K-Buffers: A Plug-in Method for Enhancing Neural Fields with Multiple Buffers

Haofan Ren[1], **Zunjie Zhu**[1*], **Xiang Chen**[1], **Ming Lu**[2], **Rongfeng Lu**[1] and **Chenggang Yan**[1]
[1]Hangzhou Dianzi University
[2]Intel Labs China
{hfren, zunjiezhu, cchenx, rongfeng-lu, cgyan}@hdu.edu.cn, lu199192@gmail.com

Abstract

Neural fields are now the central focus of research in 3D vision and computer graphics. Existing methods mainly focus on various scene representations, such as neural points and 3D Gaussians. However, few works have studied the rendering process to enhance the neural fields. In this work, we propose a plug-in method named K-Buffers that leverages multiple buffers to improve the rendering performance. Our method first renders K buffers from scene representations and constructs K pixel-wise feature maps. Then, We introduce a K-Feature Fusion Network (KFN) to merge the K pixel-wise feature maps. Finally, we adopt a feature decoder to generate the rendering image. We also introduce an acceleration strategy to improve rendering speed and quality. We apply our method to well-known radiance field baselines, including neural point fields and 3D Gaussian Splatting (3DGS). Extensive experiments demonstrate that our method effectively enhances the rendering performance of neural point fields and 3DGS.

1 Introduction

Since the pioneering work NeRF [Mildenhall *et al.*, 2020], neural fields have become the core research problem in 3D vision and computer graphics. Although NeRF can achieve high fidelity, it suffers from several drawbacks such as long training time and slow rendering speed. Instant-NGP [Müller *et al.*, 2022] uses a multiresolution hash-grid to achieve fast training and rendering speed at the expense of huge memory consumption. PlenOctrees [Yu *et al.*, 2021] can render 800×800 images at more than 100 FPS, but the model size has increased by nearly 100 times compared with the original NeRF. 3D Gaussian [Kerbl *et al.*, 2023] is the most recent representation that achieves state-of-the-art visual quality while maintaining competitive training and rendering times. However, the model size of 3D Gaussian is still much larger than the original NeRF. This presents a challenge in balancing computational cost and model size of neural fields. Conversely, neural point fields, due to their intrinsic properties, hold promise in alleviating this challenge.

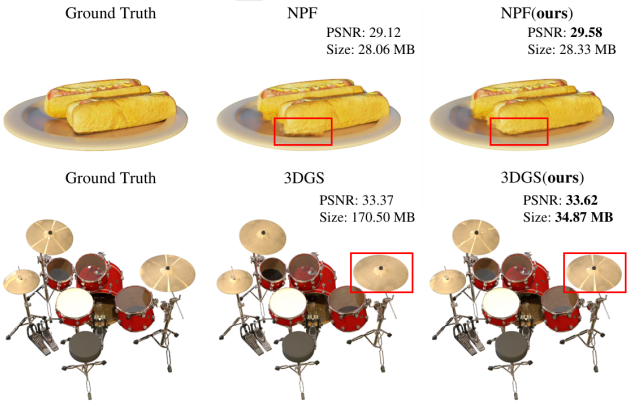

Figure 1: As shown in the figure, our method can simultaneously enhance neural point fields(NPF) and 3DGS.

However, neural point fields also have some limitations. A core limitation is that they are non-trivial to render since the point clouds are noisy. Therefore, neural point fields usually fail to achieve competitive performance compared with NeRF since volume rendering is more robust than point cloud rasterization. Several recent works have been proposed to solve this limitation from different aspects. BPCR [Huang *et al.*, 2023] imposes radiance mapping borrowed from NeRF [Mildenhall *et al.*, 2020] to construct point features. FrePCR [Zhang *et al.*, 2023] leverages a HyperNetwork [Ha *et al.*, 2017] and narrows the positional encoding interval to improve the point features. READ [Li *et al.*, 2023] propose an improved U-Net [Ronneberger *et al.*, 2015] named ω - net to filter the neural point descriptors on different scales to fill the holes in the rasterized image. SNP [Zuo and Deng, 2022] uses the spherical harmonics functions to construct point-wise features and a shallow U-Net [Ronneberger *et al.*, 2015] without a normalization layer to remove noise in the feature map. However, they all ignore the core difference between volume rendering and point cloud rasterization. The key insight of volume rendering is to integrate the colors of multiple points along a ray, rather than a single point in rasterization. This motivates us to consider using multiple points to determine the rendered

*Corresponding author. The source code is available: https://github.com/renhaofan/k-buffers

