OGRMPI: An Efficient Multiview Integrated Multiplane Image based on Occlusion Guided Residuals

Dae Yeol Lee Guan-Ming Su Peng Yin
Dolby Laboratories, Sunnyvale, CA, USA
dvlee@dolby.com guanmingsu@ieee.org pyin@dolby.com

Abstract

Multiplane Image (MPI) is a volumetric scene representation method that uses multiple layers of texture (RGB) and alpha (A) planes. It offers high deployability due to its compatibility with standard codecs and low rendering complexity. While existing MPI methods have shown promising results, they are constrained by either a limited range of pose span or necessitates transmitting multiple MPIs. In this paper, we present a novel framework to generate an efficient single MPI that integrates the information from multiple views. The key idea is to utilize the surface opacity estimates (A) to locate and retrieve occluded RGB pixels from other camera views that share matching depth, which we call Occlusion Guided Residuals (OGR). Additionally, we introduce an inter-layer texture filler, which is a learned RGB texture on intermediate depth between MPI layers to deal with scenes with continuous depth with a limited number of MPI layers. We composite MPI using the aforementioned RGB textures and refine alpha layers through training with multiview rendering supervision. Thus, through iterations of training, we jointly optimize scene opacity (A) and textures (RGB) leading to an accurate MPI representation. Experiments on various multiview image and video datasets demonstrate that the proposed method achieves state-of-the-art performances with data efficiency. Notably, with just 16 layers, the proposed method attains performance on par with other methods that use twice the layer number or fuse multiple MPIs.

1. Introduction

Volumetric scene representation has been a persistent area of research interest. Especially with the recent advancements of technologies in autonomous driving or virtual reality, there is a growing interest in reconstructing volumetric images or videos from real-world captures.

Recently, neural scene representation has gained wide attention with the introduction of NeRF [1]. The NeRF models the scene as a continuous function of color and density through a neural network and offers the capability to simulate view-dependent effects like light reflectance. However, it faces issues in deployability particularly due to its limited generalizability to unseen scenes. Each scene or frame necessitates separate training. Moreover, the requirement to transmit Multi-Layer Perceptron (MLP) weights for every frame, coupled with time-consuming MLP computations during inference poses significant challenges for practical application. To address this issue, various works [2]-[8] are being proposed to reduce the model size and complexity. [6] and [7] accelerate the training process by directly optimizing on voxels. [2] and [3] use neural hash grids to use simpler models.

Multiplane Image (MPI) is a layer-based method that represents the scene with a discrete number of 2D layers. It stores fronto-parallel planes of a scene at a discretely sampled range of depths. Due to its high compatibility with conventional image/video codecs and its low computational requirements for view rendering, it is by far the most deployable solution among other volumetric representations. In [9]-[11] the methods used single view image to generate MPI representation. Especially, AdaMPI [11] uses self-attention operation to generate MPI layers with adaptive depth distances. This adaptive depth optimizes the allocation of layers per scene and facilitates efficient layer utilization. However, the method exhibits a limited range of pose span due to the restricted information from a single view. In [12]-[14], the authors use two or more views to generate MPIs that are robust to a larger pose span. LLFF [14] takes the multiple MPI fusion approach where it selects and fuses the optimal set of MPIs for each given pose. While being able to handle a broader span, its individual MPIs exhibit issues such as holes in cumulative alpha, making it challenging to use it independently. Consequently, rendering a view with LLFF often necessitates sending multiple MPIs, resulting in higher data transmission requirements. There are also works that expand beyond the traditional MPI definition of fronto-parallel RGBA layers. NeX [15] uses view-dependent coefficients and learned neural basis functions instead of RGB pixels. MINE [16] combines neural field with MPI. S-MPI [17] uses adaptively posed planes instead of fronto-parallel ones. But they tend to increase the data size or rendering complexity.

In this paper, we overcome the limitations of the existing MPI methods by generating an efficient single MPI that integrates information from multiple views. The main

contributions of the proposed method are as follows:

- We propose a MPI generation framework that can identify and retrieve occluded RGB pixels from other views. We denote these retrieved pixels as Occlusion Guided Residuals (OGR).
- Through iterations of training, the proposed framework jointly optimizes both scene opacity (A) and textures (RGB) leading to an accurate MPI representation.
- We introduce an inter-layer texture filler, which is a learned RGB texture to represent intermediate depth between MPI layers. It plays a vital role, especially in a scene with continuous depth.

We showcase the effectiveness of the proposed method, demonstrating its ability to render high-quality novel views that are on par with other methods that use twice the layer number or fuse multiple MPIs.

2. Preliminaries

MPI representation: The MPI can be collectively expressed as $\{(\boldsymbol{C}_i^s, \boldsymbol{A}_i^s)\}_{i=0}^{N_l-1}$, where three channel RGB plane of the i^{th} layer at camera pose s is denoted as $\boldsymbol{C}_i^s$ and an alpha plane is denoted as $\boldsymbol{A}_i^s$. The N_l indicates the number of layers. Throughout the paper, we denote the closest layer as layer 0 and the furthest as layer $N_l - 1$.

