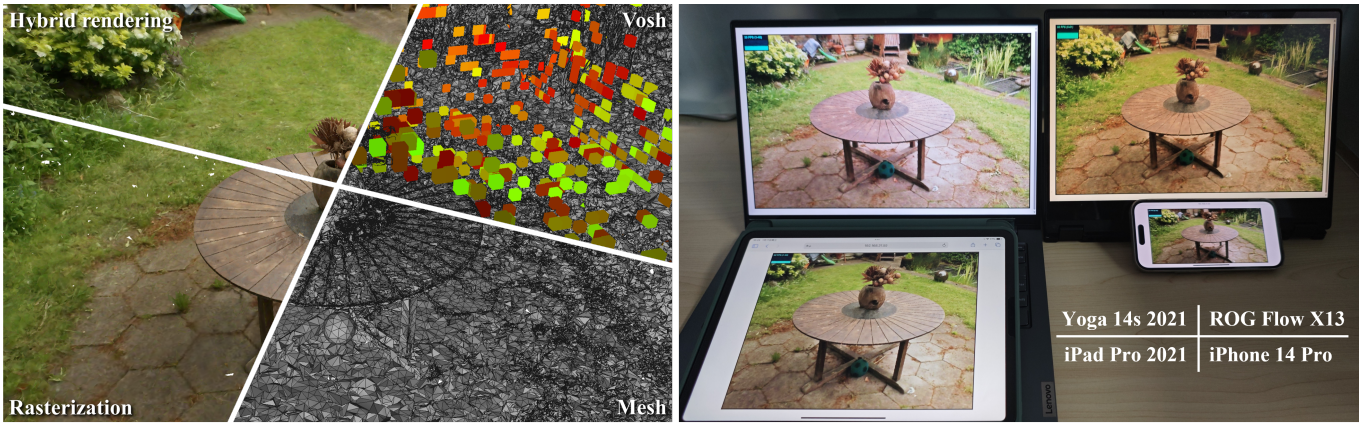
Vosh: Voxel-Mesh Hybrid Representation for Real-Time View Synthesis

CHEN-HAO ZHANG, Beijing Institute of Technology, China
YONG-YANG ZHOU, Beijing Institute of Technology, China
LEI ZHANG*, Beijing Institute of Technology, China



Voxel-mesh hybrid representation and rendering

Real-time view synthesis on different mobile devices

Fig. 1. **Left:** The hybrid representation *Vosh*, combines both voxel component (highlighted with pseudo colors) and mesh component (indicated by gray triangles) in hybrid rendering. **Right:** The proposed method facilitates real-time view synthesis on various mobile devices, e.g., laptops and mobile phones.

The neural radiance field (NeRF) has emerged as a prominent methodology for synthesizing realistic images of novel views. While neural radiance representations based on voxels or mesh individually offer distinct advantages, excelling in either rendering quality or speed, each has limitations in the other aspect. In response, we propose a pioneering hybrid representation named *Vosh*, seamlessly combining both voxel and mesh components in hybrid rendering for view synthesis. Vosh is meticulously crafted by optimizing the voxel grid of NeRF, strategically with selected voxels replaced by mesh. Therefore, it excels in fast rendering scenes with simple geometry and textures through its mesh component, while simultaneously enabling high-quality rendering in intricate regions by leveraging voxel component. The flexibility of Vosh is showcased through the ability to adjust hybrid ratios, providing users the ability to control the balance between rendering quality and speed based on flexible usage. Experimental results demonstrates that our method achieves commendable trade-off between rendering quality and speed, and notably has real-time performance on mobile devices.

CCS Concepts: • **Computing methodologies** → **Reconstruction**; *Neural networks*; *Volumetric models*.

Additional Key Words and Phrases: Neural Radiance Field, View Synthesis, Real-Time Rendering.

1 INTRODUCTION

The novel view synthesis based on the Neural Radiance Field (NeRF) leverages neural networks to model three-dimensional scenes and generates images from specified viewpoints [Mildenhall et al. 2021]. In general, NeRF-based approaches demonstrate proficiency in producing high-quality rendering for novel views with sparse inputs, effectively handling materials with specular and translucent properties. This versatility positions them for diverse applications in autonomous navigation [Kwon et al. 2023], virtual reality/augmented reality [Li et al. 2022a], and beyond [Jiang et al. 2023; Sun et al. 2022b]. Despite these merits, current methods face challenges in achieving a balance between high-quality rendering and real-time rendering, particularly when deployed on mobile devices.

The rendering speed of neural radiance fields is significantly influenced by the chosen representation, with notable implications for sampling requirements and query costs [Yu et al. 2021]. Typically, the original NeRF uses deep multilayer perceptrons (MLPs) to model neural radiance fields, relying on dense sampling for querying to approximate integrals in volume rendering. Unfortunately, this approach typically causes excessively long rendering times. A substantial amount of effort has explored alternatives such as implicit voxel representations [Fridovich-Keil et al. 2022; Hedman et al. 2021; Liu et al. 2020; Müller et al. 2022; Reiser et al. 2021; Sun et al. 2022a] and explicit mesh representations [Chen et al. 2023; Rakotosaona et al. 2023; Tang et al. 2023; Yariv et al. 2023], aiming to exclude irrelevant regions for computation. These methods reduce the number of sampling points and employ smaller MLPs to expedite rendering. However, even though approaches based on implicit voxels can achieve high-quality rendering in complex scenes, the dense sampling in non-empty regions is prohibitively expensive.

Methods based on mesh representation have demonstrated significantly accelerated rendering speeds, enabling real-time rendering

*Corresponding author.