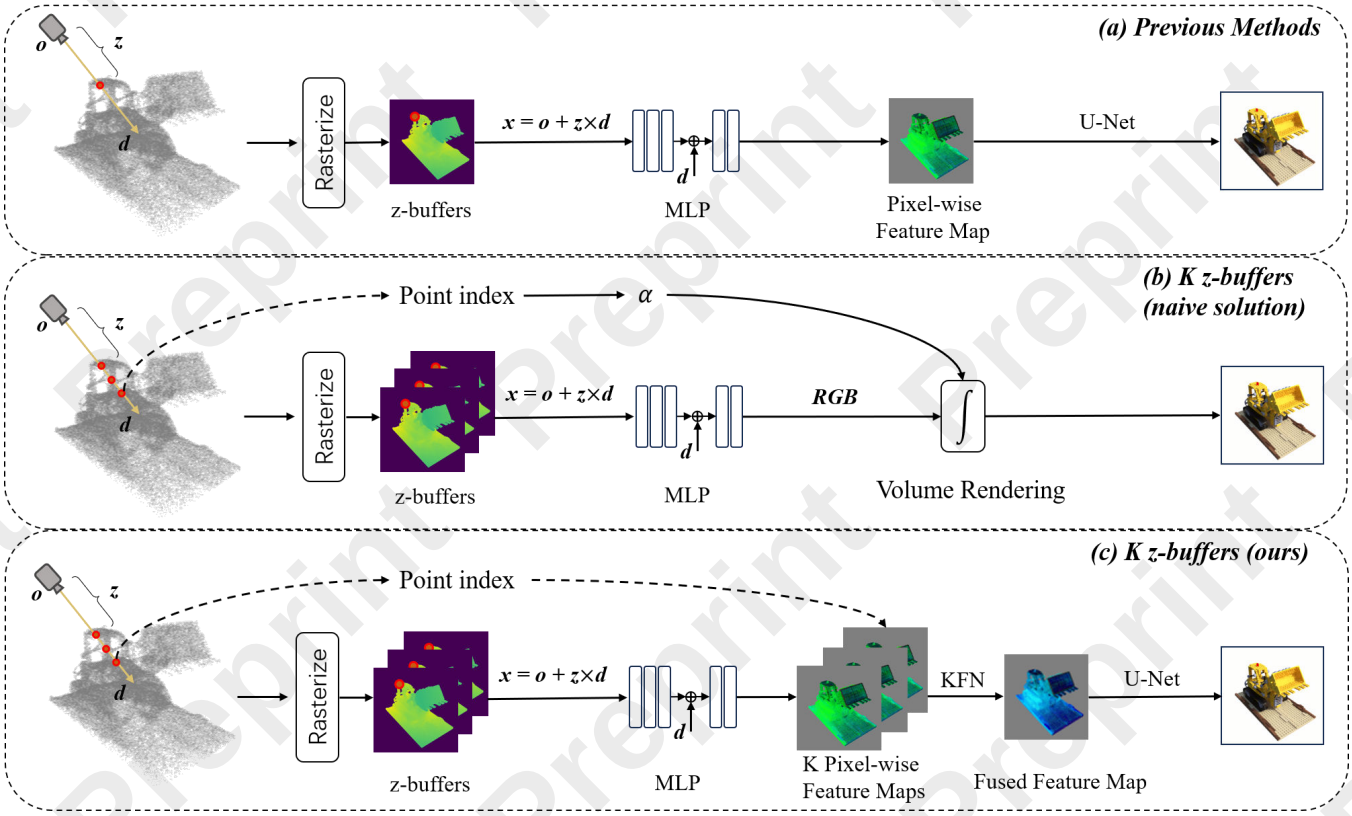

Figure 2: **The motivation of our method.** (a) Previous neural point fields first render the pixel-wise z-buffers and then use a decoder to generate the image from the pixel-wise feature map. However, they are sensitive to the noisy point cloud. (b) A naive solution is to render K z-buffers and use volume rendering to integrate the K colors. However, this solution achieves overfitted results. (c) Our method uses KFN to integrate the K pixel-wise feature maps, significantly enhancing the performance of neural point fields.

color of a pixel.

A naive solution is that, after obtaining K z-buffers, we can follow the approach of NeRF and use volume rendering to integrate the K colors predicted by MLPs as shown in Figure. 2(b). Our experiments revealed that this solution suffers from severe overfitting issues. Although data augmentation methods like random crop could alleviate the overfitting issue, the rendering quality still falls short of expectations. Besides, the memory requirement increases dramatically, restricting high-resolution rendering. The main problem of this naive solution is that it simply integrates the colors along the ray, still failing to solve the noise problem of neural point fields. Instead, we propose a novel method named K-Buffers to enhance the rendering quality by integrating the K feature maps. Specifically, we first encode the K z-buffers into K feature maps and then introduce a K-Feature Fusion Network (KFN) to fuse the K feature maps in latent space. In this manner, we can reduce the noise caused by noisy neural point fields. In the process of generating the K feature maps, we introduce a strategy to improve rendering speed. Finally, we use a decoder to generate the final rendered image. Our method significantly improves the robustness against noise, whether it is inherent in the point cloud or introduced during rasterization. Furthermore, our method can also help 3DGS to improve the visual quality and reduce the model size due to its rasterization process. In summary, our contributions can be summarized as follows:

- We propose a novel method to enhance the performance of neural point fields with multiple z-buffers.
- We design a tiny K-Feature Fusion Network (KFN) to reduce the noises caused by the point cloud geometry and rasterization process.
- We propose a novel acceleration strategy to speed up the rendering time of the proposed method.
- We conduct comprehensive experiments to demonstrate the effectiveness and advantages of our method both for neural point fields and 3DGS.

2 Related Work

Neural Radiance Fields [Mildenhall *et al.*, 2020] utilizes a 5D implicit function for scene modeling via a continuous volumetric approach, allowing it to estimate both density and radiance at any given position and direction. In this way, NeRF marked a paradigm shift in scene representation and the synthesis of realistic novel views.

Accelerated Neural Radiance Fields

While NeRF yields remarkable results, this comes at the cost of large computation, due to the need to evaluate a large MLP

model hundreds of times for each pixel. Subsequent work has proposed enhanced data structures to accelerate. Plenoxels [Fridovich-Keil *et al.*, 2022] is a sparse voxel grid where each occupied voxel corner stores a scalar opacity and a vector of spherical harmonic (SH) coefficients for each color channel. PlenOctrees [Yu *et al.*, 2021] make use of octree and a modified NeRF that is trained to output spherical basis functions to accelerate rendering speed. InstantNGP [Müller *et al.*, 2022] uses multi-resolution hash grids to accelerate the training speed, reducing training time to a few seconds. KiloNeRF [Reiser *et al.*, 2021] utilizes thousands of tiny MLPs to accelerate the rendering speed. TensorRF [Chen *et al.*, 2022] proposes to factorize the volume field tensor into multiple compact low-rank components to accelerate training speed. The above methods all use various types of data structures to store features, making them more efficiently organized in 3D space. In a sense, this is a trade-off between computation and memory.

