City-on-Web: Real-time Neural Rendering of Large-scale Scenes on the Web

Kaiwen Song[1], Xiaoyi Zeng[1], Chenqu Ren[2], and Juyong Zhang[1]*

[1] University of Science and Technology of China
[2] East China Normal University

Abstract. Existing neural radiance field-based methods can achieve real-time rendering of small scenes on the web platform. However, extending these methods to large-scale scenes still poses significant challenges due to limited resources in computation, memory, and bandwidth. In this paper, we propose City-on-Web, the first method for real-time rendering of large-scale scenes on the web. We propose a block-based volume rendering method to accommodate the independent resource characteristics of web-based rendering, and introduce a Level-of-Detail strategy combined with dynamic loading/unloading of resources to significantly reduce memory demands. Our system achieves real-time rendering of large-scale scenes at 32FPS with RTX 3060 GPU on the web and maintains quality comparable to the current state-of-the-art novel view synthesis methods. Project page: https://ustc3dv.github.io/City-on-Web/

Keywords: Real-time rendering · Neural rendering · Large-scale scene

1 Introduction

Neural radiance field (NeRF) has advanced the field of reconstruction, showing an unparalleled ability to capture complex details across diverse environments. Existing works demonstrate its ability to render small scenes with exceptional quality and performance in real-time [2, 10, 16, 23, 26, 29, 35, 41, 45]. NeRF has also been successfully applied to the various scenarios, including human reconstruction, object-centric scenes, and large-scale scenes in offline settings, achieving exceptional visual fidelity and generating intricately detailed results [9, 12, 13, 19, 28, 34, 37, 40, 41, 43].

Despite these successes, real-time neural rendering of large scenes on the web remains profoundly challenging due to inherent computational power, memory, and bandwidth limitations on commodity devices. MERF [30] has recently achieved significant progress by employing a baking technique to reduce query network calls in the rendering pipeline, thereby enabling real-time rendering of small-scale scenes on the web. However, MERF struggles to capture intricate details in large scenes due to its limited resolution. A naive solution would be to simply increase the volumetric representation's resolution, but this approach

* Corresponding Author

would lead to unacceptable increases in memory usage, scaling with $O(N^3)$, and a significant decrease in rendering speed.

To overcome these limitations, we integrate MERF with a block-based strategy [37] for reconstructing large scenes, a method supported by numerous studies [34, 37, 49]. This approach not only improves the model's representational ability but also controls memory growth at an $O(N^2)$ rate. However, there are certain challenges associated with a resource-independent block-based rendering approach on the web. Specifically, rendering on the web faces limitations on the number and resolution of texture units that can be loaded into a shader, which prevents loading all block resources into a single shader. Consequently, we load the rendering resources of different blocks into their respective shaders. Nevertheless, rendering with different shaders causes issues with 3D consistency. Specifically, when a ray traverses multiple blocks, sampling points might belong to different blocks loaded by different shaders, preventing standard volume rendering. We are thus compelled to render each block sequentially and subsequently combine the rendering results of the different blocks. To this end, we propose a block-based volume rendering strategy and demonstrate that this method of sequential block rendering is equivalent to volume rendering, thereby ensuring correct occlusion and 3D consistency of the rendering results. Notably, unlike existing block-based methods like Block-NeRF [34] and Mega-NeRF [37] that require the resources of all blocks to be loaded simultaneously for rendering, our strategy make each block can be rendered independently with its own texture in its own shader.

Moreover, when viewing from a higher altitude viewpoint, the rendering resources of all scene blocks are needed. Nonetheless, loading all blocks for rendering is impractical due to the excessive memory usage that would surpass the capacity of standard consumer devices. To address this issue, we draw inspiration from traditional graphics techniques [6–8, 14, 17, 22, 25] to create Level-of-Detail (LOD) for each block's resources, dynamically selecting resources for rendering based on the camera's position and field of view. This approach significantly reduces the resource demands during rendering, paving the way for smoother user experiences even on less capable devices.

In summary, the contributions of this paper include the following aspects:

- We propose a block-based multi-shader volume rendering method, which can be proven to be equivalent to standard volume rendering. This stategy is designed to accommodate resource-independent environments and enabling high-fidelity rendering of large-scale scenes.
- We employ an LOD strategy and dynamic loading/unloading strategies to adaptively manage rendering resources and significantly reducing the quantity of resources loaded and ensuring efficient resource utilization for large-scale scene rendering.
- Our experiments demonstrate that our system achieves real-time rendering of large-scale scenes at approximately 32FPS with 1080P resolution on an RTX 3060 GPU, while maintaining rendering quality comparable to the current state-of-the-art (SOTA) methods for large-scale scenes.

