MixRT: Mixed Neural Representations For Real-Time NeRF Rendering

Chaojian Li *
Georgia Tech
cli851@gatech.edu

Bichen Wu †
Gen AI, Meta
wbc@meta.com

Peter Vajda
Gen AI, Meta
vajdap@meta.com

Yingyan (Celine) Lin
Georgia Tech
celine.lin@gatech.edu

Project Page: https://licj15.github.io/MixRT

Abstract

Neural Radiance Field (NeRF) has emerged as a leading technique for novel view synthesis, owing to its impressive photorealistic reconstruction and rendering capability. Nevertheless, achieving real-time NeRF rendering in large-scale scenes has presented challenges, often leading to the adoption of either intricate baked mesh representations with a substantial number of triangles or resource-intensive ray marching in baked representations. We challenge these conventions, observing that high-quality geometry, represented by meshes with substantial triangles, is not necessary for achieving photorealistic rendering quality. Consequently, we propose MixRT, a novel NeRF representation that includes a low-quality mesh, a view-dependent displacement map, and a compressed NeRF model. This design effectively harnesses the capabilities of existing graphics hardware, thus enabling real-time NeRF rendering on edge devices. Leveraging a highly-optimized WebGL-based rendering framework, our proposed MixRT attains real-time rendering speeds on edge devices (over 30 FPS at a resolution of 1280 × 720 on a MacBook M1 Pro laptop), better rendering quality (0.2 PSNR higher in indoor scenes of the Unbounded-360 datasets), and a smaller storage size (less than 80% compared to state-of-the-art methods).

1. Introduction

Neural Radiance Field (NeRF), first introduced by [23], has been established as the state-of-the-art (SotA) technique in novel view synthesis tasks, owing to its superior ability to deliver photorealistic rendering quality. Despite its remarkable capabilities, the practical application of NeRF, especially in immersive interactions on edge devices, has been significantly hampered due to its slow rendering speed. Recognizing this limitation, several prior works have proposed various methods to enhance the efficiency of NeRF. These methods, such as baking NeRF into more efficient representations like mesh [8] or sparse voxels [16], have achieved impressive results, demonstrating real-time rendering speed (greater than 30 FPS) on edge devices. Unfortunately, these methods often fall short when applied to larger-scale real-world scenes, either yielding unacceptably slow rendering speeds or requiring prohibitive storage resources. Efforts to overcome these challenges have typically focused on baking NeRF into a high-quality geometry representations (for instance, more than 10 million triangles [38]) or resorting to computationally costly ray marching in the baked representations [28]. Despite offering partial solutions to the challenges, these approaches still suffer from inherent drawbacks related to efficiency and resource requirements.

Upon careful examination, we observe a critical insight that differs from the established conventions. We identify that a high-quality mesh is not necessary in the baked representations concerning rendering quality. Our observation suggests that it is feasible to trade-off the complexity of the baked mesh for a more refined representation of color

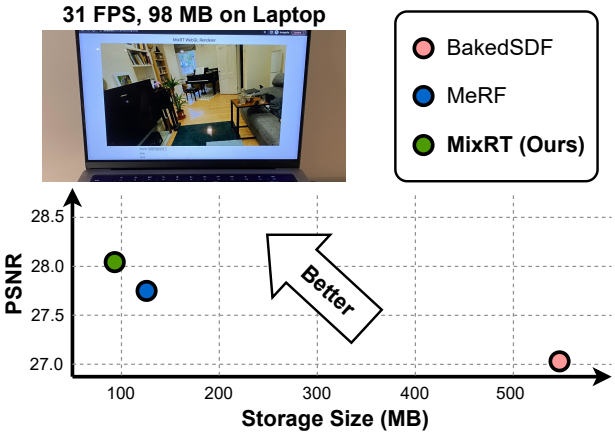

Figure 1. Our proposed MixRT can enable real-time rendering (> 30 FPS) at a resolution of 1280 × 720 on a Macbook M1 Pro laptop with better rendering quality and smaller storage size compared to SotA works on real-time NeRF rendering [28, 38].

*Work done while interning at Meta. †Corresponding Author.

fields (such as NeRF), thereby achieving a more favorable balance between rendering quality and efficiency. Guided by these observations, we propose MixRT, a unique NeRF representation, that mixes different neural representations for real-time NeRF rendering. Specifically, the propsoed MixRT consists of (1) a low-quality mesh (approximately 15 MB compared to over 300 MB in BakedSDF [38]) that provides coarse geometric information of the scene, (2) a view-dependent displacement map to calibrate the ray-mesh intersection points before fetching the corresponding color, and (3) a compressed NeRF model, in the format of an Instant-NGP [24] that provides the density and color of each sampled point. This innovative combination not only ensures the preservation of rendering quality but also maximizes the efficient utilization of available hardware resources, including Rasterizer, Texture Mapping Units, and Single Instruction Multiple Data (SIMD) Units. This balance of resource allocation enables us to achieve real-time rendering speeds on edge devices while minimizing storage requirements, making it an ideal solution for performance-conscious applications.

In summary, our key contributions are as follows:

- Through our observations, we have discovered that achieving high rendering quality in novel view synthesis tasks does not require high-complexity geometry represented by meshes with a vast number of triangles. This revelation has sparked the concept of simplifying the baked mesh and combining various neural representations. As a result, we have experienced substantial improvements in both efficiency and flexibility, paving the way for more efficient and versatile rendering techniques.
- We introduce an innovative NeRF representation, which consists of three essential components: a low-quality mesh, a view-dependent displacement map, and a compressed NeRF model. This carefully crafted design is specifically optimized to fully harness the capabilities of rasterizers, texture mapping units, and SIMD units in current graphics hardware. As a result, it empowers us to achieve real-time NeRF rendering on edge devices without compromising on rendering quality.
- In addition, we develop a highly optimized WebGL-based rendering framework, which allows our proposed MixRT to achieve SotA rendering quality (e.g., PSNR) vs. efficiency (e.g., FPS and storage size) trade-offs.