Neural Point Fields

Although the techniques above demonstrate remarkable effectiveness, it is difficult to adapt them to model large environments, which presents a challenge. An alternative method involves utilizing point clouds to represent the geometric structure of the scene [Chang *et al.*, 2023a], [Li *et al.*, 2023], [Chang *et al.*, 2023b]. Point clouds can vary in density, allowing for the allocation of computational resources where necessary, and effectively skipping the empty space. Thus, it is not prone to generate floating artifacts, which is a server problem in neural radiance fields.

There are two major categories of methods for neural point fields: those based on ray casting and those based on rasterization. The ray-casting-based methods achieved high rendering quality. Point-NeRF [Xu *et al.*, 2022] and developed work [Sun *et al.*, 2023] usually sample hundreds or thousands of points for each ray. To perform volume rendering, the feature of each sampled point is queried in the local neighborhood of point clouds to produce density and color. NPLF [Ost *et al.*, 2022] efficiently represents scenes with just one radiance evaluation per ray by promoting sparse point clouds to neural implicit light fields. However, the computational burden of these approaches remains high, making it challenging to achieve real-time rendering. HashPoint [Ma *et al.*, 2024] proposes to accelerate ray-tracing point cloud rendering. However, even with the acceleration, it is still challenging to achieve real-time rendering. Additionally, the model sizes are several times larger than NeRF.

On the contrary, the rasterization-based methods get a great balance between model size and rendering speed. NPBG [Aliev *et al.*, 2020] rasterize the points with neural descriptors at different scale resolutions and a U-Net followed to obtain the rendered image. NPBG++ [Rakhimov *et al.*, 2022] demonstrated that point embedding can be directly extracted from input images, allowing their method to render unseen point clouds without per-scene optimization. ADOP [Rückert *et al.*, 2022] combines neural rendering with a differentiable point rasterizer to minimize the difference between rendered images and the ground truth. Dai *et al.* [Dai *et al.*, 2020] aggregate points into multi-plane images, combined with a 3D-convolutional network. TriVol [Hu *et al.*, 2023] proposes a novel 3D representation called triple volumes to get point embedding. BPCR [Huang *et al.*, 2023] combines implicit radiance mapping and rasterized z-buffers to achieve excellent results. The improved approach in FrePCR [Zhang *et al.*, 2023] utilizes a hypernetwork to enhance radiance mapping. However, almost all rasterization-based methods do not specifically address noise introduced by rasterization, such as holes. The usual practice is to use a post-processing U-Net that involves downsampling and then upsampling the rasterized feature map. Unfortunately, such methods are not always effective. The noise mentioned above is the main reason why rasterization-based methods often have lower rendering quality compared to ray-casting-based methods. 3DGS [Kerbl *et al.*, 2023] utilizes anisotropic Gaussians with a geometry optimization strategy to allow a fast and high-quality radiance field. However, it still suffers from the noise issues mentioned above and significantly increases storage requirements. In this work, we focus on improving the rendering quality of rasterization-based methods without too much increase in model size and computational requirement. Additionally, we aim to enhance the robustness of these methods against noise.

3 Method

In this paper, we propose a novel strategy to enhance the rasterization pipeline of point-based neural rendering based on K z-buffers. First, we save the K z-buffers during the depth test instead of the closest point to the camera position for each pixel. Then we encode queried points to latent space, which are the so-called neural points. However, if we directly encode all queried points of the K z-buffers, the number of neural points will increase with the growth of K. Therefore, we propose a solution to significantly reduce the number of neural points, even fewer than a single z-buffer. After obtaining the neural points, we reorganize them into K pixel-wise feature maps. Finally, we utilize KFN to merge and denoise K feature maps and a U-Net to decode into colors.

3.1 Feature Extraction

The NeRF representation [Mildenhall *et al.*, 2020] takes the 3D coordinate $\boldsymbol{x} = (x, y, z)$ and view direction $\boldsymbol{d} = (\theta, \phi)$ as inputs and output the color $\mathbf{c}$ and density σ using a Multi-Layer Perceptron (MLP) $\boldsymbol{F}_\Theta$, parameterized by Θ:

$$\boldsymbol{c}, \sigma = \boldsymbol{F}_\Theta(\boldsymbol{x}, \boldsymbol{d}). \tag{1}$$

Inspired by above equation, the point-based rendering methods BPCR [Huang *et al.*, 2023], FrePCR [Zhang *et al.*, 2023] use the radiance mapping $\boldsymbol{L}$ as feature encoder:

$$\boldsymbol{L} = \boldsymbol{F}_\Theta(\boldsymbol{x}, \boldsymbol{d}). \tag{2}$$

Different from NeRF, point-based rendering approaches provide explicit geometry prior. As a consequence, $\boldsymbol{x}$ can be acquired by Eq. 3:

$$\boldsymbol{x} = \boldsymbol{o} + z\boldsymbol{d}, \tag{3}$$

where $\boldsymbol{o}$, $\boldsymbol{d}$ denote camera position and normalized ray direction respectively. In practice, the z value is obtained by querying the z-buffer via rasterization and depth test.

