SteerNeRF: Accelerating NeRF Rendering via Smooth Viewpoint Trajectory

Sicheng Li Hao Li Yue Wang Yiyi Liao* Lu Yu**

Zhejiang University

Abstract

Neural Radiance Fields (NeRF) have demonstrated superior novel view synthesis performance but are slow at rendering. To speed up the volume rendering process, many acceleration methods have been proposed at the cost of large memory consumption. To push the frontier of the efficiency-memory trade-off, we explore a new perspective to accelerate NeRF rendering, leveraging a key fact that the viewpoint change is usually smooth and continuous in interactive viewpoint control. This allows us to leverage the information of preceding viewpoints to reduce the number of rendered pixels as well as the number of sampled points along the ray of the remaining pixels. In our pipeline, a low-resolution feature map is rendered first by volume rendering, then a lightweight 2D neural renderer is applied to generate the output image at target resolution leveraging the features of preceding and current frames. We show that the proposed method can achieve competitive rendering quality while reducing the rendering time with little memory overhead, enabling 30FPS at 1080P image resolution with a low memory footprint.

1. Introduction

Novel View Synthesis (NVS) is a long-standing problem in computer vision and computer graphics with applications in navigation [40], telepresence [60], and free-viewpoint video [51]. Given a set of posed images, the goal is to render the scene from unseen viewpoints to facilitate viewpoint control interactively.

Recently, Neural Radiance Fields (NeRF) have emerged as a popular representation for NVS due to the capacity to render high-quality images from novel viewpoints. NeRF represents a scene as a continuous function, parameterized by a multilayer perceptron (MLP), that maps a continuous 3D position and a viewing direction to a density and view-dependent radiance [23]. A 2D image is then obtained via volume rendering, i.e., accumulating colors along each ray.

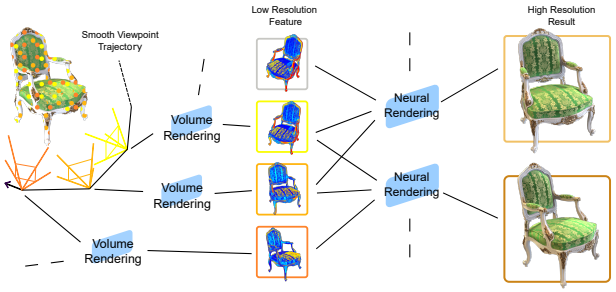

Figure 1. **Illustration**. We exploit smooth viewpoint trajectory to accelerate NeRF rendering, achieved by performing volume rendering at a low resolution and recovering the target image guided by multiple viewpoints. Our method enables fast rendering with a low memory footprint.

However, NeRF is slow at rendering as it needs to query the MLP millions of times to render a single image, preventing NeRF from interactive view synthesis. Many recent works have focused on improving the rendering speed of NeRF, yet there is a trade-off between rendering speed and memory cost. State-of-the-art acceleration approaches typically achieve fast rendering at the expense of large memory consumption [13, 56], e.g., by pre-caching the intermediate output of the MLP, leading to hundreds of megabytes to represent a single scene. While there are some attempts to accelerate NeRF rendering with a low memory footprint [16, 25], the performance has yet to reach cache-based methods. In practice, it is desired to achieve faster rendering at a lower memory cost.

To push the frontier of this trade-off, we propose to speed up NeRF rendering from a new perspective, leveraging the critical fact that the viewpoint trajectory is usually smooth and continuous in interactive control. Unlike existing NeRF acceleration methods that reduce the rendering time of each viewpoint *individually*, we accelerate the rendering by exploiting the information overlap between *multiple* consecutive viewpoints. Fig. 1 illustrates our SteerNeRF, a simple yet effective framework leveraging the *S*moo*T*h vi*E*wpoint traj*E*cto*R*y to speed up NeRF rendering. Here, "steer" also refers to a user smoothly controlling the movement of a camera during interactive real-time rendering.

* Corresponding author. ** Co-corresponding author.

Exploiting the smooth view trajectory, we can accelerate volume rendering by reducing the number of sample points while maintaining image fidelity using efficient 2D neural rendering. More specifically, our method comprises a rendering buffer, neural feature fields, and a lightweight 2D neural renderer. We first render a low-resolution feature map at a given viewpoint via volume rendering. The sampling range along each ray is reduced by fetching a depth map from the rendering buffer and projecting it to the current view. This effectively reduces the volume rendering computation as both the number of pixels and the number of samples for the remaining pixels are reduced. Next, we combine preceding and current feature maps to recover the image at the target resolution using a 2D neural renderer, i.e., a 2D convolutional neural network. The neural feature fields and the 2D neural renderer are trained jointly end-to-end. The combination of low-resolution volume rendering and high-resolution neural rendering leads to fast rendering, yet maintains high fidelity and temporal consistency at a low memory cost.

We summarize our contributions as follows. i) We provide a new perspective on NeRF rendering acceleration based on the assumption of smooth viewpoint trajectory. Our method is orthogonal to existing NeRF rendering acceleration methods and can be combined with existing work to achieve real-time rendering at a low memory footprint. ii) To fully exploit information from preceding viewpoints, we propose a simple framework that combines low-resolution volume rendering and high-resolution 2D neural rendering. With end-to-end joint training, the proposed framework maintains high image fidelity. iii) Our experiments on synthetic and real-world datasets show that our method achieved a rendering speed of nearly 100 FPS at an image resolution of 800×800 pixels and 30 FPS at 1920×1080 pixels. It is faster than other low-memory NeRF acceleration methods and narrows the speed gap between low-memory and cache-based methods.