Novel view synthesis: Each MPI layer $(\boldsymbol{C}_i^s, \boldsymbol{A}_i^s)$ needs to be warped from the source view s to the novel target view t. This is done through applying a homography warping which establishes the correspondence between the source pixel coordinates (x^s, y^s) and the target pixel coordinates (x^t, y^t) given as:

$$[x^s, y^s, 1]^{\mathrm{T}} = \boldsymbol{K}^s \left(\boldsymbol{R} - \frac{\boldsymbol{t}\boldsymbol{n}^T}{z_i^s} \right) (\boldsymbol{K}^t)^{-1} [x^t, y^t, 1]^{\mathrm{T}}, \quad (1)$$

where $\boldsymbol{K}^s$ and $\boldsymbol{K}^t$ are the intrinsic camera parameters at the source (s) and target (t) positions, respectively. $\boldsymbol{R}$ and $\boldsymbol{t}$ are the extrinsic camera parameters describing rotation and translation between two camera positions. The $\mathbf{n}$ is the normal vector $[0,0,1]^{\mathrm{T}}$. The depth distance between the i^{th} layer to the reference camera position is z_i^s. The distance between two neighboring layers does not need to be fixed equal interval. They can be adaptive distances based on different contents to provide optimal novel view rendering [11]. We can render a novel view $\boldsymbol{I}^{(s\to t)}$ using warped MPI layers via:

$$\boldsymbol{I}^{(s\to t)} = \sum_{i=0}^{N_l-1} \boldsymbol{C}_i^{(s\to t)} \boldsymbol{W}_i^{(s\to t)}, \quad (2)$$

where,

$$\boldsymbol{W}_i^{(s\to t)} = \boldsymbol{A}_i^{(s\to t)} \cdot \prod_{j=0}^{i-1} \left(1 - \boldsymbol{A}_j^{(s\to t)}\right). \quad (3)$$

The $\boldsymbol{W}_i^{(s\to t)}$ represents the i^{th} layer's visibility weight.

Depth and Disparity: Depth information can take multiple forms of representation. It can be presented as the 'disparity' which is the horizontal shift between left and right images of a stereo pair. Representative depth databases [18]-[20] provide stereo image pairs along with their disparity maps, typically normalized between 0 and 1, for use in various computer vision tasks. Our framework is also based on this disparity representation. The depth at which each MPI layer is placed is specified using a disparity vector $\boldsymbol{d}^s = \{d_i^s\}_{i=0}^{N_l-1}$. The depth maps of multiple views are provided in the form of disparity maps.

When applying homography warping (1) on MPIs, we need to convert the MPI's disparity vector $(\boldsymbol{d}^s)$ to depth vector $(\boldsymbol{z}^s)$ which requires the scene's depth range $([z_{near}, z_{far}])$. This range is usually provided alongside the camera parameters, either within the dataset or predicted using methods such as COLMAP [21],[22]. We first convert the depth ranges to a corresponding disparity representation as, $d_{near} = 1/z_{near}$ and $d_{far} = 1/z_{far}$. Then, we apply min-max normalization on the disparity vector $(\boldsymbol{d}^s)$ to obtain the rescaled disparity vector $(\tilde{\boldsymbol{d}}^s)$. Denote the i-th element of the $\boldsymbol{d}^s$ as d_i^s and i-th element of the $\tilde{\boldsymbol{d}}^s$ as $\tilde{d}_i^s$. Then,

$$\tilde{d}_i^s = \left(\frac{d_i^s - \min(\boldsymbol{d}^s)}{\max(\boldsymbol{d}^s) - \min(\boldsymbol{d}^s)} \right) (d_{near} - d_{far}) + d_{far}. \quad (4)$$

By taking reciprocal

$$z_i^s = 1/\tilde{d}_i^s, \quad (5)$$

we obtain the depth value for each i-th MPI layer which covers the depth range $[z_{near}, z_{far}]$ of the scene.

3. Method

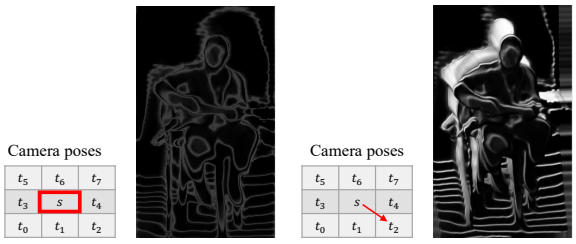
In this section, we start with the introduction to Occlusion Guided Residuals (OGR) which is a key component of our method. Then, we present the overall framework and the training procedures.

3.1. Occlusion Guided Residual

One of the key challenges in MPI representation is to reconstruct RGB textures that are occluded by the objects from previous layers. We solve this by collecting relevant textures from other views, which we call Occlusion Guided Residuals (OGR). Here, the views include the source view s which is the reference camera position that we construct MPI representation on. The rest of the views are referred to as target views t_v, $v \in [0, N_v - 2]$, where N_v is the total number of views available. The upper bound is set as $N_v - 2$ since we exclude the source view from here. For each view, we assume to have a corresponding image, a disparity map, and a camera pose.

