HS-Surf: A Novel High-Frequency Surface Shell Radiance Field to Improve Large-Scale Scene Rendering

Jiongming Qin
jiongming@whu.edu.cn
Wuhan University
School of Computer Science
Wuhan, Hubei, China

Fei Luo*
luofei@whu.edu.cn
Wuhan University
School of Computer Science
Wuhan, Hubei, China

Tuo Cao
maplect@whu.edu.cn
Wuhan University
School of Computer Science
Wuhan, Hubei, China

Wenju Xu
xuwenju123@gmail.com
Amazon, Palo Alto, CA 94301
Sunnyvale, USA

Chunxia Xiao*
cxxiao@whu.edu.cn
Wuhan University
School of Computer Science
Wuhan, Hubei, China

Abstract

Previous neural radiance fields often struggle to preserve high-frequency textures in urban and aerial large-scale scenes due to insufficient model capacity on the scene surface. This is attributed to their sampling locations or grid vertices falling in empty areas. Additionally, most models do not consider the drastic changes in distances. To address these issues, we propose a novel high-frequency surface shell radiance field, which uses depth-guided information to create a shell enveloping the scene surface under the current view, and then samples conic frustums on this shell to render high-frequency textures. Specifically, our method comprises three parts. Initially, we propose a strategy to fuse voxel grids and information of distance scales to generate a coarse scene at different distance scales. Subsequently, we construct a shell based on the depth information to carry out compensation to incorporate texture details not captured by voxels. Finally, the smooth and denoise post-processing further improves the rendering quality. Substantial scene experiments and ablation experiments demonstrate that our method achieves the obvious improvement of high-frequency textures at different distance scales and outperforms the state-of-the-art methods.

CCS Concepts

• **Computing methodologies → Image-based rendering**.

Keywords

large-scale scenes, high-frequency shell, surface rendering, high-frequency textures

*Corresponding authors

MM '24, October 28-November 1, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0686-8/24/10
https://doi.org/10.1145/3664647.3681436

ACM Reference Format:
Jiongming Qin, Fei Luo, Tuo Cao, Wenju Xu, and Chunxia Xiao. 2024. HS-Surf: A Novel High-Frequency Surface Shell Radiance Field to Improve Large-Scale Scene Rendering. In *Proceedings of the 32nd ACM International Conference on Multimedia (MM '24), October 28-November 1, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3664647.3681436

1 Introduction

Rendering urban and aerial large-scale scenes has many applications like AR/VR and digital navigation. Previous neural radiance fields [1–3] (NeRFs) have tried improving the rendering quality, which can be categorized into two streams. The first [4–7] divides the scene or camera poses into multiple sub-regions or groups, and each unit is represented by a NeRF. This increases the number of NeRF modules, indirectly enhancing model capacity on the surface. However, those NeRFs sample along the entire ray, including empty spaces. The second [8, 9] reconstructs a coarse scene or density field to guide subsequent sampling, concentrating on high-density areas near the surface. Nevertheless, some samples inevitably fall into empty regions, such as the sampling interval ends.

Multi-layer perceptron (MLP) based NeRFs [1, 2] frequently sample along rays, resulting in many points falling into empty regions. Therefore, much model capacity for storing geometry and appearance is used to represent these meaningless spaces rather than the scene surface. Additionally, voxel [10] and grid-based [11–13] NeRFs only have a few vertices to be sampled near the target surface, leading to an upper bound on the model capacity allocated to the surface. These inefficient samplings result in a significant waste of model capacity, lacking enough model capacity on the scene surface to render high-frequency textures. Moreover, most NeRFs [4, 5, 7, 8] have not considered the drastic changes in distances between the camera and the scene surface, which is prone to generate blurry rendering results at various distances.

To overcome the inefficient sampling and enhance the quality of high-frequency textures, we propose a novel high-frequency surface shell radiance field (HS-Surf) to efficiently increase model capacity on scene surfaces. It constructs a shell enveloping the scene surface based on the current view's scene depth. As shown in Figure 1(a), the shell's width increases with the depth to represent the distance, and conic frustums sampled on the shell are used to render high-frequency textures at different distance scales. We call this shell as High-Frequency Shell (HS). HS confines the sampling and rendering to scene surfaces, greatly enhancing the utilization of model capacity. Additionally, to model geometry and appearance at different distances, we propose a feature fusion strategy to embed conic frustums representing distances into voxel grids.

Our HS-Surf consists of three stages: initialization, compensation, and post-processing. The initialization uses hash-based voxel grids to generate coarse geometry and appearance. To model distances with drastic changes in large-scale scenes, the proposed feature fusion strategy embeds positional encoding of conic frustums into voxel grids. The compensation generates high-frequency textures at different distance scales. It first augments the coarse scene depth under the current view and constructs an HS based on the augmented depth. Conic frustums are then exclusively sampled on the shell to generate high-frequency textures lost in the coarse appearance. The post-processing uses a convolutional neural network (CNN) to smooth and denoise the rendering results to achieve a better visual effect.

The experimental results indicate that HS-Surf greatly improves high-frequency textures (see Figure 1(b)) and achieves state-of-the-art rendering quality. Additionally, we observe that our rendering speed is 2× to 4× faster than previous NeRFs, achieving double improvement of the rendering effect and computation efficiency. Our contributions can be summarized as follows:

- Our proposed high-frequency shell overcomes the sampling inefficiency of previous methods, efficiently increasing model capacity on scene surfaces to render high-frequency textures.
- The proposed feature fusion strategy embeds conic frustums into voxels to represent the distance scales, enabling the voxel to model the scene at various distances.