2. Related Work

Advances in NeRF: Neural radiance fields [23] have received significant attention with photorealistic novel view synthesis performance. Meanwhile, the vanilla NeRF has several limitations. Many works have been conducted to address the limitations of NeRF, including unseen scene generalization [6, 21, 47, 57], dynamic scene representation [18, 19, 29–32, 34], sparse view training [8, 26], surface reconstruction [28, 46, 53, 54], and training acceleration [5, 39, 55]. In addition to representing a single scene, NeRF is widely applied in generative modeling [3, 4, 9, 12, 27, 37], and robotics [38, 61]. In this paper, we focus on rendering acceleration, which is critical for practical applications, e.g., interactive viewing control.

NeRF Rendering Acceleration: There are two common ways to accelerate NeRF rendering: reduce computation per sample or reduce the number of sample points.

In the first category, one line of works reduces the computation by caching MLP output [7, 11, 13, 14, 42, 45, 49, 56, 58] or using a voxel grid to represent the scene [39, 55]. These methods achieve fast rendering by retrieving the prestored information instead of querying a deep network. Another line of work replaces large MLPs with thousands of smaller ones [10, 35, 50] for reducing computation per point. Despite achieving fast rendering, all these methods scarify memory over time, indicating the trade-off between rendering speed and memory cost.

The second category reduces the number of sampling points on each ray based on the content to speed up rendering without increasing memory cost. Existing works in this area demonstrate that the number of sampling points can be effectively reduced to a small number [16, 25, 33], or even to one as in neural light fields based methods [1, 44]. However, these methods have not yet achieved the same speed as tabulation-based ones. Concurrent to our work, an efficient volumetric rendering toolbox NeRFAcc [17, 41], offers substantial speedups to NeRF by leveraging empty space skipping and early ray termination.

In contrast to the aforementioned methods, our method combines low-resolution volume rendering and high-resolution neural rendering using preceding frames to reduce rendering time. It is compatible with existing acceleration methods and has a small additional memory cost due to the lightweight neural renderer. More importantly, when combined with tabulation-based approaches for volume rendering, the resolution of the voxel grid for pre-caching could be reduced sufficiently since there is no need to render high-resolution content during the volume rendering stage.

Super-resolution for Rendered Content: A few research works focus on leveraging super-resolution techniques for rendering acceleration, particularly for game engine content. Xiao et al. [52] propose a temporal super-resolution design that takes low-resolution texture, depth, and motion vectors from the game engine as input. Another similar work is DLSS [2], a proprietary software of NVIDIA that is not fully disclosed. Unlike the above works, our method cannot obtain a high-precision motion vector for super-resolution. Instead, we take a volume-rendered noisy depth map to warp the preceding frame to align with the current frame and train the low-resolution rendering and the super-resolution network end-to-end. Note that Instant-NGP integrates DLSS in their recent update. Despite enhancing the rendering speed with convincing visual quality, we demonstrate in our experiments that its PSNR drops in highly textured regions. Another related work, NeRF-SR [43], generates higher-resolution images with low-resolution supervision but is not applicable for real-time rendering.

3. Method

In this work, we propose fully exploiting the smoothly changing viewpoints to accelerate the rendering process of NeRF. In general, we achieve rendering acceleration by reducing the total number of 3D points that need to be queried in volume rendering for each frame.

Fig. 2 gives an overview of our proposed pipeline consisting of a rendering buffer, neural feature fields, and a 2D neural renderer. Specifically, the rendering buffer saves low-resolution feature maps and depth maps of previous viewpoints. From the current viewpoint, a low-resolution feature map and depth map is rendered, accelerated by the rendering buffer. Next, the lightweight neural renderer takes the preceding and the current feature maps as input to generate the output image at the target resolution.

In the following, we first introduce preliminaries of NeRF model in Section 3.1. Next, we present the accelerated volume rendering in Section 3.2, the buffer-guided neural rendering in Section 3.3, and the training procedure in Section 3.4. Finally, we describe implementation details in Section 3.5.

3.1. Background

NeRF represents a scene as a continuous function f_θ parameterized by learnable parameters θ that maps a 3D point $\mathbf{x} \in \mathbb{R}^3$ and a viewing direction $\mathbf{d} \in \mathbb{S}^2$ and to a volume density σ and a color value $\mathbf{c}$:

$$f_\theta : (\mathbf{x} \in \mathbb{R}^3, \mathbf{d} \in \mathbb{S}^2) \mapsto (\sigma \in \mathbb{R}^+, \mathbf{c} \in \mathbb{R}^3) \tag{1}$$

Given a target viewpoint, the color $\mathbf{c}_r$ and depth d_r at a camera ray r is obtained via volume rendering integral approximated by the numerical quadrature [22]:

$$\mathbf{c}_r = \sum_{i=1}^{N} T_r^i \alpha_r^i \mathbf{c}_r^i \quad d_r = \sum_{i=1}^{N} T_r^i \alpha_r^i t_r^i \tag{2}$$

$$\alpha_r^i = 1 - \exp(-\sigma_r^i \delta_r^i) \quad T_r^i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{3}$$

where T_r^i and α_r^i denote transmittance and alpha value of a sample point $\mathbf{x}_i$.

Rendering Time: The rendering time of NeRF is proportional to the amount of computation required to render an image. Let $H \times W$ denote the target image resolution, N the number of samples on each ray and F the FLOPs of querying one sample's color and density. We can roughly estimate the amount of computation as $H \times W \times N \times F$. Existing NeRF acceleration strategies mainly focus on how to decrease the FLOPs F of each query by pre-caching the output of f_θ or directly using a voxel grid [13] [11] [42] [56] [55], thus leading to large memory consumption. There are a few attempts to reduce the sampling points N along the ray via early ray termination, empty space skipping [49] or adaptive sampling [16], yet using these techniques alone has not yet reached the performance of pre-cache based methods.