2 Related Work

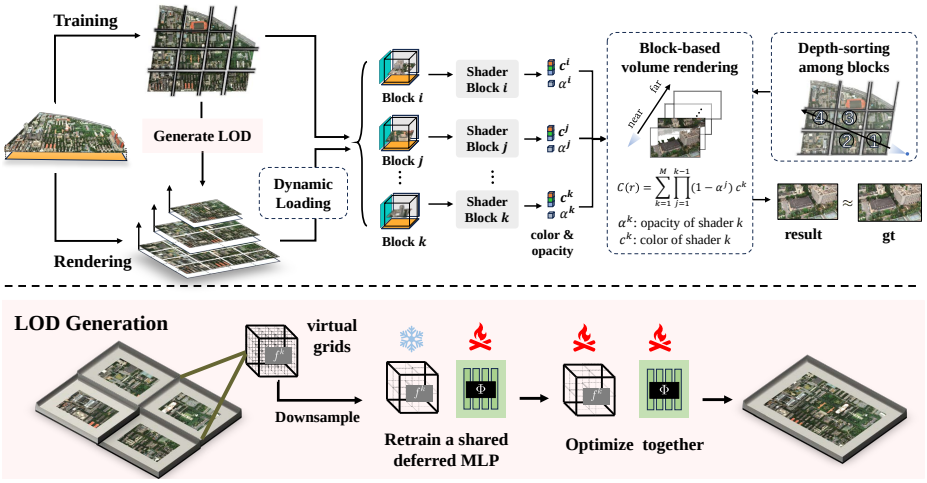


Fig. 1: Overview of City-on-Web pipeline. During the training phase, we uniformly partition the scene and reconstruct it at the finest LOD. To ensure 3D consistency, we use a resource-independent block-based volume rendering strategy (Sec. 4.2). For LOD generation, we downsample virtual grid points and retrain a coarser model (Sec. 4.4). This approach supports subsequent real-time rendering by facilitating the dynamic loading of rendering resources.

Large-scale Scene Reconstruction. For large-scale scene reconstruction, a key issue is enhancing the model's ability to adequately capture and render extensive scenes. Some works [11, 34, 37] address this by adopting a divide-and-conquer strategy, segmenting expansive scenes into smaller blocks, and applying localized NeRF processing to each. This approach significantly improves both the reconstruction quality and the model's scalability to larger scenes. Switch-NeRF [49] employs a gating network to dispatch 3D points to different NeRF sub-networks. Grid-NeRF [43] utilizes a compact multiresolution feature plane and combines the strengths of smoothness from vanilla NeRF with the local detail capturing ability of feature grid-based methods [4, 27, 31], efficiently reconstructing large scenes with fine details. NeRF++ [47] enhances the reconstruction of unbounded scenes through its innovative multi-spherical representation. On the other hand, Mip-NeRF 360 [1] introduces a scene contraction function to effectively represent scenes that extend to infinity, addressing the challenge of vast spatial extents. F2-NeRF [38] takes this a step further by implementing a warping function for local spaces, ensuring a balance of computational resources and training data across different parts of the scene.

Real-time Rendering. Early works mainly focus on the real-time rendering of a simple single object. NSVF [23] improves NeRF by introducing a more effi-

cient sparse voxel field, significantly accelerating rendering speed while maintaining high-quality output. KiloNeRF [29] utilizes thousands of small MLPs, each responsible for a tiny region, significantly reducing network evaluation time. In contrast, MERF [30] improves upon SNeRG [15] by utilizing a hybrid representation to reduce memory usage. Oblique-MERF [46] enhances real-time rendering performance over MERF by using an occupancy plane to skip empty space. MobileNeRF [5] introduces the polygon rasterization rendering pipeline, running NeRF-based novel view synthesis in real-time on mobile devices. BakedSDF [44] bakes volumetric representation into meshes and utilizes spherical harmonics for representing view-dependent color, while NeRF2Mesh [35] iteratively refine both the geometry and appearance of the mesh. Furthermore, several methods [36, 39] exploit the real-time rendering attributes of mesh representations alongside the robust representational potential of volume representations, particularly for rendering hair, translucent materials, and similar entities. These hybrid methods facilitate the achievement of high-fidelity real-time rendering. Recently, 3D Gaussian splatting [20] achieves real-time rendering by utilizing a novel 3D Gaussian representation and rasterization-based rendering pipeline. However, extending this representation to large scenes is challenging due to its substantial memory consumption.

Level of Detail. Substantial works are devoted to integrating LOD methods into the fabric of traditional computer graphics [6–8,14,17,22,24,25], aiming to streamline rendering processes, reduce memory footprint, bolster interactive responsiveness. Recently, some works have begun to apply LOD to neural implicit reconstruction. NGLoD [33] represents LOD through a sparse voxel octree, where each level of the octree corresponds to a different LOD, allowing for a finer discretization of the surface and more detailed reconstruction as the tree depth increases. Takikawa *et al*. [32] efficiently encode 3D signals into a compact, hierarchical representation using vector-quantized auto decoder method. BungeeNeRF [41] employs a hierarchical network structure, where the base network focuses on learning a coarse representation of the scene, and subsequent residual blocks are tasked with progressively refining this representation. TrimipRF [18] and LOD-Neus [50] leverage multi-scale triplane and voxel representations to capture scene details at different scales, effectively implementing anti-aliasing to enhance the rendering and reconstruction quality.

3 Background

Our exploration begins with an in-depth analysis of two influential works, SNeRG [15] and MERF [30]. SNeRG precomputes and stores a NeRF model in a sparse 3D voxel grid. Additionally, an indirection grid is used to enhance rendering by either indicating empty macroblocks or pointing to detailed texels in a 3D texture atlas. This representation allows real-time rendering on standard laptop GPUs.

The indirection grid assists in raymarching through the sparse 3D grid and selectively accessing non-zero densities σ_i, diffuse colors $\boldsymbol{c}_i$, and feature vectors

$\boldsymbol{f}_i$ from baked textures. Integrating along each ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$, they compute the sum of the weights, which can be considered as the pixel's opacity:

$$\alpha(\boldsymbol{r}) = \sum_i w_i,\ w_i = \prod_{j=1}^{i-1}(1-\alpha_j)\alpha_i, \alpha_i = 1 - e^{-\sigma_i \delta_i}. \tag{1}$$

The step size δ_i during ray marching is the voxel width for an occupied voxel. The color $\boldsymbol{C}_d(\boldsymbol{r})$ and specular feature $\boldsymbol{F}_s(\boldsymbol{r})$ along the ray are accumulated using the same weights to compute the final diffuse color and specular feature of ray:

$$\boldsymbol{C}_d(\boldsymbol{r}) = \sum_i w_i \boldsymbol{c}_i, \quad \boldsymbol{F}_s(\boldsymbol{r}) = \sum_i w_i \boldsymbol{f}_i. \tag{2}$$