Authors' addresses: Chen-hao Zhang, Beijing Institute of Technology, Beijing, China, zachzhang07@163.com; Yong-yang Zhou, Beijing Institute of Technology, Beijing, China, 3220231274@bit.edu.cn; Lei Zhang, Beijing Institute of Technology, Beijing, China, leizhang@bit.edu.cn.

on common mobile devices. However, the challenge lies in the reconstruction of scene-level mesh structure, particularly in areas with intricate geometry and textures (see Figure 1), such as thin structures or undetermined regions like backgrounds [Reiser et al. 2023]. It is observed that meshes can match the rendering quality akin to voxels in regions with simple geometry and textures, as well as offering highly efficient rendering. Therefore, a promising solution involves employing hybrid representations for reconstruction towards both high-quality rendering and fast speed. Existing works on hybrid representations [Guo et al. 2023; Wang et al. 2023], focus primarily on object or proximal region reconstruction, thus limiting flexibility to represent complex scenes and control over different parts of the hybrid representation. Alternatively, we leverage the strengths of voxels and mesh, to design a hybrid representation towards a better balance between quality and speed, especially enabling high-quality rendering on mobile devices (see Figure 1).

In summary, the main contribution of this paper is a new hybrid representation namely *Vosh* (VOxel-meSH), by combing voxels and mesh for NeRF-based real-time view synthesis. It also allows for a dynamic control of the balance between voxel and mesh components by voxel adjustment. Such flexibility results in a versatile spectrum of NeRF representations, catering to the varied computational capabilities that facilitate real-time view synthesis across different devices. Experimental results exhibit the super performance of our method compared to state-of-the-art methods.

2 RELATED WORK

There are numerous published approaches to view synthesis and NeRF, and we refer interested readers to [Gao et al. 2022; Tewari et al. 2022] for a comprehensive overview. In this context, we focus on the real-time view synthesis and NeRF representations most related to our method.

2.1 Real-time view synthesis

NeRF [Mildenhall et al. 2021] as a well-known technology for view synthesis, has been the focus of numerous endeavors aimed at enhancing its rendering speed while maintaining high-quality rendering. SNeRG [Hedman et al. 2021] achieves fast rendering without CUDA, using sparse voxel representations and a compact neural network for deferred rendering. Plenoxels [Fridovich-Keil et al. 2022] and PlenOctrees [Yu et al. 2021] expedite ray marching with sparse voxel representation, refining voxels near object surfaces. MERF [Reiser et al. 2023] enhances rendering speed and spatial occupancy by replacing pure voxel grids with a low-resolution voxel grid and a high-resolution triplane projection representation [Chan et al. 2022]. These methods mostly excel on high-end GPUs, whereas real-time rendering on mobile devices still remains a challenge.

MobileNeRF [Chen et al. 2023] achieves fast rendering by constraining NeRF to triangles and using rasterization. NeRF2Mesh [Tang et al. 2023] uses NeRF to initialize geometric structures, iteratively refining mesh attributes to improve rendering quality. BakedSDF [Yariv et al. 2023] optimizes a blended neural signed distance function representation, enabling real-time rendering of complex scenes on a high-quality mesh. While mesh-based methods offer fast rendering, they may suffer quality loss compared to voxel-based methods.

3D Gaussian Splatting [Kerbl et al. 2023] creates a 3D Gaussian representation around each point starting from a point cloud model, utilizing the Splatting method for real-time rendering of high-resolution scenes. But it suffers elongated and popping artifacts, and high memory usage for deployment on mobile phones. In contrast, our hybrid representation combines the strengths of voxel and mesh representations, allowing high-quality rendering of complex scenes on mobile devices.

2.2 Hybrid Scene Representation

Combining explicit mesh representation with implicit fields introduces functionalities such as NeRF editing and rendering acceleration. NeuMesh [Yang et al. 2022] encodes a neural implicit field with disentangled geometry and texture on mesh vertices, enabling a range of neural rendering-based editing operations. EyeNeRF [Li et al. 2022b] integrates an explicit surface with an implicit volume representation to model the human eye. Mesh-Aware-RF [Qiao et al. 2023] embeds a known polygonal mesh into NeRF, employing ray tracing and ray marching alternately to update radiance along rays.

The hybrid representation like Adaptive Shells [Wang et al. 2023] and VMesh [Guo et al. 2023] has been used in view synthesis. Adaptive Shells confines volume rendering to a narrow band around the object for effective NeRF rendering. VMesh, closely related to our work, uses a textured mesh and auxiliary sparse voxels but differs by employing SDF field and volume density field to simultaneously learn the entire scene. However, VMesh relies on gradient descent optimization that lacks controllability and flexibility. Indeed, VMesh mainly focuses on rendering individual objects or bounded scenes, while the proposed method in this paper can handle more scenes, such as unbounded outdoor scenes.

3 METHOD OVERVIEW

We start with an overview of our methodology to establish a context for the core algorithms described in the next section. As depicted in Figure 2, our method contains two key phases: the training phase for constructing a hybrid representation and the inference phase for view synthesis rendering. In the training phase, a set of multi-view images is employed to train a voxel-mesh hybrid representation, and the associated assets are exported. In the inference phase, our method achieves real-time rendering based on the baked hybrid representation assets.

Concretely, the training phase consists of three stages: grid training, voxels to mesh conversion and optimization. In the grid training stage, we train a voxel grid model through volume rendering. Then, leveraging the trained voxel grid, we extract an explicit surface and train the surface using differentiable surface rendering for both appearance and geometry optimization. This converts the initial voxel grid into a combination of voxel and mesh components. Finally, the voxels and mesh are further optimized through a training process involving hybrid rendering and voxels adjustment to obtain the desired hybrid representation. Once training is complete, we export assets containing sparse voxels and textured meshes. Subsequently, real-time view synthesis on mobile devices is implemented using WebGL. Next, we elaborate the algorithm details in the training and inference phases.

