Retinal-NeRF: Real-time Rendering of Neural Point Fields Across Platform

1st Mandun Zhang
Xinjiang College of Science & Technology
School of Artificial Intelligence
Hebei University of Technology
Tianjin International Joint Center for Virtual Reality and Visual Computing
zhangmandun@scse.hebut.edu.cn

2nd Yuhan Liu
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China
lyhgz@outlook.com

3rd Shujia Cheng
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China
amazingmaozi@163.com

4th Yonghui Pang
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China
2602119659@qq.com

5th Jiale Shi
School of Artificial Intelligence
Hebei University of Technology
Tianjin, China
1804622570@qq.com

6th Zhidong Xiao*
National Centre for Computer Animation
Bournemouth University
Bournemouth, United Kingdom
zxiao@bournemouth.ac.uk

***Abstract*—Neural Radiance Fields (NeRFs) has emerged as a promising method for 3D reconstruction and novel view synthesis. However, NeRF-based methods rely on implicit encoding and heavy spatial sampling, which differ significantly from the widely used polygon mesh rasterization method. This discrepancy leads to a lack of support from common 3D software and hardware, resulting in inefficient rendering. In this work, we introduce a novel method called Retinal-NeRF, which combines point clouds and signed distance fields, for real-time rendering of 3D scenes. Our goal is achieved by converting NeRF into a representation that can be efficiently processed by standard graphics pipelines. To enhance the geometric accuracy of the scene, we propose two SDF regularization terms to improve the quality of the generated mesh. To ensure real-time rendering, we employ appearance decomposition technique to minimize the size of the MLP within the rendering pipeline. Notably, our findings indicate that the rendering speed of Retinal-NeRF is more than five times faster than existing real-time rendering techniques, without a significant loss in the quality of novel view synthesis.**

***Index Terms*—NeRF, Point Clouds, Signed Distance Fields**

**corresponding autho: Zhidong Xiao zxiao@bournemouth.ac.uk*

I. Introduction

Precise reconstruction and real-time rendering of 3D scenes are vital in fields such as medical imaging, lane navigation, and augmented/virtual reality (AR/VR), where these applications depend on accurate 3D geometry for physics-based simulation and require efficient rendering to meet the demands for real-time performance. Recently, Neural Radiance Fields (NeRF) [1]–[5] and 3D Gaussian Splat-ting (GS) [6] have emerged as significant breakthroughs in 3D reconstruction technologies, capable of rendering new viewpoint images with high fidelity, realism, and detail. Traditional implementations of NeRF utilize volume rendering algorithms, which require hundreds of sample points along each ray in the scene to traverse a sizeable multilayer perceptron (MLP), necessitating hundreds of millions of neural network evaluations to render a single pixel and resulting in a rendering time of approximately 30 seconds per image on mainstream high-end GPUs. Although Gaussian Splatting utilizes the explicit radiance field for 3D reconstruction in real time and provides editability to the application, the high computational cost on GPU memory poses significant challenges for rendering on mobile devices due to computational constraints. In stark contrast, polygon meshes, the most common representation in 3D vision, benefit from robust support across most graphics hardware, particularly in terms of high-speed rendering. This enables real-time rendering of high-fidelity 3D models in web browsers. Nonetheless, methods are still lacking for extracting meshes from NeRF and rendering them in parallel with GPUs under various lighting angles within browsers.

Fig. 1. Testing on the Synthetic 360° dataset [3], our method not only improved the rendering quality but also increased rendering speed by 20% compared to MobileNeRF.

To address these challenges, we introduce the Retinal-NeRF,

This work was supported by the National Key Research and Development Program of China under Grant 2022YFB3303800

a novel framework specifically designed for real-time rendering of NeRF across various platforms, as depicted in Figure 1. This framework's primary innovation is the decomposition of illumination into a compact MLP, which is then embedded into RGB textures. This approach utilizes the parallelism offered by fragment shaders to facilitate real-time scene rendering. For real-time, view-dependent rendering, we separate the appearance into a view-independent diffuse component and a view-dependent specular component. Consequently, the diffuse color is exported as a standard RGB image texture, while the specular color is output as a feature texture. When combined with the current viewing direction and processed through the compact MLP embedded in the fragment shader, this configuration enables the dynamic generation of view-dependent colors.