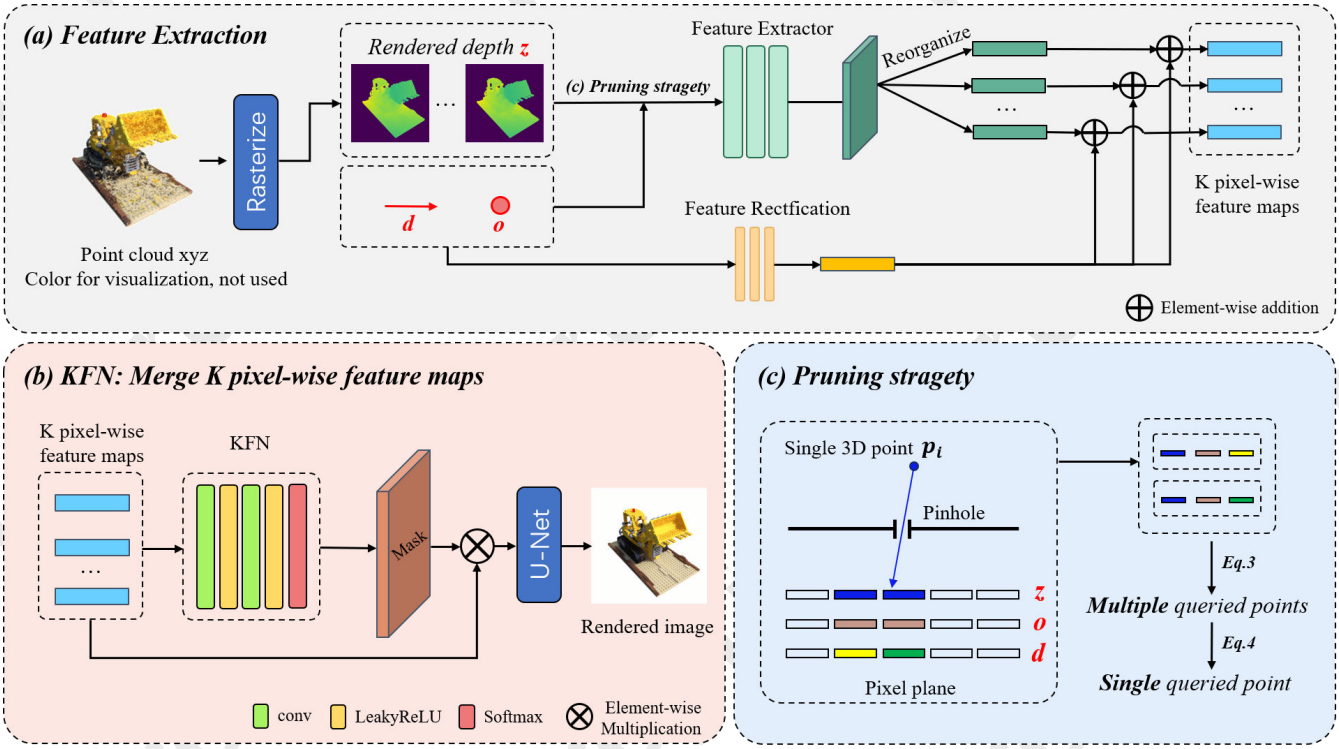

Figure 3: The Overview pipeline with our model. (a) depicts how to obtain the neural descriptors from noisy point clouds. (b) describe how to fuse K pixel-wise features and ultimately obtain the rendering result. (c) illustrates the reason why a single 3D point will generate multiple queried points and how to reduce the number of them by our pruning strategy.

By recording the point closest to the camera, we can achieve a single z-buffer. To obtain K z-buffers, we only need to store the previous K z-values during depth testing, which does not introduce excessive computational complexity compared with a single z-buffer.

The inspiration behind using multiple layers of z-buffers is that we observe different layers exhibit strengths and weaknesses in local geometry. However, due to the characteristics of rasterization, multiple layers of z-buffers as input would lead to a sharp increase in computational complexity and memory consumption. To solve this issue, we propose a novel method to prune redundant queried points.

Prune redundant queried points

Each point is expanded to a disk of radius τ during rasterization as point clouds are discrete. During rasterization, a single 3D point will splash on multiple pixels' positions. These pixels have the same z value and camera position $\boldsymbol{o}$, but different directions $\boldsymbol{d}_j$. We take a one-dimensional example to illustrate this phenomenon. As illustrated in Figure. 3(c), a 3D point $\boldsymbol{p}_i$ of point cloud splashed onto pixel plane and occupies two pixels. These two pixels with different directions will be encoded into different queried points through Eq. 3. In this way, a single 3D point generates two queried points. However, each queried point will be encoded into a neural point by Eq. 2. As a result, the number of neural points is greater than the number of points in the point clouds.

We only reserve one direction $\boldsymbol{d}_m$ for each 3D point to generate the queried point. The queried points generated by other directions are referred to as **redundant queried points**.

In particular, each pixel's ID j can be calculated by $W * \text{pixel.x} + \text{pixel.y}$, where W means the width of rendered image, pixel.x and pixel.y represent the indices in the pixel coordinate system respectively. For each 3D point $\boldsymbol{p}_i$ observed by the current view, we can deduce ID collection $\boldsymbol{A}_{\boldsymbol{p}_i}$ of all pixels occupied by $\boldsymbol{p}_i$. We only keep the direction $\boldsymbol{d}_m$ corresponding to the smallest pixel ID to generate queried point:

$$\boldsymbol{d}_m = \boldsymbol{d}_j, j = \min \boldsymbol{A}_{\boldsymbol{p}_i} \tag{4}$$

In the example shown in Figure. 3(c), five pixels' IDs are 0, 1, 2, 3, and 4, respectively. Thus, $\boldsymbol{A}_{\boldsymbol{p}_i} = \{1, 2\}$ and $\boldsymbol{d}_m = \boldsymbol{d}_1$.

Feature rectification

Simply pruning the queried point using Eq. 4 leads to identical features across different pixel positions occupied by $\boldsymbol{p}_i$, which diminishes the sensitivity of radiance mapping to variations in pixel positions.