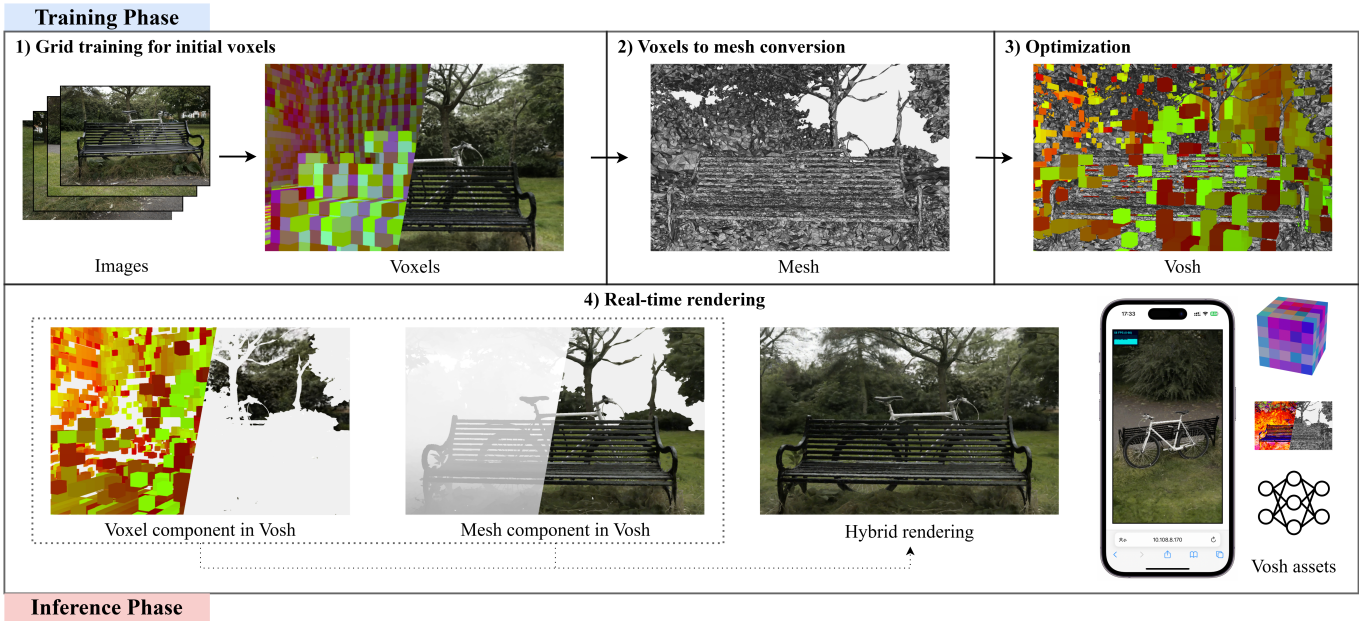


Fig. 2. **An overview of the proposed methodology**. The training phase starts from grid training for obtaining initial voxels. Then, a portion of voxels becomes mesh in the voxels to mesh conversion. Subsequently, the combination of voxels and mesh are optimized through hybrid rendering and voxels adjustment to obtain the final hybrid representation Vosh. The inference phase realizes real-time hybrid rendering with Vosh even on mobile phones.

4 HYBRID REPRESENTATION CONSTRUCTION

The construction of a hybrid representation in the training phase commences with the initialization of a volume comprising solely of voxels. This volume is then subject to a conversion process wherein a portion of the voxels is replaced with an explicit mesh according to rendering quality on par with the original voxel representation.

4.1 Grid training for initial voxels

Initially, the entire scene is reconstructed using a high-resolution voxel grid. It retains sufficient geometric and appearance information essential for subsequent mesh extraction and hybrid representation. Here, we employ SNeRG++ (an improved version of SNeRG [Hedman et al. 2021] in MERF [Reiser et al. 2023]) as the benchmark for our voxel grid, due to its capability to effectively model unbounded scenes and generate real-time rendering assets.

Concretely, SNeRG++ is built upon the volume rendering principles of vanilla NeRF, which involves sampling along the direction of a ray originating from the ray origin o along the direction d to obtain sample points $\mathbf{x}_i = o + t_i\mathbf{d}$. Based on the spatial position of these sample points, it uses the neural radiance field to calculate volumetric density σ_i, resulting in rendering weights w_i:

$$w_i = \alpha_i T_i, \qquad T_i = \prod_{j=1}^{i-1}(1-\alpha_j), \qquad \alpha_i = 1 - e^{-\sigma_i \delta_i}, \tag{1}$$

where T_i, α_i and δ_i represent the transmittance, opacity and spacing of sampling point i, respectively. SNeRG++ adopts the concept of deferred rendering, decomposing the appearance information of the neural radiance field into diffuse color $\mathbf{c}_{d,i}$ and view-dependent features f_i. By cumulatively sampling the appearance information along the ray, as shown in Figure 4a, we obtain the diffuse color C_d and view-dependent features F of the ray as

$$\mathbf{C}_d = \sum_i w_i \mathbf{c}_{d,i}, \qquad \mathbf{F} = \sum_i w_i \mathbf{f}_i. \tag{2}$$

Finally, the view-dependent color is calculated by inputting the diffuse color $\mathbf{C}_d$, view-dependent features F, and the ray direction d into a miniature MLP. The result is then added to the diffuse color $\mathbf{C}_d$ to obtain the final color C of the ray.

Besides, in order to enhance the modeling of regions situated far from the center within unbounded scenes and facilitate efficient ray marching calculations, SNeRG++ incorporates a contraction function founded on piecewise projection. This contraction function is defined as follows:

$$\text{contract}(\mathrm{x})_j = \begin{cases} x_j, & \text{if } ||x||_\infty \le 1 \\ \frac{x_j}{||x||_\infty}, & \text{if } x_j \ne ||x||_\infty > 1 \\ (2 - \frac{1}{|x_j|})\frac{x_j}{|x_j|}, & \text{if } x_j = ||x||_\infty > 1 \end{cases}, \tag{3}$$

where $||\cdot||_\infty$ denotes the infinity norm, defined as $||x||_\infty = \max_j |x_j|$.

4.2 Voxels to mesh conversion

Based on the above high-resolution voxel grid, we employ Marching Cubes [Lorensen and Cline 1998] to extract an explicit mesh. However, surfaces derived solely from density information in volume rendering often exhibit roughness and are unsuitable for direct use in hybrid representation. Inspired by NeRF2Mesh [Tang et al. 2023], we adopt a synergistic approach involving differentiable rendering and surface operations to refine the explicit mesh.