2. Related Works

2.1. NeRF on Large-Scale Scenes

Previous works on NeRF rendering for large-scale real-world scenes can be divided into two main categories. Works in the first category divide the entire space into multiple sub-spaces, assigning individual NeRFs with specific radii to each. Specifically, [32, 35] align the training of NeRFs for different sub-spaces with collected images in varying lighting conditions; [37] takes this a step further, using different NeRFs for views at varying scales, allowing city-scale scene rendering; [13, 40] enhance the rendering quality by selecting or fusing outputs from multiple sub-space NeRF models. Works in the second category map the entire space into a specific bounded space. In particular, [4] introduces the concept of using a contraction function to fold the unbounded scene domain into a finite sphere; [28] later refines this function, making it piecewise for efficient computation of ray-AABB intersections; [5, 33] subsequently improve the contraction function further to better handle multisample isotropic Gaussian and voxel representations, respectively.

In our approach, MixRT, we employ the contraction function outlined in [4] to configure the NeRF model for the mapped finite sphere. Meanwhile, we retain the low-quality mesh and view-dependent displacement map in their original space. This allows us to capitalize on the optimized rasterization pipeline which is common to most graphics hardware.

2.2. Real-Time NeRF Rendering

Real-time rendering or view synthesis is a vital and challenging problem in computer vision and graphics, given its significance in immersive interaction applications [1]. Early techniques for real-time rendering are either dependent on a vast number of images from densely sampled viewpoints or compromised on rendering quality due to the lack of fine-grained geometry proxies during reconstruction. For instance, [15, 19, 22] exploit light fields to interpolate target images from densely sampled images directly, while [14, 30, 31] utilize multi-view stereo and structure-from-motion pipelines to construct triangle meshes for real-time rendering. NeRF[23], on the other hand, employs a continuous volumetric field, represented in a multi-layer perceptron (MLP) network format for scene reconstruction, achieving state-of-the-art rendering quality thanks to the ease of optimizing MLP representations through gradient descent.

Following NeRF's trailblazing results, subsequent works have proposed "baking" (i.e., pre-computing intermediate results and storing them in buffers) NeRF models into more efficient representations to achieve NeRF's high-quality rendering with real-time speeds. These efficient representations are well-optimized on existing graphics hardware and include triangle meshes or sparse voxels.

In particular, [16, 28, 36, 39] bake NeRF models into sparse voxels with compact storage formats, enabling real-time rendering speeds with existing CUDA or WebGL APIs. On the other hand, [8, 25, 27, 38] adopt triangle meshes in the rendering pipeline, distilling them from pre-trained NeRF models or training them from scratch with differentiable rendering frameworks. Moreover, [6, 20] have

developed either fully convolution-based or MLP-based networks to reconstruct light fields and enable real-time NeRF rendering on mobile devices. However, their wider application is limited either by platform-dedicated deployment tools [2] or is constrained to synthesizing front views only. There also exist point-cloud-based works like [18] that utilize point clouds as scene representations for faster rendering speeds. However, their approaches, heavily relying on custom CUDA kernels for computational efficiency, face limitations in terms of broader adaptability. Specifically, the lack of compatibility with downstream computer graphics toolchains (e.g., editing and making collision animation in Blender [11]) limits their utility across a diverse range of edge devices

Our proposed MixRT is unique among the mentioned real-time NeRF rendering methods, combining a low-quality mesh, a view-dependent displacement map, and a compressed NeRF model in the Instant-NGP format [24]. By leveraging the rasterizers, texture mapping units, and SIMD units accessible by WebGL APIs on most existing devices, MixRT can achieve SotA rendering quality with real-time rendering speeds, suitable storage size, and memory requirements for large-scale real-world scenes (e.g., Unbounded-360 dataset [4]). Specifically, thanks to the adoption of a rasterization-based rendering pipeline, our proposed MixRT not only supports multiple devices through a cross-platform graphics library but is also compatible with existing computer graphics toolchains (e.g., collision detection from [17])[1].

3. Preliminaries

3.1. NeRF Rendering Pipeline

NeRF [23] offers photorealistic novel views by encoding a continuous volumetric field of points, which intercept and emit light rays, within the parameters of an MLP network. The rendering process with NeRF involves three steps. (1) To render each pixel in the target novel view, a ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ is cast from the origin (such as the camera's center) of the target novel view $\mathbf{o}$ along direction $\mathbf{d}$, which passes through the respective pixel. Here, t denotes the distance between sampled points along this ray and the origin $\mathbf{o}$. (2) For each point distanced t_k from the view origin $\mathbf{o}$, its location $\mathbf{o} + t_k\mathbf{d}$ and direction $\mathbf{d}$ serve as inputs to the MLP network $(\mathbf{o} + t_k\mathbf{d}, \mathbf{d}) \rightarrow (\sigma_k, \mathbf{c}_k)$, which then outputs the corresponding density σ_k and an RGB color $\mathbf{c}_k$. These represent the extracted features of that specific point. (3) Adhering to the principles of classical volume rendering [21], the color $\mathbf{C}(\mathbf{r})$ of the pixel corresponding to the ray $\mathbf{r}$ can be computed by integrating the features of the points along the ray. The following equation expresses this process:

$$\mathbf{C}(\mathbf{r}) = \sum_{k=1}^{N} T_k(1 - \exp(-\sigma_k(t_{k+1} - t_k)))\mathbf{c}_k,$$
$$\text{where } T_k = \exp(-\sum_{j=1}^{k} \sigma_j(t_{j+1} - t_j)), \quad (1)$$

where N denotes the number of sampled points along the ray $\mathbf{r}$ and T_k indicates the accumulated transmittance along the ray $\mathbf{r}$ to the point $\mathbf{o} + t_k\mathbf{d}$. This transmittance represents the likelihood of the ray reaching this point without encountering any other points.