Our solution can elevate rendering speed from a new perspective, reducing the number of pixels $H \times W$ and the number of samples N for volume rendering via fully utilizing the smooth viewpoint trajectory. Besides, our solution can cooperate with the existing work to achieve high-speed rendering, leading to a higher rendering framerate while maintaining visual quality.

3.2. Accelerating Volume Rendering

We propose to learn neural feature fields that render a low-resolution feature map suited for the subsequent neural renderer. The acceleration of our framework comes from 1) rendering the feature map at a lower resolution and 2) reducing the sampling range guided by the rendering buffer.

Low-Resolution Feature Rendering: We render a feature map via volume rendering. Our neural feature fields maps the input $\mathbf{x}$ and the viewing direction $\mathbf{d}$ to a density value and a feature vector $\mathbf{f}$:

$$f_\theta : (\mathbf{x} \in \mathbb{R}^3, \mathbf{d} \in \mathbb{S}^2) \mapsto (\sigma \in \mathbb{R}^+, \mathbf{f} \in \mathbb{R}^K) \tag{4}$$

where K is the number of channels of our feature vector. Despite providing more information, rendering extra channels leads to very little overhead in time as we only expand the last layer of the MLP to predict more channels.

We can obtain a feature vector $\mathbf{f}_r$ at each ray r via volume rendering.

$$\mathbf{f}_r = \sum_{i=1}^{N} T_r^i \alpha_r^i \mathbf{f}_r^i \tag{5}$$

We render the feature vector at a subset of the rays, yielding a feature map $\mathbf{F} \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times K} = \{\mathbf{f}_r\}$. The corresponding low-resolution depth value $\mathbf{D} \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4}}$ is also rendered. Both the feature map $\mathbf{F}$ and $\mathbf{D}$ are stored in our rendering buffer for subsequent frames.

Buffer-Guided Sampling Range Reduction: The depth information of adjacent frames rendered previously provides a coarse scene geometry. Thus, warped depth from the preceding frame could be used as guidance to determine sampling positions and thus accelerate rendering. Specifically, given a low-resolution depth map $\mathbf{D}_{t',t'<t}$ of the previous frame, it is first unprojected to a 3D point cloud and then projected to the current viewpoint. More formally, the following unprojection and projection functions are applied to each pixel (u, v) of $\mathbf{D}_{t'}$ with depth $d_{t'}$:

$$\mathbf{p} = \boldsymbol{\xi}_{t'}^{-1} \mathbf{K}_l^{-1} d_{t'} [u, v, 1]^T \tag{6}$$

$$d_{t' \to t} [u_{t' \to t}, v_{t' \to t}, 1]^T = \mathbf{K}_l \boldsymbol{\xi}_t \mathbf{p} \tag{7}$$

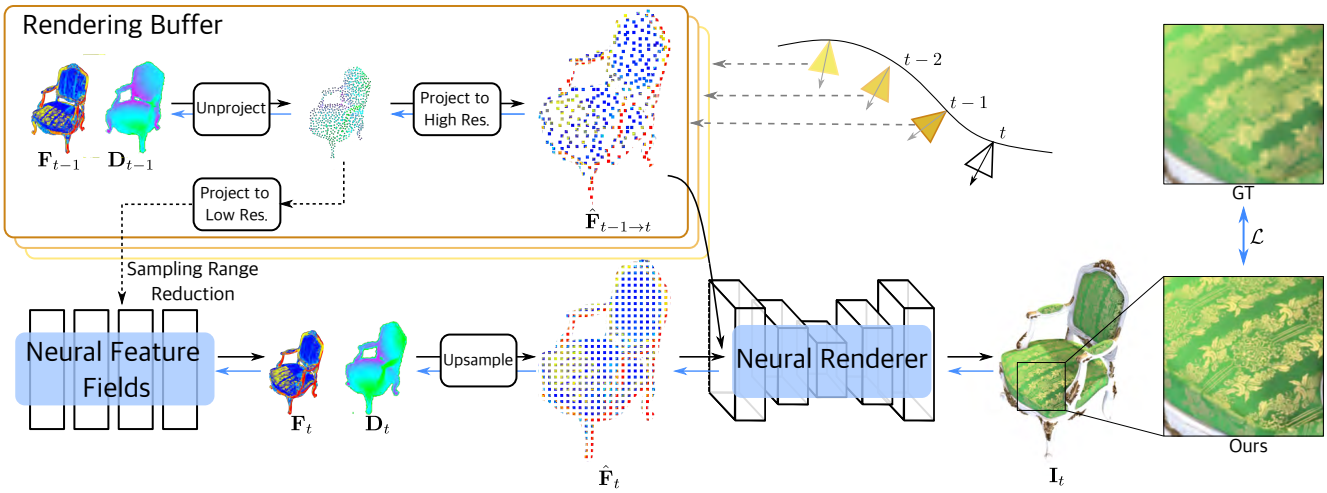

Figure 2. **SteerNeRF**. The rendering buffer saves low-resolution feature maps $\{\mathbf{F}_{t-L},\dots,\mathbf{F}_{t-1}\}$ and depth maps $\{\mathbf{D}_{t-L},\dots,\mathbf{D}_{t-1}\}$ of previous L viewpoints. At the current viewpoint t, a low-resolution feature map $\mathbf{F}_t$ and a depth map $\mathbf{D}_t$ are rendered accelerated by the rendering buffer. Next, the lightweight neural renderer takes as input the reprojected features maps at the high resolution $\{\hat{\mathbf{F}}_{t-L\to t},\dots,\hat{\mathbf{F}}_{t-1\to t}\}$ and the upsampled feature map $\hat{\mathbf{F}}_t$ to generate the output image $\mathbf{I}_t$. As illustrated by the blue arrows, during training, we apply the reconstruction loss $\mathcal{L}$ to an image patch and jointly optimize the entire model end-to-end, including preceding frames in the rendering buffer.