The objective is to convert a subset of voxels that demonstrate similar rendering quality in both voxel and explicit mesh representations into a refined mesh. Typically, these converted voxels correspond to regions in the scene characterized by uncomplicated geometry and texture. The surface training specifically entails the coarse selection of voxels and subsequent surface refinement using differentiable rendering techniques.

Coarse selection of voxels. By analyzing the rendering results of NeRF2Mesh [Tang et al. 2023] (Figure 5) and BakedSDF [Yariv et al. 2023], we found that using mesh representation to reconstruct the unbounded background region far from the camera presents a formidable challenge. Achieving rendering quality comparable to voxels in this remote area is inherently demanding. Moreover, influenced by the contraction function, triangles extracted from voxels in the contraction region tend to enlarge and may break apart as they extend away from the camera. The presence of large triangles can adversely affect rendering quality, particularly on WebGL platforms, while introducing subdivision for these large triangles to enhance rendering quality incurs additional storage overhead.

Thus, we only convert voxels with surfaces located in the region near the camera into mesh. Criteria such as edge length and the number of triangles in isolated patches guide the decision on which explicit surfaces to retain. By default, we preserve the cube area with 90% of the length of the bounding box in outdoor scenes. This conversion strategy, depicted in Figure 3, contrasts with indiscriminate conversion of voxels containing implicit surfaces into triangular meshes, which results in diminished rendering quality. So these explicit surfaces are further refined in the next differentiable appearance optimization.

(a) Conversion w/o coarse selection (b) Conversion w/ coarse selection

Fig. 3. The voxels to mesh conversion without (w/o) coarse selection of voxels obtains worse rendering results than with (w/) coarse selection, as shown by the diffuse of Vosh for voxel and mesh components respectively.

Differentiable surface refinement. We perform differentiable rasterization operations on the mesh to obtain intersections between rays and surfaces. Appearance information is obtained by querying the voxel representation based on the corresponding spatial positions. We also employ nvdiffrast [Laine et al. 2020] for differentiable rendering to optimize the appearance information of the mesh, while simultaneously tracking the rendering error corresponding to each face for surface optimization. It is worth noting that we do not enforce the mesh to represent the entire scene. Therefore, during differentiable rendering, we only calculate the loss for pixels corresponding to rays that intersect with the surface. We also perform face merging and mid-point subdivision operations based on the normal change of the faces and rendering errors. In contrast to NeRF2Mesh [Tang et al. 2023], which solely relies on rendering errors for operations, we additionally utilize the local normal change rate of faces as an optimization criterion. This enhances face density in regions with complex geometry and reduces storage overhead for faces in areas with simple geometry.

Following the conversion from voxels to mesh, we have obtained an initial hybrid representation that includes both voxels and mesh. Nevertheless, recognizing significant potential for optimization in rendering quality and volume occupancy for both representations, we are committed to further refining the hybrid representation through further fine-tuning.

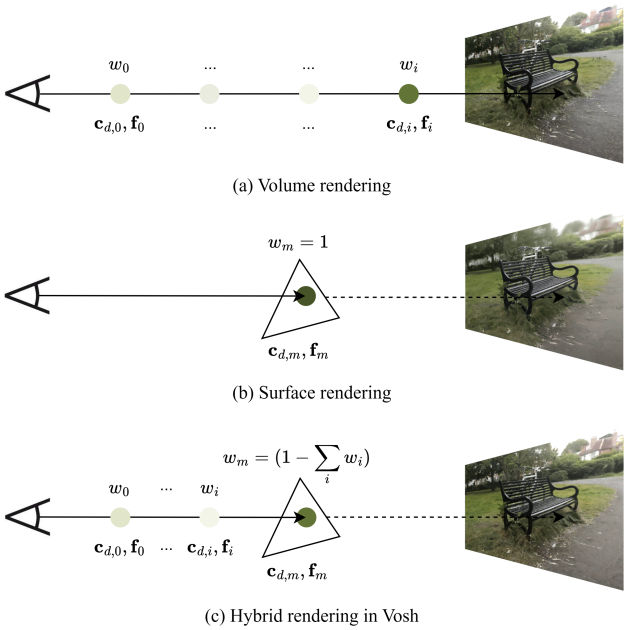


Fig. 4. The proposed hybrid rendering (c) integrates volume rendering (a) and surface rendering (b) for the Vosh.

4.3 Optimization

After the mesh conversion and optimization stage, we obtain an explicit surface suitable for representing regions in the scene with simple geometry and textures, as well as a optimized high-resolution voxel grid. By combining these two components, we can obtain a coarse hybrid representation with a significant amount of redundancy between the components. This may result in low rendering efficiency and potential rendering conflicts. So we further perform hybrid rendering optimization and voxel adjustment based on the components for a more compact and efficient representation.

Compared to volume rendering, surface rendering only considers the intersection points between rays and surface, significantly reducing the time cost associated with ray marching. This inspires us to confine the ray marching in volume rendering to occur only before the rays hit the surface. Simultaneously, performing volume rendering between the viewpoint and the surface also provides possibilities for enhancing surface rendering quality. Therefore, we straightforwardly combine volume rendering and surface rendering, treating the surface points obtained through rasterization as the final points reached by ray marching in volume rendering. The hybrid rendering can be expressed by the following three equations:

$$\mathbf{C}_{vosh} = \mathbf{C}_{vosh,d} + \mathrm{MLP}(\mathbf{C}_{vosh,d}, \mathbf{F}_{\mathrm{vosh}}, \mathbf{d}), \tag{4}$$

$$\mathbf{F}_{vosh} = \sum_i w_i \mathbf{f} + w_m \mathbf{f}_m, \tag{5}$$

$$\mathbf{C}_{vosh,d} = \sum_i w_i \mathbf{c}_{d,i} + w_m \mathbf{c}_{d,m}, \tag{6}$$

where $\mathbf{C}_{vosh}$, $\mathbf{C}_{vosh,d}$ and $\mathbf{F}_{vosh}$ represent the color, diffuse color, and view-dependent features after hybrid rendering, respectively. $w_m = (1 - \sum_i w_i)$ represents the rendering weight for the mesh, which is the remaining portion of the cumulative voxel weights. $\mathbf{c}_{d,m}$ and $\mathbf{f}_m$ represent the diffuse color and view-dependent features at the intersection point where the ray hits the surface. Three different rendering methods are shown in Figure 4.