To summarize, our contributions are listed below.

- The presented Retinal-NeRF framework, which is five times faster than SNeRG on all test datasets, and surpasses MobileNeRF on synthetic datasets.
- Our approach utilizes a single HTML page with Threejs for rendering 3D models, boasting excellent cross-platform capabilities and compatibility with all tested devices.
- By incorporating an SDF (Signed Distance Function) and an appearance decomposition module, we represent learned scenes as meshes and feature textures. This approach significantly reduces memory usage and empowered our method to operate at real-time frame rates even on low-power mobile devices.

II. Related Work

A. Neural Radiance Fields

Traditional methods for 3D scene reconstruction typically rely on implicit functions, but recent advances have shifted focus on using multilayer perceptrons (MLPs), which provide greater expressive capabilities and require less memory. A prominent example is NeRF, which merges neural implicit functions with volume rendering. This approach utilizes a continuous five-dimensional function to accurately represent a static scene, resulting in notably high-quality renderings. Subsequent variations, such as Ref-NeRF [1] and Point-NeRF [4], have further extended this framework. However, NeRF-based rendering faces significant challenges due to its high computational demands. The demands of specialized rendering algorithms along with intense process of spatial sampling for estimating and integrating density and radiance substantially impede the performance of real-time rendering.

B. Efficient Rendering of Neural Radiance Fields

To address the slow rendering speeds associated with neural radiance fields, various research efforts have been presented. FastNeRF [8] stores a deep radiance map for each spatial position, then using the directional rays for querying this map to estimate the pixel values in the rendered images. PlenOctrees [9] employs an octree representation to compute ray intersections with voxels, outputting sequences of ray segments that are partitioned according to voxel boundaries, and subsequently applying NeRF's rendering formula. Instant NGP [10] maps input coordinates to multi-scale hash table indices of trainable feature vectors, enabling both accelerated training and rendering with a smaller neural network while maintaining quality. SNeRG bakes the NeRF model onto a sparse discrete grid, transitioning the high granularity of training representations into real-time, efficient rendering-ready representations. Similar to our approach, MobileNeRF [11] decomposes the MLP component of NeRF into an Encoder and Decoder, stores the feature vectors produced by the Encoder beforehand, and during the real-time rendering phase, directly retrieves these vectors using solely the Decoder for inference. MobileNeRF generates multiple rough mesh approximations, unlike our method, which employs the SDF zero-level set to create a high-precision traditional mesh, thus achieving broader compatibility. Additionally, our method incorporates an appearance decomposition step during training to abstract diffuse and specular colors, a feature absent in MobileNeRF.

In considering of real-time performance, only SNeRG and MobileNeRF have demonstrated the capacity to function on low-power devices without the need for CUDA support. Given that our method is designed for rendering on diverse platforms, we primarily compare it with these two approaches in our experiments.

III. Methodology

In this section, we introduce a framework designed for cross-platform real-time rendering of radiance fields (see Fig. 2), compatible with most devices that support OpenGL. Our model comprises two branches that produce volume density and color, respectively. The volume density branch aligns with the design principles presented by Point-NeRF. Differing from it, our MLP G predicts SDF values, denoted as d.These values are crucial for initializing mesh geometry as outlined in subsection A. Meanwhile, the color branch utilizes two shallow MLPs to separate the appearance into view-independent diffuse color c_d and view-dependent specular color c_s, while storing weights of these MLPs and the baked diffuse texture images (see subsection B). After training, the mesh, texture maps and shader weights are saved as OBJ file, PNG file and compact JSON files respectively. Subsequently, real-time rendering is executed on an HTML page using JavaScript, with the specified viewing direction (see subsection D).