Occlusion Map Construction: The first step of generating OGR is to construct an Occlusion Map $(\boldsymbol{O}^s)$, which indicates occluded regions within each layer due to the opacity of preceding layers. Let $\boldsymbol{A}^s$ be the initial alpha layers estimates of the source view. A vector indicating the disparity at which each layer is placed at is also given as $\boldsymbol{d}^s = \{d_i^s\}_{i=0}^{N_l-1}$. The $\boldsymbol{O}^s$ is defined as:

$$\boldsymbol{O}_i^s = (1 - \boldsymbol{T}_i^s)\, \boldsymbol{A}_i^s, \quad (6)$$



(a) Source view (b) Other target view

Figure 1: Visualization of the rendered occlusion map at various camera poses.

where,

$$\boldsymbol{T}_i^s = \prod_{j=0}^{i-1}\left(1 - \boldsymbol{A}_j^s\right). \tag{7}$$

This $\boldsymbol{T}_i^s$ is the transmittance of a ray up to the i-th layer. Thus, the term $(1 - \boldsymbol{T}_i^s)$ in equation (6) represents the ray obstruction up to the i-th layer. The $\boldsymbol{O}^s$ is constructed by identifying regions with both high ray obstructions $(1 - \boldsymbol{T}_i^s)$ and high alpha $(\boldsymbol{A}_i^s)$ values, which indicates opaque surfaces in each layer that were not revealed due to the ray obstruction from previous layers. Note that $\boldsymbol{O}^s$ is the same dimension as $\boldsymbol{A}^s$ given as $N_l \times 1 \times h \times w$, where h and w refer to vertical and horizontal resolutions of the source view $(\boldsymbol{I}^s)$, respectively.

Fig. 1 presents a visualization of the rendered occlusion map $(\boldsymbol{O}^s)$ on different camera poses. When rendering $\boldsymbol{O}^s$ at its source camera pose, as in Fig. 1(a), we observe a sparse map since occluded regions are not expected to be revealed much from the source camera perspective. However, when we warp to a new camera pose, as shown in Fig. 1(b), we begin to observe the revealed occluded regions, which is information we can use to extract relevant pixels from.

OGR Extraction: Fig. 2 illustrates the initial steps of the OGR extraction process. The occlusion map $(\boldsymbol{O}^s)$ is first warped to other target views following the procedures outlined in (1), (4), and (5). This warped occlusion map $\boldsymbol{O}^{(s\to t_v)}$, hold information on where the occluded region of each layer is placed on the target views. We form initial OGR $(\boldsymbol{I}_{\boldsymbol{OGR}}^{t_v})$ by first pulling every pixel from target image $(\boldsymbol{I}^{t_v})$ on to regions specified by $\boldsymbol{O}^{(s\to t_v)}$. However, as shown in Fig. 2, $\boldsymbol{I}_{\boldsymbol{OGR}}^{t_v}$ includes some elements that do not match the layer's disparity. For example, the presence of a "guitarman" in later background layers should be removed.

Fig. 3 outlines the process of generating masks for the disparity-based pixel removal process. We first generate disparity fidelity weights $(\boldsymbol{W}_{\boldsymbol{DF}}^{t_v})$ using the disparity map of the target view $(\boldsymbol{D}^{t_v})$ and the disparity vector of the warped MPIs $(\boldsymbol{d}^{(s\to t_v)})$. The disparity at which each warped MPI plane is placed at is derived as follows:

$$d_i^{(s\to t_v)} = d_i^s \cos(\theta_y)\cos(\theta_p). \tag{8}$$

where θ_y and θ_p refer to the yaw and pitch angles from the

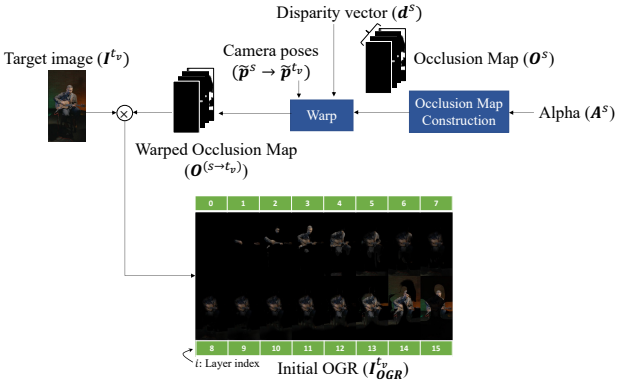


Figure 2: Initial OGR extraction procedure.

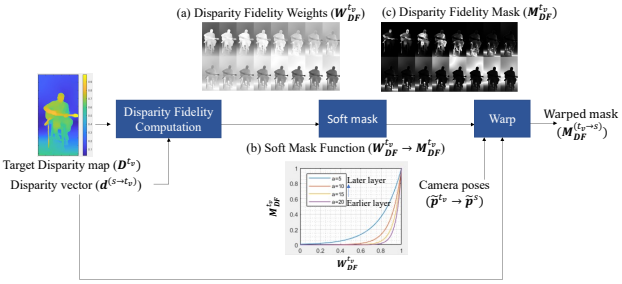


Figure 3: Disparity fidelity mask generation procedure

relative rotation between camera s and t_v.

