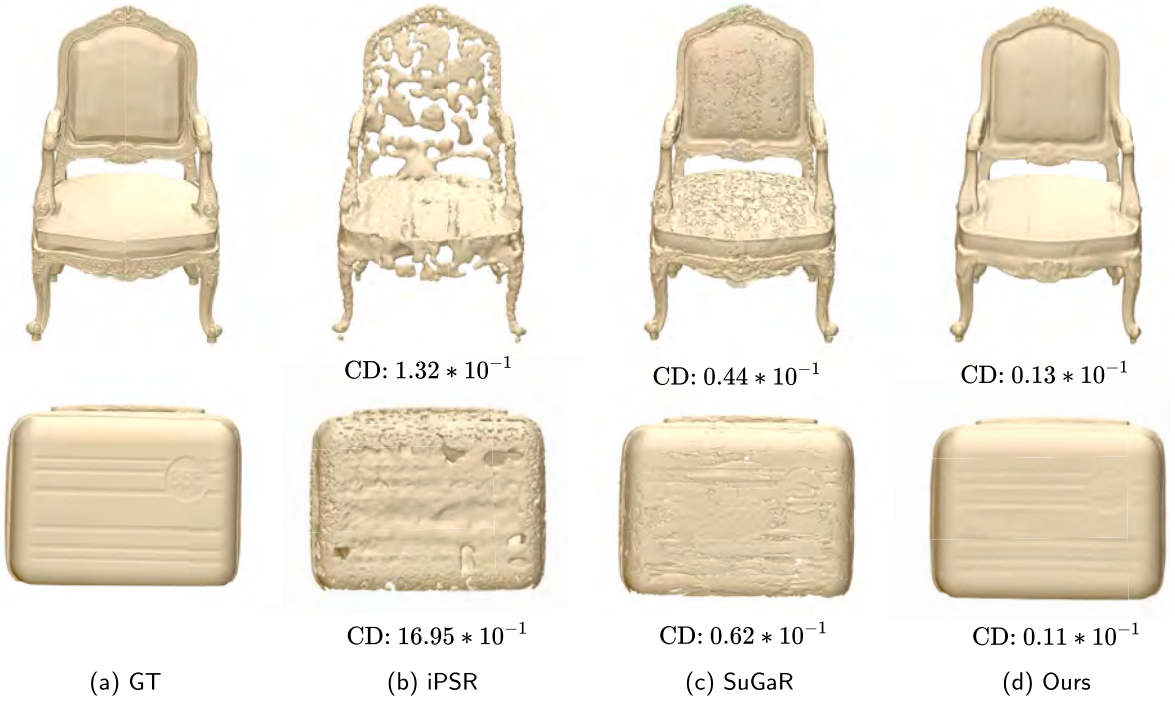

Fig. 12. Comparison with 3DGS+iPSR (Hou et al., 2022) and SuGaR (Guédon and Lepetit, 2023) in terms of geometry quality.

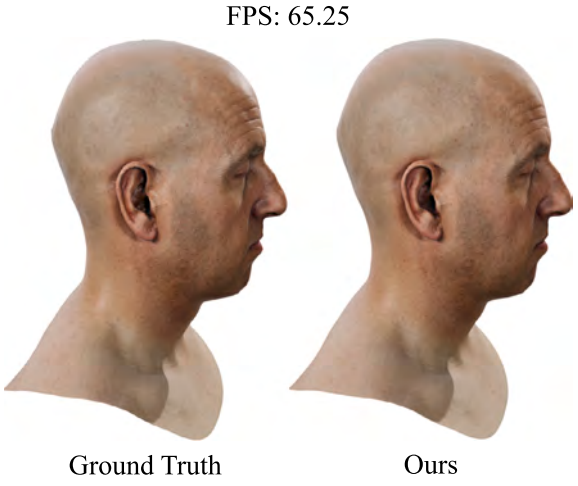

Fig. 13. Real-time rendering of 2K images.

