Oblique-MERF: Revisiting and Improving MERF for Oblique Photography

Xiaoyi Zeng[1] Kaiwen Song[1] Leyuan Yang[1] Bailin Deng[2] Juyong Zhang[1, 3*]
[1]University of Science and Technology of China [2]Cardiff University
[3]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
zxy1908542805, SA21001046, ly_1207@mail.ustc.edu.cn, DengB3@cardiff.ac.uk, juyong@ustc.edu.cn

Abstract

Neural radiance fields (NeRF) have established a new paradigm for 3D scene reconstruction, with subsequent work achieving high-quality real-time rendering. However, reconstructing large-scale scenes from oblique aerial photography presents unique challenges, such as varying spatial scale distributions and a constrained range of tilt angles, often resulting in high memory consumption and reduced rendering quality at extrapolated viewpoints. To address these issues, we propose a novel approach named Oblique-MERF to accommodate the distinctive characteristics of oblique photography datasets and support real-time rendering on various common devices. Firstly, an innovative adaptive occupancy plane is proposed to constrain the sampling space. Additionally, we propose a smoothness regularization loss for view-dependent color to enhance the MLP's ability to generalize to untrained viewpoints. Experimental results demonstrate that Oblique-MERF reduces VRAM usage by approximately 40% *while maintaining competitive rendering quality compared to baseline methods, and achieves higher frame rates with more realistic rendering even at untrained extrapolated viewpoints. Project page: https://ustc3dv.github.io/Oblique-MERF/*

1. Introduction

Reconstructing a 3D scene for high-fidelity rendering from freely chosen viewpoints has been a longstanding challenge in computer vision. NeRF [30] accomplishes this via a novel implicit representation parameterized by multi-layer perceptrons (MLP). In the realm of large-scale scene reconstruction for oblique aerial photography datasets, various works have sought to improve upon NeRF by employing strategies such as spatial partitioning [29, 35, 43], advanced sampling methods [47, 50], and efficient data structures [32, 38, 42, 57]. Furthermore, recent works bake models into meshes with optimized textures [6, 45, 56] or feature grids [15, 36], to enable interactive rendering frame rates on commercial devices.

[*]Corresponding author.

	Memory efficient	Query speed	Loss aware
Occupancy Grid [32]	×	✓	×
Proposal MLP [2]	✓	×	✓
Occupancy Plane (Ours)	✓	✓	✓

Table 1. Difference among sampling representations.

However, these advancements have not been specifically tailored to address the unique challenges posed by oblique aerial photography data. When applied to such extensive scenes, these methods often result in significant memory consumption and fail to deliver high-fidelity rendering from all viewing angles. Specifically, oblique aerial photography datasets typically cover vast areas, spanning hundreds of thousands of square meters horizontally, yet extend only a few hundred meters vertically. The commonly used cubic grid systems struggle with memory complexity and adaptability. Additionally, these datasets may introduce visual artifacts such as floaters due to inadequate constraints on the sampling space. Moreover, because these scenes are typically captured from a limited range of pitch angles relative to the ground, they can exhibit abnormal highlights and shadows when viewed from angles not covered in the training data. In this work, we enhance the performance of current NeRF-based real-time rendering methods on oblique aerial photography datasets, focusing on two key aspects: the representation of the sampling space and the issue of extrapolated view synthesis.

Current popular methods, such as 3D grids or implicit proposal sampling networks [2, 32], may encounter challenges when applied to such vast scenes. Grid-based representations, constrained by $\mathcal{O}(n^3)$ memory complexity, struggle to efficiently represent sampling space at high resolution; moreover, they often rely on heuristic optimization methods that are not inherently tied to rendering quality. These limitations can hinder precise occupancy optimization and lead to redundant sample points that slow down the rendering pipeline and affect the rendering quality. On the other hand, network-based representations are computationally expensive, as they require evaluating numerous

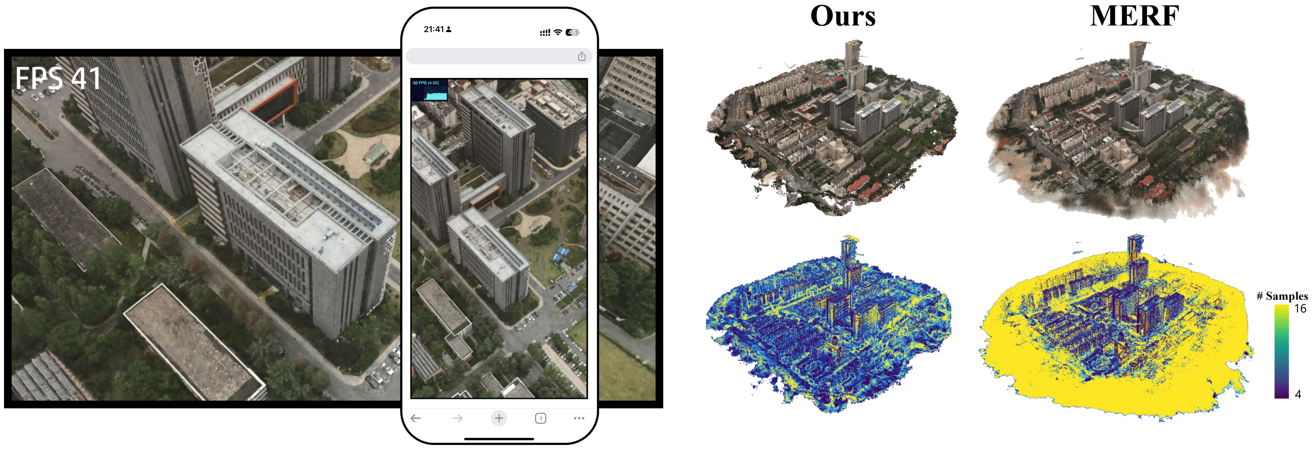


Figure 1. Oblique-MERF enables real-time synthesis of novel views for oblique photography on diverse commodity devices, including tests on an NVIDIA GTX 1650 ($\sim$ 41FPS) and an iPhone 14 Pro Max ($\sim$ 45FPS) (left). We introduce a novel compact sampling representation that markedly reduces sampling points and emphasizes significant regions, enhancing rendering quality and increasing frame rates. We present the rendering output and the visualization of the number of samples (right).

candidate points to determine the final sample points. To overcome these drawbacks, we propose a novel sampling strategy that models the occupancy space as a zone bounded by two height field surfaces conforming to the ground. Furthermore, we integrate this adaptive representation with volume rendering and optimize it in a differentiable manner to ensure awareness of the photometric loss. As shown in Tab. 1, our explicit representation of the sampling space not only reduces memory complexity to $\mathcal{O}(n^2)$ but also facilitates rapid querying, paving the way for real-time rendering on various commodity devices.