Subsequently, diffuse color and specular feature, along with the positional encoding $PE(\cdot)$ of the ray's view direction, are concatenated to pass through a lightweight deferred MLP Φ to produce a view-dependent residual color:

$$\boldsymbol{C}(\boldsymbol{r}) = \boldsymbol{C}_d + \Phi(\boldsymbol{C}_d, \boldsymbol{F}_s, PE(\boldsymbol{d})). \tag{3}$$

While SNeRG achieves impressive real-time rendering results, its voxel representation demands substantial memory, which poses limitations for further applications. MERF presents a significant reduction in memory requirements. By leveraging hybrid low-resolution sparse voxel and 2D high-resolution triplanes, MERF optimizes the balance between performance and memory efficiency. Moreover, it incorporates two pivotal strategies to bridge the gap between training and rendering performance. MERF simulates finite grid approach during training, querying MLPs at virtual grid corners and simulates quantization during training to mimic the rendering pipeline closely.

4 Method

In this section, we present a method for rendering large scenes on the web. Our approach utilizes a block and LOD strategy for rendering large-scale scenes, as described in (Sec. 4.1). A block-based volume rendering approach is introduced for the seamless blending of blocks, utilized both during the training and rendering stages to ensure consistency (Sec. 4.2). Sec. 4.3 details the optimization strategies. The generation and refinement of LODs(Sec. 4.4) are also explained. Sec. 4.5 elucidates the baking strategy suitable for the representation.

4.1 Large-scale Radiance Field

Although real-time rendering methods like MERF can achieve high-quality real-time rendering for small-scale scenes, they face representational capacity challenges when applied to larger scenes. As mentioned in Sec. 1, utilizing a single MERF model to represent vast scenes is problematic due to its limited resolution, especially in terms of detailed and accurate reconstruction. Therefore,

we represent scenes using multiple blocks. However, this approach necessitates employing an LOD strategy to reduce the number of resources that need to be loaded during the rendering phase. Thus, we adopt a block-based and LOD strategy for representing the whole scene in the rendering stage. We will elaborate on the representation used in the rendering stage and provide the representation used to reconstruct the scene in the finest LOD in the training stage.

Training Stage. In the training stage, we only represent and optimize the finest LOD. We uniformly partition the entire scene into K blocks $\{\mathcal{B}_k\}_{k=1}^{K}$, each centered at $\mathbf{c}_k = (x_k, y_k)$, on the xy plane (i.e., the ground plane). This approach stems from the observation that large scenes typically exhibit smaller scales in the z-direction compared to the xy-plane, prompting us to partition based on the ground plane and avoid subdividing along the z-axis. For a point $\mathbf{p} = (p_x, p_y, p_z) \in \mathbb{R}^3$, we determine its corresponding block $\mathcal{B}_k$ based on its xy coordinates, denoted as $\mathbf{p}_{proj} = (p_x, p_y)$:

$$\mathbf{p} \in \mathcal{B}_k, k = \arg\min_k \|\mathbf{p}_{proj} - \mathbf{c}_k\|_\infty \tag{4}$$

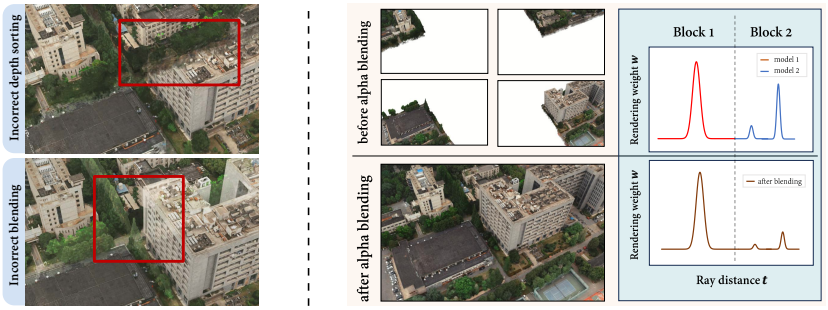
Within block k, the following trainable components are introduced: (1) f^k is an attribute query function that adopts a hash encoding and an MLP decoder that outputs attributes of points such as densities, diffuse color and specular feature (2) Φ^k is a tiny deferred MLP account for view-dependent effects. (3) ψ^k is a proposal MLP for sampling.

Rendering Stage. In the rendering stage, our scene representation includes hierarchical L LODs representation for the scene. Specifically, as shown in the right figure, we merge 2×2 blocks into one block between two consecutive LODs. As a result, for LOD l, where $l \in \{1, 2, \ldots, L\}$, there are $K/4^{l-1}$ blocks. In each block, the following baked textures are used for rendering: (1) $f^{k,l}$ is an attribute query function that takes the coordinates of a sample point as input and directly accesses the opacity, diffuse color and specular features of the sample point from the baked sparse voxel and triplane textures. (2) $\Phi^{k,l}$ represents a tiny deferred MLP that accounts for view-dependent effects. (3) $\psi^{k,l}$ is used as a multi-level occupancy grid for sampling.

4.2 Block-based Volume Rendering

In the rendering stage, we create multiple shaders to render distinct blocks. Specifically, one shader is allocated for storing the texture of an individual block. Each block subsequently renders an image respective to the current camera view. However, a simplistic averaging of these resultant rendering outputs can lead to discernible seams and does not ensure correct occlusion at the inter-block boundaries as shown in Fig. 2a. Therefore, we employ a block-based volume rendering strategy and combine it with depth sorting followed by alpha blending to ensure seamless boundaries and correct occlusion at the edges.