where $\mathbf{p}$ denotes a 3D point and $\mathbf{K}_l$ is the intrinsic matrix of the low-resolution image. This yields the reprojected depth map $\mathbf{D}_{t'\to t}$ where

$$\mathbf{D}_{t'\to t}(\lfloor u_{t'\to t}\rceil,\lfloor v_{t'\to t}\rceil)=d_{t'\to t}. \tag{8}$$

Note that here we simply round $(u_{t'\to t},v_{t'\to t})$ and observe a negligible impact on the performance. Given $\mathbf{D}_{t'\to t}$, the sampling range at frame t can be limited to the depth interval $[\mathbf{D}_{t'\to t}-\epsilon,\mathbf{D}_{t'\to t}+\epsilon]$ for rendering $\mathbf{F}_t$ and $\mathbf{D}_t$ at the camera viewpoint $\boldsymbol{\xi}_t$. This simple strategy further decreases the number of 3D sample points and accelerates the rendering of the low-resolution feature map.

3.3. Buffer-Guided Neural Rendering

Given the rendering buffer consisting of L preceding feature maps $\{\mathbf{F}_{t-L},\dots,\mathbf{F}_{t-1}\}$ and depth maps $\{\mathbf{D}_{t-L},\dots,\mathbf{D}_{t-1}\}$, we combine them with the feature map at the current viewpoint t to recover the output image $\mathbf{I}_t$ using a 2D neural renderer. We first project frames in the rendering buffer to the current viewpoint. Next, we use a 2D neural renderer to recover the target image.

Preceding Frames Reprojection: Inspired by natural video superresolution approaches, our method warps previous frames to align with the current frame to ease the task of the subsequent neural renderer. Instead of reprojecting the depth map to the low-resolution image as in Eq. 7, we directly project the 3D point cloud to the target resolution to achieve higher precision, i.e., maintain sub-pixel precision in terms of the low-resolution image:

$$d_{t'\to t}[\hat{u}_{t'\to t},\hat{v}_{t'\to t},1]^T=\mathbf{K}_h\boldsymbol{\xi}_t\mathbf{p} \tag{9}$$

where $\mathbf{K}_h$ denotes the intrinsic matrix of the high-resolution image. This allows us to obtain the reprojected high-resolution feature map $\hat{\mathbf{F}}_{t'\to t}\in\mathbb{R}^{H\times W\times K}$:

$$\hat{\mathbf{F}}_{t'\to t}(\lfloor\hat{u}_{t'\to t}\rceil,\lfloor\hat{u}_{t'\to t}\rceil)=\mathbf{F}_{t'}(u,v). \tag{10}$$

Neural Renderer: We use a lightweight 2D convolutional network for fast inference. The reprojected high-resolution feature maps $\{\hat{\mathbf{F}}_{t'\to t}\}$ are concatenated with the upsampled feature map $\hat{\mathbf{F}}_t$ and mapped to the output target image:

$$g_\theta:(\{\hat{\mathbf{F}}_{t'\to t}\},\hat{\mathbf{F}}_t)\mapsto\mathbf{I}_t\in\mathbb{R}^{H\times W\times 3} \tag{11}$$

In practice, we choose a simple modified U-Net as our neural renderer. Compared to traditional U-Net, we reduce the number of convolution layers for high-resolution features and increase the depth of convolution layers for low-resolution features. The simple adjustment allows us to greatly reduce the inference time and keep visual quality when the number of parameters is almost the same. We use an off-the-shelf inference acceleration toolbox, NVIDIA TensorRT, to optimize neural renderer to reduce the inference time.

3.4. Training

The training strategy is crucial to achieving high-quality novel view synthesis. In practice, we first pre-train our neural feature fields and then train the full model jointly in an end-to-end training fashion.

Pre-training: We pre-train our neural feature fields on the target resolution $H\times W$. Here, we render a high-resolution

feature map $\tilde{F} \in \mathbb{R}^{H \times W \times K}$ and apply an $L2$ reconstruction loss on the first three channels supervised by the high-resolution ground truth image. We leave other output channels without the constraint of supervision as pretraining on the first three channels is sufficient to learn reasonable volume density.

End-to-end Joint Training: With the pre-trained neural feature fields, we train our full model end-to-end using an $L2$ loss $\mathcal{L}$ on the final output image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$. Note that we do not apply reconstruction loss to the rendered feature map during end-to-end training and let the neural feature fields learn features suited for the 2D neural renderer. During training, the loss $\mathcal{L}$ is applied to image patches. As the training viewpoints are scattered in the space without a smooth trajectory, we generate a short sequence of preceding camera poses for each training image to train the buffer-based neural renderer. Note that this process does not introduce additional supervision as we only apply the loss $\mathcal{L}$ to the training viewpoints despite taking preceding feature maps as input.

3.5. Implementation Details

Network Architecture: Our method is compatible with different NeRF approaches for learning the neural feature fields. In this work, we implement our neural feature fields based on Instant-NGP [24] using a third-party PyTorch implementation[1]. This allows more efficient feature map rendering than the vanilla NeRF. We follow the original architecture of Instant-NGP that uses multi-resolution hash tables where the table length at each resolution is fixed to 2^{19}. Following Instant-NGP, empty space skipping and early ray termination is applied when rendering the low-resolution feature map. Regarding the 2D neural renderer, we adopt a shallow U-Net [36] with the detailed architecture described in the supplementary.

Distillation: When the number of training views is relatively small, the 2D neural renderer tends to overfit the training views, yielding degenerated performance on the test poses. In this case, we leverage a pre-trained NeRF model to synthesize more viewpoints as our pseudo ground truth by randomly sampling viewpoints within the available viewing zone. Adding the randomly sampled pseudo ground truth alleviates the overfitting problem.