To further accelerate NeRF's reconstruction process, Instant-NGP [24] replaces the MLP network of vanilla NeRF [23] with a 3D embedding grid stored as a compact 1D hash table. As a result, the computationally heavy MLP inferences in the standard NeRF, involving about 1 million FLOPs, are transformed into significantly less demanding embedding interpolation operations, requiring fewer than 0.00005 million FLOPs. Specifically, for each queried point along the rays passing through the pixels of training images, the embeddings of its eight nearest vertices in the 3D embedding grid are retrieved from the compact 1D hash table using their respective table index that is determined by their coordinates. The embeddings of the queried point are then obtained through trilinear interpolation of these eight embeddings. After retrieving the embeddings for the queried points along the rays passing through the pixels as described above, these embeddings are fed into a smaller MLP model to obtain the corresponding density and view-dependent color.

Unlike the vanilla NeRF which employs an MLP with 10 layers, each with 256 hidden units, this smaller MLP comprises only 2 layers with 64 hidden units each [24]. As discussed in Sec. 3.2 and recent real-time NeRF rendering studies [28, 38], Instant-NGP[24] retains the storage efficiency of vanilla NeRF due to the compact 1D hash table but can only achieve real-time rendering speeds on high-end GPUs, such as RTX 3090Ti [26]. The challenge remains to enable real-time rendering of large-scale real-world scenes using Instant-NGP while maintaining storage efficiency. As we analyze in Sec. 4.1, directly combining low-quality meshes with Instant-NGP retains storage efficiency, but fails to achieve real-time rendering speeds. Informed by the profiling in Sec. 4.2, our proposed MixRT modifies the model structure of Instant-NGP to better align with the WebGL framework, making it more accessible for most existing devices equipped with browsers.

3.2. Discussion on Existing NeRF Representations

As detailed in Sec.2.2, prior works have explored the adoption of alternative, more efficient NeRF representations in

[1]The real-time online collision demonstration of our proposed MixRT is available at https://licj15.github.io/MixRT/collision_viewer/

Table 1. Overview of Commonly-Used NeRF Representations

Representations	Rendering Quality	FPS	VRAM Efficiency	Storage Efficiency	Hardware
MLP Network [23, 32]	High	Low	High	High	SIMD Units
Triangle Mesh [8, 27, 38]	Medium	High	Low	Low	Rasterizer
Sparse Voxels [16, 39]	Medium	Medium	Low	Low	Texture/SIMD Units
Plane/Vector [7, 28]	Medium	Medium	Medium	Medium	Texture/SIMD Units
Hash Table [5, 24]	High	Low	High	High	SIMD Units

lieu of MLP networks for the purpose of real-time rendering. Yet, to date, no NeRF representation has been able to simultaneously meet the criteria of delivering high rendering quality (e.g., measured by PSNR), ensuring real-time frame rates, optimizing VRAM efficiency (which translates to minimized memory allocation during the rendering process), and maximizing storage efficiency (implying a compact model size that's conducive for efficient data transmission between users). This collective performance assessment is comprehensively summarized in Tab. 1.

Specifically, the **MLP network** used in the vanilla NeRF model [23] excels in photorealistic rendering quality. Furthermore, it is highly efficient in terms of storage and memory usage, requiring only about 5 MB of network weights for each scene in the NeRF-Synthetic dataset [23]. As such, it is a popular choice in subsequent research focusing on high-fidelity, large-scale NeRF rendering [4, 32]. However, it has a significant limitation: there are no well-optimized accelerators available in the current graphics hardware to run this type of network efficiently (i.e., only SIMD units such as CUDA cores can execute the model). This results in slower rendering speeds, which restricts its application in scenarios requiring real-time interactions.

Driven by the fact that most existing edge devices support **triangle mesh** effectively within their hardware rasterizer, studies such as [8, 27, 38] construct their rendering pipelines based on the mesh rasterization process. Utilizing triangle mesh as NeRF representations significantly improves FPS, enabling real-time rendering speeds even on mobile devices [8], while maintaining a respectable rendering quality (i.e., only 0.1 lower PSNR than that of vanilla NeRF on the NeRF-Synthetic dataset [8]). Nevertheless, the approach's scalability remains a concern for large-scale real-world scenes, as the requirements for storage and memory increase proportionally with the scale of the scene, leading to over 400 MB of disk usage on the Unbounded-360 dataset [38].

In pursuit of a better balance between the MLP network with costly volumetric ray casting and the triangle mesh with efficient rasterization, prior works like [16, 39] have proposed replacing the MLP network with **sparse voxels**, while still employing volumetric ray casting for the rendering process. Leveraging the compressed format of sparse voxels (for instance, densely packed 3D texture in [16]) and the same volumetric ray casting technique used by vanilla NeRF, these works achieve a respectable compromise between rendering quality and FPS. They utilize either the texture mapping units accessible via WebGL API or the SIMD units accessible through CUDA APIs. However, akin to the triangle mesh representations, scaling these methods to large-scale scenes can pose significant challenges in terms of memory and storage efficiency, as pointed out by [28].

In the effort to enhance the memory and storage efficiency of sparse voxels, studies such as [7, 28] propose using **plane/vector** as NeRF representations, which can be perceived as the low-rank decomposed format of 3D voxels. By employing a similar rendering pipeline and hardware as used by sparse voxels, yet with a more compact representation alongside a distinct decoding method (for instance, matrix-vector outer product in [7]) for embeddings of the sampled points, these studies manage to maintain similar or even superior rendering quality vs. FPS trade-offs when compared to those using sparse voxels. In addition, they significantly reduce the memory and storage requirements (e.g., only requiring 188 MB vs. 3785 MB of disk space as per [28]).

Following [24] to employ a **hash table** as a new NeRF representation, numerous subsequent studies have sought to enhance its balance between rendering quality and efficiency, given its state-of-the-art training speed. As validated by [24, 28], hash tables exhibit superior memory and storage efficiency compared to sparse voxels or plane/vector (for instance, only requiring $\sim$100 MB vs. $\sim$200 MB or $\sim$400 MB of disk space in the Unbounded-360 dataset [24, 28]). Consequently, they are often used as the NeRF representation during training, before being translated into other representations [24, 28]. Additionally,[5] also affirms that a hash table representation with a well-designed point sampling strategy in ray casting can yield superior rendering quality compared to MLP network representations. Nonetheless, the hash table proposed in Instant-NGP[24] is constrained by its limited compatibility with most devices, thus only achieving real-time rendering speeds on high-end GPUs with customized CUDA kernel for accessing SIMD units in graphic hardware.