To address this problem, we utilize a tiny MLP $\boldsymbol{T}_\Psi$ with 3,656 trainable params parameterized by Ψ to rectify the features of queried points via Eq. 2. Thus, our rectified feature map is formulated as Eq. 5. Besides, we make use of different encoding methods for $\boldsymbol{F}_\Theta$ and $\boldsymbol{T}_\Psi$. In practice, we apply position encoding introduced in NeRF [Mildenhall *et al.*, 2020] to $\boldsymbol{x}_m$ and $\boldsymbol{d}_m$, while employing sphere harmonics encoding for $\boldsymbol{d}_j$. The multi-resolution hash encoding is applied to $\boldsymbol{o}$. Our rectified features can be formulated by Eq. 5:

$$\boldsymbol{L}_{rect} = \boldsymbol{F}_{\Theta}(\boldsymbol{x}_m, \boldsymbol{d}_m) + \boldsymbol{T}_{\Psi}(\boldsymbol{o}, \boldsymbol{d}_j). \tag{5}$$

3.2 K-Feature Fusion Network

Due to multiple z-buffers as input, the key is how to fuse and denoise them. A similar research area [Godard *et al.*, 2018], [Mildenhall *et al.*, 2018], [Xia *et al.*, 2020] is called burst denoise. It takes a burst of noisy images captured with a handheld camera as input, and the aim is to produce a clean image. KPN [Mildenhall *et al.*, 2018](Kernel Prediction Network) is a kind of classic and widely used technique in this area. KPN predicts a feature vector for each pixel, which is then reshaped into a set of spatially varying kernels applied to the input burst images. A naive idea is to direly employ KPN [Mildenhall *et al.*, 2018] to process multiple z-buffers and merge them into a single clean z-buffer, which is then fed into the point-based rendering methods [Huang *et al.*, 2023], [Zhang *et al.*, 2023]. Nevertheless, this approach can easily lead to unstable training scenarios.

We designed a tiny network to integrate multiple K z-buffers in the latent space. In other words, we merge K pixel-wise feature maps and denoise to generate cleaner feature maps, where the denoise operation happens in latent space rather than the aspect of the z-buffer. In practice, we first generate K pixel-wise feature maps from K z-buffers. Then, we propose a module called **K**-Feature **F**usion **N**etwork(KFN) to fuse K pixel-wise feature maps. Different from the U-Net structure of KPN, KFN is very simple and consists of two convolution layers, two activation layers, and a softmax normalization(see Figure. 3(b)). Besides, KFN predicts a pixel-wise scalar mask range in $[0, 1]$ to merge K pixel-wise feature maps rather than the spatially varying kernels. Besides, previous work [Hedman *et al.*, 2018] also uses CNNs to estimate blending weights for novel-view synthesis, which results in temporal flickering. However, our method does not suffer such a problem(see (Table 5)). The results also demonstrate the good denoising effects with KFN(see (Figure. 4)).

3.3 Overall Training Process

Neural Points Fields

Our pipeline could be divided into five stages: rasterization, neural points construction, pruning, K pixel-wise feature maps fusion, and refinement. Firstly, through rasterization, all points $\boldsymbol{P}$ observed by the current view will be splashed into the pixel plane to generate K z-buffers. Since this process does not require differentiability, it can be implemented using hardware-accelerated frameworks such as OpenGL [Shreiner *et al.*, 2009] and run in real-time. At the same time, we record the pixel positions $\boldsymbol{A}_{\boldsymbol{p}_i}$ occupied by each unique point $\boldsymbol{p}_i \in \boldsymbol{P}$ for reorganization.

For each pixel, we generate the queried points $\boldsymbol{x}$ from K z-buffers by Eq. 3. Afterward, we remove redundant points by Eq. 4, and the remaining points $\boldsymbol{x_m}$ are passed through Eq. 5 to obtain latent features $\boldsymbol{L}_{rect} = N \times C$, where N and C refer to the number of queried points $\boldsymbol{x}_m$ and the dimension of latent features respectively. Using the pre-reserved set $\boldsymbol{A}_{\boldsymbol{p}_i}$, we reorganize the $\boldsymbol{L}_{rect}$ back to K pixel-wise feature maps with shape $K \times H \times W \times C$, while we assign zero to the feature of those unoccupied pixels. We use KFN to predict a mask with the same dimensions as K pixel-wise feature maps. The fused features would be obtained by the element-wise multiplication between the predicted mask and K pixel-wise feature maps. Finally, a U-Net is employed to decode the fused features into the rendered image in the shape of $H \times W \times 3$.

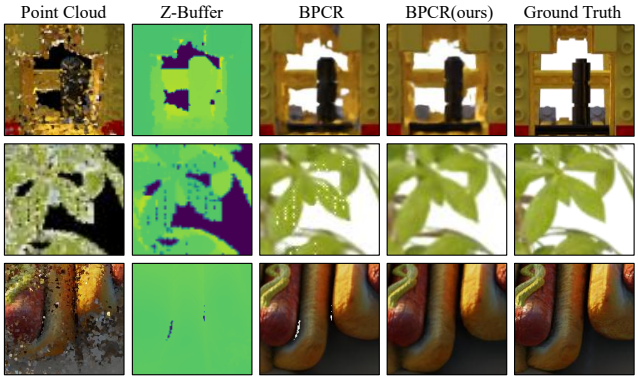

Figure 4: Illustration of the z-buffer defect. The color of the point cloud has not been used, which is only for visualization purposes.

3DGS

Each of the 3D Gaussians is parametrized by its potision $\boldsymbol{\mu}$, covariance matrix $\boldsymbol{\Sigma}$, opacity $\boldsymbol{\alpha}$ and high dimension feature $\boldsymbol{f}$. We use a tiny MLP $\boldsymbol{H}_{\Gamma}$ parameterized by Γ to generate the radiance mapping $\boldsymbol{L}_{rect}$, which is constructed by blending the multiple ordered Gaussians as:

$$\boldsymbol{L}_{rect} = \sum_{i=1}^{N} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)\, \boldsymbol{l}_i^{rect} \tag{6}$$

$$\boldsymbol{l}_i^{rect} = \boldsymbol{H}_{\Gamma}(\boldsymbol{d}_i) + \boldsymbol{f}_i \tag{7}$$

where $\boldsymbol{f}_i$, α_i and $\boldsymbol{d}_i$ represent the feature, opacity and view direction of queried Gaussian i.