Inference Optimization: Optimizing trained neural renderer for real-time inference and lower memory footprint is necessary. Thus, we leverage NVIDIA TensorRT to optimize the 2D neural renderer. Prior to testing, we optimize the 2D neural renderer into two versions in FP16 and INT8 precision separately.

[1] https://github.com/kwea123/ngp_pl

Measurement Platform: Our test system consists of an NVIDIA GTX 3090 consumer GPU, an Intel Core i7-10700K CPU with 3.80GHz, and 32GB of RAM.

4. Experiments

In this section, we evaluate our method's performance through quantitative comparisons with prior work, runtime breakdown analysis of two representative datasets, and extensive ablation studies to validate our design decisions.

Datasets: We evaluate our performance using the NeRF-Synthetic dataset, consisting of eight synthetic scenes rendered by Blender at 800×800 resolution, and a subset of the Tanks & Temples dataset [15], a real-world dataset at 1920×1080 resolution. We follow NSVF [20] in selecting a subset and cropping image backgrounds.

Baselines: We consider two groups of baseline methods based on their real-time rendering capabilities. The first group consists of vanilla NeRF and NSVF, while the second group comprises real-time rendering methods including PlenOctree, DIVeR, and Instant-NGP. We use the real-time version of DIVeR called DIVeR32 (RT). For Instant-NGP, we mainly compare to the third-party PyTorch implementation (denoted as Instant-NGP*) for a fair comparison. We further evaluate the official implementation of Instant-NGP to compare our relative speed gain with DLSS [2].

Metrics: We evaluate our method from efficiency, quality, and memory usage. The efficiency is measured by the number of frames per second. The quality is measured by PSNR, SSIM [48], and LPIPS [59]. Memory usage is measured by megabytes.

4.1. Comparisons to Baseline

We first compare our method with all baselines in Tab. 1. Here, we consider Instant-NGP* for a fair comparison. Tab. 1 (left) reports the results on Tanks & Temples dataset. All baseline and our methods show similar visual quality. However, our method shows the best FPS, outperforming the best baseline method, PlenOctree. In the meantime, the memory usage of our method is less than 2% of PlenOctree. Instant-NGP, which has similar memory usage to ours, can only render views at 5 FPS. NeRF and NSVF, both employing a large MLP, perform much slower in rendering efficiency. With such a trade-off between memory usage and rendering efficiency, our method satisfies 3D interaction along smooth viewport trajectories at 1080P resolution and 30FPS, as demonstrated by our qualitative results on Tanks & Temples in Fig. 3.

On NeRF-Synthetic dataset in Tab. 1 (right), all methods exhibit similar visual quality scores. PlenOctree achieves the highest FPS, albeit with a memory usage one to two orders of magnitude higher due to its network-free nature. In