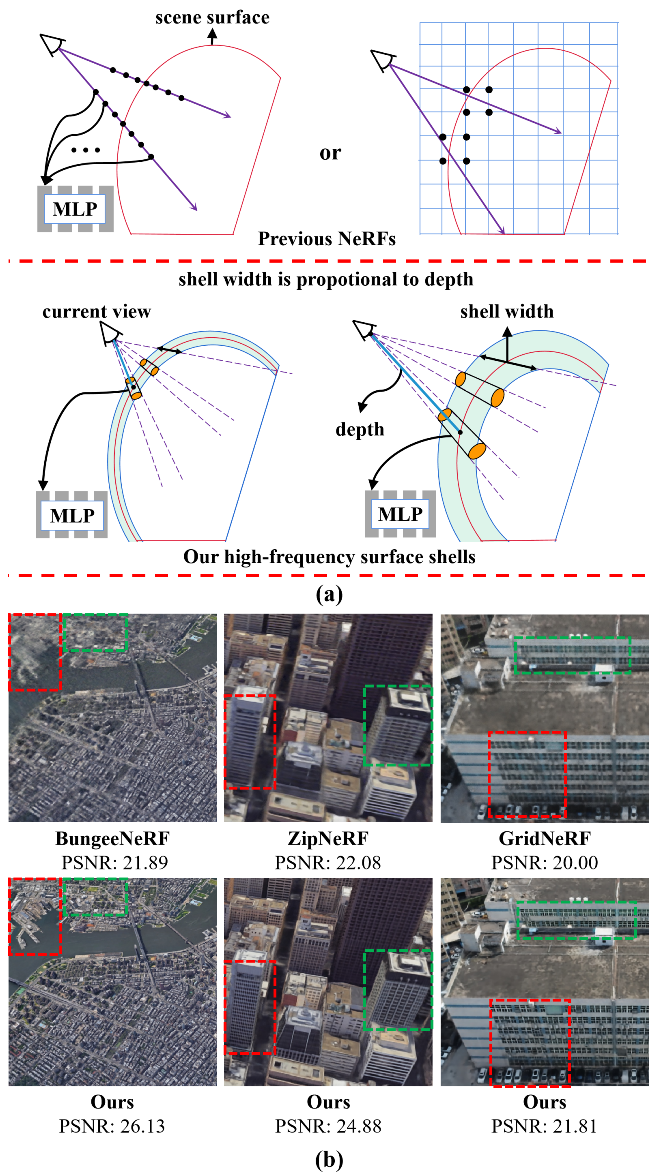

Figure 1: (a) Top: Previous NeRFs sample along the entire ray or within the grid, which usually falls in the contentless areas or is limited by the grid resolution. It leads to rendering blur and model capacity wasting. Bottom: HS-Surf constructs a shell on the scene surface based on the current view's depth, fully using model capacity on texture-rich areas to improve rendering quality. (b) Our method could render more high-frequency details on the scene surface to improve the clarity of textures compared to the SOTA NeRFs.

2 Related Work

2.1 Neural Radiance Fields

NeRF [1, 3] employs MLPs to model volume density and color of spatial points. A lot of NeRF variants [14–24] are proposed to enhance rendering fidelity, rectify camera poses, and accelerate rendering. There are also models [25, 26] designed for unbounded scenes. To speed up rendering, some methods replace MLPs in NeRF with voxel [10, 27] or plane grids [11–13], but these increase GPU memory consumption. Recent methods [28–30] map the voxel vertices into smaller hash tables, which achieve more compact representations. Adaptive shells [31] focus on small-scale scenes, creating a fixed shell based on the geometry generated by NeuS [3]. In contrast, HS-Surf creates a shell for each viewpoint based on the scene's depth. This dynamic shell can better represent different distance scales. Additionally, our compensation enhances the depth maps to obtain more reliable shells.

MipNeRF [2] samples conic frustums along rays, and uses integrated positional encoding (IPE) of the frustums to represent the distance scales. However, the local continuous space of the frustum is incompatible with interpolation operations in grids. ZipNeRF [9] simulates the local space by sampling six discrete points within a conic frustum, while our feature fusion embeds the continuous positional encoding of the frustum into the voxel grid. Therefore, our feature fusion is theoretically superior to the discrete ZipNeRF, and subsequent experiments also validate this conclusion. 3D Gaussian [32] is another recent method with different mechanisms for scene representation, which involves fitting a large number of ellipsoids to approximate the target scene and render novel views.

2.2 Large-scale Scene Rendering

Some traditional methods [33–40] have been proposed to reconstruct the large-scale scenes. Their working pipeline usually needs three stages: keypoint detection, feature matching, and bundle adjustment. Keypoint detection [41–43] looks for unique and easily identifiable regions in images and constructs corresponding feature descriptions. Then, the features of key points are matched to compute camera poses and locations of 3D points. Finally, the camera poses and 3D points are jointly optimized by bundle adjustment [44, 45]. These methods can roughly reconstruct the target scene and synthesize novel views [46, 47], but the results often contain artifacts and holes.

Recently, a lot of NeRFs are used for rendering large-scale scenes. BungeeNeRF [6] divides camera poses based on their heights, and recovers texture details at lower heights by adding NeRF modules. BlockNeRF [4] and MegaNeRF [5] geometrically divide the target scene into multiple sub-regions, with each region represented by a separate NeRF. VastGaussian [48], based on different principles, also uses a similar approach to render large-scale scenes. By partitioning the camera poses or the scene, BungeeNeRF, BlockNeRF, and MegaNeRF reduce the target regions for each sub-NeRF, increasing the model capacity on scene surfaces. However, their performance improvements are limited because NeRF still needs to sample the entire ray, including empty regions. The key issue of low utilization and allocation of model capacity on the scene surface remains unresolved. URF [49] leverages depth data of radar as auxiliary information to reconstruct street-level scenes. SwitchNeRF [7] classifies sampled points on rays through a gating network, thereby obtaining a learnable region partition.

GridNeRF [8] uses planar grids (grid branch) to construct a coarse scene, guiding the NeRF branch to add sampling points in high-density regions near the scene surface. However, this approach still leads to some points falling into empty regions, particularly at the ends of the sampling interval. Moreover, the points in the first sampling operation are distributed across the entire ray, which increases the consumption of model capacity by empty regions. As a result, GridNeRF still lacks sufficient model capacity on the scene surface to render high-frequency texture details. In contrast, HS-Surf has different motivation and working mechanism, which constructs high-frequency shells on the scene surface. These shells confine the computation of MLPs to the surface while excluding the surrounding empty regions, greatly enhancing the utilization of model capacity. Thus, our method has more power to render high-frequency textures.

3 Method

The overview of HS-Surf is illustrated in Figure 2(a), consisting of three stages: initialization, compensation, and post-processing. The initialization reconstructs the coarse geometry and appearance using hash-based voxel grids. IPE encoding of conic frustums and grid features are fused to model the target scene at different distance scales. The compensation first augments the coarse depth map under the current view to ensure a more accurate surface geometry. Subsequently, it constructs an HS based on the depth map, and samples conic frustums on HS to compensate for the lost high-frequency textures in the coarse appearance. This step is very important because the texture is attached to the geometry. Thus, good depth and its subsequent product of HS can confine sampling to effective texture areas. In the post-processing, a lightweight CNN is employed to smooth and denoise the rendering results of the compensation.

3.1 Initialization of Geometry and Appearance

Hash-based voxel grid is suitable for large-scale scenes as uniformly distributed vertices ensure that model capacity is reasonably allocated across the entire scene to generate a coarse radiance field. Additionally, hash tables are very useful for reducing GPU occupancy for high-resolution voxel grids.