The i-th layer plane of $\boldsymbol{W}_{\boldsymbol{DF}}^{t_v}$ is formulated as:

$$\boldsymbol{W}_{\boldsymbol{DF},i}^{t_v} = 1 - \frac{\left|\boldsymbol{D}^{t_v} - d_i^{(s\to t_v)}\right|}{\max\limits_{x\in[0,w-1],y\in[0,h-1],i\in[0,N_l-1]}\left(\left|\boldsymbol{D}^{t_v}(x,y) - d_i^{(s\to t_v)}\right|\right)}. \tag{9}$$

For each layer, eq. (9) computes the normalized absolute difference between the target view's disparity map $(\boldsymbol{D}^{t_v})$ and the current layer's disparity $(d_i^{(s\to t_v)})$ and subtracts this term from one. As depicted in Fig. 3(a), each layer will yield a plane $(\boldsymbol{W}_{\boldsymbol{DF},i}^{t_v})$ where the regions with higher value (closer to one) indicate the region with higher matching disparity to the current i-th layer's disparity. We further apply soft thresholding on $\boldsymbol{W}_{\boldsymbol{DF}}^{t_v}$ using exponential weighting function to highlight only the areas with a high degree of disparity fidelity. The soft masking procedure is formulated as

$$\boldsymbol{M}_{\boldsymbol{DF},i}^{t_v} = \frac{exp\left(a_i\, \boldsymbol{W}_{\boldsymbol{DF},i}^{t_v}\right)}{exp\,(a_i)}, \tag{10}$$

where we apply layer-adaptive parameter $a_i, i \in [0, N_l]$. This adaptive thresholding adjusts sensitivity across different layers. Since the errors in the front layers significantly impact the rendered scene, we assign a higher value of a_i to strictly threshold OGRs and minimize the transfer of inaccurate residuals. However, enforcing high a_i value on deeper layers restricts the overall OGR transfer, reducing disocclusion capability. Thus, we set a_i adaptively to take larger values on earlier layers and gradually decrease on later layers as follows:

$$a_i = a_{max} - \frac{i}{(N_l - 1)}(a_{max} - a_{min}). \quad (11)$$

For 16 layer representation, we set $a_{max} = 20$ and $a_{min} = 5$. Fig. 3(c) visualizes the resulting disparity fidelity mask ($\boldsymbol{M}_{\boldsymbol{DF}}^{t_v}$) with 16 layers. We see that only the regions with a high degree of disparity fidelity are indicated bright and the rest of the regions are indicated dark.

The derived $\boldsymbol{M}_{\boldsymbol{DF}}^{t_v}$ is then multiplied to the initial $\boldsymbol{I}_{\boldsymbol{OGR}}^{t_v}$, allowing only the pixels with matching disparities in each layer to remain while removing the rest. The processed final OGR ($\boldsymbol{I}_{\boldsymbol{OGR}}^{t_v}$) is then warped back to the source camera pose s which is denoted as $\boldsymbol{I}_{\boldsymbol{OGR}}^{(t_v \to s)}$. We also warp back the $\boldsymbol{M}_{\boldsymbol{DF}}^{t_v}$ to the source camera s which we denote as $\boldsymbol{M}_{\boldsymbol{DF}}^{(t_v \to s)}$. This warped back disparity fidelity mask is used for various purposes including OGR combination and inter-layer texture filler ($\boldsymbol{I}_{\boldsymbol{f}}$) masking.

Disparity Fidelity-based Combination: The collected $\boldsymbol{I}_{\boldsymbol{OGR}}^{(t_v \to s)}$ from multiple target views ($v \in [0, N_v - 2]$) are combined using disparity fidelity as the criteria. More specifically, for each layer's pixel location, we compare the $\boldsymbol{M}_{\boldsymbol{DF}}^{(t_v \to s)}$ values across all views and choose the view with the highest value. Again, the higher $\boldsymbol{M}_{\boldsymbol{DF}}^{(t_v \to s)}$ value indicates closer proximity to the corresponding MPI layer's disparity. The i-th layer of the combined OGR ($\hat{\boldsymbol{I}}_{\boldsymbol{OGR},i}$) is expressed as follows:

$$\hat{\boldsymbol{I}}_{\boldsymbol{OGR},i}(x, y) = \boldsymbol{I}_{\boldsymbol{OGR},i}^{(t_{Q(x,y,i)} \to s)}(x, y) \quad (12)$$

where

$$Q(x, y, i) = \underset{v}{\operatorname{argmax}} \left(\widetilde{\boldsymbol{M}}_{\boldsymbol{DF},i}^{(t_v \to s)}(x, y)\right), v \in [0, N_v - 2], \quad (13)$$

In (13), the $Q(\cdot)$ function serves as a target view index selector where, for each pixel position (x, y) and layer index (i), it compares the disparity fidelity values from all available view $v \in [0, N_v - 2]$. Then, it outputs the view index that gives the maximum value. Based on these selected view indices for each (x, y, i), we construct combined OGR ($\hat{\boldsymbol{I}}_{\boldsymbol{OGR}}$) by extracting pixel values from the respective view's $\boldsymbol{I}_{\boldsymbol{OGR}}^{(t_v \to s)}$.