Specifically, in the rendering stage, for a ray $\boldsymbol{r}(t)$ passing through $\boldsymbol{M}$ blocks with a total of $\boldsymbol{N}$ samples, where each block k has $\boldsymbol{n}_k$ samples, we perform volume rendering within each block to obtain its individual rendering diffuse



(a) Incorrect blending **(b)** Ours alpha blending

Fig. 2: Visualization comparison between the alpha blending method and others. (a) Top image: incorrect occlusion without depth sorting. Bottom image: incorrect rendering results when simply using $\alpha_i/(\sum_j \alpha_j)$ as blending weights. (b) Left: rendering results of four separate blocks and the final blending result. Right: visualization of sample points' rendering weights before and after alpha blending.

color $\boldsymbol{C}_d^k$, specular feature $\boldsymbol{F}^k$ and opacity α^k of the ray in block $\mathcal{B}_k$ according to Eq. (1) and Eq. (2). Then we get final rendering color $\boldsymbol{C}^k$ of block $\mathcal{B}_k$ according to Eq. (3). Subsequently, to correctly handle occlusion in rendering, we depth-sort the blocks and apply volume rendering across multiple blocks in sequence, using opacity to generate the blending weights:

$$\boldsymbol{C}(\boldsymbol{r}) = \sum_{k=1}^{M} \prod_{j=1}^{k-1} (1 - \alpha^j) \boldsymbol{C}^k. \tag{5}$$

Under the Lambertian surface setting where the specular color is zero, the color obtained from volume rendering on the total of $\boldsymbol{N}$ ray samples from Eq. (2) is equal to the results produced by our approach of conducting volume rendering within each block followed by inter-block volume rendering Eq. (5). The proof is given in the supplementary. Thus, our rendering approach maintains correct occlusion and keeps 3D consistency when using multiple shaders rendering on the web as shown in Fig. 2b.

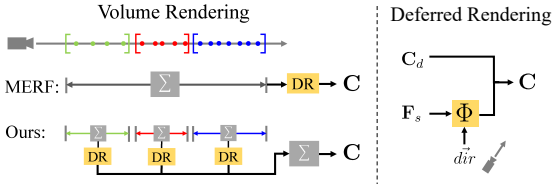


Fig. 3: Block-based volume rendering. "DR" denotes deferred rendering. Φ represents the deferred MLP.

The volume rendering process in MERF involves integrating all sample points together, followed by deferred rendering. In contrast, as shown in Fig. 3, our block-based volume rendering is fundamentally based on segmented integration.

Without the deferred rendering process, it is entirely equivalent to traditional volume rendering. However, if we adhere to the MERF rendering pipeline during the training process, it will lead to discrepancies between the rendering results during the training and rendering phases, ultimately affecting the rendering quality. To minimize this gap, we adopt the same rendering pipeline during the training stage as we do in the rendering stage.

Specifically, in the training stage, for ray $\boldsymbol{r}(t)$, we uniformly sample between the near and far boundaries based on the scene's bounding box. Then, we determine that this point is inside $\mathcal{B}_k$ according to Eq. (4) and query the corresponding proposal MLP ψ^k of $\mathcal{B}_k$ to sample probability distributions along the ray. Similarly, we also query the corresponding f^k to obtain the attributes of the rendering sample points. Lastly, like in the rendering stage, we render each block sequentially to obtain the color and opacity for the ray in $\mathcal{B}_k$ and use Eq. (5) to derive the final rendering result.

4.3 Optimization

In the training stage, we reconstruct the finest LOD model by optimizing it with various losses:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{cb}} + \mathcal{L}_{\text{global}} + \lambda_1 \mathcal{L}_{\text{s3im}} + \lambda_2 \mathcal{L}_{\text{prop}} + \lambda_3 \mathcal{L}_{\text{dist}} + \lambda_4 \mathcal{L}_{\text{s}} + \lambda_5 \mathcal{L}_{\text{opacity}}. \quad (6)$$

Here, we use Charbonnier loss [3] $\mathcal{L}_{cb}$ for reconstruction and S3IM loss [42] $\mathcal{L}_{s3im}$ to assist model in capturing high-frequency details. Additionally, we use the interlevel loss $\mathcal{L}_{prop}$ to provide a supervision signal for proposal MLP and distortion loss $\mathcal{L}_{dist}$ to reduce floaters like Mip-nerf 360 [1].

Sparsity Loss. We random uniform sample points set $\mathcal{P}$ within the bounding box of the scene and apply L_1 regularization on the opacity of sample points α_i to encourage model to predict sparse occupied space:

$$\mathcal{L}_{\text{sparse}} = \frac{1}{|\mathcal{P}|} \sum_{p_i \in \mathcal{P}} |\alpha_i| \quad (7)$$

Opacity Loss. We introduce a regularization term for the opacity of the block. This regularization encourages the opacity of the block to be as close to 0 or 1 as possible, implying either full transparency or full opaqueness:

$$\mathcal{L}_{\text{opacity}} = -\sum_k (\alpha^k log(\alpha^k) + (1-\alpha^k) log(1-\alpha^k)). \quad (8)$$

Regularization of Deferred MLPs. In the deferred rendering context, various combinations of specular and diffuse colors can satisfy multi-view consistency constraints. This situation often leads to incorrect disentanglement of these color components. Our training process involves using a deferred MLP within each block, but this approach does not guarantee the smoothness of specular color across block boundaries and the multitude of possible combinations of specular and diffuse colors.

Table 1: Quantitative comparison. We report PSNR, LPIPS, and SSIM on the test views. The **best** and second best results are highlighted.