A. SDF-based Neural Point Fields

Unlike previous method [12] that use MVS (Multi-View Stereo) to provide geometric information priors for NeRF, we utilize Point-NeRF, which employs the initial point cloud provided by TransMVS [13], to initialize the neural point cloud. The method initially utilizes a TransMVS network to predict depth. Then a 2D CNN is used to extract 2D feature maps from the input image to obtain per-point features. After aggregating the depth maps, we obtain a neural point cloud:

$$P = \{(p_i, f_i)\} \tag{1}$$

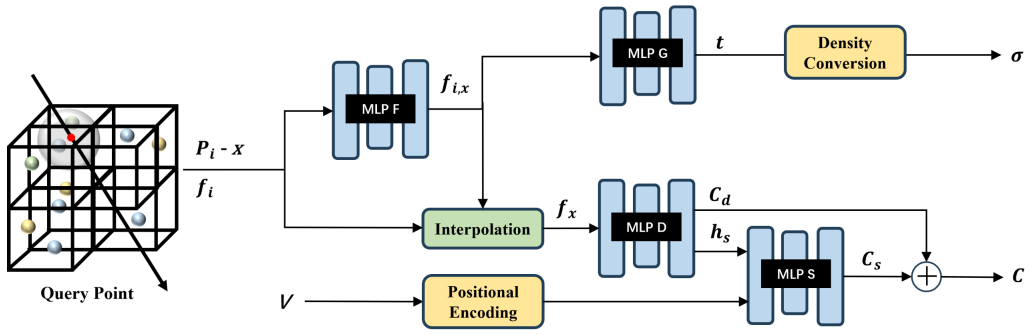

Fig. 2. A visualization of Retinal-NeRF's architecture. For simplicity, we have omitted the construction process of the neural point cloud.

where p_i and f_i denote point locations and point features respectively. During the point querying process, we established an explicit link between the shallow MLP F and MLP G to calculate the volumetric density of the eight surrounding neural points for the sampled point. To acquire a high-quality smooth surface from the zero level set of the predicted SDF values, we apply the method described in [14] to let the geometric MLP G predict the SDF value t at the sampled point x, instead of predicting the volumetric density σ, as done by Point-NeRF. Consequently, the model of our MLP G becomes:

$$d(x) = \sum_i \frac{w_{p_i}}{\Sigma w_{p_i}} G(f_{i,x}) \tag{2}$$

For a given point x in space, the Signed Distance Function (SDF) returns the distance t to the nearest surface, a value typically obtained via a large Multilayer Perceptron (MLP) [15], [16]. Based on this indirect method, we can execute Marching Cubes [17] over the predicted SDF values to obtain a high-quality, smooth surface. It is worth noting that, in contrast to approaches like BakedSDF [18], which employ region growing to address voids in meshes during the extraction phase, our method utilizes point growth operations throughout the training phase to fill in gaps present in the original point cloud. To render the color of a pixel, the SDF value t of the sampled point x along the ray can be converted into a density value for volumetric rendering. VolSDF [15] provides a method for converting density with a cumulative distribution function (CDF) that follows a Laplace distribution:

$$\sigma_\beta(t) = \begin{cases} \frac{1}{2\beta} exp(\frac{t}{\beta}) & \text{if } t \leq 0, \\ \frac{1}{\beta}(1 - \frac{1}{2} exp(-\frac{t}{\beta})) & \text{if } t > 0. \end{cases} \tag{3}$$

Where σ is the volumetric density, and β is a learnable parameter. By combining the predicted color $\mathcal{C}(x_i)$ of the sampling points along the ray (for more details, see section 3.2), the color $\mathcal{C}(r)$ of the current ray r can be computed according to the volumetric rendering equation of NeRF:

$$C(r) = \sum_i c(x_i)(exp(-\sum_{j=1}^{i-1} \sigma(x_j)\delta_j)\,(1 - exp(-\sigma(x_j)\delta_j))) \tag{4}$$

B. Appearance Decomposition

NeRF typically does not presuppose specific lighting conditions, instead, it implicitly fits them into a large MLP, which takes spatial coordinates as input to produce color and density. Although this approach significantly improves the image quality of novel view rendering, it poses challenges in separating illumination from model textures. To overcome this challenge, we utilize two shallow MLPs to separate the appearance into view-independent diffuse color c_d and view-dependent specular color c_s:

$$C_d, h_s = MLP_D(f_x) \tag{5}$$

$$C_s = MLP_S(h_s, V) \tag{6}$$

We introduce the viewing direction V as an additional input to the specular MLP, resulting in the final radiance $c(x)$ for each sampled point on the ray becoming:

$$c(x) = MLP_D(f_x) + MLP_S(h_s, V) \tag{7}$$

As shown in Fig. 3, we have successfully separated the diffuse and specular reflection components. The diffuse reflection color can be conveniently baked into an RGB image texture. Simultaneously, the specular reflection features h_s can also be baked into a texture, and to ensure the performance of the rendering pipeline, we insert the smallest possible MLP S into the fragment shader following the method described in MobileNeRF. Consequently, the specular reflection color can also be exported and rendered later (for more details, please refer to subsection D). It is important to acknowledge that, although our method bake specular reflection components into textures, previous research has indicated that this procedure may compromise rendering quality in real-world datasets [19], [20].

Final Color Diffuse Color Specular Color

Fig. 3. Renders separated diffuse and specular colors.

C. SDF Regularizations

Previous research [21] applied large-scale continuous MLPs to implicitly represent the SDF values of complete scenes, so Eikonal regularization is satisfied from the loss function. According to [22], [23], our approach may cause discontinuities in the projected SDF values due to the use of discrete neural point clouds. To better represent continuous smooth surfaces, we have introduced two regularization terms for the predicted SDF values. The first regularization is aimed at stabilizing the SDF prediction for points within the surface. Since SDF represents the distance of a point in space to its nearest surface, our model requires a stable prediction of negative SDF values within the surface region. To achieve this goal, we applied Cauchy loss to the density values obtained from the SDF transform:

$$\mathcal{L}_n = \frac{1}{N}\Sigma_i log\left(1 + \frac{(1-\beta\sigma_i^2}{c^2}\right) \tag{8}$$

Here, β is the parameter used in Eqn.3, and c is a hyperparameter. This regularization term is uniformly applied to all N sampled points in each iteration. The second regularization method is designed to improve continuity around the zero level set of the Signed Distance Function. Frequent fluctuations in the predicted SDF values near the zero level set can significantly affect the smoothness of the surface; hence, to ensure a stable decreasing curve of our estimated SDF when approaching the surface, we introduce a slope suppression regularization technique. This regularization penalizes the segments of the SDF curve that have positive slopes, as detailed in the subsequent equation:

$$\mathcal{L}_d = \sum_i \max(\Delta d_i, 0)\frac{\Delta d_i}{\delta_i^2 + \Delta d_i^2} \tag{9}$$

In this equation, $\Delta d_i = d(x_{i+1}) - d(x_i)$ represents the difference in the predicted SDF values between two sampling points, and δ_i is the actual distance between the two sampling points. Our loss function is essentially the same as the NeRF method, but with the two additional regularization terms described above:

$$\mathcal{L} = \mathcal{L}_{color} + \lambda_n\mathcal{L}_n + \lambda_d\mathcal{L}_d \tag{10}$$

Where $\mathcal{L}_{color}$ is the L1 loss between the ground truth and the rendered image, and λ_n and λ_d are loss hyperparameters. By combining the two additional regularization terms mentioned above, we can obtain more precise SDF values.

D. Rendering

The final output of our framework is a surface mesh compatible with common 3D hardware and software, along with two separate images, I_d and I_s. The images I_d and I_s are baked respectively from the diffuse color c_d and specular highlights h_s. Our exported mesh can be rendered in real-time using Three.js in an HTML webpage, and the diffuse texture I_d can be rendered on most modern browser-capable devices by interpreting it as an RGB texture. To render specular colors, we store the weights of MLP S as a JSON file and incorporate them into the fragment shader during rendering. This custom shader can run in parallel on all pixels and ultimately adds specular reflections to the diffuse component, thus achieving real-time specular reflection effects.

TABLE I
TESTING DEVICES. OUR TEST EQUIPMENT IS REPRESENTATIVE OF A GROUP OF LOW-POWER DEVICES. OUR DEVICES INCLUDE CELL PHONES, TABLETS, LAPTOPS, AND VR HEADSETS.

Device	Type	OS	Browser
IPhone	Phone	IOS 16	Safari
One plus	Phone	Android 12	Chrome
IPad	Tablet	IPadOS 16	Safari
Hololens 2	Headset	Windows Holographic	Edge
Lenovo	Laptop	Windows 11	Chrome

IV. EXPERIMENTS

We assessed the performance of Retinal-NeRF across a variety of datasets and devices by comparing it with well-known real-time rendering techniques.