Due to the drastic changes of distance scales in large-scale scenes, sampling points along rays can easily result in blurry rendering results at different distances. Inspired by MipNeRF [2], we sample conic frustums within the target scene and use IPE encoding to model distance scales. Specifically, a frustum is approximated by mean and covariance, which are then fed into IPE to generate the corresponding encoding [2]. Like MipNeRF, the radius of conic containing the frustums at image plane $o + dir$ is set to $\dot{r}$, and $\dot{r}$ is the width of the pixel in world coordinates scaled by $2/\sqrt{12}$.

To leverage the advantages of both the hash-based voxel grids and conic frustums, we propose a feature fusion strategy to reconstruct the coarse scene at different distance scales. As shown in Figure 2, the center coordinates x of the frustums are used to query features in the density grids $DG(\cdot)$ and the color grids $CG(\cdot)$. The IPE encoding E_i of the frustums is fed into a two-layer MLP $M_i(\cdot)$, which is then fused with density features and color features to compute density σ and color rgb as follows:

$$\begin{aligned} \sigma &= M_\sigma(concat(DG(x), M_i(E_i))) \\ rgb &= M_c(concat(CG(x), M_i(E_i)), dir), \end{aligned} \tag{1}$$

where $M_\sigma(\cdot)$ and $M_c(\cdot)$ are small MLPs for generating density and color, and dir is the ray direction. Volume rendering subsequently generates scene depth and rendering results.

The initialization stage can be described in two steps. The first step samples conic frustums along rays to render pixel colors C_1. The second step conducts finer sampling based on existing sample densities to obtain pixel depths d_c and colors C_2. Both steps utilize the same voxel grids and hash tables. To ensure that the initialization results contain less artifacts such as "floaters" and "background collapse", we add an interval-based regularization loss L_{dist} in the fine step, which is proposed in MipNeRF360 [25].

$$\begin{aligned} L_{dist}(s, w) = \sum_{i,j} \omega_i \omega_j \mid \frac{s_i + s_{i+1}}{2} - \frac{s_j + s_{j+1}}{2} \mid \\ + \frac{1}{3} \sum_i \omega_i^2 (s_{i+1} - s_i), \end{aligned} \tag{2}$$

where s and w represent the (normalized) ray distances and weights of conic frustums in volume rendering, respectively. The role of L_{dist} is to concentrate the frustums with high density into a narrower region. Then, the loss of the initialization stage is as follows:

$$L_{init} = \lambda_1 \parallel C_1 - C_{gt} \parallel_2^2 + \parallel C_2 - C_{gt} \parallel_2^2 + \lambda_2 L_{dist}, \tag{3}$$

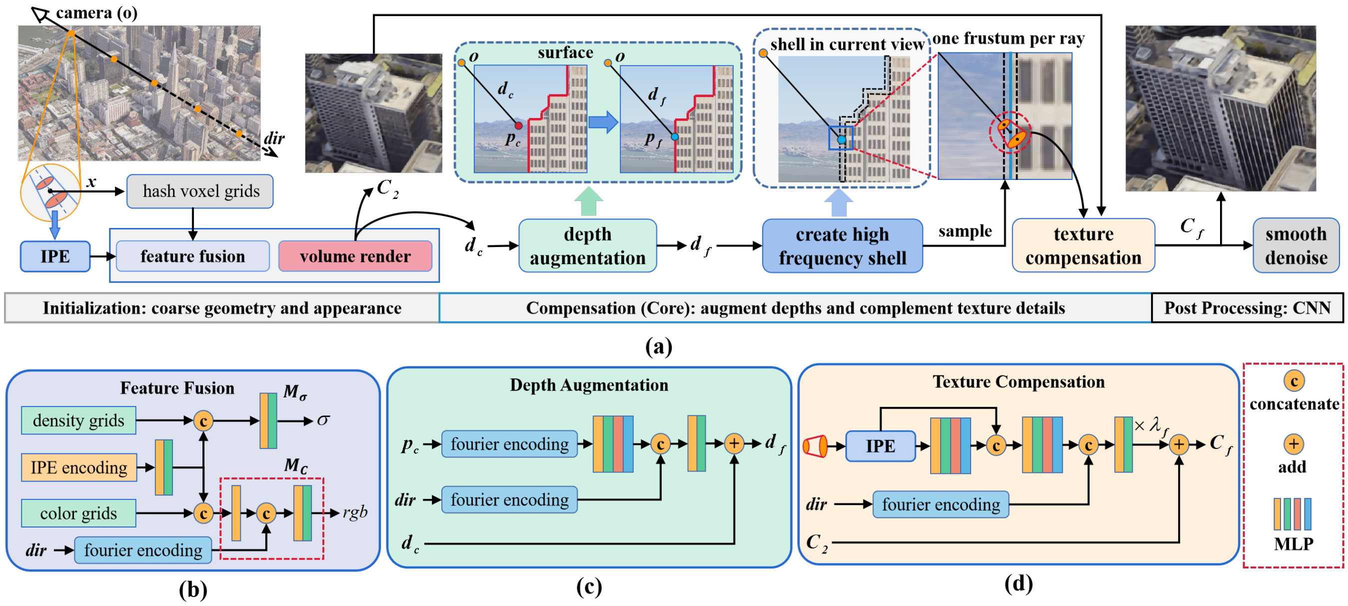

Figure 2: Overview of HS-Surf. (a) is the method pipeline. The initialization employs the feature fusion to embed IPE encoding of conic frustum into the grid feature, generating the coarse geometry d_c and texture C_2 at different distance scales. The compensation stage first augments the depth d_c to obtain a more accurate depth d_f, and constructs a high-frequency shell of the current view based on d_f. Then, for each ray, a conic frustum sampled on the shell is fed into the texture compensation to render the high-frequency textures lost in C_2. Finally, the post-processing smooths and denoises the output of the compensation stage to achieve a better visual effect. (b), (c) and (d) are structures of feature fusion, depth augmentation and texture compensation.

where C_{gt} is the real pixel color. λ_1 and λ_2 are set to 0.1 and 0.001 in all experiments.