In addition, we tackle the extrapolated view issue that arises from the limited range of view directions in the training set. In novel extrapolated views, rendering results often display aberrant colors due to the absence of supervision. These anomalies typically stem from the non-smooth behavior of specular color with respect to varying viewpoints. To address it, we introduce a novel smoothing term to regularize view-dependent color. Specifically, our approach targets the variation of the specular component relative to the viewing directions, which leads to a more natural handling of view-dependent color during rendering from most views.

In summary, our primary contributions are:

- We introduce a novel sampling strategy that integrates the proposed 2D occupancy plane with volume rendering, minimizing memory usage while ensuring high fidelity.
- Utilizing the smooth nature of specular reflections, we present a regularization that enhances the rendering quality of extrapolated viewpoints, yielding smoother and more realistic visuals.
- Extensive experiments demonstrate that our method decreases VRAM usage by 40%, and boosts the frame rate by 35% and 300% for low-altitude and high-altitude viewpoints, respectively, while achieving better rendering quality compared to the baseline.

2. Related Work

3D Reconstruction for Oblique Photography. Traditional 3D modeling techniques include using Structure-from-Motion [1, 10, 39, 52] to estimate camera poses and obtain sparse point clouds, followed by surface reconstruction through dense multi-view stereo [11, 12, 17]. These methods struggle to reconstruct view-dependent colors. With the advent of NeRF [30], neural rendering for novel view synthesis has been increasingly applied to large-scale scene reconstruction. Some approaches [13, 29, 40, 43, 46] adopt a divide-and-conquer strategy, segmenting the scene into chunks to perform parallel reconstruction, and then merging them to represent the entire scene. BungeNeRF [53] employs residual networks to learn multi-scale features, fitting scenes with dramatic changes in altitude. To accommodate oblique photography, GridNeRF [55] combines a multi-resolution ground feature plane representation with vanilla NeRF incorporating position-encoded inputs, while our method uses a occupancy plane to model the height field for effective sampling.

Real-Time Rendering. Some methods accelerate rendering by reducing the volume of network queries [20, 33, 41], decomposing large MLP to facilitate parallel processing [35], or introducing explicit structures that store features, such as voxel grids or octrees, to replace partial or entire network [5, 21, 32, 38, 42, 57]. Other approaches seek to circumvent extensive network queries by baking models into explicit structures [15]. On the contrary, certain works [3, 6, 14, 26, 37, 45, 48, 56] utilize optimized surfaces with textures to represent scenes, integrating into modern computer graphics rendering pipelines. While achieving interactive frame rates on commercial devices, they fall short in rendering quality and memory efficiency. Some work [36, 59] employs a memory-efficient triplane combined with a sparse grid to represent the features of spatial points, striking a balance between rendering quality and memory consumption. Recently, 3D Gaussian Splatting(3DGS) [18] utilizes 3D ellipsoids as representation primitives. High-quality and fast reconstruction and rendering are achieved by splatting ellipsoids onto 2D images using point-based rendering [60]. Subsequent works [24, 27] have extended this representation to large-scale scenes, yet still face challenges such as significant memory consumption and optimization difficulties.

Sampling in Rendering. Various methods enhance rendering efficiency by refining the sampling strategy. NeRF [30] and Mip-NeRF 360 [2] employ a coarse-to-fine strategy to concentrate on significant regions. DDNeRF [7] adopts the Gaussian function instead of a piecewise-constant probability function, achieving accurate density representation. DONeRF [33] and ENeRF [23] use depth information to reduce the number of sampling points. NeuSample [9] maps rays to sampling points through a single inference. In contrast to the network-based sampling, Instant-NGP [32] skips empty space by explicit multi-resolution occupancy grids. While Adaptive Shells [50] and HybridNeRF [47] refine the sampling interval size across different spatial locations by optimizing the spatially-varying parameter, compressing the sampling area.

Regularizations of Neural Fields. In addition to these advancements, subsequent works on NeRF have introduced various regularization terms to enhance its performance. Plenoxels [38] proposes a total variation loss to minimize differences in features among adjacent voxels, and RegNeRF [34] aims to encourage the continuity of volumetric density changes. Additionally, sparse regularization [36, 57] is employed in many real-time rendering schemes to eliminate irrelevant features. In the realm of neural fields, numerous studies have incorporated regularization terms to encourage the smoothness of networks, such as penalties on the norms of Jacobians and Hessians [8, 31] or encouragement of smaller Lipschitz constants for weights [25].

3. Background

NeRF [30] employs an MLP to represent a scene as a continuous volumetric function $\mathcal{F} : (\mathbf{p}, \mathbf{d}) \mapsto (\sigma, \mathbf{c})$, which maps positional encoding of a 3D point $\mathbf{p} \in \mathbb{R}^3$, and a normalized direction $\mathbf{d} \in \mathbb{S}^2$, to volumetric density $\tau(\mathbf{p})$ and color $\mathbf{c}(\mathbf{p}, \mathbf{d})$. To render the corresponding color of a pixel, a ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ is first emitted from the origin $\mathbf{o}$ along the view direction $\mathbf{d}$, where dozens of points $\{\mathbf{p}_i = \mathbf{o} + t_i\mathbf{d} \mid i = 1, \ldots, n, t_i < t_{i+1}\}$ are sampled to estimate their volumetric density and features. These estimates are integrated to synthesize the final color [28]:

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum\nolimits_{i=1}^{n} \omega_i \mathbf{c}_i, \quad \omega_i = T_i \alpha_i, \tag{1}$$

where $\alpha_i = (1 - e^{-\sigma_i \delta_i})$ is the opacity of the sample $\mathbf{p}_i$, with $\delta_i = t_{i+1} - t_i$ being the distance between adjacent samples; $T_i = \prod_{j=1}^{i-1}(1 - \alpha_j)$ is the accumulated transmittance from $\mathbf{p}_1$ to $\mathbf{p}_{i-1}$.