	Matrix City			*Campus*			*Rubble*			*Building*		
	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑
NeRFacto	24.95	0.688	0.456	23.47	0.689	0.255	19.02	0.512	0.538	17.70	0.442	0.502
Instant-NGP	23.55	0.629	0.597	21.91	0.549	0.478	20.37	0.478	0.629	-	-	-
Mega-NeRF	25.43	0.674	0.517	22.28	0.565	0.472	**23.68**	0.525	0.558	**20.93**	0.504	0.547
Ours	**25.87**	**0.734**	**0.332**	**24.73**	**0.736**	**0.192**	21.32	0.539	**0.482**	20.13	**0.397**	**0.578**

Inspired by Grid-NeRF [43], which utilizes the smoothness of MLP to regularize explicit grid representations. We also utilize a global deferred MLP to regularize the rendering outputs from smaller, block-specific deferred MLPs, ensuring the global smoothness of specular color. In particular, we combine the specular color generated by this global deferred MLP with the diffuse color to obtain the final rendering result. We then supervise the rendering result using ground truth images in the form of a Charbonnier loss, denoted as $\mathcal{L}_{global}$, to regularize the smaller deferred MLPs. Notably, this global deferred MLP is significantly larger and thus possesses sufficient representational capacity compared to the smaller deferred MLPs designated for each block. Therefore, the global MLP does not limit the model's representational capacity.

4.4 LOD Generation

To ensure high-quality rendering from elevated viewpoints and reduce resource usage for distant scene blocks, our method generates multiple LODs for the scene. One conventional approach to generate LODs would be to retrain the entire scene using fewer blocks, that is, at a lower representation resolution, but this method extends the training time a lot. Additionally, considering the specialized photography techniques employed for capturing large scenes, usually from aerial or top-down perspectives, it is challenging to ensure appearance consistency across models trained separately for extrapolated views if we retrain the entire scene from scratch.

Therefore, we generate LODs based on the scene's finest LOD acquired during the training stage. Specifically, we simulate the virtual grid to store attributes like MERF in the training stage. As merging $M \times M$ blocks into $\frac{M}{2} \times \frac{M}{2}$ blocks to generate LODs, we initially downsample the resolution of the virtual grid in each block by a factor of 2. Subsequently, we freeze the training of the query function f^k within these submodels and retrain a new shared deferred MLP $\Phi^{k,l}$ across merged blocks. Finally, we continue to jointly optimize these submodels and the deferred MLP to adapt to lower-resolution voxels and triplanes.

4.5 Baking

For LOD l, we merge $M \times M$ blocks into $\frac{M}{2^l} \times \frac{M}{2^l}$ blocks. Every $2^l \times 2^l$ blocks can be baked into a single texture asset $f^{k,l}$ in the baking stage thanks to the

Fig. 4: Qualitative comparisons with existing SOTA methods. By testing different methods across diverse scales and environments, it clearly reveals that our approach excels in recovering finer details and achieves a higher quality of reconstruction.

dowmsampling when generating LODs. Thus, in the rendering stage, a single shader is responsible for rendering these $2^l \times 2^l$ blocks.

Specifically, we render all training rays to collect ray samples initially. Samples with opacity and weight values above a certain threshold are retained, and samples below the threshold are discarded. The preserved samples are used to mark the adjacent eight grid points as occupied in the binary occupancy grids $\phi^{k,l}$. After generating binary grids to identify occupied voxels, we follow MERF by baking high-resolution 2D planes and a low-resolution 3D voxel grid in each block to get the attribute function $f^{k,l}$ used in the rendering stage. Only the non-empty 3D voxels are stored using a block-sparse format. We downsample the occupancy grid with max-pooling for efficient rendering. To further save storage, we compress textures into the PNG format.

5 Experiments

5.1 Experiments setup

Dataset and Metric. Our experiments span across various scale and environments. We have incorporated a real-world urban scene dataset (*Campus*), public datasets consisting of real-world scenes (*Rublle* and *Building*) [37] and synthetic city-scale data (*MatrixCity*) [21]. Our datasets were recorded under uniform,

Table 2: Comparison with existing real-time methods. We divide the scene into four blocks and 16 blocks. We split the data into four parts, with each part being reconstructed by an individual MERF for a fair comparison. We present the results of Gauss splatting [20] after training 30,000 and 100,000 iterations for demonstration. VRAM and DISK are denoted in megabytes (MB).

	PSNR↑	SSIM↑	LPIPS↓	VRAM↓	DISK↓	FPS↑
MobileNeRF	19.99	0.516	0.712	712.3	242.1	68
BakedSDF	22.24	0.627	0.413	544.8	515.3	**223**
MERF(4 blocks)	24.02	0.713	0.254	592.4	121.2	58
GS(30k iters)	23.78	0.745	0.263	1469.3	1469.1	77
GS(100k iters)	24.94	**0.783**	0.227	1467.1	1467.1	58
Ours(4 Blocks)	24.82	0.741	0.190	**526.6**	**114.4**	46
Ours(16 Blocks)	**25.13**	0.779	**0.167**	2040.7	464.5	34

cloudy lighting conditions to minimize variation. To obtain precise pose information, we employed an annular capturing approach, which has a higher overlap rate compared to grid-based capturing methods. The dataset covers an area of 1200×800 square meters. It includes a total of 6515 images. We use 99% data for training, and the rest is used as the test dataset. To assess the quality and fidelity of our reconstructions, we employ various evaluation metrics, including **PSNR**, **SSIM** and **LPIPS** [48].