3.2 Compensation of Depth and Texture

The initialization includes lots of voxel vertices far from the scene surface, leading to the insufficient model capacity on the surface and the loss of high-frequency textures. To recover the lost textures, we construct a high-frequency shell for each view to efficiently increase the capacity on the surface. The details are as follows:

Depth Augmentation. The limited model capacity on the surface leads to coarse geometry with noises and holes. Therefore, before constructing the high-frequency shell, we propose a depth augmentation module to improve the depth map under the current view. As shown in Figures 2(a) and 2(c), the coarse depth d_c is utilized to compute the coordinate p_c of the scene surface. The depth augmentation then employs a four-layer MLP $D(\cdot)$ to predict the distance from p_c to surface along the ray direction dir. The output of $D(\cdot)$ is added with d_c to obtain a more accurate depth d_f:

$$d_f = D(\gamma(p_c), \gamma(dir)) + d_c, \tag{4}$$

where γ represents the Fourier encoding [1]. More accurate depths and positions of the scene surface guarantee fewer errors in the inputs of subsequent modules.

We optimize the parameters of depth augmentation using both depth loss and subsequent rendering loss L_{render}. The depth loss ensures that the module preserves the basic geometry and structure of the scene, and the rendering loss L_{render} is used to refine the scene depth for more accurate surface representation. Without depth ground truth, we take an augmentation manner to improve depth. Like the pyramid image processing, we take a down-sample on the coarse depth d_c to get a low-resolution depth map, which could filter out certain noises while preserving the major depth information. The low-resolution depth map is obtained by a mask W, and is then compared with the output d_f of depth augmentation to compute the depth loss L_{depth}. Specifically, we downsample d_c by a factor of 3, where the mask W selects pixels with coordinates (x, y) can be divided by 3:

$$\begin{aligned} &W = (x\%3 == 0) \ and \ (y\%3 == 0) \\ &L_{depth} = W \cdot \parallel d_f - d_c \parallel_2^2 . \end{aligned} \tag{5}$$

High-frequency Shell. After obtaining accurate scene depth d_f under the current view, we need to construct a high-frequency shell on the surface based on d_f. As shown in Figure 1(a), the shell's width t_{range} along the ray determines the enclosed space on the ray $\{x|x = o + t \times dir, t \in [d_f - 0.5 \times t_{range}, d_f + 0.5 \times t_{range}]\}$. An infinite cone is constructed based on camera position, ray direction and pixel. Then, we truncate the interval $[d_f - 0.5 \times t_{range}, d_f + 0.5 \times t_{range}]$ on conic axis to obtain a conic frustum Δ.

As the depth increases, the high-frequency textures on the surface should become blurred or even disappear. Therefore, the width of Δ needs to increase with depth to suppress the high-frequency components in IPE encoding. The relationship between shell's width t_{range} and depth d_f follows a linear function. First, we calculate the normalized width r:

$$r = \lambda_r \cdot (k \cdot \frac{d_f - near}{far - near} + b), \tag{6}$$

where $\lambda_r = 0.1$ is a scaling factor to stabilize the model. $Near$ and far are the distances traveled along the ray from camera to enter and exit the target scene. $k \geq 0$ and $b \geq 0$ are estimated by a four-layer MLP, and the input of MLP is the concatenation of $near$ and far. The length t_{range} of the shell in the world coordinate system is as follows:

$$t_{range} = max(min(r, \frac{1}{50}), \frac{1}{2000}) \cdot (far - near). \tag{7}$$

The normalized length r needs to be clipped to $[1/2000, 1/50]$ for more stable results.

Texture Compensation. After sampling a conic frustum Δ for each ray on the high-frequency shell, we utilize continuous MLPs to complete the high-frequency textures lost in the voxel grids during initialization. As illustrated in Figures 2(a) and 2(d), the frustum Δ is located at the surface coordinate p_f, and its width equals to the shell's width along the ray. IPE encoding of Δ and ray direction dir are then fed into an eight-layer MLP $F(\cdot)$ to compute the high-frequency texture details missing in the voxel grids. The generated details are then added with the coarse rendering C_2 to obtain the complete appearance C_f as follows:

$$C_f = \lambda_f \cdot F(IPE(\Delta), \gamma(dir)) + C_2, \tag{8}$$

where γ is the Fourier encoding [1]. $\lambda_f = 0.2$ is a scale factor to stabilize the model. The output of $F(\cdot)$ is the texture residual instead of the complete rendering result. The reason is that this method can reduce the learning burden of MLPs and focus the attention of model on the generation of texture details.

Since the scope of inputs in the texture compensation is confined to the scene surface, the model capacity of MLP is dedicated to rendering textures on the target surface. This significantly improves both the capacity utilization on surface and the ability to render high-frequency details. The rendering loss L_{render} is as follows:

$$L_{render} = \lambda_3 \parallel C_f - C_{gt} \parallel_1 + \parallel C_f - C_{gt} \parallel_2^2, \tag{9}$$

where C_{gt} is the real pixel color, and λ_3 is set to 0.1. The loss function $L_{texture}$ of the compensation stage includes rendering loss L_{render} and depth loss L_{depth}:

$$L_{texture} = L_{render} + L_{depth}. \tag{10}$$

3.3 Post-Processing of Smooth and Denoise

As the initialization and compensation stages calculate each pixel individually, the generated results may contain noises and are not continuously smooth. Therefore, a lightweight CNN-based post-processing is constructed to deal with them. The network contains two residual blocks to adjust the features of the original image C_f. Sigmoid activation is used in the final layer of convolution, which limits the output range to $[0, 1]$, whose details are provided in the supplementary material. We opt for a CNN due to the necessity of incorporating correlations between neighboring pixels in the smoothing and denoising processes. Convolution kernels offer a natural way to introduce such information. The loss function L_{img} of the post-processing stage is as follows:

$$L_{img} = \lambda_4 \parallel C - C_{gt} \parallel_1 + \parallel C - C_{gt} \parallel_2^2, \tag{11}$$

where C and C_{gt} are the predicted and real colors, and $\lambda_4 = 0.1$.

3.4 Scene Division and Details

The compensation stage can further enhance the quality of high-frequency textures by partitioning the target scene. Assuming the scene is uniformly divided into N sub-regions, and each contains a depth augmentation and a texture compensation, with corresponding MLPs represented as $D_i(\cdot)$ and $F_i(\cdot)$. The outputs of all sub-regions are combined into the final results. Therefore, the output d_f of depth augmentation in Equation (4) is modified as follows:

$$d_f = \frac{\sum_{i=1}^{N} M_i \cdot (D_i(\gamma(p_c), \gamma(dir)) + d_c)}{\sum_{i=1}^{N} M_i}, \tag{12}$$

where $M_i = 1$ indicates that p_c is located in the i-th region, otherwise $M_i = 0$. The output C_f of texture compensation in Equation (8) is modified as follows:

$$C_f = \frac{\sum_{i=1}^{N} M_i \cdot (\lambda_f \cdot F_i(IPE(\Delta), \gamma(dir)) + C_2)}{\sum_{i=1}^{N} M_i}, \tag{13}$$

where $M_i = 1$ indicates that p_f is located in the i-th region, otherwise $M_i = 0$. The increment in the number of MLPs leads to an augmentation in model capacity on the scene surface.