MERF [36] employs a low-resolution voxel grid $\mathbf{V} \in \mathbb{R}^{L \times L \times L \times 8}$ and a high-resolution triplane $\{\mathbf{P}_i \in \mathbb{R}^{R \times R \times 8} \mid i = x, y, z\}$ to represent the scene. For each sample point $\mathbf{p}$, the feature is obtained by adding the results of trilinear interpolation on the sparse grid and bilinear interpolation on the three planes, respectively:

$$\mathbf{t}(\mathbf{p}) = \mathbf{V}(\mathbf{p}) + \mathbf{P}_x(\mathbf{p}) + \mathbf{P}_y(\mathbf{p}) + \mathbf{P}_z(\mathbf{p}). \tag{2}$$

Similar to SNeRG [15], the feature $\mathbf{t}(\mathbf{p})$ is split into $[\tilde{\tau}, \tilde{\mathbf{c}}^d, \tilde{\mathbf{f}}]$. Exponential $\exp(\cdot)$ and sigmoid activation function $\sigma(\cdot)$ are applied to obtain volumetric density $\tau \in \mathbb{R}$, diffuse color $\mathbf{c}^d \in \mathbb{R}^3$, and specular features $\mathbf{f} \in \mathbb{R}^4$:

$$\tau = \exp(\tilde{\tau}), \quad \mathbf{c}^d = \sigma(\tilde{\mathbf{c}}^d), \quad \mathbf{f} = \sigma(\tilde{\mathbf{f}}). \tag{3}$$

After alpha composition as in Eq. (1), the diffuse color and specular features are concatenated with the direction and fed into the Deferred MLP $\mathcal{G}$ to obtain the final color:

$$\begin{aligned} \hat{\mathbf{C}}(\mathbf{r}) &= \sum\nolimits_{i=1}^{n} \omega_i \mathbf{c}_i^d + \mathcal{G}(\mathbf{F}, \mathbf{d}), \\ \mathbf{F} &= \left[\sum\nolimits_{i=1}^{n} \omega_i \mathbf{c}_i^d, \sum\nolimits_{i=1}^{n} \omega_i \mathbf{f}_i\right]. \end{aligned} \tag{4}$$

After training, a full-resolution rendering of all images is performed to determine the occupied space, and the volumetric density, diffuse colors, and specular features within this space are stored for rendering on the web.

4. Method

In this paper, we present a high-fidelity, real-time rendering approach tailored for oblique photography. Sec. 4.1 details the occupancy plane and its optimization for efficient sampling. Sec. 4.2 introduces a smooth regularization to address the view extrapolation issue. Sec. 4.3 describe the entire optimization process. Sec. 4.4 covers the rapid baking and real-time rendering based on the proposed model. The pipeline of our method is shown in Fig. 2.

4.1. Occupancy Plane

When reconstructing scenes from aerially captured data, we align the world coordinate system's xy-plane with the ground, with the z-axis perpendicular to the ground and pointing upward. Under this configuration, we identify two key observations:

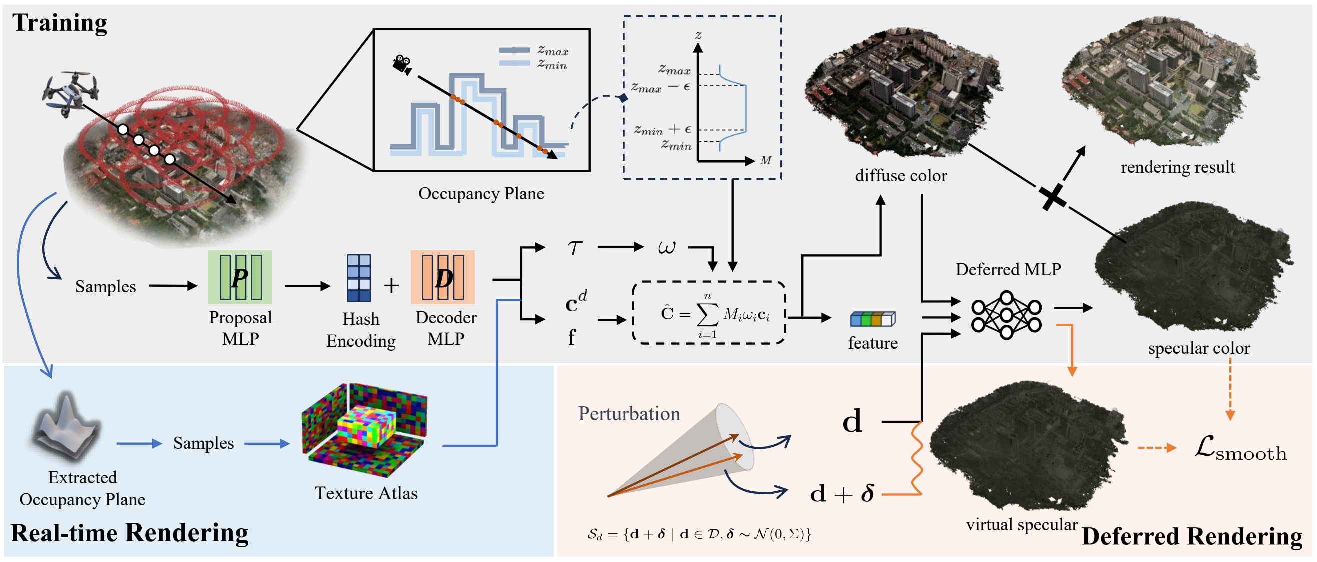
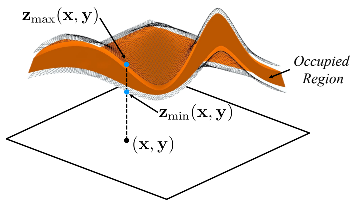


Figure 2. Overview of our Oblique-MERF pipeline. During training, we introduce a 2D plane to represent the occupied space as a sandwiched region between two height field surfaces. For sampling points on rays, occupancy masks retrieved from the occupancy plane, are used as multipliers in the volume rendering process(Sec. 4.1). Additionally, we incorporate a smoothness regularization for view-dependent color to minimize variations in specular color with viewing direction(Sec. 4.2). Post-training, spatial occupancy information is directly extracted from the occupancy plane, and corresponding features are stored for real-time rendering(Sec. 4.4).

1. For any point on the xy-plane, scene elements exist within a single continuous interval along the z-axis, without isolated mid-air objects;
2. As the z-value increases, the occupancy within the scene gradually becomes sparser, signifying a decrease in elements or structures at higher elevations.

Based on these observations, widely used 3D grids are unable to simultaneously achieve high precision, efficient querying, and low memory usage due to their cubic spatial complexity and regular structure, while network-based representations are unsuitable for real-time rendering owing to the heavy computational load.