In summary, for neural point fields, we incorporate the pruning queried points strategy and feature rectification into the radiance mapping $\boldsymbol{L}_{rect}$, which will be reorganized into K pixel-wise feature maps. As for 3DGS, the radiance mapping is generated by Eq. 6. With the radiance mapping, a proposed lightweight module named KFN is used to merge K pixel-wise feature maps. Finally, similar to the previous method, a U-Net is used to decode and obtain the rendered image. The loss function remains completely consistent with the baseline without any modifications.

3.4 Benchmark Evaluation

Settings and Compared Methods

Since our work needs z-buffers rasterized by point cloud, we choose three methods as our baseline to evaluate: 1) BPCR [Huang *et al.*, 2023]: A simple but effective architecture point-based rendering method. 2) FrePCR [Zhang *et al.*, 2023]: The improved version of BPCR. For a fair comparison, we don't narrow the positional encoding interval as described in FrePCR but rather keep it the same as BPCR. 3) 3DGS [Kerbl *et al.*, 2023]: An efficient point-based rendering method, which represents the scene with 3D Gaussians

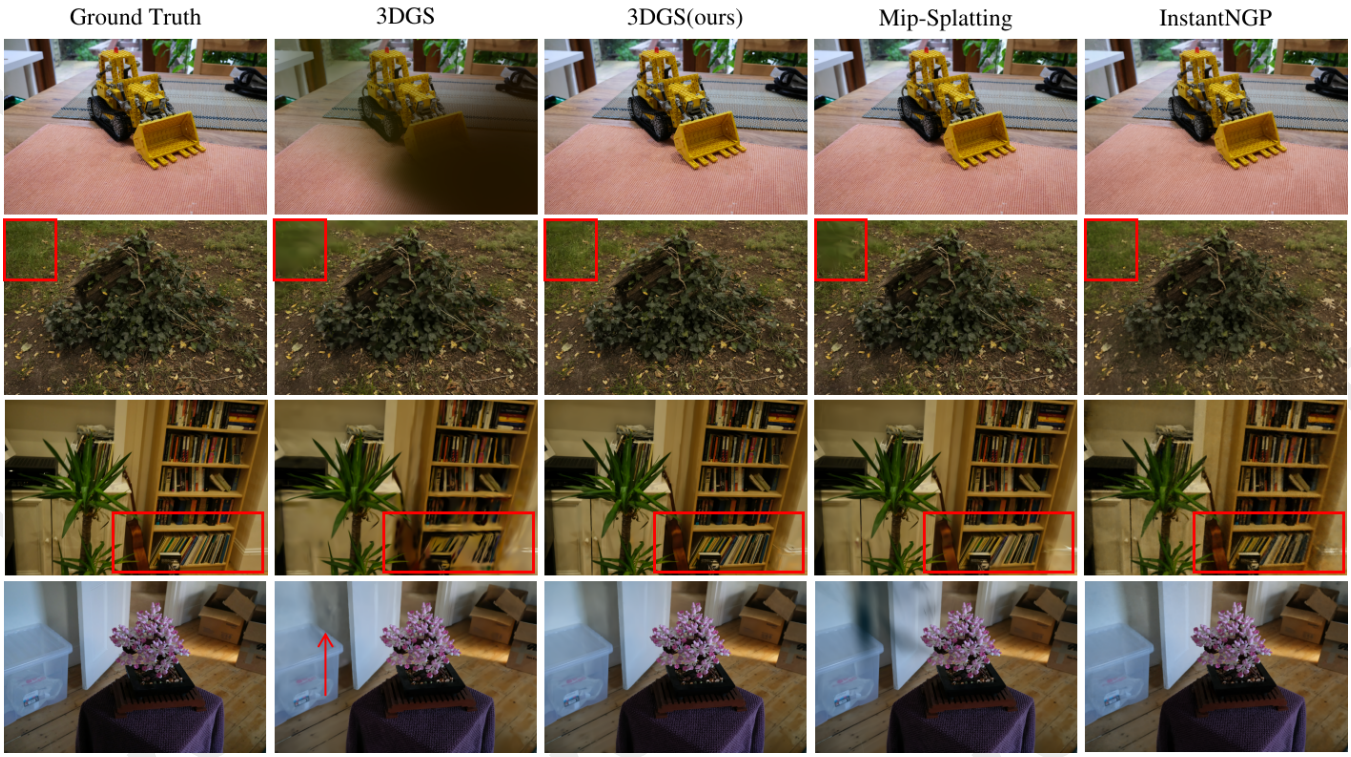

Figure 5: We compare the rendered novel views between ours and previous methods. The scenes are, from the top down: *kitchen*, *stump*, *room*, *bonsai* from Mip-NeRF 360 dataset.

Metric	PSNR ↑	SSIM ↑	LPIPS ↓	Storage(MB) ↓
InstantNGP	26.67	0.738	0.369	38.57
Mip-Splatting	29.02	0.871	0.224	768.86
3DGS	28.93	**0.869**	0.136	735.57
3DGS+Ours	**29.19**	0.859	**0.126**	**383.43**

Table 1: Quantitative comparisons on Mip-NeRF 360 dataset.

to optimize the geometry. We keep $\boldsymbol{F}_\Theta$ and U-Net exactly the same as the baseline for experimentation. We evaluate on three datasets: NeRF-Synthetic [Mildenhall *et al.*, 2020], ScanNet [Dai *et al.*, 2017], DTU [Jensen *et al.*, 2014]. The point cloud initialization is same as the BPCR. Besides, we also evaluate the 3DGS on Mip-NeRF 360 [Barron *et al.*, 2022], which is compared with InstantNGP [Müller *et al.*, 2022] and Mip-Splatting [Yu *et al.*, 2024].