One thing to note from eq. (13) is that we used lowpass filtered disparity fidelity mask ($\widetilde{\boldsymbol{M}}_{\boldsymbol{DF}}^{(t_v \to s)}$) computed as

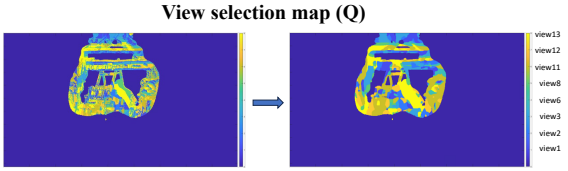
$$\widetilde{\boldsymbol{M}}_{\boldsymbol{DF},i}^{(t_v \to s)}(x, y) = \sum_{j=-L}^{L} \sum_{k=-M}^{M} G(j, k) \cdot \boldsymbol{M}_{\boldsymbol{DF},i}^{(t_v \to s)}(x + j, y + k), \quad (14)$$

where

$$G(j, k) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{j^2 + k^2}{2\sigma^2}\right). \quad (15)$$

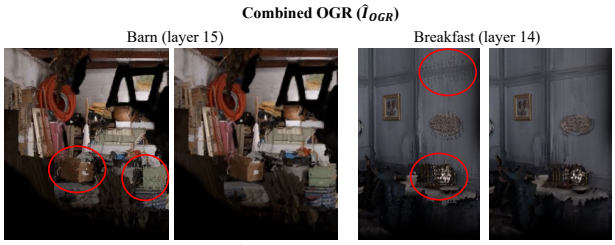
We set $\sigma = 3$ in (15) and $L = 6$, $M = 6$ in (14) to have Gaussian kernel window sampled out to two standard deviations for both horizontal and vertical directions.

As $\boldsymbol{M}_{\boldsymbol{DF}}^{(t_v \to s)}$ is reliant on the disparity map ($\boldsymbol{D}^{t_v}$) from eq. (9), local noise or inaccuracies on the provided $\boldsymbol{D}^{t_v}$ may significantly affect the quality of the combined OGR, leading to scatter artifacts where objects appear scattered and deformed. Thus, we applied the low pass operation in (14) to minimize local fluctuations in $\boldsymbol{M}_{\boldsymbol{DF}}^{(t_v \to s)}$. Fig. 4 (a) and (b) each visualizes view selection map (Q) before and after applying low pass operation on $\boldsymbol{M}_{\boldsymbol{DF}}^{(t_v \to s)}$ on content with noisy disparity map. Fig. 4 (a) shows severe local variations of view selections. If this view selection is used to construct $\hat{\boldsymbol{I}}_{\boldsymbol{OGR}}$ as using (12), it leads to scatter artifacts as depicted in Fig. 5(a) and (c). After the low pass operation, the OGRs are pasted in larger, more coherent clusters coming from the same view as shown in Fig. 4(b) which alleviated the artifacts as shown in Fig. 5 (b) and (d). Note that we are not applying the low pass operation on the OGR pixels but on the combination criteria $\boldsymbol{M}_{\boldsymbol{DF}}^{(t_v \to s)}$ in (13). Thus, there won't be any blurs on OGR pixels.



(a) Before low pass (b) After low pass

Figure 4: Visualization of the view selection map for combined OGR (a) before and (b) after applying low pass on disparity fidelity mask.



(a) Before low pass (b) After low pass (c) Before low pass (d) After low pass

Figure 5: Visualization of the combined OGR: (a),(c) before applying low pass operation and (b),(d) after applying low pass on disparity fidelity mask. Red circles indicate scatter artifact regions.

Alongside the $\hat{\boldsymbol{I}}_{\boldsymbol{OGR}}$ generated from equation (12), we also generate combined disparity fidelity mask ($\hat{\boldsymbol{M}}_{\boldsymbol{DF}}$). Similar to how we combined the OGRs using equation (12), we can form $\hat{\boldsymbol{M}}_{\boldsymbol{DF}}$ as follows:

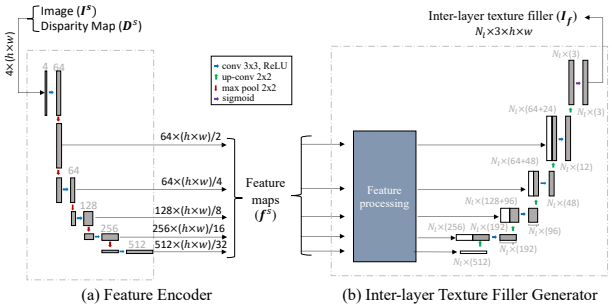
$$\hat{\boldsymbol{M}}_{\boldsymbol{DF},i}(x, y) = \boldsymbol{M}_{\boldsymbol{DF},i}^{(t_{Q(x,y,i)} \to s)}(x, y). \quad (16)$$

The resulting $\hat{\boldsymbol{M}}_{\boldsymbol{DF}}$ is a volume of dimension $N_l \times 1 \times h \times w$ that contains information on the extent to which $\hat{\boldsymbol{I}}_{\boldsymbol{OGR}}$ maintain disparity accuracy for each layer and will be used in the following composite process of the MPI RGB layers.