$\mathbf{z}_{\max}(\mathbf{x},\mathbf{y})$
Occupied Region
$\mathbf{z}_{\min}(\mathbf{x},\mathbf{y})$
$(\mathbf{x},\mathbf{y})$

As an optimal solution, we represent the occupied space as a region sandwiched between two height field surfaces that bound the lower and upper limits of z-coordinates (see inset figure). To parameterize it, we sample the xy-plane with a high-resolution $M\times M$ grid, and store the heights $z_{\min}(x,y)$ and $z_{\max}(x,y)$ of the two surfaces at each sample grid point (x,y), resulting in a representation $\mathcal{P}_o\in\mathbb{R}^{M\times M\times 2}$. We refer to this as the *occupancy plane* in the following. The heights $z_{\min}(x,y)$ and $z_{\max}(x,y)$ define a continuous occupied internal $\mathcal{I}_{(x,y)}=[z_{\min}(x,y),z_{\max}(x,y)]$ for the z-coordinates corresponding to each grid point (x,y). We set the occupancy probability outside this interval to be zero, so that we can exclude the points outside it when computing the color for a ray. To maximize memory efficiency, we would like to compress the interval as much as possible. Thus, we introduce a loss term to penalize the span of the interval:

$$\mathcal{L}_{\text{occ}}=\sum\nolimits_{(x,y)\in\mathcal{S}}(z_{\max}(x,y)-z_{\min}(x,y))^2, \tag{5}$$

where $\mathcal{S}$ is the set of grid points for the occupancy plane.

However, simply compressing the occupancy intervals using $\mathcal{L}_{\text{occ}}$ can affect the reconstruction quality if significant elements in the scene are left outside the sandwiched region. To avoid this issue, we integrate the occupancy plane into the volume rendering process, to ensure the final occupancy intervals are sufficient to represent the scene. Specifically, for the vertical line over each sample grid point (x,y), we derive a differentiable occupancy function for the points along the line based on their z-coordinates and the occupancy interval $[z_{\min}(x,y),z_{\max}(x,y)]$ (below we ignore the arguments (x,y) for $z_{\min}$ and $z_{\max}$ for brevity):

$$\begin{aligned}&M_{(x,y)}(z;\mathcal{P}_o)\\&=\begin{cases}1 & \text{if } z\in[z_{\min}+\epsilon,z_{\max}-\epsilon],\\0 & \text{if } z\in(-\infty,z_{\min})\cup(z_{\max},\infty),\\(z-z_{\min})^q/\epsilon^q & \text{if } z\in[z_{\min},z_{\min}+\epsilon],\\(z_{\max}-z)^q/\epsilon^q & \text{if } z\in[z_{\max}-\epsilon,z_{\max}].\end{cases}\end{aligned} \tag{6}$$

Here we use a threshold ϵ to introduce two buffer zones $[z_{\min},z_{\min}+\epsilon]$ and $[z_{\max}-\epsilon,z_{\max}]$ around the bounds, to ensure smooth transitions from zero occupancy outside these limits to full occupancy within using power functions. In this paper, We set the exponent parameter $q = 2$. During volume rendering, for a sample point $\mathbf{p}_i = (x_i, y_i, z_i)$ on a ray $\mathbf{r}$, we project it onto the XY plane and find the nearest sample grid point $(\overline{x}_i, \overline{y}_i)$ to obtain occupancy probability:

$$M_{(x_i,y_i)}(z_i; \mathcal{P}_o) = M_{(\overline{x}_i,\overline{y}_i)}(z_i; \mathcal{P}_o). \quad (7)$$

The value is then combined with the weight from volume rendering, to determine the contribution of the feature at $\mathbf{p}_i$ to the final color:

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum\nolimits_{i=1}^{n} M_{(x_i,y_i)}(z_i; \mathcal{P}_o)\omega_i \mathbf{c}_i. \quad (8)$$

This is used to define a photometric loss $\mathcal{L}_{\text{rgb}}$ in Eq. (12) that penalizes the deviation between the predicted color and the ground truth. The occupancy value in Eq. (7) correlates the predicted color with the occupancy interval, such that the photometric loss suppresses further compression of the occupancy interval when its endpoints with high weights, ensuring optimal rendering quality and memory efficiency. In addition, the occupancy plane reduces computational overhead by filtering out numerous sampling points (see Fig. 1).

4.2. Smoothness Regularization for View-Dependent Color

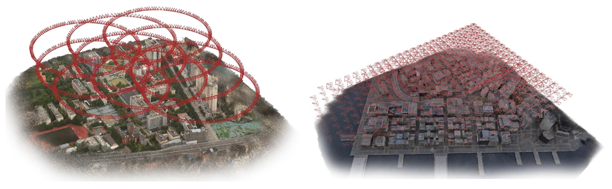

(a) Surround-Style (b) Grid-Style

Figure 3. Two camera trajectories for oblique photography.

Another challenge in large-scale scene reconstruction with oblique photography is the extrapolation issue for view-dependent colors. As shown in Fig. 3, when a drone flies through the scene, the conventional approach involves capturing images with limited pitch angles, either by orbital capture at a consistent angle relative to the ground plane or by employing a multi-camera system to photograph the scene in a grid pattern. The limited range of pitch angles in these photography methods often results in unnatural specular colors when viewed from a horizontal or upward perspective that is outside the range of view angles in the captured images (see Fig. 4(a)(b)).

In our context, the Deferred MLP $\mathcal{G}$ that generates the color (see Eq. (4)) is supervised from a sparse set of input viewpoints. When rendering under extrapolated viewpoints, the specular component reveals significant uncertainty, leading to extreme highlights in the images. Our key observation is that adjacent viewpoints in real-world scenes often exhibit similar specular reflection colors, aligning with the local consistency seen in BRDF on smooth surfaces. We incorporate this prior into our model to encourage

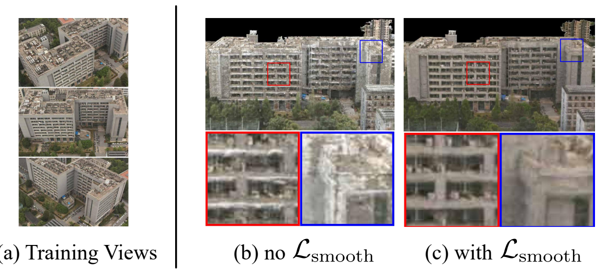

(a) Training Views (b) no $\mathcal{L}_{\text{smooth}}$ (c) with $\mathcal{L}_{\text{smooth}}$

Figure 4. In (a), the training views are captured at limited tilt angles. (b) and (c) illustrate the rendering results from a novel extrapolation viewpoint without and with $\mathcal{L}_{\text{smooth}}$, respectively.