In Figure 2, the depth d_c generated by the initialization needs to go through a gradient stop. Otherwise, the depth augmentation performance may decrease. In the initialization, the first coarse and second fine step samples 64 and 128 inters along a ray, respectively. The minimum resolution of the voxel grids is 256^3, and the maximum resolution is 8192^3 after 15 increments. The size of hash table is $2^{21} \times 4$ or $2^{22} \times 4$, and the hidden nodes of MLPs are 128. For the compensation stage, the hidden nodes of MLPs in the depth augmentation, estimation of high-frequency shell, and texture compensation are set to 256, 64, and 512. The channel of CNN in the post-processing is set to 32.

The training of HS-Surf consists of two stages. The first stage involves joint training of the initialization and compensation. The second stage only trains the post-processing. Their losses are as follows:

$$L_{stage1} = L_{init} + L_{texture} \tag{14}$$

$$L_{stage2} = L_{img}. \tag{15}$$

The learning rate is $1e-4$ for both stages and decays exponentially to $1e-5$ during training. More details can be found in the supplementary material.

4 Experiments and results

4.1 Experiment Setup

HS-Surf is compared with the previous state-of-the-art NeRFs, including BungeeNeRF [6], MegaNeRF [5], and GridNeRF [8]. MipNeRF [2] and ZipNeRF [9] are also selected, because they are basic neural rendering methods and can model different distance scales in large-scale scenes. All models are implemented by Python and

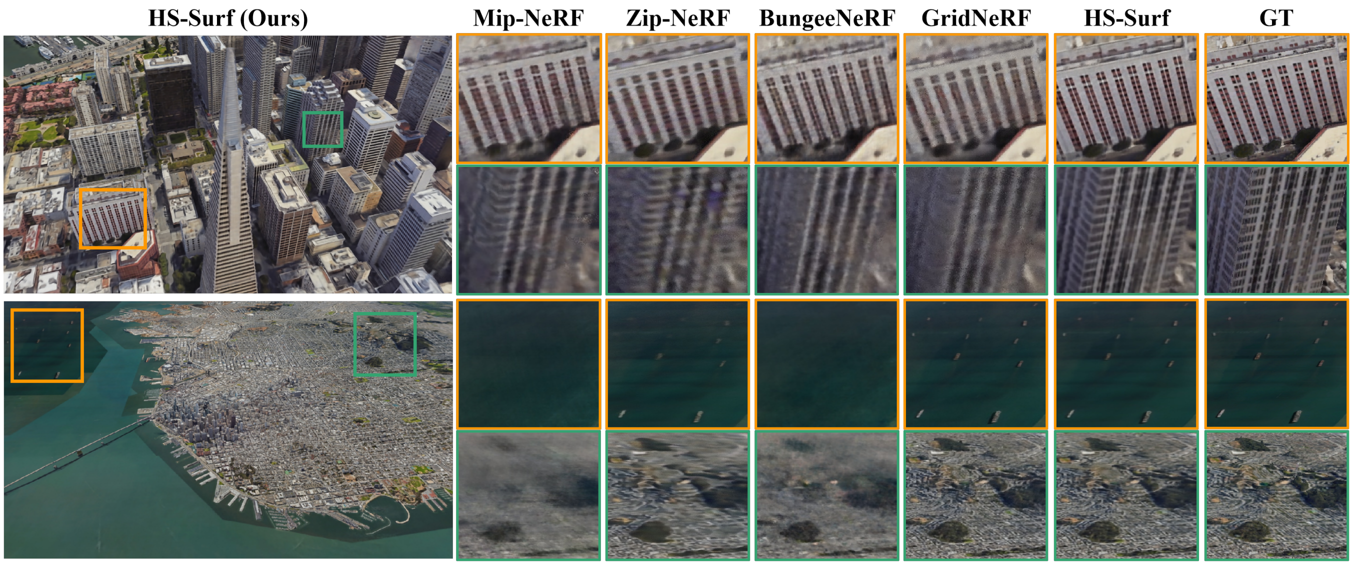

Figure 3: A visual comparison example from the aerial orbiting photography of *transamerica*. The top row is near the ground and the bottom row is far from the ground. The zoom-in images include different distances (near and far) and appearances (floater and buildings).