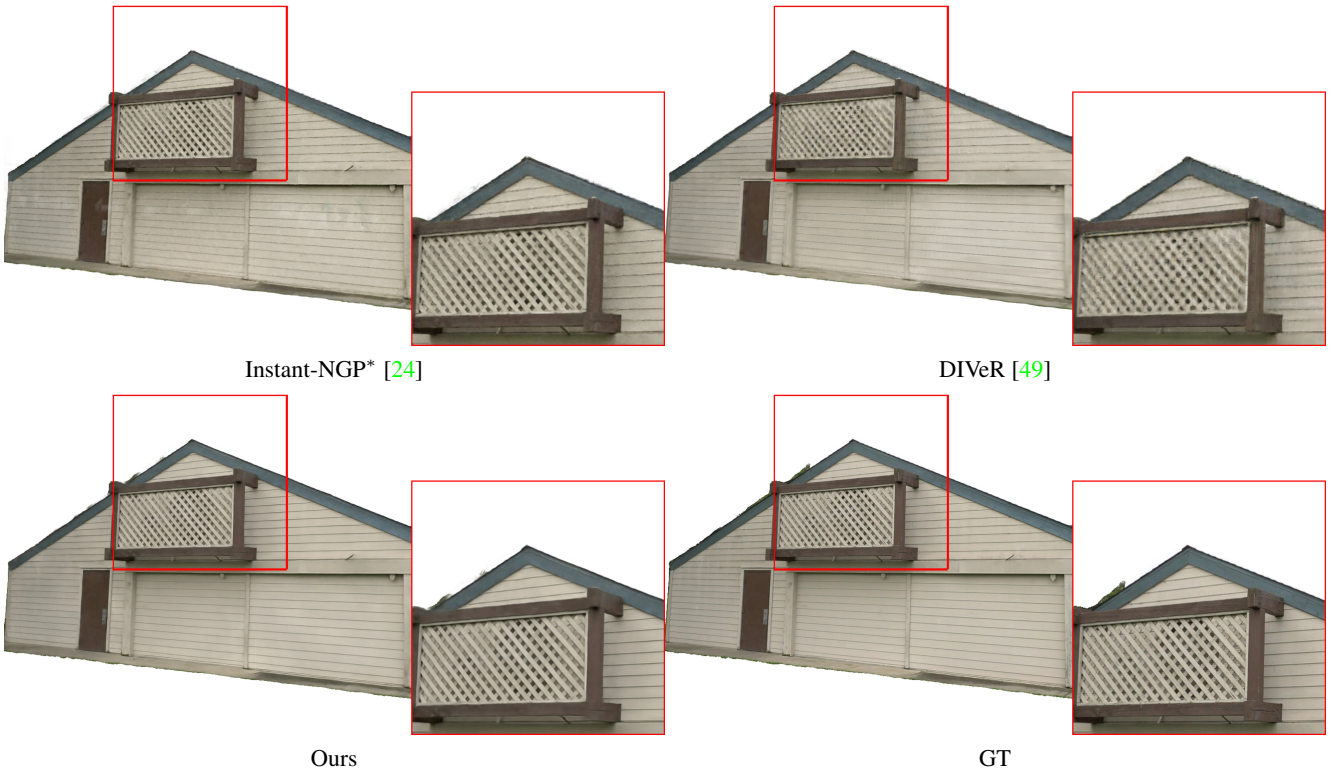

Figure 3. **Qualitative comparison on Tanks & Temples.**

Method	Tanks & Temples				NeRF-Synthetic				Mem.(MB)↓
	PSNR(dB)↑	SSIM↑	LPIPS↓	FPS↑	PSNR(dB)↑	SSIM↑	LPIPS↓	FPS↑	
NeRF [23]	28.32	0.890	0.198	0.005	31.01	0.947	0.081	0.02	**5**
NSVF [20]	28.40	0.900	0.153	0.06	31.74	0.953	0.047	0.23	-
KiloNeRF [35]	28.41	0.900	**0.092**	10.95	31.00	0.950	**0.030**	38.50	161
PlenOctree [56]	27.99	0.917	0.131	20.84	31.71	**0.958**	0.053	**167.70**	1930
DIVeR [49]	28.18	0.912	0.116	-	32.12	**0.958**	0.033	74.00	68
Instant-NGP*	**28.77**	0.918	0.136	5.00	**32.79**	0.957	0.055	35.41	25.2
Ours (FP16, L=0)	28.51	0.922	0.125	26.58	31.42	0.949	0.060	77.04	29.4
Ours (FP16)	28.65	**0.924**	0.121	27.24	31.60	0.954	0.058	75.19	29.4
Ours (INT8)	28.44	0.919	0.129	**30.90**	30.97	0.948	0.065	86.97	27.3

Table 1. **Quantitative results on Tanks & Temples and NeRF-Synthetic** show that our method could achieve high framerate rendering while keeping the memory footprint not too large. (**Best**, Second Best). Instant-NGP* refers to the third-party python implementation, which we also adopt to implement our neural feature fields. Our method has two versions (FP16, INT8), indicating two optimization precision for 2D neural renderer with NVIDIA TensorRT. Ours (FP16, L=0) refers to our method without preceding frames.

contrast, our method outperforms the others in FPS while maintaining similar memory usage. Notably, we observe two differences in FPS compared to the previous dataset: KiloNeRF is slower than Instant-NGP and the margin of our method against the other methods is relatively smaller. We explain both differences by the bandwidth of GPU, which may not be fully maximized due to the lower image resolution i.e. fewer times of ray marching or neural network inference. In summary, we consider that the parallelism of our method is able to push the frontier of the efficiency-memory trade-off, especially for high-resolution rendering. We present corresponding qualitative results on NeRF-Synthetic in Fig. 4.

We further compare our method to Instant-NGP with DLSS in Tab. 2. Note that the two groups in the table follow different codebases and thus are not directly compara-