smoothness in view-dependent colors in unseen viewpoints. A typical approach is to impose constraints on the network parameters to achieve this continuity [8, 16, 25, 31]. For example, LipschitzMLP [25] achieves smoothness of the output by constraining the Lipschitz constant of the network. However, it enforces smoothness with respect to all inputs of the MLP (i.e., the viewing direction, the diffuse color, and the specular feature), whereas we only require smoothness with respect to the viewing direction; this could lead to over-constraints for the MLP parameters and may hinder the optimization process. Thus we propose a smoothness regularization for view-dependent color. Specifically, for a known viewing direction $\mathbf{d} \in \mathcal{D}$ in the training set, we apply a small Gaussian perturbation to define a sampling space for the viewing direction.

$$\mathcal{S}_d = \{\mathbf{d} + \boldsymbol{\delta} \mid \mathbf{d} \in \mathcal{D}, \boldsymbol{\delta} \sim \mathcal{N}(0, \Sigma)\}, \quad (9)$$

where Σ is a hyperparameter that determines the sampling range. Then, we introduce a loss to penalize large changes between the deferred MLP's outputs for $\mathbf{d}$ and $\mathbf{s} \in \mathcal{S}_d$

$$\mathcal{L}_{\text{smooth}} = \sum_{\mathbf{d}\in\mathcal{D}} \sum_{\mathbf{s}\in\mathcal{S}_d} S(\mathbf{d}, \mathbf{s}) \|\mathcal{G}(\mathbf{F}, \mathbf{s}) - \mathcal{G}(\mathbf{F}, \mathbf{d})\|_2^2, \quad (10)$$

where $S(\cdot, \cdot)$ denotes cosine similarity. In our experiments, this regularization term significantly enhances the robustness of the Deferred MLP across interpolated and extrapolated viewpoints (see Fig. 4(c)). It effectively mitigates the instability of view-dependent colors under new viewpoints while ensuring rendering quality and 3D consistency.

4.3. Optimization

Our model is trained using a loss function written as a weighted sum:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{rgb}} + \lambda_2 \mathcal{L}_{\text{S3IM}} + \lambda_3 \mathcal{L}_{\text{distortion}} + \lambda_4 \mathcal{L}_{\text{interval}} + \lambda_5 \mathcal{L}_{\text{sparsity}} + \lambda_6 \mathcal{L}_{\text{entropy}} + \lambda_7 \mathcal{L}_{\text{occ}} + \lambda_8 \mathcal{L}_{\text{smooth}}. \quad (11)$$

Here $\mathcal{L}_{\text{occ}}$ and $\mathcal{L}_{\text{smooth}}$ are defined in Eq. (5) and Eq. (10), respectively. $\mathcal{L}_{\text{rgb}}$ is a photometric loss that penalizes the disparity between the rendered images and the ground truth images, using the Charbonnier loss [4] as a robust norm:

$$\mathcal{L}_{\text{rgb}} = \sum\nolimits_{\mathbf{r}\in\mathcal{R}} \sqrt{\|\mathbf{C}(\mathbf{r}) - \hat{\mathbf{C}}(\mathbf{r})\|_2^2 + \epsilon_c}, \tag{12}$$

where $\mathcal{R}$ is the set of training rays, and $\epsilon_c = 10^{-6}$ is a parameter to ensure smoothness. $\mathcal{L}_{\text{S3IM}}$ is the S3IM loss from [54] to enforce structural similarity between the rendered and input images. The interval loss $\mathcal{L}_{\text{interval}}$ and the distortion loss $\mathcal{L}_{\text{distortion}}$ are both adopted from [2] to rationalize the sample point distribution. $\mathcal{L}_{\text{sparsity}}$ is a sparsity loss defined using randomly sampled points in the occupied space to encourage lower opacity:

$$\mathcal{L}_{\text{sparsity}} = \frac{1}{|\mathcal{P}|} \sum\nolimits_{\mathbf{p_i}\in\mathcal{P}} \alpha_i. \tag{13}$$

Additionally, following [19], we randomly sample a number of rays $\mathcal{R}$ from high altitudes to the ground, calculate the opacity $\{\alpha_i\}$ of sample points $\{\mathbf{p_i}\}$ along each ray $\mathbf{r}$, and combine them with the previously defined occupancy values to derive an entropy loss:

$$\mathcal{L}_{\text{entropy}} = -\frac{1}{|\mathcal{R}|} \sum\nolimits_{\mathbf{r}\in\mathcal{R}} \sum\nolimits_{\mathbf{p}_i\in\mathbf{r}} p(\mathbf{p_i}) \log(p(\mathbf{p_i})), \tag{14}$$

where $p(\mathbf{p_i}) = M_i\alpha_i / (\sum_{\mathbf{p}_i\in\mathbf{r}} M_i\alpha_i)$. This encourages the occupancy probability of spatial points to approach either 0 or 1, which ensures consistency between training and real-time rendering.

During training, we initialize the weight for $\mathcal{L}_{\text{occ}}$ to a small value and gradually increase it. In this way, the training first focuses on the reconstruction of the scene, and then gradually compresses the occupied space while maintaining the reconstruction quality. Further details can be found in the supplementary materials.

4.4. Real-Time Rendering

After training, we store the features in the occupied area for subsequent rendering. Existing approaches [15, 36] construct a 3D voxel grid by evaluating volume density or weight, requiring dozens of hours for large-scale, high-resolution scenes. In contrast, we efficiently determine occupancy information based on the high-resolution occupancy plane obtained during training, which is completed in just a few minutes. Subsequently, we store voxel grids as sparse blocks and the deferred MLP as floating-point arrays. For the 2D triplane, we clip along the the z direction, according to the upper and lower bounds of the occupancy plane. To skip empty space efficiently, we employ 2D max pooling to obtain multiple low-resolution occupancy planes. All these features are encoded in PNG format.

After the baking process, we follow MERF [36] for real-time rendering but diverge in ray marching. Unlike the 3D binary grid used to determine the current point's occupancy status, the occupancy plane directly stores the start and end positions of the point's sampling interval in the z direction. As the ray traverses the blank area, it can efficiently reach the next grid point or the starting sampling position of the current grid point through bounding box intersection detection. Experimental results indicate that this sampling strategy is notably faster than previous approaches, especially when overlooking the entire scene.

5. Experiments

5.1. Real-Time Rendering on Oblique Photography

Datasets. We employ two distinct datasets for evaluation: 1) *Matrix City* [22], a synthetic dataset, captured in a grid-style format typical of classic oblique photography; 2) *Campus-Oblique*, a real-world dataset we captured using a surround-style approach. It encompasses three distinct scenes on a university campus. *Scene-1* and *Scene-2* cover an area of about 120,000 square meters and comprise over 1,000 images each. *Scene-3* is even more extensive, covering approximately 600,000 square meters and containing over 7,000 images. For each scene, we use 99% of the images for training and the remaining ones for testing.