and we will consider solving this problem in the future through higher order terms such as Hessian. In addition, to accelerate the training, we cache the neighboring node information and some parameters that are fixed in the optimization after each subdivision of the octree. These caches will take up GPU memory, and under the Tesla V100 32GB, subdividing octree to $l = 10$ has reached the bottleneck of the GPU, making it difficult for us to process datasets in 4K and higher. During real-time rendering, the octree size usually takes 1GB of memory at $l = 9$ and 2GB at $l = 10$. In the future, we will reduce the memory requirement for both training and rendering such as trying different data structure like OpenVDB (Museth, 2013) to try datasets with higher resolution images. It is noteworthy that our explicit representation is based on octree structures, and facilitates easier editing of scenes and objects compared to network-based implicit methods, which will be a direction for our future exploration. In addition, we conducted our experiments on the relatively low-performance Tesla V100 device. We believe that with the advent of more advanced hardware, our method will achieve superior performance.

Fig. 14. Additional 3D reconstruction results from the NeRF-Synthetic dataset (row 1) and the Omni-Intense Light dataset (row 2).

CRediT authorship contribution statement

Jiaze Li: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology. **Luo Zhang:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology. **Jiangbei Hu:** Writing – review & editing, Writing – original draft, Validation, Supervision, Conceptualization. **Zhebin Zhang:** Funding acquisition. **Hongyu Sun:** Funding acquisition. **Gaochao Song:** Data curation. **Ying He:** Writing – review & editing, Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code and associated data for this study are accessible online. They can be downloaded from the following URL: https://github.com/LaoChui999/Octree-VolSDF.

Acknowledgement

This project was partially supported by the Ministry of Education, Singapore, under its Academic Research Fund Grants (MOE-T2EP20220-0005 & RT19/22), and a gift fund from OPPO.

References

Atkinson, K., Han, W., 2012. Spherical Harmonics and Approximations on the Unit Sphere: an Introduction, vol. 2044. Springer Science & Business Media. https://api.semanticscholar.org/CorpusID:118983975.

Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P., 2021. Mip-NeRF: a multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5855–5864.

Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H., 2022. Tensorf: tensorial radiance fields. In: European Conference on Computer Vision. Springer, pp. 333–350.

Chen, Z., Funkhouser, T., Hedman, P., Tagliasacchi, A., 2023. MobileNeRF: exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16569–16578.

Darmon, F., Bascle, B., Devaux, J.C., Monasse, P., Aubry, M., 2022. Improving neural implicit surfaces geometry with patch warping. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6260–6269.

Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A., 2022. Plenoxels: radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5501–5510.
Fridovich-Keil, S., Meanti, G., Warburg, F.R., Recht, B., Kanazawa, A., 2023. K-Planes: explicit radiance fields in space, time, and appearance. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12479–12488.
Fu, Q., Xu, Q., Ong, Y.S., Tao, W., 2022. Geo-NeuS: geometry-consistent neural implicit surfaces learning for multi-view reconstruction. Adv. Neural Inf. Process. Syst. 35, 3403–3416. https://doi.org/10.48550/arXiv.2205.15848.
Gao, Q., Xu, Q., Su, H., Neumann, U., Xu, Z., 2023. Strivec: sparse tri-vector radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).
Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J., 2021. FastNeRF: high-fidelity neural rendering at 200fps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 14346–14355.
Guédon, A., Lepetit, V., 2023. SuGaR: surface-aligned Gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. arXiv preprint. arXiv:2311.12775.
Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P., 2021. Baking neural radiance fields for real-time view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5875–5884.
Hou, F., Wang, C., Wang, W., Qin, H., Qian, C., He, Y., 2022. Iterative Poisson surface reconstruction (iPSR) for unoriented points. ACM Trans. Graph. 41, 1–13. https://doi.org/10.48550/arXiv.2209.09510.
Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., Aanæs, H., 2014. Large scale multi-view stereopsis evaluation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 406–413. https://ieeexplore.ieee.org/document/6909453.
Johari, M.M., Lepoittevin, Y., Fleuret, F., 2022. GeoNeRF: generalizing nerf with geometry priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18365–18375.
Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G., 2023. 3D Gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. 42, 1–14. https://doi.org/10.48550/arXiv.2308.04079.
Li, Z., Muller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H., 2023. Neuralangelo: high-fidelity neural surface reconstruction. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8456–8465.
Liu, L., Gu, J., Zaw Lin, K., Chua, T.S., Theobalt, C., 2020. Neural sparse voxel fields. Adv. Neural Inf. Process. Syst. 33, 15651–15663. https://doi.org/10.48550/arXiv.2007.11571.
Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R., 2021. NeRF: representing scenes as neural radiance fields for view synthesis. Commun. ACM 65, 99–106. https://doi.org/10.48550/arXiv.2003.08934.
Min, C., Gibou, F., 2008. New finite difference methods in quadtree grids. https://api.semanticscholar.org/CorpusID:202612679.
Müller, T., Evans, A., Schied, C., Keller, A., 2022. Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. 41, 1–15. https://doi.org/10.1145/3528223.3530127.
Museth, K., 2013. VDB: high-resolution sparse volumes with dynamic topology. ACM Trans. Graph. 32, 1–22. https://doi.org/10.1145/2487228.2487235.
Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A., 2011. Kinectfusion: real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality. Ieee, pp. 127–136.
Niemeyer, M., Geiger, A., 2021. Giraffe: representing scenes as compositional generative neural feature fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11453–11464.
Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A., 2020. Differentiable volumetric rendering: learning implicit 3d representations without 3d supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3504–3515.
Oechsle, M., Peng, S., Geiger, A., 2021. Unisurf: unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5589–5599.
Reiser, C., Peng, S., Liao, Y., Geiger, A., 2021. KiloNeRF: speeding up neural radiance fields with thousands of tiny mlps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 14335–14345.
Rosu, R.A., Behnke, S., 2023. PermutoSDF: fast multi-view reconstruction with implicit surfaces using permutohedral lattices. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8466–8475.
Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M., 2016. Pixelwise view selection for unstructured multi-view stereo. In: Computer Vision–ECCV 2016: 14th European Conference. Amsterdam, the Netherlands, October 11-14, 2016, Proceedings, Part III 14. Springer, pp. 501–518.
Shu, Z., Yi, R., Meng, Y., Wu, Y., Ma, L., 2023. RT-Octree: accelerate plenoctree rendering with batched regular tracking and neural denoising for real-time neural radiance fields. In: SIGGRAPH Asia 2023 Conference Papers, pp. 1–11.
Sun, C., Sun, M., Chen, H.T., 2022a. Direct voxel grid optimization: super-fast convergence for radiance fields reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5459–5469.
Sun, C., Sun, M., Chen, H.T., 2022b. Improved direct voxel grid optimization for radiance fields reconstruction. arXiv preprint. arXiv:2206.05085.
Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H., 2022. Block-NeRF: scalable large scene neural view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8248–8258.
Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P., 2022. Ref-NeRF: structured view-dependent appearance for neural radiance fields. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 5481–5490.
Wang, L., Zhang, J., Liu, X., Zhao, F., Zhang, Y., Zhang, Y., Wu, M., Yu, J., Xu, L., 2022a. Fourier plenoctrees for dynamic radiance field rendering in real-time. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13524–13534.
Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W., 2021a. NeuS: learning neural implicit surfaces by volume rendering for multi-view reconstruction. Adv. Neural Inf. Process. Syst. 34, 27171–27183. https://doi.org/10.48550/arXiv.2106.10689.
Wang, Y., Skorokhodov, I., Wonka, P., 2022b. HF-NeuS: improved surface reconstruction using high-frequency details. Adv. Neural Inf. Process. Syst. 35, 1966–1978. https://doi.org/10.48550/arXiv.2206.07850.
Wang, Y., Han, Q., Habermann, M., Daniilidis, K., Theobalt, C., Liu, L., 2023a. NeuS2: fast learning of neural implicit surfaces for multi-view reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).
Wang, Z., Wu, S., Xie, W., Chen, M., Prisacariu, V.A., 2021b. NeRF–: neural radiance fields without known camera parameters. arXiv preprint. arXiv:2102.07064.
Wang, Z., Shen, T., Nimier-David, M., Sharp, N., Gao, J., Keller, A., Fidler, S., Müller, T., Gojcic, Z., 2023b. Adaptive shells for efficient neural radiance field rendering. ACM Trans. Graph. 42. https://doi.org/10.1145/3618390.
Weng, C.Y., Curless, B., Srinivasan, P.P., Barron, J.T., Kemelmacher-Shlizerman, I., 2022. HumanNeRF: free-viewpoint rendering of moving people from monocular video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16210–16220.
Whelan, T., Salas-Moreno, R.F., Glocker, B., Davison, A.J., Leutenegger, S., 2016. Elasticfusion: real-time dense slam and light source estimation. Int. J. Robot. Res. 35, 1697–1716. https://api.semanticscholar.org/CorpusID:21124365.
Wilhelms, J., Van Gelder, A., 1992. Octrees for faster isosurface generation. ACM Trans. Graph. 11, 201–227. https://doi.org/10.1145/130881.130882.
Wu, T., Zhang, J., Fu, X., Wang, Y., Ren, J., Pan, L., Wu, W., Yang, L., Wang, J., Qian, C., Lin, D., Liu, Z., 2023a. OmniObject3D: large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).
Wu, T., Wang, J., Pan, X., Xu, X.X., Theobalt, C., Liu, Z., Lin, D., 2023b. Voxurf: voxel-based efficient and accurate neural surface reconstruction. In: International Conference on Learning Representations (ICLR).