3.2. Inter-layer Texture Filler

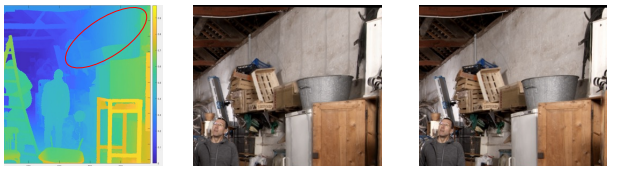
The conventional composite formulation [9], [11], [12] for constructing MPI RGB layers are as follows:

$$\boldsymbol{C}_i^s = \boldsymbol{T}_i^s \, \boldsymbol{I}^s + (1 - \boldsymbol{T}_i^s) \, \boldsymbol{I}_{\boldsymbol{b},i} \quad (17)$$



(a) Feature Encoder (b) Inter-layer Texture Filler Generator

Figure 6: Network architecture of (a) Feature Encoder (b) Inter-layer Texture Filler Generator.



(a) Continuous disparity along the side wall (b) Without inter-layer texture filler(I_f) (c) With inter-layer texture filler(I_f)

Figure 7: Effect of Inter-layer texture filler (I_f). (a) Content with a continuous disparity on the side wall, indicated in red circle. (b) Rendered novel view (b) without I_f and (c) with I_f.

where I_b refers to RGB textures that are occluded in the reference view. T_i^s is a composite weight primarily influenced by A^s as defined in equation (7). A typical approach of constructing I_b is to train an inpainting network. However, the inpainting network has quality limitations and may introduce artifacts related to temporal inconsistency when generating MPIs for multiple frames.

In our case, we have $\hat{I}_{OGR}$ which are RGB textures collected from multiple views. We additionally introduce inter-layer texture filler (I_f) representing the learned RGB textures capable of filling in surfaces that are not covered by source view or OGR pixels. Consequently, we formulate the RGB composition as follows:

$$C_i^s = T_i^s I^s + (1 - T_i^s)\{\hat{M}_{DF,i}\hat{I}_{OGR,i} + (1 - \hat{M}_{DF,i})I_{f,i}\}. \quad (18)$$

Here, we formulate the occluded textures I_b, from (17) as the convex combination of $\hat{I}_{OGR}$ and I_f, with greater weights assigned to $\hat{I}_{OGR}$ pixels in regions characterized by high $\hat{M}_{DF}$ values. This is rational as these regions closely match disparities of the current layer, making them the most suitable candidates for texture placement in those regions. The regions not covered by either $\hat{I}_{OGR}$ or I^s are presented as $(1 - T_i^s)(1 - \hat{M}_{DF,i})$ which serves as a mask for I_f.

Fig. 6 shows the architectures of the network components for generating I_f. We first input the source view's image (I^s) and disparity information (D^s) to a Feature Encoder in Fig. 6(a) to map the RGBD information into multi-resolution features (f^s). We use widely adopted U-net architecture [11], [12], [16]. Then, these features are fed into the Inter-layer Texture Filler Generator module in Fig. 6(b) which is a decoder component of the U-net. In the initial feature processing stage, each resolution component of the feature maps (f^s) are replicated N_l times, effectively expanding the final representation to N_l layers. The module receives weight supervision from gradients that propagate the aforementioned mask, adapting I_f to the regions with relevant textures accordingly.

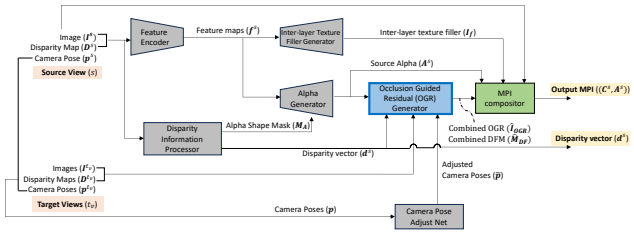


Figure 8: Overall flowchart of the proposed method. Zoom in for better clarity.

Fig. 7 demonstrates the effectiveness of the current MPI composition in eq. (18). Fig. 7(a) presents a disparity map of content with a continuous disparity along the side wall. Such continuous disparity poses challenges to MPI representation with a discrete set of disparities. Fig. 7(b) shows an example of a rendered novel view using MPI generated without inter-layer texture filler (I_f). More specifically, the MPI is generated using (17) where $I_b = \hat{I}_{OGR}$. In the figure, we observe inter-layer artifacts. In contrast, Fig. 7(c) demonstrates how I_f effectively learned to fill these intermediate disparity textures.