Experiment Settings. We conduct comprehensive comparisons with several offline and real-time methods to evaluate the performance of our method. For offline models, we compare with 1) Instant-NGP [32]; 2) NeRFacto [44], an implementation of MipNeRF 360 [2] based on the Nerfstudio framework [44] that additionally integrates multi-level hash encoding; 3) Mega-NeRF [46] with 2×2 partitions; 4) Grid-NeRF [55]. For real-time models, we compare with mesh-based rasterization methods like MobileNeRF [6] and BakedSDF [56], and the splat-based rasterization method 3DGS [18]. Our Oblique-MERF and the baseline MERF [36] are built upon the Nerfstudio framework [44], augmented with the tiny-cuda-nn extension. To ensure fairness, both MERF and Oblique-MERF models follow the same training hyperparameters and architectural designs. We set the triplane resolution R to 2048, the sparse grid resolution L to 512 and the occupancy plane resolution M to 512. Our real-time viewer is implemented as a JavaScript web application, utilizing GLSL for rendering.

Metrics. We evaluate the rendering quality using established metrics: peak signal-to-noise ratio (PSNR), SSIM [49], and LPIPS [58]. Additionally, we use GPU memory usage (VRAM), frames per second (FPS), and on-disk storage (DISK) as metrics for the efficiency of real-time rendering methods. The evaluation is carried out at 1920×1080 resolution on an NVIDIA GTX 1650.

Results. Tab. 2 present the quantitative comparison results, demonstrating that our approach achieves competitive or superior rendering quality. As shown in Fig. 5,

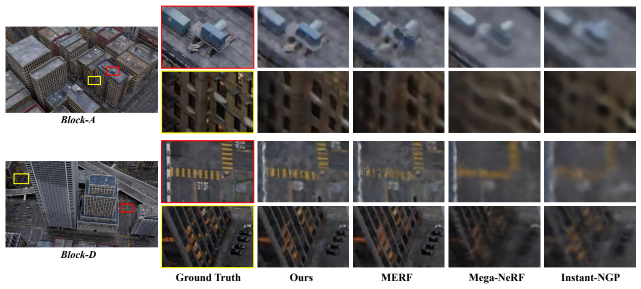


Figure 5. Qualitative comparison between Oblique-MERF and other methods on the *Matrix City* [22] dataset.

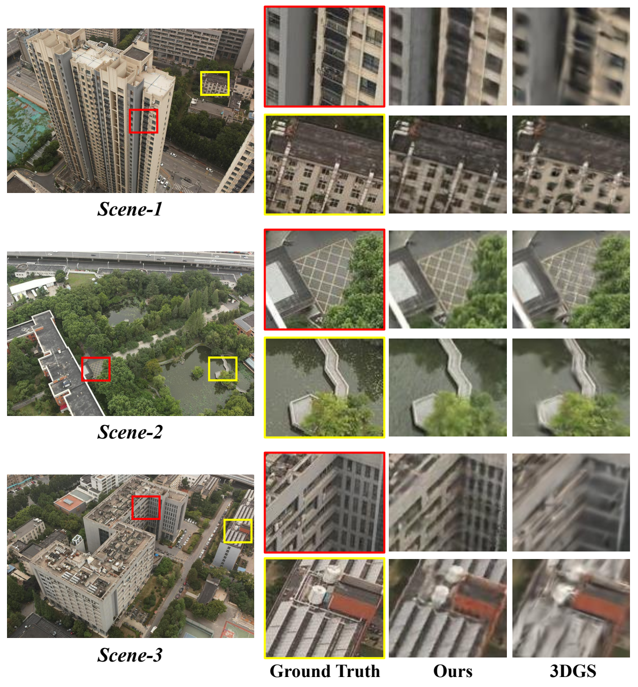


Figure 6. Visual comparison between Oblique-MERF and 3DGS [18] on the *Campus-oblique* datasets.

	Campus-Oblique			*Matrix City* [22]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
InstantNGP [32]	23.08	0.600	0.505	23.10	0.612	0.707
Nerfacto [2]	22.20	0.618	0.335	23.40	0.674	<u>0.444</u>
Mega-NeRF [46]	<u>23.46</u>	0.591	0.466	24.70	0.650	0.600
Grid-NeRF [55]	-	-	-	<u>25.06</u>	<u>0.704</u>	0.512
MobileNeRF [6]	20.84	0.443	0.499	19.02	0.517	0.608
BakedSDF [56]	21.24	0.493	0.573	22.09	0.582	0.652
3DGS [18]	23.08	**0.673**	<u>0.321</u>	-	-	-
MERF [36]	22.84	0.617	0.344	24.50	0.677	0.456
Ours	**23.69**	<u>0.662</u>	**0.298**	**25.30**	**0.706**	**0.406**

Table 2. Quantitative results on the *Campus-Oblique* and *Matrix City* [22] datasets. The best results are bolded, while the second-best results are underlined.

other methods exhibit excessive smoothness in details such as building surfaces and pedestrian crosswalk markings. Leveraging feature grids and regularization of sampling

	VRAM↓ (MB)	DISK↓ (MB)	FPS↑	
			high-altitude	low-altitude
MobileNeRF [6]	1142.0	400.3	**60**	44
BakedSDF [56]	582.0	537.3	<u>54</u>	**60**
3DGS [18]	1356.9	1356.9	14	37
MERF [36]	<u>177.5</u>	<u>92.8</u>	9	37
Ours	**111.9**	**78.3**	37	<u>45</u>

Table 3. The performance for our model and other real-time methods on *Scene-1* of *Campus-Oblique* dataset.

space, our method effectively concentrates sampling points on the scene surfaces, thereby better recovering local geometry and appearance information. While 3DGS [18] provides excellent perceptual quality with high SSIM, its reconstruction efficiency, limited by the number of primitives, rapidly decreases when extended to larger-scale scenes (see the cluttered geometry in *Scene-3* of Fig. 6). In contrast, our method maintains stable rendering quality and does not rely on accurate point cloud initialization.

We evaluate the real-time rendering performance of our Oblique-MERF and other real-time rendering methods in Tab. 3. Despite the inherently lower frame rates of volumetric rendering compared to mesh rasterization, our method excels in disk and VRAM efficiency while delivering exceptional real-time rendering quality. In contrast to 3DGS [18], which requires a large number of explicit Gaussian ellipsoids to fit large-scale scenes, we achieve competitive rendering quality with significantly lower storage requirements. Compared to the baseline MERF [36], our compact occupancy space design significantly reduces storage requirements and consistently improves rendering frame rates across different scales. Further comparisons can be found in the supplementary material.