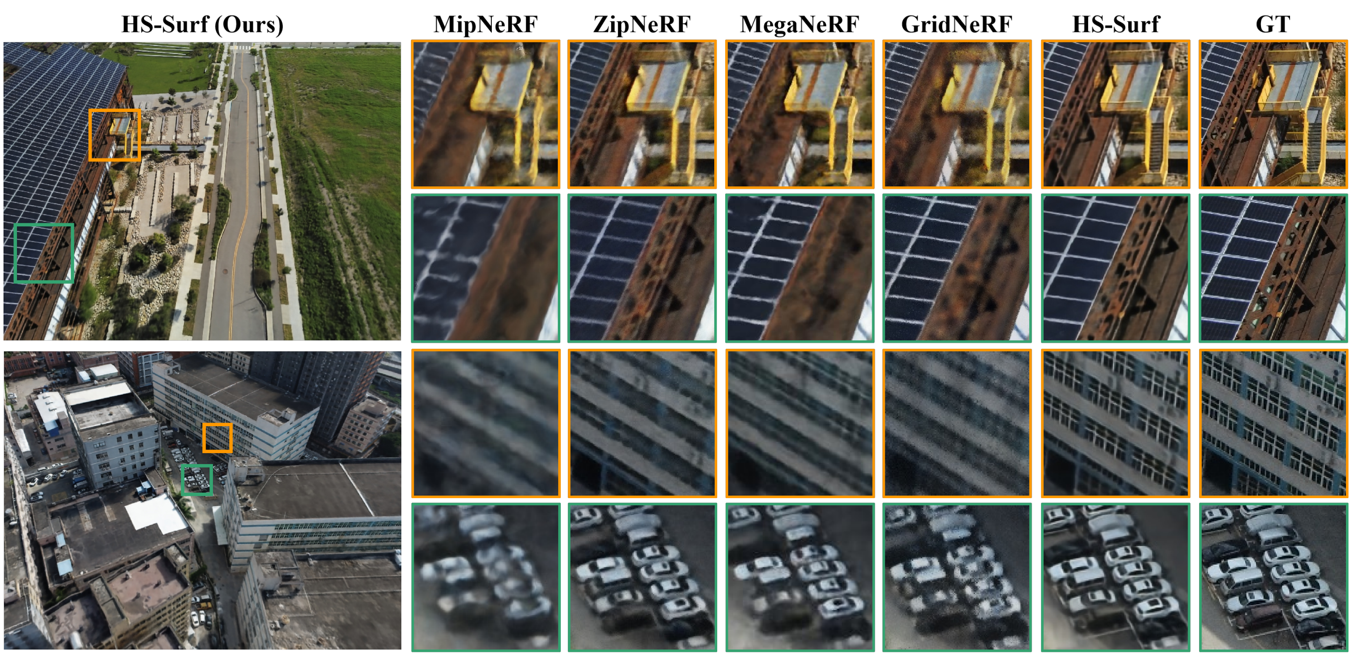

Figure 4: Two visual comparison examples from the drone shooting with fixed altitude and route. The top row is the *building* and the bottom row is the *residence*, respectively. The zoom-in images include the complex building structure, vertical and horizontal planes, and the vehicles with tiny height differences.

PyTorch on a single RTX3090 24G GPU. Additionally, HS-Surf is also compared with 3D Gaussian [32], and the results are presented in our supplementary material.

Six scenes are selected in our experiments. *Transamerica* and 56*Leonard* are two synthetic scenes from the satellite level to the ground level [6]. *Building*, *rubble*, *residence*, and *campus* are four real aerial data. The first two are from Mill 19 [5], and the remaining two are from UrbanScene3D [50]. Since each image in the aerial data has a different exposure and white balance, we refer to NeRF-in-the-wild [51] to assign a 48-dimensional appearance embedding

Table 1: Performance comparison of HS-Surf with previous NeRFs on large-scale scenes

Model	Transamerica				56Leonard				Building			
	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s)	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s)	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s)
MipNeRF [2]	22.12	0.6016	0.4856	50.90	21.87	0.5754	0.4883	51.48	19.44	0.3853	0.6499	44.32
ZipNeRF [9]	23.34	0.7092	0.4327	38.73	24.39	0.7864	0.3376	39.31	20.47	0.5282	0.5010	36.74
BungeeNeRF [6]	22.40	0.6216	0.4812	92.16	22.15	0.6015	0.4839	93.12	×	×	×	×
MegaNeRF [5]	×	×	×	×	×	×	×	×	20.69	0.4738	0.5544	251.96
GridNeRF [8]	23.22	0.6769	0.4640	89.30	23.47	0.6875	0.4605	90.41	21.00	0.5055	0.5259	80.85
HS-Surf	**25.59**	**0.8304**	**0.2941**	**25.42**	**26.41**	**0.8679**	**0.2363**	**24.51**	**21.88**	**0.6039**	**0.4417**	**27.35**
Model	Rubble				Residence				Campus			
	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s)	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s)	PSNR ↑	SSIM ↑	LPIPS ↓	Time (s)
MipNeRF [2]	22.12	0.3933	0.6761	44.20	20.21	0.4504	0.6582	55.66	20.89	0.3687	0.7631	55.19
ZipNeRF [9]	23.68	0.5536	0.5169	37.15	21.00	0.5424	0.5240	44.77	20.61	0.4013	0.6591	48.58
BungeeNeRF [6]	×	×	×	×	×	×	×	×	×	×	×	×
MegaNeRF [5]	23.10	0.4591	0.6003	232.34	20.45	0.4869	0.5796	341.34	21.71	0.4028	0.6981	271.03
GridNeRF [8]	23.20	0.4752	0.5897	81.11	20.85	0.4967	0.5883	103.38	20.00	0.3863	0.6596	100.32
HS-Surf	**24.24**	**0.5824**	**0.4943**	**27.01**	**22.12**	**0.5982**	**0.5015**	**34.06**	**21.97**	**0.4639**	**0.6183**	**34.36**

for each image to model the lighting information. Details of training and testing sets, and more model configurations can be found in the supplementary material.

4.2 Experiment results

In Table 1, we use PSNR, SSIM, LPIPS (VGG), and the time of rendering a frame to compare the rendering performance. The data distribution in *transamerica* and 56*Leonard* [6] is uneven because the cameras make loop shoot for the centers of scenes. Therefore, *transamerica* and 56*Leonard* are not divided. MegaNeRF [5] is not suitable for this mode, so it has no corresponding results. For the four aerial photography data, MegaNeRF and HS-Surf divide each scene into 8 sub-regions evenly. Since the drone always keeps a stable flight height, BungeeNeRF [6] cannot divide the camera poses according to the height from the camera to the ground, so it has no corresponding results on the later four datasets.

In *transamerica* and 56*Leonard* [6], the distance scales undergo drastic changes, HS-Surf achieves noticeable improvements in all metrics compared to models designed for variable distance scales (MipNeRF [2], ZipNeRF [9] and BungeeNeRF [6]), as shown in Table 1. The LPIPS errors of our method are reduced by 30%-40%. Figure 3 demonstrates a visual comparison example in the *transamerica* from the satellite level to the ground level. HS-Surf renders more high-frequency texture details for objects with different distances and shapes, which benefits from that HS-Surf embeds conic frustums into voxels to model distances and uses compensation to render high-frequency details at different distance scales.

In aerial photography scenes (*building*, *rubble*, *residence*, *campus*) with stable heights, HS-Surf also demonstrates better performance, with LPIPS error decreasing by 10%-20% compared to MegaNeRF [5] and GridNeRF [8]. Figure 4 presents visual comparison examples in *building* (top) and *residence* (bottom), HS-Surf synthesizes the most accurate views, especially regarding high-frequency textures. In Figure 5, we visualize the surface of *building* and the sampling strategies of three models (MipNeRF [2], GridNeRF [8], HS-Surf). MipNeRF and GridNeRF have many samples falling into empty regions in the fine stage. While the shell of HS-Surf tightly wraps around the scene surface, which validates the advantage of our method in confining the rendering to the surface and improving the utilization of model capacity.