Motivated by the aforementioned comparison, discus-

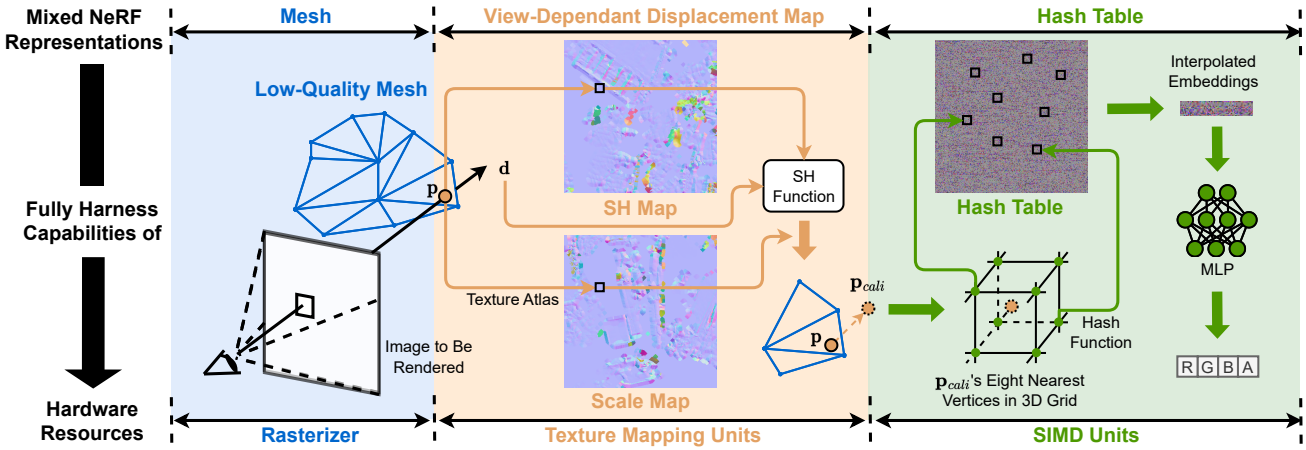

Figure 2. An overview of our proposed MixRT rendering pipeline: MixRT integrates three core components: a low-quality **mesh**, a **view-dependent displacement map**, and a NeRF model compressed into a **hash table**. This combination aims to maximize utilization of diverse hardware resources. To render an image pixel: (1) We use **rasterizer** hardware to perform mesh rasterization, determining the ray-mesh intersection point, $\mathbf{p}$. (2) Leveraging **texture mapping units**, we use texture coordinates to access maps containing the spherical harmonics (SH) coefficients and scale, computing the calibrated point, $\mathbf{p}_{cali}$. (3) Lastly, $\mathbf{p}_{cali}$ is processed by **SIMD units**, retrieving embeddings for its eight closest vertices from the 3D grid stored as a hash table. A small MLP network then converts these interpolated embeddings into the final rendered color.

sion, and analysis of previous works, we propose a new form of NeRF representation. Our proposed MixRT mixes a low-quality mesh, a view-dependent displacement map, and a compressed NeRF model in Instant-NGP's hash table [24] format. This configuration is purposefully designed to leverage the inherent strengths of rasterizers, texture mapping units, and SIMD units in current graphics hardware. The proposed method empowers real-time NeRF rendering on edge devices while maintaining better rendering quality (e.g., 0.2 PSNR higher on indoor scenes of the Unbounded-360 dataset), and staying within smaller memory and storage parameters (e.g., consuming only 80% of [28]'s disk usage), as compared to SotA methods.

4. Method

In this section, we first examine the relationship between mesh quality and rendering quality in Sec.4.1, finding that, with the help of color fields that are represented by hash tables, high-quality novel view synthesis doesn't necessarily demand meshes with extensive triangles. We then perform the runtime profiling analysis on hash tables in Sec.4.2, pinpointing bottlenecks to inform hash table configuration adjustments for improved FPS. Finally, we unveil MixRT, with the detailed design described in Sec. 4.3, comprising: (1) a low-quality mesh, (2) a view-dependent displacement map, and (3) an compressed NeRF model stored in a hash table. This design ensures MixRT's rendering pipeline, illustrated in Fig. 2, can be specifically optimized to fully harness the capabilities of rasterizers, texture mapping units, and SIMD units in current graphics hardware, enabling real-time NeRF rendering on edge devices without rendering quality sacrifice.

4.1. Observations on the Effect of Mesh Quality

Previous real-time NeRF rendering research that uses triangle mesh as the NeRF representation highlights the importance of mesh geometry quality for photorealistic rendering [34, 38]. For instance, [34] delves into refining the surface by adjusting vertex positions and face density. Yet, we note that an ultra-detailed mesh, packed with a vast number of triangles, isn't mandatory for photorealistic rendering outcomes.

In particular, we made the above observations by (1) simplifying the SotA high-quality mesh from [38] via the classical vertex clustering [41] and (2) querying the color of the ray-mesh intersection points from a color field represented by Instant-NGP [24]'s hash table. As summarized in Tab. 2, the low-quality mesh contains more than 5 $\times$ fewer triangles and faces as compared to the high-quality one but can achieve $\sim$ 0.3 higher PSNR than the high-quality mesh by equipping a hash table as a color filed to query the color from. Thanks to the high memory and storage efficiency of Instant-NGP [24]'s hash table, the combination described above only consumes 0.26 $\times$ storage size of the high-quality mesh. However, as suggested in prior works [28, 38] and the discussion in Sec. 3.2, the achieved FPS by the combination of simplified mesh and hash table can not satisfy the real-time rendering requirements. The set of conducted experiments implies that (1) the high-quality geometry information represented by mesh with massive triangles is not

necessary for achieving high rendering quality and (2) replacing high-quality mesh with the combination of the simplified low-quality mesh and a hash table as the NeRF representation can achieve better PSNR vs. storage efficiency trade-offs.