Based on the optimized mesh, we attempt to dynamically adjust the voxel grid, forming a high-quality hybrid representation. Since the trained mesh already possesses the capability to represent regions with simple geometry and textures, we need to consider how to simplify the voxel representation for the entire scene. We focus on retaining voxels in regions with complex geometry and textures to better complement the rendering of the hybrid representation with the mesh. Inspired by PlenOctrees [Yu et al. 2021], we propose a voxel adjustment loss based on surface mesh weights. Specifically, we calculate the rendering weight of mesh w_m for rays hitting the surface and force it close to 1. The implementation is as follows:

$$\mathcal{L}_{voxel} = \lambda_{voxel} \frac{1}{K} \sum_k (1 - \exp(w_m - 1)), \tag{7}$$

where λ_{voxel} represents the weight of the voxel adjustment loss, and K is the number of samples taken in the rays hitting the surface.

The number of retained voxels is an important factor in determining the balance between rendering quality and speed. Concretely, retaining more voxels can enhance rendering quality but slow down rendering speed, while retaining fewer voxels can accelerate rendering speed but compromise rendering quality. Setting different voxel adjustment weights to limit the volumetric density during ray marching is one of the methods to achieve different trade-offs between speed and quality. Additionally, voxels and mesh may overlap at the same spatial location, potentially affecting rendering quality and the volume of exported assets. To address this issue, we first calculate the occupancy grid of the meshes under the training poses. During voxel adjustment, we ignore the voxels in the mesh-occupied grid. We also adjust the ratio of sparse voxels in the hybrid representation by setting different resolutions r_{mesh} for the mesh-occupancy grid.

5 REAL-TIME RENDERING

The real-time rendering based on Vosh is realized as a JavaScript 3D web application, leveraging the capabilities of the MERF real-time renderer. The rendering process is also hybrid and structured into two passes: the rasterization pass and the ray marching pass. Both passes are employed to make rendering result consistency between training on the GPU and inference on the web.

In the rasterization pass, meshes are uniformly shaded. The fragment shader takes charge of sampling appearance textures, incorporating diffuse color and view-dependent feature. The output of the rasterization pass includes corresponding diffuse, view-dependent features and depth map. These output is then used in the subsequent ray marching pass.

In the ray marching pass, we also employ a multi-resolution hierarchical structure for the occupancy grid introduced in SNeRG [Hedman et al. 2021] to mark the position of occupied voxels. This structure creates binary mask images in multiple resolution levels through max-pooling. We use the multi-resolution occupancy grid to perform spatial skips during ray marching. Simultaneously, the depth map obtained in the rasterization pass further reduces the sampling distance.

For each spatial coordinate after ray marching, we compare the transformed depth with the depth map from the rasterization pass to determine whether the ray has intersected the surface position. If the ray reaches the surface of the mesh, the marching stops, and the accumulated result of ray marching is weightedly combined with the appearance result from the rasterization pass. Leveraging the surface depth information from the rasterization pass proves instrumental in significantly diminishing the number of required sampling points during ray marching, thereby effectively reducing the computational cost associated with volume rendering.

6 EXPERIMENTS

We have implemented real-time view synthesis using the proposed hybrid representation Vosh, and carried out comprehensive evaluations encompassing both rendering quality and speed. We also compare our method with state-of-the-art (SOTA) methods, like offline classical methods (NeRF [Mildenhall et al. 2021], Mip-NeRF 360 [Barron et al. 2022], Instant-NGP [Müller et al. 2022]) and real-time methods (Mobile-Nerf [Chen et al. 2023], NeRF2Mesh [Tang et al. 2023], BakedSDF [Yariv et al. 2023], MERF [Reiser et al. 2023], SNeRG++ [Hedman et al. 2021]). As done by SOTA methods, we used the challenging unbounded dataset Mip-NeRF 360 [Barron et al. 2022] for evaluation. Next, we elaborate the details of the experimental results.

6.1 Rendering quality comparison

We conduct both quantitative and qualitative evaluation of rendering quality for view synthesis obtained by the proposed method and SOTA methods. Quantitative evaluation uses the metrics of PSNR, SSIM [Wang et al. 2004] and LPIPS [Zhang et al. 2018] for the assessment of rendering quality. Table 1 shows the statistics of the quantitative evaluation of different methods.

Here, we classify these methods based on their ability to achieve real-time rendering on mobile devices like laptops and mobile phones.

Table 1. Quantitative evaluation of the proposed method and SOTA methods with 3 outdoor scenes and 4 indoor scenes from Mip-NeRF 360 dataset [Barron et al. 2022]. The methods that can render in real-time on mobile phones are highlighted.

Methods	Representations	Outdoor			Indoor			GTX 1660Ti	iPhone 14 Pro
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FPS @ 1920*1080	FPS @ 344*707
NeRF	MLP	21.46	0.458	0.515	26.84	0.790	0.370	-	-
Mip-NeRF 360	MLP	**25.92**	**0.747**	**0.244**	**31.72**	**0.917**	**0.179**	-	-
Instant-NGP	Voxel	23.90	0.648	0.369	29.47	0.877	0.273	-	-
SNeRG++$_{1024}$	Voxel	24.11	0.625	0.377	27.32	0.802	0.276	9.0	-
MERF	Voxel + Plane	24.41	0.678	0.298	27.80	0.855	0.271	9.7	25.3
BakedSDF	Mesh	23.40	0.577	0.351	27.20	0.845	0.300	131.0	-
MobileNeRF	Mesh	22.90	0.524	0.463	25.75	0.757	0.453	92.7	153.3
NeRF2Mesh	Mesh	22.74	0.497	0.463	25.20	0.721	0.340	231.0	210.0
Vosh-Light	Voxel + Mesh	23.63	0.586	0.392	27.02	0.792	0.294	79.7	70.0
Vosh-Base	Voxel + Mesh	**23.74**	**0.592**	**0.389**	**27.07**	**0.791**	**0.291**	38.0	33.7