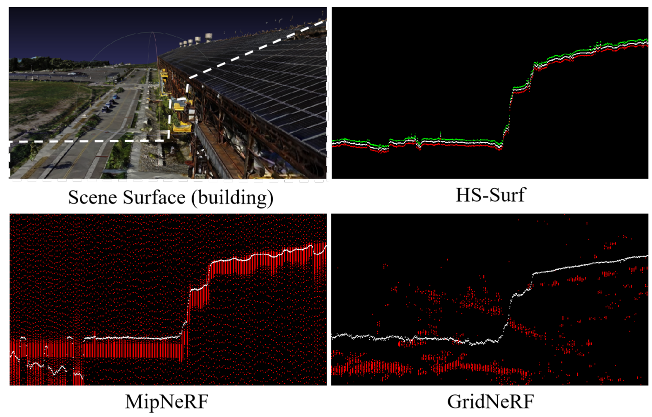

Figure 5: Demonstration of surface shell and sampling strategies. The white dashed line represents the actual scene surface, and the white solid lines represent the reconstructed surface. The surface shell of HS-Surf is represented by green and red lines. For MipNeRF and GridNeRF, the positions of samples added in the fine stage are indicated by red dots.

We find that the rendering of HS-Surf is 2× to 4× faster than other NeRFs in Table 1. Except for benefiting from the voxel grids, there are two other contributing reasons: 1) The compensation stage only samples a conic frustum for each ray. Thus, each ray only needs to be calculated once. 2) The hidden channel of shallow CNN in the post-processing is only 32. MipNeRF, BungeeNeRF, MegaNeRF and GridNeRF query multiple samples along a ray in each sampling stage, resulting in multiple computations for rendering a pixel. ZipNeRF samples six points in a conic frustum, bringing it a huge cost and a slow speed. Additionally, a comparison of model sizes is presented in the supplementary materials.

4.3 Ablation Study

In Table 2, we present an ablation study of HS-Surf. The post-processing stage is represented by PP. We summarize the findings as follows. F) The removal of feature fusion in the initialization results in a decreased modeling ability for distance scales. In supplementary material, we demonstrate the impacts of feature fusion

Table 2: Performance comparison of ablation experiments

Model	Transamerica			56 Leonard			Building			Residence		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
A) w/o fusion & PP	23.77	0.7411	0.4087	24.43	0.7877	0.3476	20.79	0.4971	0.5060	21.41	0.5275	0.5394
B) w/o compensate & PP	24.95	0.7966	0.3528	23.24	0.7063	0.4528	20.50	0.4911	0.5151	21.68	0.5639	0.5089
C) w/o dea & PP	25.15	0.8089	0.3309	25.05	0.8262	0.2896	21.25	0.5521	0.4681	19.53	0.5010	0.5509
D) w/o HS & PP	25.18	0.8105	0.3332	25.98	0.8515	0.2613	20.61	0.5274	0.4852	21.61	0.5549	0.5106
E) w/o PP	25.25	0.8147	0.3251	26.12	0.8566	0.2537	21.52	0.5678	0.4537	21.99	0.5797	**0.4952**
F) w/o fusion	24.50	0.7814	0.3383	25.30	0.8240	0.2875	21.35	0.5567	0.4788	21.70	0.5725	0.5281
G) w/o compensate	25.29	0.8149	0.3146	23.80	0.7446	0.3722	20.89	0.5428	0.5011	21.81	0.5867	0.5128
H) w/o dea	25.49	0.8250	0.2983	25.56	0.8431	0.2623	21.66	0.5911	0.4552	19.99	0.5268	0.5584
I) w/o HS	25.51	0.8268	0.2988	26.28	0.8634	0.2419	21.14	0.5648	0.4711	21.76	0.5797	0.5185
J) complete	**25.59**	**0.8304**	**0.2941**	**26.41**	**0.8679**	**0.2363**	**21.88**	**0.6039**	**0.4417**	**22.12**	**0.5982**	0.5015

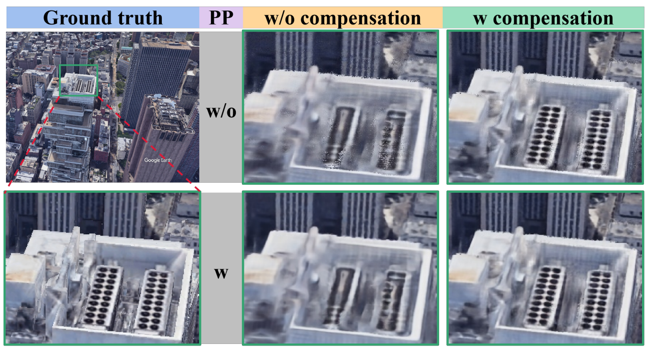

Figure 6: Ablation study on compensation. Without compensation, the rendering results lose a lot of high-frequency textures. The post-processing (PP) just removes noises, but cannot generate the lost textures.

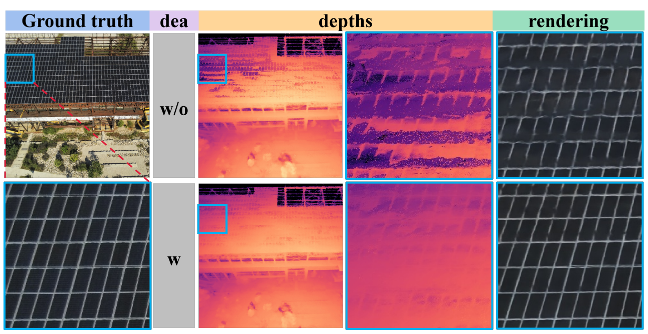

Figure 7: Ablation study on the depth augmentation. With depth augmentation, the depth map is further improved, alleviating the holes and depth texture-copy.

on scene representation. G) The removal of compensation leads to a significant decrease in model performance. In Figure 6, the results without compensation lose a lot of high-frequency textures at the top of the building. For more results of depth and rendering, please refer to the supplementary material. H) Removing depth augmentation (dea) in compensation leads to poor geometry and holes, and it also affects rendering quality, as shown in Figure 7. I) Replacing the high-frequency shell with a single spatial point in compensation results in the framework's inability to model distance scales, as shown in Figure 8. A)-E) remove the post-processing based on F)-J).

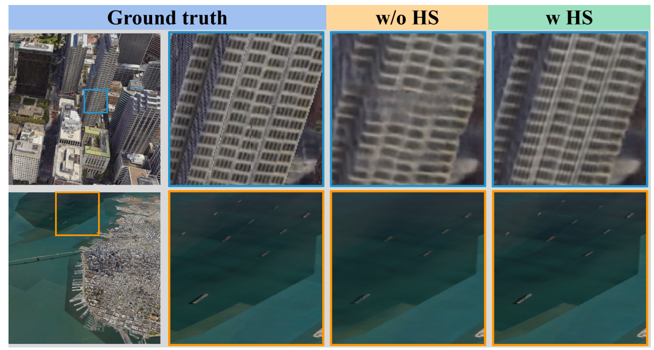

Figure 8: Ablation study on the HS. When removing the HS and directly sampling a point on the surface, the targets like windows and ships-like become unclear.