From the above observations, it is evident that the primary limitation of merging a low-quality mesh with other representations is the rendering speed. Therefore, we conduct an in-depth runtime profiling analysis on the hash table representation which is known for its memory and storage efficiency, as detailed in Sec. 4.2.

4.2. Runtime Profiling Analysis

Since there is no existing runtime breakdown analysis tool for WebGL, we perform the runtime profiling analysis on hash tables by tuning the model structure and observing the resulting FPS. Specifically, as summarized in Tab. 3, the effect of varying (1) hash table size, (2) number of levels, and (3) MLP architectures on FPS implies that the number of levels is the runtime bottleneck, i.e., tuning it can significantly change the resulting FPS, while the other two factors are not.

Motivated by the profiling analysis above, we propose to modify the default model structure of Instant-NGP's hash table by shrinking the number of levels while enlarging the hash table size. As suggested in Tab. 4, such modifications can boost the rendering speed to > 30 FPS while maintaining hash tables' memory or storage efficiency.

4.3. Mixing Mesh, Texture Map, and NeRF

With the observation that high-quality mesh is not necessary for achieving high rendering quality (see Sec. 4.1) and the profiling-inspired hash table configuration can achieve both high rendering speed and high memory or storage efficiency (see Sec. 4.2), we propose a type of NeRF representation that comprises (1) a low-quality mesh, (2) a view-dependant displacement map, and (3) a compressed NeRF model in Instant-NGP [24]'s hash table format. Such a design can not only leverage the commonly agreed high rendering quality, high renderings speed, and high memory and storage efficiency of existing NeRF representations as discussed above but also fully leverage the rasterizers, texture mapping units, and SIMD units in graphics hardware. We summarize the rendering pipeline of our proposed MixRT in Fig. 2 and detail the design of each part in our proposed MixRT as follows.

4.3.1 Triangle Mesh

We leverage the standard triangle mesh format to include the information on geometric vertices coordinates, texture coordinates, and polygonal face elements. Unlike [38], we do not need to store the per-vertex appearance parameters because the color of the intersection points will be fetched from the hash table. Following [38], the mesh is post-processed by vertex order optimization [29] to allow higher cache hit rates for accessing neighboring triangles.

4.3.2 View-Dependant Displacement Map

Inspired by the commonly-used normal map [9, 10] that fakes the lighting of bumps and dents without using more polygons, we propose a view-dependant displacement map to calibrate the coordinate of the intersection points to be inputted in the color field represented by Instant-NGP's hash table. Similar to a normal map, our proposed view-dependant displacement map can fake more accurate coordinates of the intersection points without adding new polygons to the mesh. However, our proposed one can better fit Instant-NGP [24]'s hash table and the corresponding rendering pipeline that only takes the coordinates and view directions as input instead of surface normal. While previous studies [3, 12] use neural networks to predict displacement vectors for calibrating points in volumetric rendering, this approach is unfeasible for real-time on-device rendering. In contrast, our method employs 2D maps for displacement prediction, leveraging the texture mapping units of graphics hardware.

In particular, the proposed view-dependant displacement map consists of (1) a spherical harmonics (SH) map, m_{SH}, to store the SH coefficients for guiding the encoded view directions to output a view-dependant vector and (2) a scale map m_s to scale the outputted view-dependant vector to a proper length and thus the scaled vector can be used as the calibration variable of the coordinate of the ray-mesh intersection points. The shape of the SH map m_{SH} and scale map m_s is designed to be $[R_m, R_m, 3 \times (D_{SH}+1)^2]$ and $[R_m, R_m, 1]$, respectively. R_m represents the resolution of the map and D_{SH} is the SH degree used in SH map m_{SH}.

Table 2. Comparison between (1) the **high-quality triangle mesh** from [38] and (2) the combination of the **low-quality mesh** (simplified from the high-quality mesh) and Instant-NGP's [24] **hash table**, in terms of the average PSNR vs. storage size or FPS trade-offs on the indoor scenes of Unbounded-360 dataset. The FPS was measured on a Macbook M1 Pro laptop with a resolution of 1280 × 720.

Representations	**↓ # of Vertices on**				**↓ # of Faces on**				**↑ Avg. PSNR**	**↓ Storage**	**↑FPS**
	Room	Counter	Kitchen	Bonsai	Room	Counter	Kitchen	Bonsai			
Mesh from [38]	7,060,849	11,950,574	13,539,203	13,343,679	14,110,659	23,892,064	27,056,127	26,679,898	27.06	542 MB	120
Simplified Mesh + Hash Table	946,962	1,572,959	1,778,283	1,750,341	1,893,695	3,147,635	3,557,514	3,501,683	**27.41**	**139 MB**	0.4

Table 3. Adjusting the model structure of Instant-NGP's hash table [24]. FPS was measured on a Macbook M1 Pro laptop at a resolution of 1280 × 720, and the fragment shader was set to query the hash table for color once per pixel. "Hash table size" and "# of levels" denote the maximum entries per level and the number of multi-resolution levels in Instant-NGP's hash table, respectively. The "MLP architecture" outlines the structure of the MLP responsible for transforming the embedding retrieved from the hash table into RGB color.