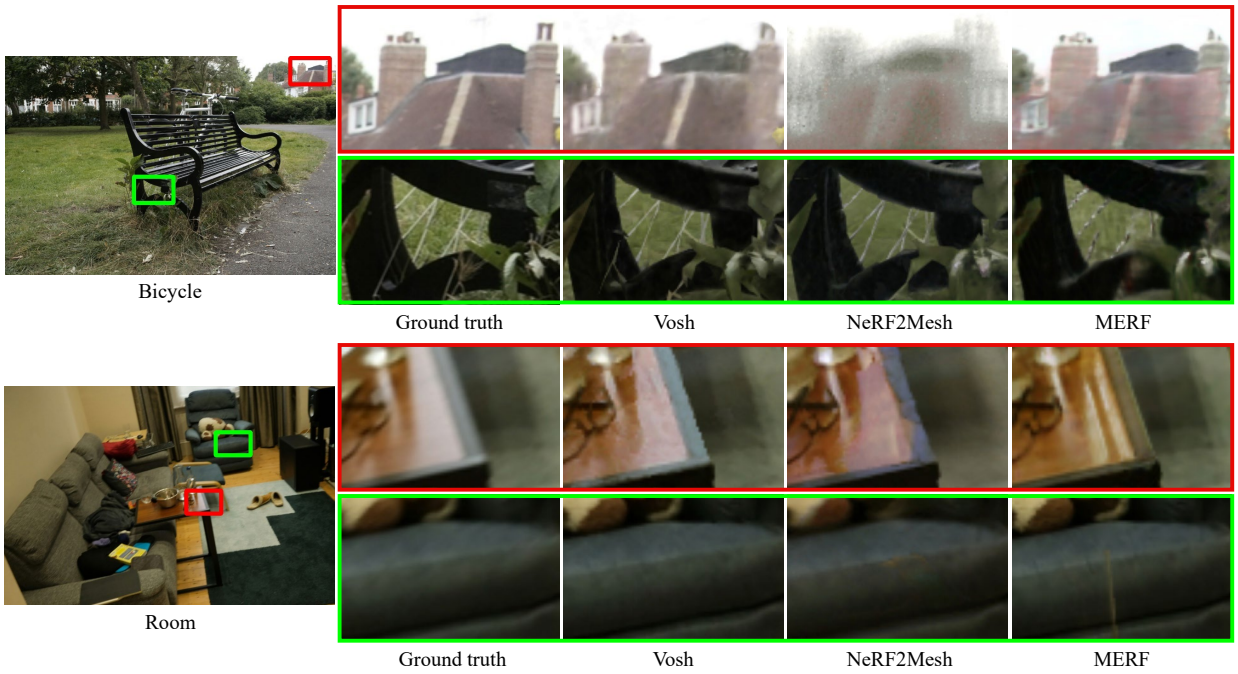


Fig. 5. The rendering results and zoomed-in images obtained by our method, as well as some SOTA methods based on voxels or mesh representation.

Methods such as SNeRG, MERF, and BakedSDF can only attain real-time rendering on select laptops with robust GPUs. It can be seen there exists an obvious gap between representations based on voxels and meshes. While BakedSDF narrows the quality gap with its millions of triangles, its substantial memory consumption makes it impractical for deployment on mobile phones.

Vosh demonstrates superior rendering quality in real-time rendering on mobile devices, even surpassing some methods that cannot achieve real-time rendering on mobile devices. For the outdoor scenes, Vosh enhances rendering quality over mesh-based methods by leveraging sparse voxels through the use of a contraction function (see Figure 5, 6 and 7). However, for the indoor scenes, methods have limitations in the representational capacity of tiny view-dependent MLPs. While Vosh achieves real-time rendering on mobile devices by employing a tiny view-dependent MLP, it has slightly lower rendering quality in indoor scenes compared to MLP-based methods.

We provide a qualitative comparison with SOTA methods based on voxels [Reiser et al. 2023] and mesh [Tang et al. 2023] on the examples in Figure 5. We select one distant view and one close-up view for local magnification for better exhibition. The mesh-based methods often lack the ability to represent background regions far from the camera location, whereas voxel-based methods necessitate higher computational and storage costs to achieve enhanced rendering quality. Contrarily, our method improves rendering speed while essentially maintaining the rendering quality comparable to

that of voxel-based methods. We refer the reader to the companion video for visual demonstrations of the results.

Comparison with VMesh. It is worth noting that VMesh [Guo et al. 2023] is also built on the hybrid representation of volume and mesh, but primarily designed for rendering individual objects or scenes with boundaries, posing challenges in handling unbounded scenes like the examples in Figure 5 and 7. Nonetheless, our method produces better rendering results than VMesh, even for the synthetic bounded object as shown in Figure 6.

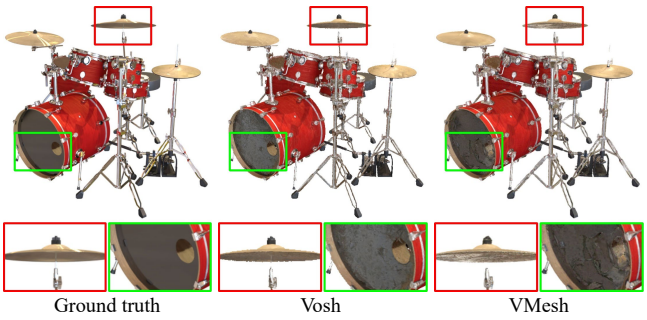


Fig. 6. The rendering results obtained by ours and VMesh [Guo et al. 2023].