Datasets. Our experiments encompassed both synthetic and real data using two benchmark datasets:

- Synthetic 360° dataset [3], comprising eight synthetic scenes with 800 x 800 pixel resolution, 100 training views, 100 validation views, and 200 test views apiece.
- Mip-NeRF 360 dataset [24], We used three publicly available unbounded real world outdoor scenes. Each scene comes in four different resolutions, with 185 images at each resolution.

Devices. Our study aims to evaluate the effectiveness of our proposed approach on mobile devices, particularly low-power gadgets like VR head-mounted displays, smartphones, and tablets that lack CUDA support. To ensure the comprehensiveness of our research, we also expanded the testing scope to include laptops running on the amd64 architecture. Detailed specifications and software configurations for each device used in the testing are listed in Table I.

A. Implementation Details

Fig. 2 illustrates the architecture of Retinal-NeRF. MLP F and MLP G consist of 4 layers and 1 layer respectively, while MLP D and MLP S both have 3 layers. We adopt the same positional encoding presented in NeRF for the input neural points' relative positions $p_i - x$ and the viewing direction V. In the training phase, we trained the model for 160k iterations and empirically set the loss hyperparameters $\lambda_n = 1e-3$; $\lambda_d = 5e-3$. As the initialization step of neural points is not the main focus of this study, for all experiments, based on TransMVS [13], the initial neural point field is obtained by employing PointNeRF [4]. All experiments were conducted on a single NVIDIA A40 GPU with 48GB of memory.

B. Main Results

In the following, we evaluate the performance of Retinal-NeRF in terms of both speed and accuracy. We initially present

TABLE II
RENDERING SPEED (FPS) ON DEVICES. WITH MOBILENERF (M-NERF) AND SNERG ACROSS VARIOUS TYPES OF DEVICES. AN ASTERISK "*" INDICATES THAT THE DEVICE'S FPS LIMIT HAS BEEN REACHED, WHILE A DASH "-" INDICATES THAT THE METHOD FAILED TO RUN.

Device	Synthetic 360°			Mip-NeRF 360		
	SNeRG	M-NeRF	Ours	SNeRG	M-NeRF	Ours
IPhone	-	55.87	60*	-	21.22	35.27
One plus	-	46.58	77.13	-	19.57	27.00
IPad	-	18.58	46.62	-	-	15.67
Hololens 2	-	12.90	14.68	-	4.30	5.50
Lenovo	28.56	120.13	149.16	-	36.85	33.33

Fig. 4. Qualitative Results on Synthetic 360°dataset.

the results from two synthetic scenes (Fig. 4), where compared to MobileNeRF, our method significantly preserves clearer object boundaries in the drumming scene. Furthermore, in the materials scene, Retinal-NeRF effectively captures more light effects such as brightness and reflections. It is noteworthy that MobileNeRF, lacking an appearance decomposition process, tends to produce holes and high-frequency noise on smooth surfaces. The study compared the frames per second (FPS) of two real-time rendering methods across five devices, as delineated in Table I, with the findings detailed in Table II. The data revealed that Retinal-NeRF's rendering speed substantially exceeds that of SNeRG and maintains a performance edge over the more recent MobileNeRF on the Synthetic 360° dataset. In synthetic environments, this performance disparity is particularly notable, with Retinal-NeRF achieving speeds up to five times faster than SNeRG. Moreover, in unbounded environments, our method improved rendering speeds by 20%. Critically, Retinal-NeRF demonstrated the capability to perform real-time rendering on various low-power devices, even achieving the maximum possible FPS supported by the hardware.

TABLE III
GPU MEMORY USAGE↓ ON SYNTHETIC 360°SCENES (UNIT IS MB).

Method	GPU Memory Usage							
	Chair	Drums	Lego	Mic	Materials	ship	Hotdog	Ficus
SNeRG	1254	4729	1253	1251	1253	2422	1253	8243
MobileNeRF	451	590	723	322	721	594	456	450
Ours	**154**	**170**	**181**	**180**	**178**	**190**	**177**	**165**

TABLE IV
DISK STORAGE↓ ON SYNTHETIC 360°SCENES (UNIT IS MB).