3.3. Overall Framework

In this section, we present the overall framework of our multi-view integrated MPI. Fig. 8 presents a high-level flow chart of our proposed method. Indicated in grey are modules with trainable network parameters. The framework here represents the flowchart at the inference stage and the procedural flow for network training will be presented in the following section. The inputs to the framework are multi-view images, corresponding disparity maps, and camera poses.

As our objective is to generate data efficient MPI, we draw inspiration from AdaMPI [11] and utilize their plane depth adjustment architecture to output a disparity vector optimized for the scene (d^s). The Disparity Information Processor module receives the source view's image (I^s) and disparity map (D^s) concatenated with initial disparity values $d_0(j)$, $j \in [0, N_l - 1]$, initially set as uniform disparity distance. Then, the module processes the inputs through ResNet-based context encoder followed by a self-attention operation to exchange feature-level geometry and appearance information among different layers. The module outputs the disparity vector (d^s), of dimension $N_l \times 1$, which indicates the disparity of each MPI layer and an Alpha Shape Mask (M_A), of dimension $N_l \times 1 \times h \times w$,

that contains approximate Alpha Shape on each MPI layer derived from $\boldsymbol{D}^s$ and $\boldsymbol{d}^s$.

The Feature Encoder module, in Fig. 6(a), receives the source view's image ($\boldsymbol{I}^s$) and disparity map ($\boldsymbol{D}^s$) as inputs. It outputs feature maps ($\boldsymbol{f}^s$) that capture hierarchical features of the given RGBD information. The $\boldsymbol{f}^s$ is then received by two types of decoders. One is the Inter-layer Texture Filler Generator covered in Sec. 3.2 and Fig. 6(b).

Another is the Alpha Generator which takes the feature maps ($\boldsymbol{f}^s$) from the Feature Encoder and the Alpha Shape Mask ($\boldsymbol{M_A}$) from the Disparity Information Processor to outputs alpha layers ($\boldsymbol{A}^s$) of dimension $N_l \times 1 \times h \times w$. This $\boldsymbol{A}^s$ is used for deriving occlusion maps and OGRs from Sec. 3.1. It is iteratively refined during the training procedure to jointly optimize the RGB textures and the surface geometry representation. It takes U-net decoder structure similar to Inter-layer Texture Filler Generator. Key differences are that the output dimension is adjusted to one-channel alpha layers and that we now take auxiliary information $\boldsymbol{M_A}$ which is concatenated to the $\boldsymbol{f}^s$.

The OGR Generator module, covered in Sec. 3.1, generates warped occlusion maps using $\boldsymbol{A}^s$ and $\boldsymbol{d}^s$ and collects OGR using target views' images, disparity maps, and camera poses. The module combines the OGRs from multiple views to output combined OGR ($\widehat{\boldsymbol{I}}_{\boldsymbol{OGR}}$) of dimension $N_l \times 3 \times h \times w$. The module also collects Disparity Fidelity Mask (DFM) from multiple views to form combined DFM ($\widehat{\boldsymbol{M}}_{\boldsymbol{DF}}$) of dimension $N_l \times 1 \times h \times w$ that contains information on the extent to which these combined OGR pixels maintain disparity accuracy for each layer.

The camera poses received by the OGR Generator are adjusted ones from the Camera Pose Adjust Net module. Similar approaches have been taken, mostly from NeRF-based works [24], [25], where they take initial estimates of camera poses obtained from structure from motion (SfM) methods and jointly optimize the neural scene and camera pose. Here, we take the COLMAP [21], [22] predictions as the initial estimates. For rotation matrices ($\boldsymbol{R}$), we convert them to quaternions ($\boldsymbol{q}$) for a simpler network and stability. The $\boldsymbol{q}$ and $\boldsymbol{t}$ each go through three layers of Fully Connected layers (FC) with skip connection to refine the initial pose estimates. The adjusted quaternions ($\tilde{\boldsymbol{q}}$) are converted back to rotation matrices ($\tilde{\boldsymbol{R}}$) and combined with the adjusted translation vectors ($\tilde{\boldsymbol{t}}$) to form the adjusted camera pose ($\tilde{\boldsymbol{p}}$).

The MPI compositor module forms the RGB layers ($\boldsymbol{C}^s$) of MPI using eq. (18) and concatenate then with alpha layers ($\boldsymbol{A}^s$) to form MPI $\{(\boldsymbol{C}_i^s, \boldsymbol{A}_i^s)\}_{i=0}^{N_l-1}$. The final outputs of the framework are the MPI $\{(\boldsymbol{C}_i^s, \boldsymbol{A}_i^s)\}_{i=0}^{N_l-1}$ and disparity vector $\{d_i^s\}_{i=0}^{N_l}$ which are all the data that is required to be sent as a bitstream to the edge device side for volumetric representation.