5.2. Color for Extrapolation Novel Viewpoints

Datasets. We employ a novel real-world dataset, named *Campus-extra*, to validate the effectiveness of our smoothness regularization term. For the training set, we follow the same technique as *Campus-Oblique* to capture 1486 images with a fixed tilt angle to the ground at approximately 60 degrees. For the test set, we simulate extrapolated viewpoints by capturing 140 images with a tilt angle of around 25 degrees. To minimize the impact of unseen scenes, we adopt the NerfBusters [51] protocol to mask test images to exclude unseen regions during training.

Experiment Settings. Our method is compared with a range of MERF variants that improve the continuity of the Deferred MLP in various ways. In "MERF+LipMLP", we replace the Deferred MLP with a Lipschitz MLP and corresponding regularization [25]. In "MERF+Gradient", we penalize the ℓ_2-norm of the specular color's gradient with respect to the viewing direction, namely $\|\frac{\partial \mathcal{G}(\mathbf{F},\mathbf{d})}{\partial \mathbf{d}}\|_2$. In "w/o

Spatial Res.	512^3 PSNR↑	VRAM↓	DISK↓	OR↓	1024^3 PSNR↑	VRAM↓	DISK↓	OR↓	2048^3 PSNR↑	VRAM↓	DISK↓	OR↓	Time↓
MERF [36]	21.83	94.8	65.4	2.54%	23.35	441.0	319.4	1.97%	24.18	2478.0	1679.1	1.34%	$\sim 6h$
Ours(vanilla baking)	**22.84**	67.6	50.3	1.57%	**24.54**	334.4	272.6	**1.15%**	**25.24**	1633.4	1338.2	**0.73%**	$\sim 6h$
Ours		**61.3**	**47.0**	**1.49%**		**322.5**	**264.9**	1.27%		**1605.0**	**1273.1**	0.89%	~ **8min**

Table 4. Ablations on baking across varied spatial resolutions. VRAM and DISK capacity is denoted in megabytes (MB).

	Training views PSNR↑	SSIM↑	LPIPS↓	Test views PSNR↑	SSIM↑	LPIPS↓
MERF [36]	25.93	0.770	0.189	22.14	0.710	0.213
+LipMLP [25]	25.86	0.768	0.188	22.42	0.729	0.201
+Gradient	25.97	0.772	0.187	**22.66**	0.732	<u>0.199</u>
w/o direction	25.94	0.771	0.187	22.32	**0.746**	0.203
w/o high freq.	<u>25.98</u>	<u>0.774</u>	<u>0.184</u>	21.23	0.724	0.203
Ours	**26.00**	**0.776**	**0.183**	**22.66**	<u>0.744</u>	**0.196**

Table 5. Effectiveness of the smoothness regularization.

	PSNR↑	SSIM↑	LPIPS↓	VRAM(MB)↓
W/o Warmup Train	21.89	0.632	0.297	106.0
Ours	24.33	0.716	0.222	111.9

Table 6. Ablations on training strategy.

direction”, we remove view dependence entirely. In “w/o high freq.”, we remove the high-frequency components for the positional encoding of $\mathbf{d}$.

Results. Tab. 5 showcases quantitative comparison under both interpolated and extrapolated viewpoints. We find that merely ensuring the network’s Lipschitz continuity slightly enhances the test view performance but can negatively impact the training views. This is potentially because it enforces smoothness on all network inputs, including diffuse and specular features, which is redundant and hinders the optimization. Applying regularization to the specular color’s gradient is effective, but it may result in undesirable floaters in some views. Eliminating directional or high-frequency encoded inputs would diminish the model’s capacity to capture color variations with changing viewpoints. Compared to these approaches, our method consistently excels on both the training and test sets, achieving an approximate improvement of 0.5 dB PSNR over the baseline. Visual comparison will be presented in the supplementary materials.

5.3. Ablation Study

Spatial Resolution. Tab. 4 compares our method with the baseline method in rendering quality, memory usage, occupancy ratio (OR), and time consumed in the baking pipeline on *Scene-1* of the *Campus-Oblique* dataset. To more clearly illustrate the compactness of the occupied space optimized by our sampling strategy, we exclusively employ sparse feature grids at resolutions from 512 to 2048, omitting the use of triplane. After training, we extract a 3D occupancy grid, contrasting it with the one obtained from MERF’s baking process. Our approach consistently yields a lower OR (approximately 60% of MERF’s), enhancing memory efficiency and rendering quality. It can also be observed that the vanilla baking strategy requires several hours for post-processing, with the time increasing proportionally to the number and resolution of training images, whereas our method produces consistent texture sets within minutes.

Training Strategy. To validate the effectiveness of warm-up training in Sec. 4.3, we perform an ablation by compressing the occupancy plane from the beginning using a fixed weighting factor of $\lambda_7 = 0.0001$ for $\mathcal{L}_{\text{occ}}$. Despite a slight memory advantage, the absence of warm-up leads to a severe drop in rendering quality, as shown in Tab. 6. This is because applying regularization to the sampling space before the model has learned the approximate geometry can result in excessive, irrecoverable compression.

6. Conclusion and Future Work

In this paper, we introduce Oblique-MERF, a compact and robust model optimized for real-time NeRFs in large-scale scenes like oblique photography. Our core contributions include an adaptive explicit occupancy plane aware of photometric loss and a smoothness regularization designed to enhance robustness under extrapolated viewpoints. Compared to existing real-time rendering techniques, Oblique-MERF delivers superior rendering quality, lower memory usage, and higher frame rates. However, scalability remains a challenge. We plan to adopt a divide-and-conquer strategy to extend our representation to larger scales. Moreover, integrating our method with physically-based rendering to accurately simulate changes in specular color represents a future research direction.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (No.62441224, No.62272433), the Fundamental Research Funds for the Central Universities (No.WK0010000090), and USTC Fellowship (No.S19582024). 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] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *2009 IEEE 12th International Conference on Computer Vision*, pages 72–79, 2009. 2

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 1, 3, 6, 7

[3] Aljaž Božič, Denis Gladkov, Luke Doukakis, and Christoph Lassner. Neural assets: Volumetric object capture and rendering for interactive environments. *arXiv preprint arXiv:2212.06125*, 2022. 2

[4] Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing*, 6(2):298–311, 1997. 6

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[6] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *CVPR*, 2023. 1, 2, 6, 7

[7] David Dadon, Ohad Fried, and Yacov Hel-Or. Ddnerf: Depth distribution neural radiance fields. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 755–763, 2023. 3