Implementations and Baselines. Our experiments focus on a single part in *Campus* dataset for comparative analysis with existing real-time rendering methods. This part contains over 1600 images, covering an area of approximately 600×400 square meters. We divide this part of the data into four parts for a fair comparison with the MERF method. Other methods did not divide the dataset when conducting experiments. Moreover, we benchmark current real-time rendering methods using three critical parameters: Peak GPU memory usage (VRAM), frames per second (FPS), and on-disk storage (DISK). We report these metrics based on tests conducted on an NVIDIA RTX 3060 GPU with 1920×1080 resolution. More details can be found in the supplement.

5.2 Results Analysis

We systematically evaluate the performance of both baseline models and our method through qualitative and quantitative comparisons in Tab. 1 and Fig. 4. Notably, our method demonstrates a remarkable enhancement in visual fidelity as reflected by the SSIM and LPIPS metrics, which indicate the extent of detail restoration. Despite a reduction in PSNR compared to the SOTA methods, this is attributable to the fact that LPIPS and SSIM are more sensitive to the recovery of fine details, whereas PSNR mainly measures pixel-wise color accuracy. Our approach achieves higher fidelity reconstructions, revealing finer details due to our partitioned reconstruction strategy.

In our evaluation, detailed in Tab. 2 and Tab. 3, we compare our method with current real-time rendering methods. The tests are conducted on a subset of the

Campus dataset and a significantly large scene, *Block ALL* and performed on an NVIDIA RTX 3060 Laptop GPU at a 1920×1080 resolution. The results demonstrate that our method excels in reconstruction quality. We represent each scene block using voxels and triplanes, and store the baked grid attributes as images. This strategy significantly reduces the memory. This reduction notably accelerates resource transmission for web-based rendering applications. However, it is observed that our frame rate during rendering is lower compared to other methods. This is attributed to their rendering pipeline based on mesh rasterization, which is in contrast to our method, which utilizes volume rendering.

Table 3: Comparison on *Block All* scene. Grid-NeRF is tested on a RTX 3090 and the other methods are tested on a RTX 3060.

	PSNR↑	SSIM↑	LPIPS↓	VRAM↓	DISK↓	FPS↑
Grid-NeRF	24.90	0.698	0.480	-	7102 MB	0.55
Gaussian-Splatting	25.42	**0.784**	**0.308**	1978 MB	1978 MB	17
Ours	**25.87**	0.734	0.332	**912 MB**	**279 MB**	**34**

5.3 LOD Result

Tab. 4 presents the quantitative rendering results at various LODs, along with the corresponding DISK and VRAM usage. With increasing LOD, the resources required for rendering significantly decrease. Notably, our method's lowest LOD still maintains high-fidelity rendering results, as demonstrated in Fig. 5. Our LOD strategy significantly streamlines the management of resource loading on web platforms, which is particularly advantageous in rendering distant blocks, as it requires less VRAM. It is worth noting that the VRAM usage presented in Tab. 4 represents the cumulative memory consumption of all blocks. Our dynamic loading strategy adaptively selects resources to load based on the camera's field of view and the distance to each block, effectively keeping the peak VRAM usage around 1100MB.

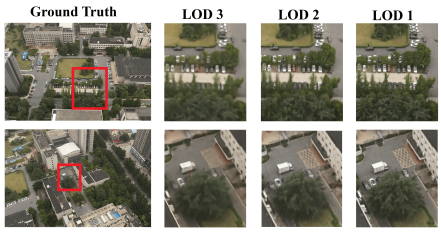


Fig. 5: Visualization of LOD result.

5.4 Ablation Study

We conduct ablation studies to demonstrate the impact of the contributions introduced to our method.

Table 4: The LOD results on the whole *Campus* dataset. VRAM and DISK are denoted in megabytes (MB).

	PSNR↑	SSIM↑	LPIPS↓	VRAM↓	DISK↓
LOD3	23.72	0.660	0.306	**132.1**	**40.2**
LOD2	24.23	0.682	0.297	841.6	201.7
LOD1	**24.73**	**0.736**	**0.192**	3970.2	1259.6

Table 5: Ablations on LOD generation. "Time" column indicates the time required to generate the LOD.

	PSNR↑	SSIM↑	LPIPS↓	Time ↓
downsample	22.43	0.614	0.362	**0 hours**
from scratch	24.17	0.722	0.224	12 hours
ours	**24.20**	**0.724**	**0.204**	4 hours

Table 6: Ablation study on our method. The result is tested on one section of the *Campus* dataset. VRAM and DISK are denoted in megabytes (MB).

	PSNR↑	SSIM↑	LPIPS↓	VRAM↓	DISK↓
model with 1024^3 Res.	24.05	0.710	0.201	540.9	147
model with 2048^3 Res.	24.42	0.751	0.184	3073.2	457
no alpha blending	24.03	0.684	0.345	565.4	153
no consistent training	24.21	0.702	0.281	514.4	110
no global deferred mlp	24.61	0.712	0.198	536.6	126
ours(4 blocks)	24.82	0.741	0.190	**526.6**	**114**
ours(16 Blocks)	**25.13**	**0.779**	**0.167**	2040.7	464

Ablation on Our Method. In Tab. 6, we conduct an ablation study of our method on one part of *Campus* dataset "ours(4 blocks)" means we use four blocks with 512^3 voxel resolution and 2048^3 triplane resolution for scene reconstruction. "ours(16 blocks)" means we use 16 blocks with 512^3 voxel resolution and 2048^3 triplane resolution for scene reconstruction. In "model with 1024^3 Res.", we train a one block MERF model with 1024^3 voxel resolution and 4096^2 triplane resolution. In "model with 2048^3 Res.", we train a one block MERF model with 2048^3 voxel resolution and 8192^2 triplane resolution. Our method has higher rendering quality and requires less storage space. In "no alpha blending", we instead our alpha blending with simply using $\alpha_i/(\sum_j \alpha_j)$ as blending weights. This non-occlusion-aware blending strategy significantly reduces the rendering quality. In "no consistent training", we use MERF's volume rendering pipeline in the training stage. In "no global deferred MLP", we remove global deferred MLP. Without the regularization of deferred MLPs, the quality of the reconstruction has decreased.