Quantitative and Qualitative Results

We present the quantitative results in Table 1 and Table 2. It can be found that our method can simultaneously help improve the rendering quality of neural point fields and 3DGS. As shown in Figure 4, the first row showcases the noise removal capability of our proposed method. The second row demonstrates the aliasing caused by unsuitable point radius and camera observation view, which can be alleviated by our method. The third row exhibits its capability for filling holes. As illustrated in Figure 5, our method can effectively help 3DGS mitigate the floater artifacts. At the same time, it can restore high-frequency details of the scene and also improve modeling in low-textured areas. Besides, we noticed that the PSNR of FrePCR on the DTU dataset is bad shown in Table 2. One possible reason is that the scenes of DTU contain relatively less high-frequency information. FrePCR increases the expression ability of neural points by AFNet(Adaptive Frequency Net). However, due to the low-frequency information of the scene, it is difficult to modulate the implicit radiance signal correctly for AFNet. Incorporating our method is equivalent to expanding the receptive field of AFNet, which enhances the comprehensive understanding of the scene. We compare with the ray-tracing-based method as shown in Table 3, which demonstrates the advantages and disadvantages of our method. We visualize the failure case of our method. As shown in Figure 6, our method blurs the details of the reflective regions.

3.5 Ablation Study

As shown in Table 4, our proposed pruning strategy significantly reduces the number of queried points, leading to a substantial decrease in memory usage and computational load. Through the pruning strategy, both computational and memory usage have been reduced by more than 3 times. It seems that feature rectification does not improve rendering quality. However, we conducted experiments on different datasets as shown in Table 7. We find that feature rectification achieves the best overall rendering quality across multiple datasets. In addition, each rendered pixel is not derived from the fixed

	NeRF-Synthetic			ScanNet			DTU		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
BPCR	29.12	0.935	0.056	25.88	**0.794**	0.415	30.81	0.904	**0.128**
BPCR+Ours	**29.58**	**0.936**	**0.054**	**26.66**	0.789	**0.383**	**30.99**	**0.907**	0.151
FrePCR	29.26	0.934	0.055	**26.44**	**0.788**	**0.384**	24.61	0.870	0.195
FrePCR+Ours	**29.83**	**0.939**	**0.051**	26.34	**0.788**	0.391	**30.61**	**0.905**	**0.153**
3DGS	33.37	0.968	0.028	24.16	0.773	**0.403**	**33.91**	0.943	**0.050**
3DGS+Ours	**33.62**	**0.969**	**0.019**	**25.55**	**0.785**	0.406	**33.91**	**0.954**	0.059

Table 2: Quantitative comparisons on NeRF-Synthetic, ScanNet and DTU dataset.

	Rasterization						Ray Tracing	
	BPCR	BPCR†	FrePCR	FrePCR†	3DGS	3DGS†	PN	PN+HP
PSNR↑	29.12	29.58	29.26	29.83	33.37	**33.62**	33.31	33.22
FPS↑	39.21	27.52	39.56	26.39	**294.70**	82.12	0.12	9.60
Storage(MB)↓	**28.46**	28.73	28.63	28.90	170.50	34.87	105.12	105.12

Table 3: Comparison of our method integrated with ray-tracing-based method PN(PointNeRF) and HP(HashPoint). The storage includes the point cloud size. † means our method.

	Number	GFLOPS	Model Size(MB)	Mem(GB)	PSNR	FPS
BPCR	203,757	**80.96**	**8.20**	9	29.12	**39.21**
BPCR(KFN)	1,613,013	386.86	8.46	33	29.95	10.12
BPCR(KFN+Pruning)	**194,225**	122.43	8.46	**7**	29.68	27.80
BPCR(KFN+Pruning+Rect)	**194,225**	122.44	8.47	**7**	**29.58**	27.52

Table 4: Complexity analysis on NeRF-Synthetic dataset. The first column of metrics means the number of queried points $\boldsymbol{x}$. Since the numbers and GFLOPS are view-dependent, here we take the first view of the test scene as an example.

number of Gaussians in 3DGS process, which hinders the efficient implementation of the pruning operation on CUDA. As a result, the pruning operation is not applied to 3DGS in our approach.

We use the StopThePop [Radl *et al.*, 2024] to evaluate the temporal consistency, utilizing MSE(Mean Squared Error) and ꟻLIP [Andersson *et al.*, 2020] as the evaluation metrics. As shown in Table 5, integrating our method does not compromise the temporal consistency of the original approach.

We conduct tests to evaluate the impact of varying z-buffer layers as shown in Table 6. It is clear that as K increases, there is an increase in FPS, model size and rendering quality. The model size does not sharply increase with the increment in K, but the computational load becomes more sensitive to K.

	Short-range		Long-range	
	MSE↓	ꟻLIP ↓	MSE↓	ꟻLIP ↓
3DGS	0.018	0.008	0.100	0.028
3DGS+Ours	0.018	0.009	0.102	0.028

Table 5: Quantitative results of temporal consistency on NeRF-Synthetic dataset. We use step 1,7 as frame offset during optical flow prediction to evaluate short-range and long-range consistency, respectively.

	K=1	K=2	K=4	K=8
PSNR↑	26.17	26.29	26.84	**27.62**
FPS↓	**51.00**	44.87	36.83	27.52
Model Size(MB)↑	**8.30**	8.33	8.37	8.47

Table 6: Quantitative results on *lego* scene of NeRF-Synthetic for different values of K.