# of Levels	Hash Table Size	MLP Architecture	FPS
8	2^{17}	2 Layers, 8 Hidden Neurons	27
8	2^{17}	**Removed**	30
👉 Observation: **MLP** is not the runtime bottleneck			
8	2^{17}	Removed	30
1	2^{17}	Removed	120
👉 Observation: **# of levels** is the runtime bottleneck			
8	2^{17}	Removed	30
8	$\mathbf{2^5}$	Removed	35
8	$\mathbf{2^{22}}$	Removed	25
👉 Observation: **Hash table size** is not the bottleneck			

Given the coordinate $\mathbf{p} \in \mathbb{R}^3$ of an intersection point, its texture coordinates $\mathbf{p}_t \in \mathbb{R}^2$, and the corresponding view direction $\mathbf{d} \in \mathbb{R}^3$, the calibrated coordinate $\mathbf{p}_{cali} \in \mathbb{R}^3$ can be computed as:

$$\mathbf{p}_{cali} = \mathbf{p} + S(m_{SH}(\mathbf{p}_t), \mathbf{d}) \times m_s(\mathbf{p}_t), \tag{2}$$

where S denotes the SH functions as used in [7]. $m_{SH}(\mathbf{p}_t) \in \mathbb{R}^{(D_{SH}+1)^2}$ and $m_s(\mathbf{p}_t) \in \mathbb{R}$ represent the feature interpolated from m_{SH} and m_s with coordinate $\mathbf{p}_t$, respectively. As such, the calibrated coordinate $\mathbf{p}_{cali}$ can be determined by the coordinate and view directions of the ray-mesh intersection point. The view-dependant displacement map is quantized into 8 bits after training for both higher rendering speeds and memory or storage efficiency.

4.3.3 Hash Table

For the hash table in the proposed MixRT, similar to the settings in Instant-NGP [24], it consists of (1) multiple levels of 1D hash tables with different corresponding 3D resolutions and (2) small MLP networks to convert the fetched embeddings from the hash table to density or color. However, as illustrated in Sec. 4.2 we modify its model structure, i.e., shrinking the number of levels and enlarging the hash table size, to improve its rendering speed. To be compatible with the 2D texture mapping units that can be accessed by WebGL, we reshape the hash tables stored in 1D format to 2D image format for exporting it to the WebGL rendering framework.

Table 4. Optimizing Instant-NGP's hash table configurations based on runtime profiling insights. FPS measurements were taken on a Macbook M1 Pro laptop at a resolution of 1280 × 720, while PSNR evaluations were conducted on the indoor scenes of the Unbounded-360 dataset.

Mesh	# of Levels	Hash Table Size	↑Avg. PSNR	↓Storage	↑ FPS
[38]	-	-	27.06	542 MB	120
Simplified	16	2^{20}	27.41	139 MB	0.4
Simplified	4	2^{21}	26.63	**74 MB**	**35**

5. Experiments

5.1. Experiments Settings

5.1.1 Baselines, Datasets, and Metrics

We benchmark the proposed MixRT on challenging large-scale indoor scenes of the Unbounded-360 dataset [4] and compare with the following three SotA real-time NeRF rendering works: (1) BakedSDF [38]: it leverages high-quality mesh with massive triangles, and gets the rendering results by mesh rasterization with appearance parameters stored in each vertex, (2) NeRFMeshing [27]: it also leverages high-quality mesh that is distilled from pre-trained NeRF model but store the appearance parameters as texture maps, and (3) MeRF [28], which adopts tri-planes as the representations to store density and color information of the scene and gets the rendering results by volumetric ray casting like vanilla NeRF [23]. The rendering quality is measured by PSNR, and the FPS is measured on a Macbook M1 Pro laptop at the resolution of 1280 × 720, following the settings used in [28]. The memory or storage efficiency is measured by the total file sizes of meshes in glTF format, textures in PNG format, and scene configurations in JSON format.

5.1.2 Implementation Details

We implement our real-time rendering pipeline with WebGL framework. Specifically, in the GLSL vertex shader, we compute the coordinates of the ray-mesh intersection points and their corresponding texture coordinates. In the GLSL fragment shader, we first calibrate the coordinate of the intersection points with the texture coordinates and view directions, following Eq. 2. Then, following the standard pipeline of Instant-NGP [24], we loop over the hash tables' number of levels to get the trilinear interpolated embeddings from each level. The interpolated embeddings from different levels are concatenated together to be the input of a small MLP model that is implemented as matrix-vector multiplication, following the implementation in [8]. In all our experiments, the hash table is configured to have four levels with a minimum level resolution of 256 and a maximum of 4096, and the hash map size is set as 2^{21}, with each entry holding a four-dimensional vector. For the view-dependant displacement map, the map resolution R_m is set as 1536 for all scenes. For the low-quality meshes, they

Table 5. Comparison between our MixRT and SotA real-time NeRF rendering techniques on the four indoor scenes from the Unbounded-360 Dataset [4]. PSNR values were measured using Mip-NeRF-360's settings [4], while FPS was measured on a Macbook M1 Pro at a resolution of 1280 × 720, consistent with the experiment settings in [28].

Method	↑ PSNR on					↑ FPS	↓ Storage
	Room	Counter	Kitchen	Bonsai	Avg.		
NeRFMeshing [27]	26.13	20.00	23.59	25.58	23.83	-	-
BakedSDF [38]	-	-	-	-	27.06	120	542 MB
MeRF [28]	-	-	-	-	27.80	30	124 MB
MixRT (Ours)	29.88	26.60	27.46	28.10	**28.01**	31	**98 MB**

are simplified from the SotA BakedSDF [38]'s mesh by the classical vertex cluster [41] algorithm in the space contracted by Mip-NeRF-360 [4]'s contraction function with the voxel size hyperparameter set as 0.01 for all scenes. The simplified meshes, along with randomly initialized view-dependent displacement maps and hash tables, are jointly trained using the loss function based on the differences between the rendered images and the ground truth images.

5.2. Comparing with SotA

We first compare our proposed MixRT with SotA real-time NeRF rendering works. As summarized in Tab. 5, the proposed MixRT achieves the highest PSNR and storage efficiency among all the methods in the benchmark, while maintaining the real-time (> 30 FPS) rendering speed. Specifically, as compared to MeRF [28], our proposed MixRT achieves 0.2 higher PSNR than it with only 80% storage cost under the same rendering speed. Please refer to Appendix B for the corresponding qualitative comparison.

5.3. Ablation Study

As highlighted in Sec. 4.1, incorporating hash tables into our MixRT framework is critical for maintaining high memory and storage efficiency without sacrificing rendering quality. Following this, we further conduct an ablation study to verify the significance of the view-dependent displacement map, another integral component of MixRT. As shown in Tab. 6, removing the view-dependent displacement map from our proposed MixRT reduces storage by approximately 24% but results in a 1.37 decrease in PSNR. Meanwhile, the rendering speed remains relatively stable,

Table 6. Comparison MixRT w/ and w/o the proposed view-dependent displacement (VDD) map, in terms of PSNR, FPS, and storage size on the indoor scenes of the Unbounded-360 dataset [4].