Yang, B., Bao, C., Zeng, J., Bao, H., Zhang, Y., Cui, Z., Zhang, G., 2022. Neumesh: learning disentangled neural mesh-based implicit field for geometry and texture editing. In: European Conference on Computer Vision. Springer, pp. 597–614.

Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., Lipman, Y., 2020. Multiview neural surface reconstruction by disentangling geometry and appearance. Adv. Neural Inf. Process. Syst. 33, 2492–2502. https://doi.org/10.48550/arXiv.2003.09852.

Yariv, L., Gu, J., Kasten, Y., Lipman, Y., 2021. Volume rendering of neural implicit surfaces. Adv. Neural Inf. Process. Syst. 34, 4805–4815. https://doi.org/10.48550/arXiv.2106.12052.

Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P.P., Szeliski, R., Barron, J.T., Mildenhall, B., 2023. BakedSDF: meshing neural sdfs for real-time view synthesis. In: ACM SIGGRAPH 2023 Conference Proceedings.

Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A., 2021. Plenoctrees for real-time rendering of neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5752–5761.

Yu, Z., Chen, A., Antic, B., Peng, S.P., Bhattacharyya, A., Niemeyer, M., Tang, S., Sattler, T., Geiger, A., 2022a. SDFStudio: a unified framework for surface reconstruction. https://github.com/autonomousvision/sdfstudio.

Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A., 2022b. MonoSDF: exploring monocular geometric cues for neural implicit surface reconstruction. Adv. Neural Inf. Process. Syst. 35, 25018–25032. https://doi.org/10.48550/arXiv.2206.00665.

Zhang, K., Riegler, G., Snavely, N., Koltun, V., 2020. NeRF++: analyzing and improving neural radiance fields. arXiv preprint. arXiv:2010.07492.