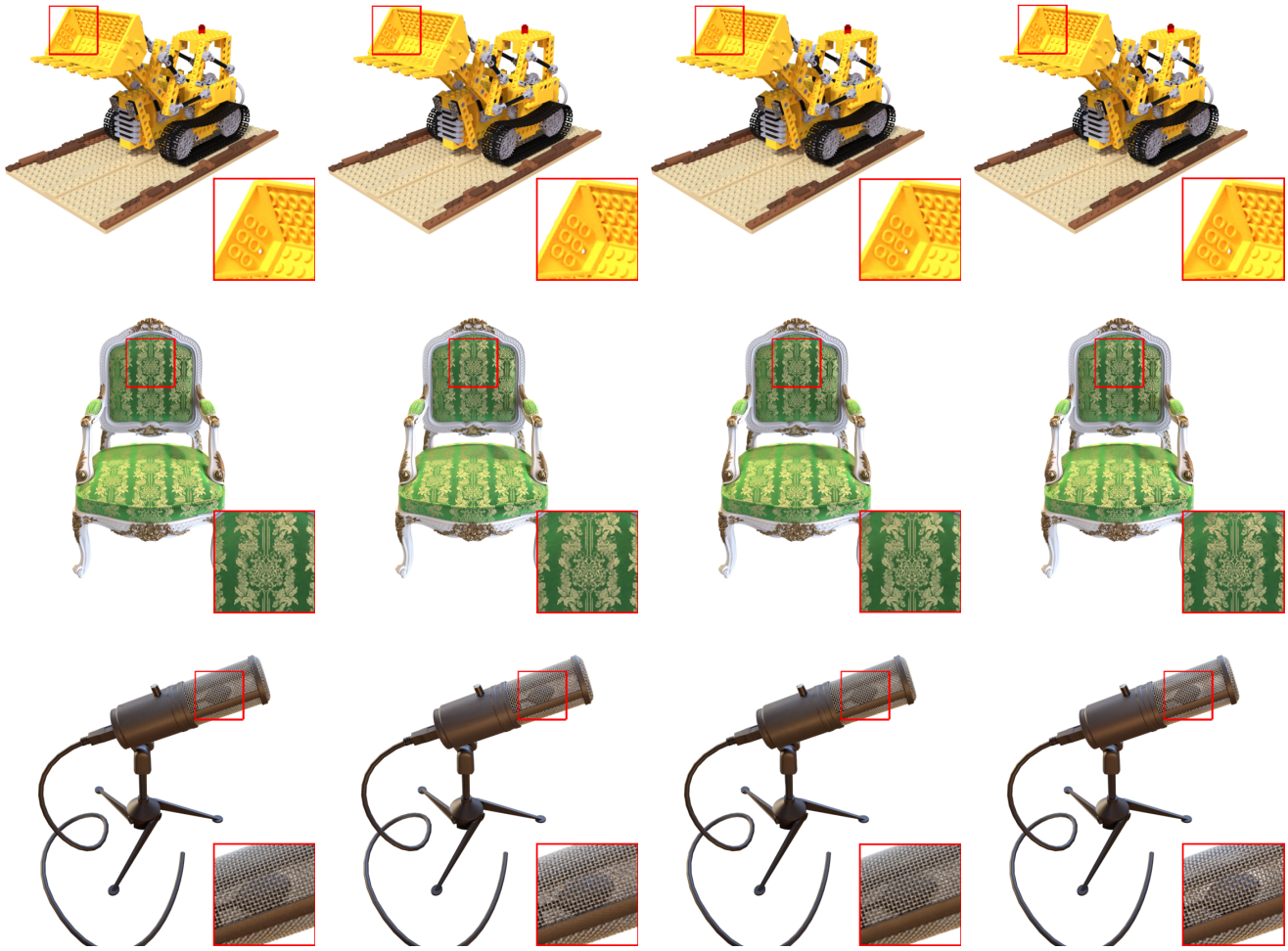

Figure 4. **Qualitative comparison on NeRF-Synthetic.**

ble. The upsampling rate of DLSS is set to 2 in the experiment. On Tanks & Temples, Instant-NGP with DLSS boosts 2× frame rate with a negligible loss in quality powered by dedicated AI processors named Tensor Core. Our method, based on a slower implementation of Instant-NGP, achieves a 5× speedup with minimal quality degradation. On NeRF-Synthetic, Instant-NGP with DLSS delivers 120+ FPS, but there is a decline in quality. Our method exhibits a better quality score than Instant-NGP with DLSS, despite not outperforming it in FPS. We leave qualitative comparisons to supplementary materials. Nonetheless, we believe our method is valuable to the community since DLSS is a closed box and our method significantly accelerates our implementation baseline in a directly comparable context.

	Tanks & Temples		NeRF-Synthetic	
	PSNR↑	FPS↑	PSNR↑	FPS↑
Instant-NGP	28.91	18.82	32.60	69.07
+ DLSS	28.70	41.30	30.75	122.59
Instant-NGP*	28.77	5.00	32.79	35.41
Our (FP16)	28.65	27.24	31.60	75.19

Table 2. **Comparsion with Instant-NGP w/ DLSS**

4.2. Runtime Breakdown

We report the average runtime of our method in Tab. 3 for Chair and Barn scenes, including the runtime of volume rendering, neural rendering, and depth reprojection. For volume rendering, $\{\mathbf{D}_{t'\to t}\}_{t'=t-1}$ refers to the depth reprojection to enable buffer-guided sampling range reduction. When this function is turned on, it reduces the volume rendering time (f_θ) to 85-90% of the original. As for the neural rendering part, the reprojection $\{\hat{\mathbf{F}}_{t'\to t}\}_{t'=\{t-2,t-1\}}$ takes longer as two frames are reprojected to a higher resolution. The reprojection can be further accelerated in future work by using customised CUDA kernels.

Module	Time(ms)			
	Barn		Chair	
$\{\mathbf{D}_{t'\to t}\}_{t'=t-1}$	0	0.11	0	0.11
f_θ	20.28	17.15	4.81	3.76
$\{\hat{\mathbf{F}}_{t'\to t}\}_{t'=\{t-2,t-1\}}$	3.21		2.40	
g_θ	12.00		3.71	
Total	35.49	32.47	10.92	9.98

Table 3. **Runtime breakdown** for Chair and Barn scenes

L: # Previous frames	0	1	2	3
PSNR(dB)	32.94	33.05	33.14	33.10
SSIM	0.959	0.964	0.968	0.967
LPIPS	0.035	0.034	0.034	0.032
Runtime(ms)	8.46	9.80	11.14	12.48

Table 4. **Comparison of number of preceding frames** on Chair.

w/o Joint K=3 Ours GT

Figure 5. **Ablation study.** Impact of different training configurations has been shown in the above visual examples.

4.3. Ablation Study

We conduct ablation studies on SteerNeRF using *Chair* scene from NeRF-Synthetic dataset.

Number of Preceding Frames: We evaluate the impact of the number of preceding frames L on visual quality and rendering time in Tab. 4. Increasing the number of preceding frames improves reconstruction quality but also leads to longer rendering times due to the additional warping operation. We observe that the quality gain per additional frame decreases as the number of preceding frames increases. Therefore, we can adjust this parameter flexibly to balance quality and rendering speed. Additionally, we investigate the impact of the sampling distance among the views during model training on reconstruction quality and provide further analysis in the supplementary materials.

Number of Feature Channels: We also verify the effect of the number of channels of feature images K. Experiments show that as the number of channels increases, the gain of visual quality decreases gradually in Tab. 5. Therefore, we ended up choosing six channels in total for our implementation. Note that employing feature rendering almost causes no rendering time overhead.

Joint Training: The necessity of joint training is validated in Tab. 6. The method without joint training means the parameters of neural feature fields are frozen when training the neural renderer. Joint training helps neural feature fields generate more expressive features compatible with the following neural renderer to synthesize higher-quality textures. In Fig. 5, we show visual examples of different training configurations.

K: # Feature channels	3	6	9
PSNR(dB)	32.53	33.14	33.21
SSIM	0.959	0.965	0.966
LPIPS	0.042	0.035	0.034

Table 5. **Comparison of number of feature channels** on Chair.

	PSNR	SSIM	LPIPS
Ours w/ joint training	33.05	0.965	0.038
Ours w/o joint training	31.98	0.957	0.053

Table 6. **Joint training** on Chair.