Method	Disk Storage							
	Chair	Drums	Lego	Mic	Materials	ship	Hotdog	Ficus
SNeRG	141	**44**	114	**22**	134	129	67	**43**
MobileNeRF	107	120	199	50	191	171	88	80
Ours	**54**	80	**79**	55	**64**	**79**	**46**	77

TABLE V
AVERAGE NUMBER OF VERTICES AND FACES (UNIT IS 10^3).

Method	Synthetic 360°	
	V	F
Ground Truth	631	873
SNeRG	N/A	N/A
MobileNeRF	494	224
Ours	**227**	**322**

Disk Storage and GPU Memory Usage. In Table III and IV we evaluate our method's Storage performance by comparing against SNeRG and MobileNeRF (the real-time model that produces the highest quality renderings after our own). Our method provides an unparalleled advantage in GPU Memory Usage, which is essential for achieving real-time performance across a wide range of low-power devices.

Polygon count. Table V shows the average number of vertices and faces produced by our method in comparison with SNeRG and MobileNeRF on the NeRF-synthetic dataset .

Rendering quality. We apply three metrics, peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [25], and perceptual LPIPS [26], for assessing the quality of the rendering. As presented in BakedSDF, baking specular features

TABLE VI
PSNR↑ ON SYNTHETIC 360° SCENES.

Method	PSNR↑							
	Chair	Drums	Lego	Mic	Materials	ship	Hotdog	Ficus
NeRF	33.00	25.01	32.54	32.91	**29.62**	28.65	36.18	30.13
SNeRG	33.24	24.57	33.82	32.60	27.21	27.97	34.33	29.32
MobileNeRF	34.09	25.02	34.18	32.48	26.72	29.06	35.46	30.20
Ours(re-time)	**34.24**	25.30	**34.50**	33.73	27.75	29.47	35.76	30.41
Ours(offline)	33.47	**25.70**	34.37	**34.52**	27.25	**29.82**	**36.80**	**32.47**

TABLE VII
SSIM↑ ON SYNTHETIC 360° SCENES.

Method	SSIM↑							
	Chair	Drums	Lego	Mic	Materials	ship	Hotdog	Ficus
NeRF	0.967	0.925	0.961	0.980	0.949	0.856	0.974	0.964
SNeRG	0.975	0.929	0.973	0.982	0.938	0.865	0.971	0.967
MobileNeRF	0.978	0.927	0.975	0.979	0.913	0.867	0.973	0.965
Ours(re-time)	0.976	0.927	0.976	0.982	0.931	0.874	0.976	0.969
Ours(offline)	**0.987**	**0.957**	**0.988**	**0.992**	**0.951**	**0.939**	**0.991**	**0.986**

TABLE VIII
LPIPS↓ ON SYNTHETIC 360° SCENES.

Method	LPIPS↓							
	Chair	Drums	Lego	Mic	Materials	ship	Hotdog	Ficus
NeRF	0.046	0.091	0.050	0.028	0.063	0.206	0.121	0.044
SNeRG	0.025	0.061	0.022	0.016	0.052	0.156	0.043	0.028
MobileNeRF	0.025	0.077	0.025	0.032	0.092	0.145	0.050	0.048
Ours(re-time)	0.029	0.080	0.025	0.026	0.080	0.138	0.058	0.056
Ours(offline)	**0.015**	**0.049**	**0.012**	**0.008**	**0.045**	**0.087**	**0.015**	**0.017**

into textures results in a loss of precision. Therefore, we utilize two models, an offline model (Chapter III A and B) and a real-time model (Chapter III D), for the comparison across eight objects in a synthetic dataset. Our real-time model successfully achieved a higher frame rate while maintaining comparable quality to MobileNeRF. It is worth noting that while measuring rendering quality is a supplementary experiment, our primary goal is to achieve a higher frame rate during rendering.