Method	↑ PSNR on					↑ FPS	↓ Storage
	Room	Counter	Kitchen	Bonsai	Avg.		
MixRT **w/** VDD Map	29.88	26.60	27.46	28.10	**28.01**	31	98 MB
MixRT **w/o** VDD Map	29.10	25.26	25.64	26.54	26.64	35	**74 MB**

Table 7. Comparison between our MixRT and SotA real-time NeRF rendering techniques on the three publicly available outdoor scenes from the Unbounded-360 Dataset [4].

Method	↑ PSNR on			
	Bicycle	Garden	Stump	Avg.
Volumetric-Rendering-Based Methods				
MeRF [28]	22.82	25.32	25.06	24.40
Rasterization-Based Methods				
MobileNeRF [8]	21.70	23.53	23.95	21.06
NeRFMeshing [27]	21.15	22.91	22.66	22.24
MixRT (Ours)	21.81	24.55	23.76	23.37

shifting from 31 FPS to 35 FPS. Considering that MixRT already achieved higher storage efficiency than all baselines, as demonstrated in Sec. 5.2, integrating view-dependent displacement maps in our MixRT is a better option to achieve higher PSNR vs. rendering speed trade-offs.

6. Limitation

As illustrated in Tab. 5, our proposed MixRT demonstrates better rendering quality vs. rendering speeds and storage efficiency than SotA methods in indoor scenes. However, its rendering quality is still constrained by rasterization-based rendering methods, a common limitation in rasterization-based real-time NeRF methods [8, 38]. In particular, for the more complex outdoor scenes from the Unbounded-360 dataset [4], as shown in Tab. 7, MixRT's rendering quality is 1 PSNR lower than the volumetric-rendering-based MeRF [28]. Despite this, it still achieves comparable or better quality than other rasterization-based baselines [8, 27].

7. Conclusion

We present MixRT, a NeRF representation that combines a low-quality mesh, a view-dependent displacement map, and a compressed NeRF in a hash table structure. This design emerges from our observation that achieving high rendering quality does not require high-complexity geometry represented by meshes with a vast number of triangles. This realization suggests the potential to streamline the baked mesh and incorporate diverse neural representations for rendering, memory, and storage efficiency. Through detailed runtime profiling analysis and an optimized WebGLbased rendering framework, MixRT offers state-of-the-art balance between rendering quality and efficiency.

Acknowledgement

Chaojian Li and Yingyan (Celine) Lin would like to acknowledge the funding support from the National Science Foundation (NSF) Computing and Communication Foundations (CCF) programs (Award ID: 2211815 and 2312758).

References

[1] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-time rendering*. AK Peters/crc Press, 2019. 2

[2] Inc. Apple. Core ml tools, 2023. https://github.com/apple/coremltools, accessed 2023-08-01. 3

[3] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16610–16620, 2023. 6

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2, 3, 4, 7, 8

[5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *arXiv preprint arXiv:2304.06706*, 2023. 2, 4

[6] Junli Cao, Huan Wang, Pavlo Chemerys, Vladislav Shakhrai, Ju Hu, Yun Fu, Denys Makoviichuk, Sergey Tulyakov, and Jian Ren. Real-time neural light field on mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8328–8337, 2023. 2

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. 4, 7

[8] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 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*, pages 16569–16578, 2023. 1, 2, 4, 7, 8

[9] Paolo Cignoni, Claudio Montani, Claudio Rocchini, and Roberto Scopigno. A general method for preserving attribute values on simplified meshes. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 59–66. IEEE, 1998. 6

[10] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 115–122, 1998. 6

[11] Blender Online Community. Blender - a 3d modelling and rendering package, 2018. 3

[12] Jiemin Fang, Lingxi Xie, Xinggang Wang, Xiaopeng Zhang, Wenyu Liu, and Qi Tian. Neusample: Neural sample field for efficient view synthesis. *arXiv preprint arXiv:2111.15552*, 2021. 6

[13] Jiading Fang, Shengjie Lin, Igor Vasiljevic, Vitor Guizilini, Rares Ambrus, Adrien Gaidon, Gregory Shakhnarovich, and Matthew R Walter. Nerfuser: Large-scale scene representation by nerf fusion. *arXiv preprint arXiv:2305.13307*, 2023. 2

[14] Yasutaka Furukawa, Carlos Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015. 2

[15] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996. 2

[16] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021. 1, 2, 4

[17] Stefan Hedman. cannon.js: Lightweight 3d physics for the web, 2023. https://github.com/schteppe/cannon.js/tree/master, accessed 2023-08-01. 3

[18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023. 3

[19] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996. 2

[20] Zhong Li, Liangchen Song, Celong Liu, Junsong Yuan, and Yi Xu. Neulf: Efficient novel view synthesis with neural 4d light field. *arXiv preprint arXiv:2105.07112*, 2021. 2

[21] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 3

[22] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 2

[23] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 2, 3, 4, 7

[24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 2, 3, 4, 5, 6, 7

[25] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8280–8290, 2022. 2

[26] NVIDIA LLC. GEFORCE RTX 3090 FAMILY, 2021. https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3090-3090ti/, accessed 2022-06-01. 3

[27] Marie-Julie Rakotosaona, Fabian Manhardt, Diego Martin Arroyo, Michael Niemeyer, Abhijit Kundu, and Federico Tombari. Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes. *arXiv preprint arXiv:2303.09431*, 2023. 2, 4, 7, 8

[28] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-

time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 1, 2, 3, 4, 5, 7, 8

[29] Pedro V Sander, Diego Nehab, and Joshua Barczak. Fast triangle reordering for vertex locality and reduced overdraw. In *ACM SIGGRAPH 2007 papers*, pages 89–es. 2007. 6

[30] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[31] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[32] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. 2, 4