T of hash table	PSNR(dB)	SSIM	LPIPS	Mem.(MB)
2^{19}	33.05	0.965	0.038	29.2
2^{16}	33.03	0.961	0.039	8.1
2^{14}	32.70	0.953	0.039	5.0

Table 7. **Memory usage of neural feature fields** on Chair.

Memory Usage of Neural Feature Fields: In Tab. 7, we compare the visual quality and memory usage when neural feature fields are configured with different lengths of the hash table T, i.e., learnable parameters. We find that under our framework, even greatly reducing the length of the hash table in Instant-NGP, which implements neural feature fields, makes a slight impact on the final visual quality while significantly reducing memory usage.

5. Conclusion

We propose a new perspective for NeRF rendering acceleration by considering the smooth viewpoint trajectory during interactions. The main idea is to supersample the image rendered at the current viewpoint by taking preceding low-resolution features and depths. The experiments show that our method achieves real-time even when rendering 1080P images. As a limitation, training the 2D neural renderer is time-consuming. Moreover, we need to train a specialized neural renderer from scratch for each scene individually. In the future, we plan to train a neural renderer that generalizes well on different scenes only after short-time fine-tuning or even without fine-tuning.

Acknowledgements: This work is supported by the National Natural Science Foundation of China under Grant No. U21B2004, No. 62071427, No. 62202418, Zhejiang University Education Foundation Qizhen Scholar Foundation, and the Fundamental Research Funds for the Central Universities under Grant No. 226-2022-00145.

References

[1] Benjamin Attal, Jia-Bin Huang, Michael Zollhöfer, Johannes Kopf, and Changil Kim. Learning neural light fields with ray-space embedding. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[2] Andrew Burnes. Nvidia dlss 2.0: A big leap in ai rendering. https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-2-0-a-big-leap-in-ai-rendering/, 2020. 2, 5

[3] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[4] Eric R. Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. Pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

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

[6] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2

[7] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv.org*, 2022. 2

[8] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[9] Yu Deng, Jiaolong Yang, Jianfeng Xiang, and Xin Tong. GRAM: generative radiance manifolds for 3d-aware image generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[10] Stefano Esposito, Daniele Baieri, Stefan Zellmann, André Hinkenjann, and Emanuele Rodolà. Kiloneus: Implicit neural representations with real-time global illumination. *arXiv.org*, 2022. 2

[11] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien P. C. Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2, 3

[12] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. *Proc. of the International Conf. on Learning Representations (ICLR)*, 2022. 2

[13] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul E. Debevec. Baking neural radiance fields for real-time view synthesis. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 1, 2, 3

[14] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf efficient neural radiance fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[15] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 5

[16] Andreas Kurz, Thomas Neff, Zhaoyang Lv, Michael Zollhöfer, and Markus Steinberger. Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. 2022. 1, 2, 3

[17] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: A general nerf accleration toolbox. *arXiv preprint arXiv:2210.04847*, 2022. 2

[18] Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard A. Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[19] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[20] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 5, 6

[21] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Theobalt Christian, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

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

[23] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 1, 2, 6

[24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. on Graphics*, 2022. 5, 6, 7

[25] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. *Comput. Graph. Forum*, 40(4):45–59, 2021. 1, 2

[26] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. *arXiv.org*, 2021. 2

[27] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[28] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2

[29] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2

[30] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: a higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6):238:1–238:12, 2021. 2

[31] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2

[32] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[33] Martin Piala and Ronald Clark. Terminerf: Ray termination prediction for efficient neural rendering. In *Proc. of the International Conf. on 3D Vision (3DV)*. IEEE, 2021. 2

[34] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[35] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2, 6

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015. 5

[37] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[38] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. imap: Implicit mapping and positioning in real-time. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2

[39] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[40] 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[41] 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. *arXiv preprint arXiv:2302.04264*, 2023. 2

[42] Krishna Wadhwani and Tamaki Kojima. Squeezenerf: Further factorized fastnerf for memory-efficient inference. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3

[43] Chen Wang, Xian Wu, Yuanchen Guo, Song-Hai Zhang, Yu-Wing Tai, and Shi-Min Hu. Nerf-sr: High quality neural radiance fields using supersampling. In *Communications of the ACM*, pages 6445–6454. ACM, 2022. 2

[44] Huan Wang, Jian Ren, Zeng Huang, Kyle Olszewski, Menglei Chai, Yun Fu, and Sergey Tulyakov. R2l: Distilling neural radiance field to neural light field for efficient novel view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022. 2

[45] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Fourier plenoctrees for dynamic radiance field rendering in real-time. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[46] 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. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[47] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[48] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing (TIP)*, 13(4):600–612, 2004. 5

[49] Liwen Wu, Jae Yong Lee, Anand Bhattad, Yu-Xiong Wang, and David A. Forsyth. Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 6, 7

[50] Xiuchao Wu, Jiamin Xu, Zihan Zhu, Hujun Bao, Qixing Huang, James Tompkin, and Weiwei Xu. Scalable neural indoor scene rendering. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022. 2

[51] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint

video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[52] Lei Xiao, Salah Nouri, Matthew Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. Neural supersampling for real-time rendering. *ACM Trans. Graph.*, 39(4):142, 2020. 2

[53] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[54] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. 2020. 2

[55] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3

[56] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 1, 2, 3, 6

[57] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[58] Jian Zhang, Jinchi Huang, Bowen Cai, Huan Fu, Mingming Gong, Chaohui Wang, Jiaming Wang, Hongchen Luo, Rongfei Jia, Binqiang Zhao, et al. Digging into radiance grid for real-time view synthesis with detail preservation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022. 2

[59] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[60] Yizhong Zhang, Jiaolong Yang, Zhen Liu, Ruicheng Wang, Guojun Chen, Xin Tong, and Baining Guo. Virtualcube: An immersive 3d video communication system. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2146–2156, 2022. 1

[61] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 2