Ablation on LOD Generation. Tab. 5 shows ablation study on LOD generation. We use our LOD generation method as basline. In "downsample", we simply downsample the model without re-optimization. In "from scratch", we do not use the finest LOD model to generate LOD. Instead, we trained the same resolution model from scratch.

6 Conclusion and Disscussion

In this work, we introduced City-on-Web, which to our knowledge is the first system that enables real-time neural rendering of large-scale scenes on the web

using laptop GPUs. Our choice of a block-based volume rendering strategy, tailored for resource-independent web environments, achieves seamless integration between blocks. Our carefully designed LOD generation and refinement strategy support dynamic loading, minimizing necessary resources on the web while ensuring the best visual experience. Extensive experiments have also fully proved the effectiveness of City-on-Web.

Scientific Impact. Unlike existing block-based methods that require the simultaneous loading of all block resources for rendering, our innovative method allows each block to be rendered independently using its own texture in its own shader. This strategy supports independent rendering, making it highly adaptable to various resource-independent environments, opening new avenues for research and application, and potentially impacting the efficiency and scalability of large-scale radiance field training.

For example, in multi-GPU and distributed system environments, our method enables the direct transfer of rendered color and opacity of each block among GPUs or nodes, reducing the need to transfer numerous sample attributes. This reduction in data transfer minimizes communication overhead, thereby eliminating the common bottleneck in large-scale distributed training and substantially improving training speeds. We are glad to see future work applying this strategy to the parallel or distributed training of NeRF or Gaussian splatting model.

Limitations. Since we derive alpha blending across shaders based on the Lambertian surface assumption, visible seams may occur at the boundaries between blocks on non-Lambertian surfaces, such as water surfaces. Combining physically-based rendering with multiple shaders blending may alleviate this problem. Additionally, while our approach achieves real-time rendering of large scenes on consumer-grade laptops, the inherently resource-intensive nature of large scenes makes that real-time rendering on mobile devices remains a challenge. Moreover, while our method recovers more intricate geometrical detail, it frequently results in color discrepancies with the ground truth image due to unstable lighting conditions and variable exposure, as shown in Fig. 6. Our deferred shading model has limited ability to represent view-dependent colors and cannot accurately represent lighting changes in the data as view-dependent effects. This limitation results in slightly lower PSNR values.

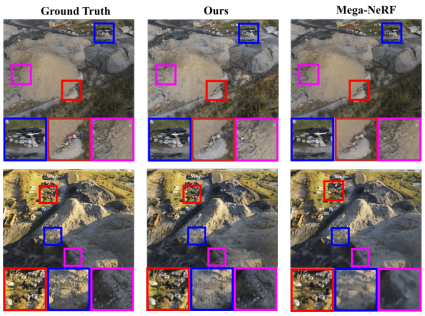


Fig. 6: Comparison to Mega-NeRF in *Rubble* dataset. The dataset presents significant variations in lighting.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (No.62122071, No.62272433), and the Fundamental Research Funds for the Central Universities (No. WK3470000021). The numerical calculations in this paper have been done on the supercomputing system in the Supercomputing Center of University of Science and Technology of China.

References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
2. Cao, J., Wang, H., Chemerys, P., Shakhrai, V., Hu, J., Fu, Y., Makoviichuk, D., Tulyakov, S., Ren, J.: Real-time neural light field on mobile devices. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8328–8337 (2023)
3. Charbonnier, P., Blanc-Féraud, L., Aubert, G., Barlaud, M.: Deterministic edge-preserving regularization in computed imaging. IEEE Transactions on image processing **6**(2), 298–311 (1997)
4. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022)
5. Chen, Z., Funkhouser, T., Hedman, P., Tagliasacchi, A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16569–16578 (2023)
6. Clark, J.H.: Hierarchical geometric models for visible surface algorithms. Communications of the ACM **19**(10), 547–554 (1976)
7. Crassin, C., Neyret, F., Lefebvre, S., Eisemann, E.: Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In: Proceedings of the 2009 symposium on Interactive 3D graphics and games. pp. 15–22 (2009)
8. Duchaineau, M., Wolinsky, M., Sigeti, D.E., Miller, M.C., Aldrich, C., Mineev-Weinstein, M.B.: Roaming terrain: Real-time optimally adapting meshes. In: Proceedings. Visualization'97 (Cat. No. 97CB36155). pp. 81–88. IEEE (1997)
9. Gao, X., Zhong, C., Xiang, J., Hong, Y., Guo, Y., Zhang, J.: Reconstructing personalized semantic facial nerf models from monocular video. ACM Transactions on Graphics (TOG) **41**(6), 1–12 (2022)
10. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14346–14355 (2021)
11. Gu, J., Jiang, M., Li, H., Lu, X., Zhu, G., Shah, S.A.A., Zhang, L., Bennamoun, M.: Ue4-nerf: Neural radiance field for real-time rendering of large-scale scene. Advances in Neural Information Processing Systems **36** (2024)
12. Guo, J., Deng, N., Li, X., Bai, Y., Shi, B., Wang, C., Ding, C., Wang, D., Li, Y.: Streetsurf: Extending multi-view implicit surface reconstruction to street views. arXiv preprint arXiv:2306.04988 (2023)