V. CONCLUSION

The Retinal-NeRF method presented in this paper is a cross-platform solution capable of real-time rendering of neural radiance fields. It can render lifelike images on low-power platforms without CUDA support at a frame rate five times higher than SNeRG. We propose an SDF regularization technique that enhances the capability to retrieve high-precision model meshes. Additionally, the appearance is baked into texture images to facilitate real-time rendering.

In conclusion, we have proposed an efficient framework capable of real-time rendering of neural radiance fields reconstructed from multi-view RGB images. In comparison with traditional NeRF-based methods, our method allows for real-time rendering on various low-power devices, and the quality of the generated meshes is sufficiently high for use in downstream applications.

REFERENCES

[1] D. Verbin, P. Hedman, B. Mildenhall, T. Zickler, J. T. Barron, and P. P. Srinivasan, "Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields," in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, Jun. 2022. doi:10.1109/cvpr52688.2022.00541.

[2] W. Hu et al., "Tri-MipRF: Tri-Mip Representation for Efficient Anti-Aliasing Neural Radiance Fields," Jul. 2023.

[3] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in Computer Vision – ECCV 2020,Lecture Notes in Computer Science, 2020, pp. 405–421. doi: 10.1007/978-3-030-58452-8_24.

[4] Q. Xu et al., "Point-NeRF: Point-based Neural Radiance Fields," in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, Jun. 2022. doi: 10.1109/cvpr52688.2022.00536.

[5] F. Zhu, S. Guo, L. Song, K. Xu, and J. Hu, "Deep Review and Analysis of Recent NeRFs".

[6] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," Aug. 2023.

[7] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking Neural Radiance Fields for Real-Time View Synthesis," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, Oct. 2021. doi: 10.1109/iccv48922.2021.00582.

[8] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin, "FastNeRF: High-Fidelity Neural Rendering at 200FPS," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, Oct. 2021. doi: 10.1109/iccv48922.2021.01408.

[9] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "PlenOctrees for Real-time Rendering of Neural Radiance Fields," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, Oct. 2021. doi: 10.1109/iccv48922.2021.00570.

[10] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," ACM Transactions on Graphics, pp. 1–15, Jul. 2022, doi: 10.1145/3528223.3530127.

[11] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi, "MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures".

[12] A. Chen et al., "MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, Oct. 2021. doi: 10.1109/iccv48922.2021.01386.

[13] Y. Ding et al., "TransMVSNet: Global Context-aware Multi-view Stereo Network with Transformers," in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, Jun. 2022. doi: 10.1109/cvpr52688.2022.00839.

[14] R. Liang, J. Zhang, H. Li, C. Yang, and N. Vijaykumar, "SPIDR: SDF-based Neural Point Fields for Illumination and Deformation," Oct. 2022.

[15] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume Rendering of Neural Implicit Surfaces," Jun. 2021.

[16] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction," arXiv: Computer Vision and Pattern Recognition,arXiv: Computer Vision and Pattern Recognition, Jun. 2021.

[17] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87, Not Known, Jan. 1987. doi: 10.1145/37401.37422.

[18] L. Yariv et al., "BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis," Feb. 2023.

[19] J. Munkberg et al., "Extracting Triangular 3D Models, Materials, and Lighting From Images," in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, Jun. 2022. doi: 10.1109/cvpr52688.2022.00810.

[20] X. Zhang, PratulP. Srinivasan, B. Deng, P. Debevec, WilliamT. Freeman, and JonathanT. Barron, "NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination," arXiv: Computer Vision and Pattern Recognition,arXiv: Computer Vision and Pattern Recognition, Jun. 2021.

[21] M. Oechsle, S. Peng, and A. Geiger, "UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, Oct. 2021. doi: 10.1109/iccv48922.2021.00554.

[22] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, Oct. 2021. doi: 10.1109/iccv48922.2021.01407.

[23] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance Fields without Neural Networks," in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, Jun. 2022. doi: 10.1109/cvpr52688.2022.00542.

[24] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields," in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, Jun. 2022. doi: 10.1109/cvpr52688.2022.00539.

[25] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," IEEE Transactions on Image Processing, pp. 600–612, Apr. 2004, doi: 10.1109/tip.2003.819861.

[26] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, Jun. 2018. doi: 10.1109/cvpr.2018.00068.