[8] H. Drucker and Y. Le Cun. Double backpropagation increasing generalization performance. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, pages 145–150 vol.2, 1991. 3, 5

[9] Jiemin Fang, Lingxi Xie, Xinggang Wang, Xiaopeng Zhang, Wenyu Liu, and Qi Tian. Neusample: Neural sample field for efficient view synthesis. *arXiv:2111.15552*, 2021. 3

[10] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building rome on a cloudless day. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, page 368–381, Berlin, Heidelberg, 2010. Springer-Verlag. 2

[11] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010. 2

[12] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1434–1441, 2010. 2

[13] Jiaming Gu, Minchao Jiang, Hongsheng Li, Xiaoyuan Lu, Guangming Zhu, Syed Afaq Ali Shah, Liang Zhang, and Mohammed Bennamoun. Ue4-nerf:neural radiance field for real-time rendering of large-scale scene. In *Advances in Neural Information Processing Systems*, pages 59124–59136. Curran Associates, Inc., 2023. 2

[14] Yuan-Chen Guo, Yan-Pei Cao, Chen Wang, Yu He, Ying Shan, and Song-Hai Zhang. Vmesh: Hybrid volume-mesh representation for efficient view synthesis. In *SIGGRAPH Asia 2023 Conference Papers*, New York, NY, USA, 2023. Association for Computing Machinery. 2

[15] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *ICCV*, 2021. 1, 2, 3, 6

[16] Judy Hoffman, Daniel A Roberts, and Sho Yaida. Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729*, 5(6):7, 2019. 5

[17] Hailin Jin, Stefano Soatto, and Anthony J. Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. *International Journal of Computer Vision*, 63:175–189, 2005. 2

[18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 3, 6, 7

[19] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *CVPR*, 2022. 6

[20] Andreas Kurz, Thomas Neff, Zhaoyang Lv, Michael Zollhöfer, and Markus Steinberger. Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. In *European Conference on Computer Vision*, pages 254–270. Springer, 2022. 2

[21] Chaojian Li, Bichen Wu, Peter Vajda, and Yingyan Lin. Mixrt: Mixed neural representations for real-time nerf rendering. In *2024 International Conference on 3D Vision (3DV)*, pages 1115–1124. IEEE, 2024. 2

[22] Yixuan Li, Lihan Jiang, Linning Xu, Yuanbo Xiangli, Zhenzhi Wang, Dahua Lin, and Bo Dai. Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3205–3215, 2023. 6, 7

[23] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia Conference Proceedings*, 2022. 3

[24] Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, and Wenming Yang. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *CVPR*, 2024. 3

[25] Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. Learning smooth neural functions via lipschitz regularization. In *SIGGRAPH '22: Special Interest Group on Computer Graphics and Interactive Techniques Conference, Vancouver, BC, Canada, August 7 - 11, 2022*, pages 31:1–31:13, 2022. 3, 5, 7, 8

[26] Jeffrey Yunfan Liu, Yun Chen, Ze Yang, Jingkang Wang, Sivabalan Manivasagam, and Raquel Urtasun. Real-time neural rasterization for large scenes. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8382–8393, 2023. 2

[27] Yang Liu, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. Citygaussian: Real-time

high-quality large-scale scene rendering with gaussians. In *European Conference on Computer Vision*, pages 265–282. Springer, 2025. 3

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

[29] Zhenxing Mi and Dan Xu. Switch-nerf: Learning scene decomposition with mixture of experts for large-scale neural radiance fields. In *International Conference on Learning Representations (ICLR)*, 2023. 1, 2

[30] 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 *The European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3

[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *CVPR*, pages 9078–9086. Computer Vision Foundation / IEEE, 2019. 3, 5

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 1, 2, 3, 6, 7

[33] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021. 2, 3

[34] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 3

[35] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, pages 14335–14345, 2021. 1, 2

[36] 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, 3, 6, 7, 8

[37] Christian Reiser, Stephan Garbin, Pratul P. Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan T. Barron, Peter Hedman, and Andreas Geiger. Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. *SIGGRAPH*, 2024. 2

[38] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 1, 2, 3

[39] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016. 2

[40] Kaiwen Song, Xiaoyi Zeng, Chenqu Ren, and Juyong Zhang. City-on-web: Real-time neural rendering of large-scale scenes on the web. In *European Conference on Computer Vision*, pages 385–402. Springer, 2025. 2

[41] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1161–1170. ACM, 2019. 2

[42] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 1, 2

[43] 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 *CVPR*, pages 8248–8258, 2022. 1, 2

[44] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 6

[45] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17739–17749, 2023. 1, 2

[46] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *CVPR*, pages 12922–12931, 2022. 2, 6, 7

[47] Haithem Turki, Vasu Agrawal, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Deva Ramanan, Michael Zollhöfer, and Christian Richardt. Hybridnerf: Efficient neural rendering via adaptive volumetric surfaces. In *Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 3

[48] Ziyu Wan, Christian Richardt, Aljaž Božič, Chao Li, Vijay Rengarajan, Seonghyeon Nam, Xiaoyu Xiang, Tuotuo Li, Bo Zhu, Rakesh Ranjan, and Jing Liao. Learning neural duplex radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8307–8316, 2023. 2

[49] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 6

[50] Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Muller, and Zan Gojcic. Adaptive shells for efficient neural radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42:1 – 15, 2023. 1, 3

[51] Frederik Warburg, Ethan Weber, Matthew Tancik, Aleksander Holynski, and Angjoo Kanazawa. Nerfbusters: Removing ghostly artifacts from casually captured nerfs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18120–18130, 2023. 7

[52] Changchang Wu. Towards linear-time incremental structure

from motion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 127–134, 2013. 2

[53] 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 *The European Conference on Computer Vision (ECCV)*, 2022. 2

[54] Zeke Xie, Xindi Yang, Yujie Yang, Qi Sun, Yixiang Jiang, Haoran Wang, Yunfeng Cai, and Mingming Sun. S3im: Stochastic structural similarity and its unreasonable effectiveness for neural fields. In *ICCV*, 2023. 6

[55] Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. In *CVPR*, pages 8296–8306. IEEE, 2023. 2, 6, 7

[56] 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. In *ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023*, pages 46:1–46:9. ACM, 2023. 1, 2, 6, 7

[57] 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. 1, 2, 3

[58] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

[59] Yuqi Zhang, Guanying Chen, and Shuguang Cui. Efficient large-scale scene representation with a hybrid of high-resolution grid and plane features. *Pattern Recognition*, 158: 111001, 2025. 3

[60] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538, 2001. 3