6.2 Rendering speed comparison

We evaluated the performance of the methods applicable for rendering on mobile devices, including SNeRG++ [Hedman et al. 2021], MERF [Reiser et al. 2023], BakedSDF [Yariv et al. 2023], MobileNeRF [Chen et al. 2023], and NeRF2Mesh [Tang et al. 2023]. For SNeRG++ and MERF, we follow their configurations with progressive upsampling turned off. We conduct testing on the iPhone 14 Pro equipped with the A16 chip with a resolution of 344*707 and on the Legion Y7000P 2019, which features NVIDIA GeForce GTX 1660Ti (Mobile version, 80W) and operates with a resolution of 1920*1080.

Rendering speed tests for all comparative methods were conducted on the Mip-Nerf 360 dataset [Barron et al. 2022] in three outdoor scenes, using identical device settings, resolutions, and viewpoints. As shown on the right side of Table 1, our method can run with the real-time frame rate (>30 FPS) on both iPhone 14 Pro and GTX 1660Ti. Particularly on the 1660Ti, our method demonstrates a considerable speed improvement compared to MERF and SNeRG. The latter two methods experience slow rendering speeds due to the necessity to sample each alive voxel traversed by every ray. BakedSDF is difficult to render on mobile phones and laptops with limited VRAM due to its substantial assets. MobileNeRF and NeRF2Mesh achieve rendering speeds that reach the maximum refresh rates of their respective devices (60Hz for Safari on iPhone 14 Pro and 144Hz for Legion Y7000P 2019). We make statistics on their potential rendering speeds using overlapping multiple passes.

6.3 Ablation Study

We conduct ablation experiments on the voxels to mesh conversion and voxel adjustment parts of the hybrid representation in Table 2. The assessment involves a quantitative comparison, considering both rendering quality and speed.

The 'full bound' row refers to the results of the variant that do not use coarse selection of voxels in the voxels to mesh conversion. The experimental results show that converting all voxels containing surfaces to mesh not only reduces the rendering quality of the final hybrid representation, but also significantly increases the storage of the assets.

In the voxel adjustment stage, we compare the complete model with variants of Vosh-Base, incorporating different loss weights or resolutions for mesh occupancy grids. The rows with λ_{voxel} show the results of different hyper-parameter weights of the loss function for voxel adjustment. The rows with r_{mesh} indicate the results of using mesh occupancy grids with different resolutions. For the Vosh-Base model, we use $\lambda = 0.001$ and $r_{mesh} = 128$ as hyper-parameters. For the Vosh-Light model, we use $\lambda = 0.1$ and $r_{mesh} = 32$ as hyper-parameters. Experimental results suggest that increasing the weight of voxel adjustment loss or reducing the resolution of the mesh occupancy grid aids in compressing the number of voxels, leading to speed benefits at the expense of rendering quality.

Table 2. The ablation study in the construction of hybrid representation.

Variants	PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑
Vosh-Base	23.74	0.592	0.389	33.7
full bound	23.49	0.583	0.395	36.0
$\lambda_{voxel} = 0$	23.73	0.591	0.389	30.0
$\lambda_{voxel} = 0.01$	23.72	0.590	0.389	38.7
$\lambda_{voxel} = 0.1$	23.69	0.590	0.390	46.7
$r_{mesh} = 0$	23.80	0.593	0.386	32.0
$r_{mesh} = 1024$	23.79	0.593	0.387	32.0
$r_{mesh} = 64$	23.70	0.590	0.389	37.7
Vosh-Light	23.63	0.586	0.392	70.0

6.4 Limitations

Although the proposed hybrid representation Vosh brings valuable advantages to real-time view synthesis, it is not without limitations. We adopt the same view-dependent MLP as SNeRG and MERF, consequently sharing some similarities with their drawbacks. Notably, it faces challenges such as the incapability to model view-dependent colors for translucent objects, and represent extremely large scenes or objects with complex reflections. These limitations, in turn, may result in potential degradation of mesh optimization quality, thereby impacting the rendering quality based on the hybrid representation.

7 CONCLUSION

We have presented a hybrid representation Vosh, composed of both voxels and mesh, for NeRF-based real-time view synthesis. By capitalizing on the strengths of voxel and mesh components in our representation, we attain a controllable balance between rendering quality and speed. Experimental evidence demonstrates that our approach outperforms state-of-the-art methods, showcasing exceptional flexibility for deployment across a range of mobile devices.

In our future work, we intend to explore more efficient data structures for organizing and storing the hybrid representation, aiming to optimize memory consumption and further enhance rendering

efficiency. Additionally, we are considering more compact integration of classic representations to achieve fast and high-quality novel view synthesis specifically tailored for mobile devices.

REFERENCES

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of CVPR*. 5470–5479.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. 2022. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of CVPR*. 16123–16133.

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2023. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of CVPR*. 16569–16578.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. In *Proceedings of CVPR*. 5501–5510.

Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. 2022. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379* (2022).

Yuan-Chen Guo, Yan-Pei Cao, Chen Wang, Yu He, Ying Shan, and Song-Hai Zhang. 2023. VMesh: Hybrid volume-mesh representation for efficient view synthesis. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. 2021. Baking neural radiance fields for real-time view synthesis. In *Proceedings of ICCV*. 5875–5884.

Kaiwen Jiang, Shu-Yu Chen, Hongbo Fu, and Lin Gao. 2023. NeRFFaceLighting: Implicit and Disentangled Face Lighting Representation Leveraging Generative Prior in Neural Radiance Fields. *ACM Transactions on Graphics* 42, 3 (2023), 1–18.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (2023), 1–14.

Obin Kwon, Jeongho Park, and Songhwai Oh. 2023. Renderable Neural Radiance Map for Visual Navigation. In *Proceedings of CVPR*. 9099–9108.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics* 39, 6 (2020), 1–14.

Chaojian Li, Sixu Li, Yang Zhao, Wenbo Zhu, and Yingyan Lin. 2022a. RT-NeRF: Real-Time On-Device Neural Radiance Fields Towards Immersive AR/VR Rendering. In *Proceedings of ICCAD*. 1–9.