13. Guo, Y., Chen, K., Liang, S., Liu, Y.J., Bao, H., Zhang, J.: Ad-nerf: Audio driven neural radiance fields for talking head synthesis. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 5784–5794 (2021)
14. Guthe, S., Wand, M., Gonser, J., Straßer, W.: Interactive rendering of large volume data sets. In: IEEE Visualization, 2002. VIS 2002. pp. 53–60. IEEE (2002)
15. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5875–5884 (2021)
16. Hong, Y., Peng, B., Xiao, H., Liu, L., Zhang, J.: Headnerf: A real-time nerf-based parametric head model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20374–20384 (2022)
17. Hoppe, H.: Progressive meshes. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. p. 99–108. SIGGRAPH '96, Association for Computing Machinery, New York, NY, USA (1996). https://doi.org/10.1145/237170.237216, https://doi.org/10.1145/237170.237216
18. Hu, W., Wang, Y., Ma, L., Yang, B., Gao, L., Liu, X., Ma, Y.: Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19774–19783 (2023)
19. Jiang, B., Hong, Y., Bao, H., Zhang, J.: Selfrecon: Self reconstruction your digital avatar from monocular video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5605–5615 (2022)
20. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (July 2023), https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/
21. Li, Y., Jiang, L., Xu, L., Xiangli, Y., Wang, Z., Lin, D., Dai, B.: Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3205–3215 (2023)
22. Lindstrom, P., Pascucci, V.: Visualization of large terrains made easy. In: Proceedings Visualization, 2001. VIS'01. pp. 363–574. IEEE (2001)
23. Liu, L., Gu, J., Zaw Lin, K., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. Advances in Neural Information Processing Systems **33**, 15651–15663 (2020)
24. Losasso, F., Hoppe, H.: Geometry clipmaps: terrain rendering using nested regular grids. In: ACM Siggraph 2004 Papers, pp. 769–776 (2004)
25. Luebke, D.: Level of detail for 3D graphics. Morgan Kaufmann (2003)
26. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7210–7219 (2021)
27. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
28. Peng, B., Hu, J., Zhou, J., Gao, X., Zhang, J.: Intrinsicngp: Intrinsic coordinate based hash encoding for human nerf. IEEE Transactions on Visualization and Computer Graphics (2023)
29. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14335–14345 (2021)

30. Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P., Mildenhall, B., Geiger, A., Barron, J., Hedman, P.: Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. ACM Transactions on Graphics (TOG) **42**(4), 1–12 (2023)
31. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5459–5469 (2022)
32. Takikawa, T., Evans, A., Tremblay, J., Müller, T., McGuire, M., Jacobson, A., Fidler, S.: Variable bitrate neural fields. In: ACM SIGGRAPH 2022 Conference Proceedings. SIGGRAPH '22, Association for Computing Machinery (2022)
33. Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M., Fidler, S.: Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11358–11367 (2021)
34. Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8248–8258 (2022)
35. Tang, J., Zhou, H., Chen, X., Hu, T., Ding, E., Wang, J., Zeng, G.: Delicate textured mesh recovery from nerf via adaptive surface refinement. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2023)
36. Turki, H., Agrawal, V., Bulò, S.R., Porzi, L., Kontschieder, P., Ramanan, D., Zollhöfer, M., Richardt, C.: Hybridnerf: Efficient neural rendering via adaptive volumetric surfaces. In: Computer Vision and Pattern Recognition (CVPR) (2024)
37. Turki, H., Ramanan, D., Satyanarayanan, M.: 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. pp. 12922–12931 (2022)
38. Wang, P., Liu, Y., Chen, Z., Liu, L., Liu, Z., Komura, T., Theobalt, C., Wang, W.: F2-nerf: Fast neural radiance field training with free camera trajectories. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4150–4159 (2023)
39. Wang, Z., Shen, T., Nimier-David, M., Sharp, N., Gao, J., Keller, A., Fidler, S., Müller, T., Gojcic, Z.: Adaptive shells for efficient neural radiance field rendering. ACM Trans. Graph. **42**(6) (2023). `https://doi.org/10.1145/3618390`, `https://doi.org/10.1145/3618390`
40. Xiang, J., Gao, X., Guo, Y., Zhang, J.: Flashavatar: High-fidelity digital avatar rendering at 300fps. arXiv preprint arXiv:2312.02214 (2023)
41. Xiangli, Y., Xu, L., Pan, X., Zhao, N., Rao, A., Theobalt, C., Dai, B., Lin, D.: Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In: European conference on computer vision. pp. 106–122. Springer (2022)
42. Xie, Z., Yang, X., Yang, Y., Sun, Q., Jiang, Y., Wang, H., Cai, Y., Sun, M.: S3im: Stochastic structural similarity and its unreasonable effectiveness for neural fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 18024–18034 (2023)
43. Xu, L., Xiangli, Y., Peng, S., Pan, X., Zhao, N., Theobalt, C., Dai, B., Lin, D.: Grid-guided neural radiance fields for large urban scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8296–8306 (2023)
44. Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P.P., Szeliski, R., Barron, J.T., Mildenhall, B.: Bakedsdf: Meshing neural sdfs for real-time view synthesis. In:

ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023. pp. 46:1–46:9. ACM (2023)
45. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5752–5761 (2021)
46. Zeng, X., Song, K., Yang, L., Deng, B., Zhang, J.: Oblique-merf: Revisiting and improving merf for oblique photography. arXiv preprint arXiv:2404.09531 (2024)
47. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020)
48. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
49. Zhenxing, M., Xu, D.: Switch-nerf: Learning scene decomposition with mixture of experts for large-scale neural radiance fields. In: The Eleventh International Conference on Learning Representations (2022)
50. Zhuang, Y., Zhang, Q., Feng, Y., Zhu, H., Yao, Y., Li, X., Cao, Y.P., Shan, Y., Cao, X.: Anti-aliased neural implicit surfaces with encoding level of detail. arXiv preprint arXiv:2309.10336 (2023)