	NeRF-Synthetic	ScanNet	DTU	Mean
BPCR	29.12	25.88	30.81	28.60
BPCR(KFN)	**29.95**	25.93	31.20	29.03
BPCR(KFN+Pruning)	29.68	26.05	**31.08**	28.94
BPCR(KFN+Pruning+Rect)	29.58	**26.66**	30.99	**29.08**
FrePCR	29.26	**26.44**	24.61	26.77
FrePCR(KFN)	29.44	26.20	29.20	28.28
FrePCR(KFN+Pruning)	29.68	26.12	30.78	28.86
FrePCR(KFN+Pruning+Rect)	**29.83**	26.34	**30.61**	**28.93**
3DGS	33.37	24.16	**33.91**	30.48
3DGS(KFN)	33.20	**25.81**	33.66	30.89
3DGS(KFN+Pruning)	/	/	/	/
3DGS(KFN+Rect)	**33.62**	25.55	**33.91**	**31.03**

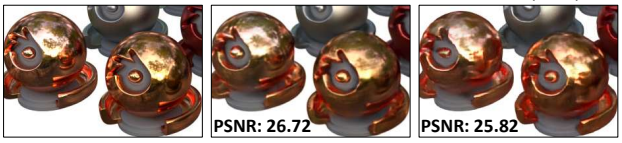

Figure 6: Our method struggles to produce accurate results in regions with specular reflections.

4 Conclusion

In this work, we propose a point cloud rendering method to enhance the neural point fields and 3DGS. We propose to utilize multiple points rather than single surface points to decide the color for each pixel. We also propose a pruning strategy and feature rectification to reduce the number of neural points. Besides, our proposed method can help 3DGS improve rendering quality and reduce storage. Experiments on major benchmarks have demonstrated the effectiveness of our method.

Acknowledgments

This work was supported by the Key R&D Program of Zhejiang under Grant (2025C03001), the Fundamental Research Funds for the Provincial Universities of Zhejiang (GK259909299001-023), the National Nature Science Foundation of China (62301198).

References

[Aliev *et al.*, 2020] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 696–712. Springer, 2020.

[Andersson *et al.*, 2020] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. ꟻLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2):15:1–15:23, 2020.

[Barron *et al.*, 2022] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

[Chang *et al.*, 2023a] Jen-Hao Rick Chang, Wei-Yu Chen, Anurag Ranjan, Kwang Moo Yi, and Oncel Tuzel. Pointersect: Neural rendering with cloud-ray intersection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8359–8369, 2023.

[Chang *et al.*, 2023b] MingFang Chang, Akash Sharma, Michael Kaess, and Simon Lucey. Neural radiance field with lidar maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17914–17923, 2023.

[Chen *et al.*, 2022] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022.

[Dai *et al.*, 2017] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[Dai *et al.*, 2020] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7830–7839, 2020.

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

[Godard *et al.*, 2018] Clément Godard, Kevin Matzen, and Matt Uyttendaele. Deep burst denoising. In *Proceedings of the European conference on computer vision (ECCV)*, pages 538–554, 2018.

[Ha *et al.*, 2017] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

[Hedman *et al.*, 2018] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018.

[Hu *et al.*, 2023] Tao Hu, Xiaogang Xu, Ruihang Chu, and Jiaya Jia. Trivol: Point cloud rendering via triple volumes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20732–20741, 2023.

[Huang *et al.*, 2023] Xiaoyang Huang, Yi Zhang, Bingbing Ni, Teng Li, Kai Chen, and Wenjun Zhang. Boosting point clouds rendering via radiance mapping. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 953–961, 2023.

[Jensen *et al.*, 2014] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[Kerbl *et al.*, 2023] 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.

[Li *et al.*, 2023] Zhuopeng Li, Lu Li, and Jianke Zhu. Read: Large-scale neural scene rendering for autonomous driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 1522–1529, 2023.

[Ma *et al.*, 2024] Jiahao Ma, Miaomiao Liu, David Ahmedt-Aristizabal, and Chuong Nguyen. Hashpoint: Accelerated point searching and sampling for neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4462–4472, 2024.

[Mildenhall *et al.*, 2018] Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2502–2510, 2018.

[Mildenhall *et al.*, 2020] 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 *ECCV*, 2020.

[Müller *et al.*, 2022] 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.

[Ost *et al.*, 2022] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition, pages 18419–18429, 2022.

[Radl *et al.*, 2024] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Transactions on Graphics*, 43(4), 2024.

[Rakhimov *et al.*, 2022] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. Npbg++: Accelerating neural point-based graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15969–15979, 2022.

[Reiser *et al.*, 2021] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021.

[Ronneberger *et al.*, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[Rückert *et al.*, 2022] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)*, 41(4):1–14, 2022.

[Shreiner *et al.*, 2009] Dave Shreiner, Bill The Khronos OpenGL ARB Working Group, et al. *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Pearson Education, 2009.

[Sun *et al.*, 2023] Weiwei Sun, Eduard Trulls, Yang-Che Tseng, Sneha Sambandam, Gopal Sharma, Andrea Tagliasacchi, and Kwang Moo Yi. Pointnerf++: A multi-scale, point-based neural radiance field, 2023.

[Xia *et al.*, 2020] Zhihao Xia, Federico Perazzi, Michaël Gharbi, Kalyan Sunkavalli, and Ayan Chakrabarti. Basis prediction networks for effective burst denoising with large kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11844–11853, 2020.

[Xu *et al.*, 2022] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022.

[Yu *et al.*, 2021] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021.

[Yu *et al.*, 2024] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19447–19456, June 2024.

[Zhang *et al.*, 2023] Yi Zhang, Xiaoyang Huang, Bingbing Ni, Teng Li, and Wenjun Zhang. Frequency-modulated point cloud rendering with easy editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 119–129, 2023.

[Zuo and Deng, 2022] Yiming Zuo and Jia Deng. View synthesis with sculpted neural points. *arXiv preprint arXiv:2205.05869*, 2022.