Gengyan Li, Abhimitra Meka, Franziska Mueller, Marcel C Buehler, Otmar Hilliges, and Thabo Beeler. 2022b. EyeNeRF: a hybrid representation for photorealistic synthesis, animation and relighting of human eyes. *ACM Transactions on Graphics* 41, 4 (2022), 1–16.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural sparse voxel fields. *Proceedings of NeurIPS* 33 (2020), 15651–15663.

William E Lorensen and Harvey E Cline. 1998. Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 347–353.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics* 41, 4 (2022), 1–15.

Yi-Ling Qiao, Alexander Gao, Yiran Xu, Yue Feng, Jia-Bin Huang, and Ming C. Lin. 2023. Dynamic mesh-aware radiance fields. In *Proceedings of ICCV*. 385–396.

Marie-Julie Rakotosaona, Fabian Manhardt, Diego Martin Arroyo, Michael Niemeyer, Abhijit Kundu, and Federico Tombari. 2023. NeRFMeshing: Distilling Neural Radiance Fields into Geometrically-Accurate 3D Meshes. *arXiv preprint arXiv:2303.09431* (2023).

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of ICCV*. 14335–14345.

Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. 2023. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics* 42, 4 (2023), 1–12.

Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022a. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of CVPR*. 5459–5469.

Jingxiang Sun, Xuan Wang, Yichun Shi, Lizhen Wang, Jue Wang, and Yebin Liu. 2022b. Ide-3d: Interactive disentangled editing for high-resolution 3d-aware portrait synthesis. *ACM Transactions on Graphics (ToG)* 41, 6 (2022), 1–10.

Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. 2023. Delicate Textured Mesh Recovery from NeRF via Adaptive Surface Refinement. In *Proceedings of ICCV*. 17739–17749.

Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. 2022. Advances in neural rendering. In *Computer Graphics Forum*, Vol. 41. 703–735.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic. 2023. Adaptive Shells for Efficient Neural Radiance Field Rendering. *ACM Transactions on Graphics* 42, 6 (2023), 1–15.

Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. 2022. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *Proceedings of ECCV*. 597–614.

Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. 2023. BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–9.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of ICCV*. 5752–5761.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of CVPR*. 586–595.

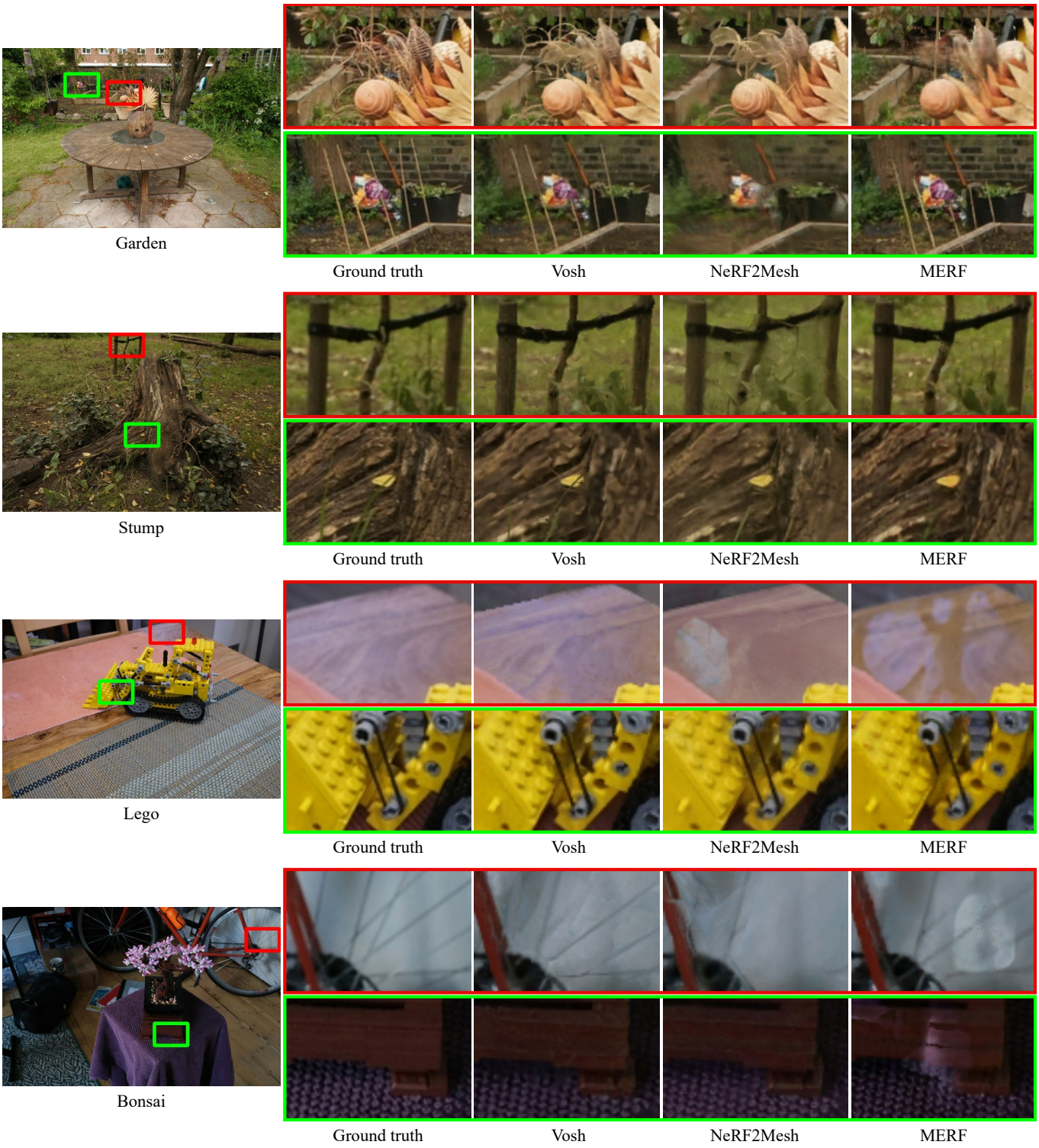


Fig. 7. More rendering results and zoomed-in images obtained by our method, as well as some SOTA methods based on voxels or mesh representation.