Therefore, the rendering results contain noise and have relatively low quality, as shown in Table 2 and Figure 6.

Note that B) retains only the feature fusion and achieves higher metrics compared to ZipNeRF [9] in Table 1. This validates that our feature fusion is an effective strategy for modeling distance scales.

5 Discussion and Conclusion

In this work, we aim to improve the quality of high-frequency textures in urban and aerial large-scale scenes by dealing with issues of inefficient sampling and various distances. The proposed HS-Surf create a high-frequency shell on the scene surface under the current view, and sample conic frustums on this shell to overcome the sampling inefficiency in previous methods. As a result, model capacity is efficiently utilized to render high-frequency textures. Additionally, to model the distances with drastic changes in large-scale scenes, we embed frustums representing distance into voxel grids to construct the scene at different distance scales.

Our HS-Surf achieves better rendering results in large-scale scenes, particularly concerning high-frequency textures. The solid ablation study experiments validate the effectiveness of each component in our model. Meanwhile, as we optimize the sampling and make better use of the model capacity, our implementation speed is also faster. In the future, we will explore integrating the high-frequency surface shell into other rendering techniques, and consider rendering texture details at different distance scales in dynamic scenes.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (No. 62372336, No. 61972298 and No. 62172309) and Wuhan University Huawei GeoInformatices Innovation Lab.

References

[1] 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. *Eur. Conf. Comput. Vis.*, pages 405–421, 2020.

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *Int. Conf. Comput. Vis.*, 2021.

[3] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

[4] 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. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8238–8248, 2022.

[5] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12912–12921, 2022.

[6] 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. *Eur. Conf. Comput. Vis.*, 2022.

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

[8] 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. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8296–8306, 2023.

[9] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *Int. Conf. Comput. Vis.*, 2023.

[10] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[11] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[12] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[13] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *Eur. Conf. Comput. Vis.*, 2022.

[14] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf : Bundle-adjusting neural radiance fields. *Int. Conf. Comput. Vis.*, 2021.

[15] Yifan Jiang, Peter Hedman, Ben Mildenhall, Dejia Xu, Jonathan T. Barron, Zhangyang Wang, and Tianfan Xue. Alignerf: High-fidelity neural radiance fields via alignment-aware training. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 46–55, 2023.

[16] Dogyoon Lee, Minhyeok Lee, Chajin Shin, and Sangyoun Lee. Dp-nerf: Deblurred neural radiance field with physical scene priors. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12386–12396, 2023.

[17] Yusen Wang, Zongcheng Li, Yu Jiang, Kaixuan Zhou, Tuo Cao, Yanping Fu, and Chunxia Xiao. Neuralroom: Geometry-constrained neural implicit surfaces for indoor scene reconstruction. *ACM Transactions on Graphics*, pages 1–15, 2022.

[18] Zongcheng Li, Xiaoxiao Long, Yusen Wang, Tuo Cao, Wenping Wang, Fei Luo, and Chunxia Xiao. Neto: Neural reconstruction of transparent objects with self-occlusion aware refraction-tracing. *Int. Conf. Comput. Vis.*, 2023.

[19] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 16569–16578, 2023.

[20] Krishna Wadhwani and Tamaki Kojima. Squeezenerf: Further factorized fastnerf for memory-efficient inference. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2716–2724, 2022.

[21] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 14153–14161, 2021.

[22] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *Int. Conf. Comput. Vis.*, 2021.

[23] Tong Xu, Ruhao Wang, Fei Luo, and Chunxia Xiao. Multi-scale implicit surface reconstruction for outdoor scenes. *Computational Visual Media*, 2024.

[24] Wenxiao Zhang, Huajian Zhou, Zhen Dong, Qingan Yan, and Chunxia Xiao. Rank-pointretrieval: Reranking point cloud retrieval via a visually consistent registration evaluation. *IEEE Trans. Vis. Comput. Graph.*, 2022.

[25] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5470–5479, 2022.

[26] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[27] Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. Voxurf: Voxel-based efficient and accurate neural surface reconstruction. *Int. Conf. Learn. Represent.*, 2023.

[28] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022.

[29] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. *Int. Conf. Comput. Vis.*, 2023.

[30] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[31] Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic. Adaptive shells for efficient neural radiance field rendering. *ACM Trans. Graph.*, 42(6), 2023.

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

[33] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 2011.

[34] Yanping Fu, Qingan Yan, Jie Liao, Huajian Zhou, Jin Tang, and Chunxia Xiao. Seamless texture optimization for rgb-d reconstruction. *IEEE Trans. Vis. Comput. Graph.*, 2021.

[35] Xiaowei Li, Changchang Wu, Christopher Zach, Svetlana Lazebnik, and Jan-Michael Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. *Eur. Conf. Comput. Vis.*, 2008.

[36] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[37] Yanping Fu, Qingan Yan, Jie Liao, Alix L. H. Chow, and Chunxia Xiao. Real-time dense 3d reconstruction and camera tracking via embedded planes representation. *The Visual Computer (TVC)*, 2020.

[38] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *SIGGRAPH*, 2006.

[39] Siyu Zhu, Runze Zhang, Lei Zhou, Tianwei Shen, Tian Fang, Ping Tan, and Long Quan. Very large-scale global sfm by distributed motion averaging. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[40] Yanping Fu, Qingan Yan, Jie Liao, and Chunxia Xiao. Joint texture and geometry optimization for rgb-d reconstruction. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[41] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 2004.

[42] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Eur. Conf. Comput. Vis.*, 2006.

[43] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. *Int. Conf. Comput. Vis.*, 2011.

[44] R. I. Hartley and A. Zisserman. Multiple view geometry in computer vision. *Cambridge University Press, second edition*, 2004.

[45] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment - a modern synthesis. *International workshop on vision algorithms*, 1999.

[46] Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *SIGGRAPH*, 2004.

[47] Qi Shan, Riley Adams, Brian Curless, Yasutaka Furukawa, and Steven M. Seitz. The visual turing test for scene reconstruction. *3DV*, 2013.

[48] 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. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[49] Konstantinos Rematas, Andrew Liu, Pratul Srinivasan, Jonathan Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[50] Liqiang Lin, Yilin Liu, Yue Hu, Xingguang Yan, Ke Xie, and Hui Huang. Capturing, reconstructing, and simulating: the urbanscene3d dataset. *Eur. Conf. Comput. Vis.*, 2022.

[51] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.