[33] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023. 2

[34] Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. Delicate textured mesh recovery from nerf via adaptive surface refinement. *arXiv preprint arXiv:2303.02091*, 2023. 5

[35] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12922–12931, 2022. 2

[36] Xiuchao Wu, Jiamin Xu, Zihan Zhu, Hujun Bao, Qixing Huang, James Tompkin, and Weiwei Xu. Scalable neural indoor scene rendering. *ACM transactions on graphics*, 41 (4), 2022. 2

[37] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *European conference on computer vision*, pages 106–122. Springer, 2022. 2

[38] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv preprint arXiv:2302.14859*, 2023. 1, 2, 3, 4, 5, 6, 7, 8

[39] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2, 4

[40] Mi Zhenxing and Dan Xu. Switch-nerf: Learning scene decomposition with mixture of experts for large-scale neural radiance fields. In *The Eleventh International Conference on Learning Representations*, 2022. 2

[41] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Vertex clustering in Open3D: A modern library for 3D data processing, 2018. http://www.open3d.org/docs/release/tutorial/geometry/mesh.html#Vertex-clustering. 5, 8

MixRT: Mixed Neural Representations For Real-Time NeRF Rendering

Supplementary Material

A. Real-Time Interactive Demonstration

To experience the real-time interactive demonstration of the proposed MixRT, please visit https://licj15.github.io/MixRT/index.html#demos. Our demo offers real-time online interaction with static scenes, as well as collision animations.

B. Visual Comparison with SotA

In addition to the quantitative comparison of our proposed MixRT and the SotA real-time NeRF rendering shown in Tab 5, we offer additional rendered image comparisons in Fig. 3 below. Consistent with the observations in Sec. 5.2, our proposed MixRT excels in two main areas: (1) accurately rendering regions with specular highlights or fine-grained geometry structures, e.g., the bowl in "Scene: Counter" and the bulldozer bucket in "Scene: Lego", and (2) eliminating ghostly effects such as the "floaters" observed on the floor of "Scene: Room" and the wall of "Scene: Bonsai".

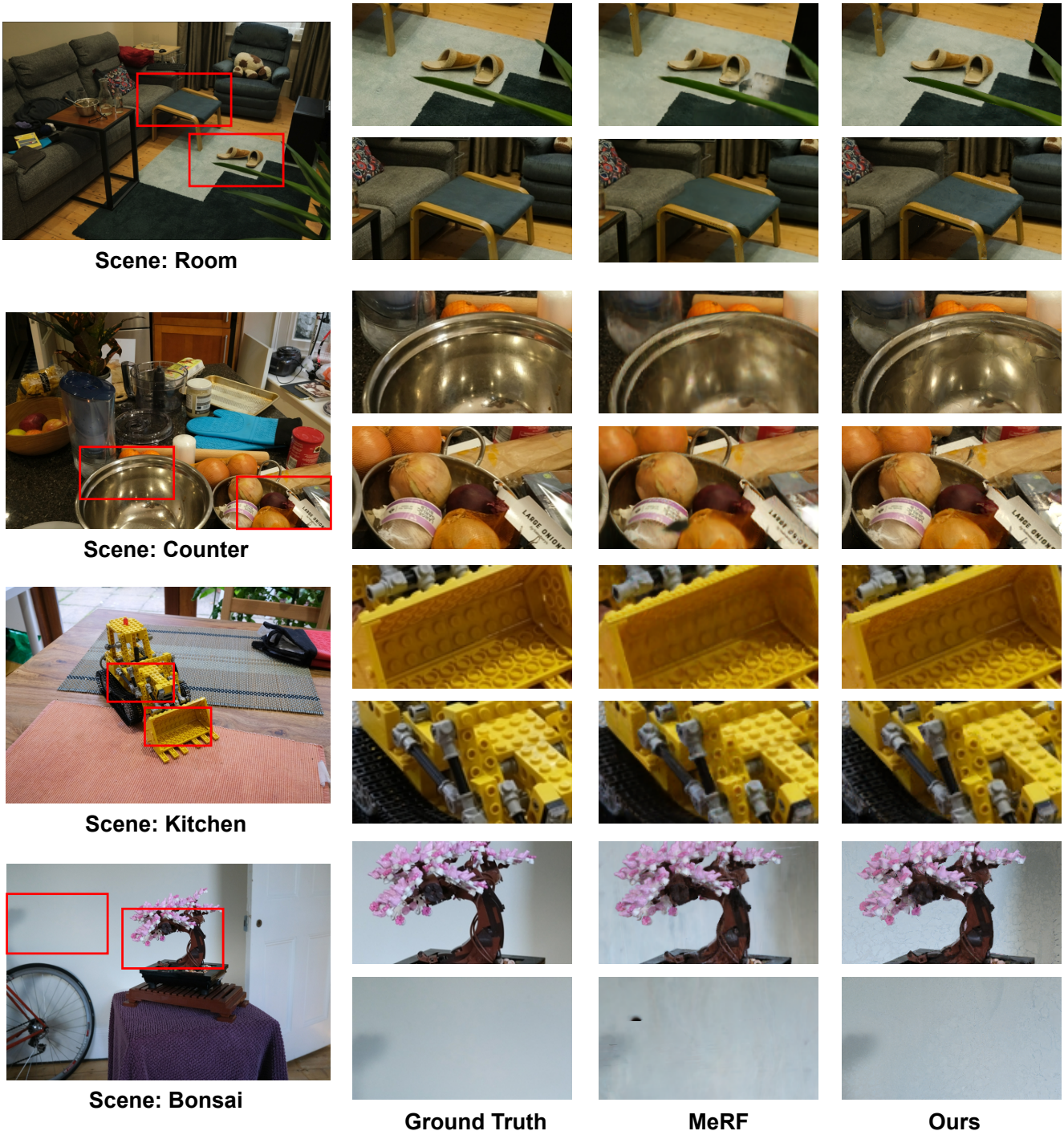

Figure 3. Visual comparison between our proposed MixRT and MeRF [28], a real-time NeRF rendering work with SotA rendering quality vs. efficiency trade-offs. The rendered images are randomly selected from the test set.