3.4. Training Networks

On every iteration of training, we select a subset of target views and form an MPI $\{(\boldsymbol{C}_i^s, \boldsymbol{A}_i^s)\}_{i=0}^{N_l-1}$ and disparity vector $\{d_i^s\}_{i=0}^{N_l}$ as in Fig. 8. Then, we warp and render the MPIs to other target views to supervise its volumetric representation capability. The loss function used for training the weights of the network modules is as follows:

$$L_{total} = L_{disp} + L_{render} + L_{occ}. \tag{19}$$

Disparity Loss (L_{disp}): This loss is a disparity-based constraint on the alpha shape ($\boldsymbol{M_A}$) and disparity vector ($\boldsymbol{d^s}$) used in [11] which is formulated as

$$L_{disparity} = \lambda_{order} L_{order} + \lambda_{mask} L_{mask}, \tag{20}$$

where

$$L_{order} = \frac{1}{N_l - 1} \sum_{i=0}^{N_l-2} max\,(0, d_{i+1}^s - d_i^s), \tag{21}$$

$$L_{mask} = \frac{1}{hwN_l} \sum_{i=0}^{N_l-1} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} \boldsymbol{M}_{\boldsymbol{A},i}(x,y) \cdot |\boldsymbol{D}^s(x,y) - d_i^s|. \tag{22}$$

Here, we set $\lambda_{order} = 1$ and $\lambda_{mask} = 10$. L_{order} is to penalize any switched orders among $\boldsymbol{d^s}$, ensuring that $\boldsymbol{d^s}$ is guided to maintain a sorted order. The L_{mask} provides disparity-based constraints on the alpha shape mask ($\boldsymbol{M_A}$). The $\boldsymbol{M_A}$ is a volume of dimension $N_l \times 1 \times h \times w$, where each layer contains the approximate alpha shape at the current disparity.

Render Fidelity Loss (L_{render}): The objective of L_{render} is to provide supervision to MPIs, ensuring that the warped and rendered target view ($\boldsymbol{I}^{(s \to t_v)}$) closely aligns with the ground truth target view ($\boldsymbol{I}^{t_v}$). For evaluating the fidelity between the two images, we employ L1 loss (L_1), SSIM [26] loss (L_{ssim}), and LPIPS [27] loss (L_{lpips}) as follows:

$$L_{render} = \lambda_1 L_1 + \lambda_{ssim} L_{ssim} + \lambda_{lpips} L_{lpips}. \tag{23}$$

Here, we set $\lambda_1 = 0.5$, $\lambda_{ssim} = 0.7$, and $\lambda_{lpips} = 1$.

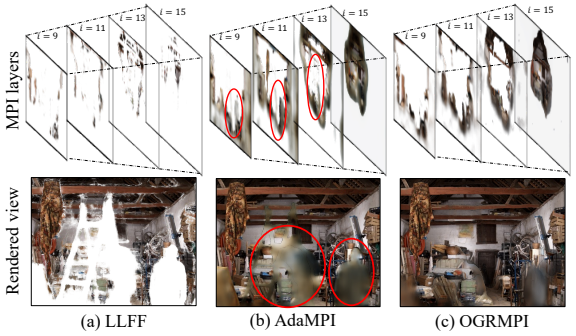
Occlusion Loss (L_{occ}): While multi-view-based training facilitates high-quality rendering on novel views, it can inadvertently introduce opacities optimized for rendering on expansive poses, potentially compromising the source view reconstruction or rendering on smaller poses. Thus, we add constraint L_{occ} to minimize unnecessary ray obstructions. L_{occ} is formulated as:

$$L_{occ} = \frac{1}{hw} \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \overline{\boldsymbol{O}}^{\boldsymbol{s}}(x,y), \tag{24}$$

where

$$\overline{\boldsymbol{O}}_i^{\boldsymbol{s}} = \sum_{i=0}^{N_l-1} \boldsymbol{O}_i^s \boldsymbol{W}_i^s. \tag{25}$$

In (25), $\boldsymbol{O}_i^s$ follows the definition in (6), and $\boldsymbol{W}_i^s$ is obtained from (3) with the target camera pose set as the source ($t = s$). The $\overline{\boldsymbol{O}}^{\boldsymbol{s}}$ measures the amount of ray obstruction present at the source view. Ideally, the



(a) LLFF (b) AdaMPI (c) OGRMPI

Figure 9: Visualization of MPI layers from different methods.

occluded regions are not revealed from the source view. Therefore, through having L_{occ}, we sparsify $\bar{\boldsymbol{O}}^{\boldsymbol{s}}$ and provide supervision for $\boldsymbol{A}^{s}$ to reduce unnecessary ray obstructions.

4. Experimental Results

4.1. Experimental Setup

Dataset: For evaluation, we used Multiview video dataset and LLFF dataset [14]. The Multiview video dataset consists of two contents 'Barn' and 'Breakfast' from MPEG immersive video (MIV) common test conditions [28] and a 'Guitarman' content. All contents provide 15 views with 30 frames. LLFF dataset is an image dataset providing 20~30 views per content. We use 'Flower', 'Horns', and 'Orchids' contents for evaluation. MIV content provides ground truth disparity maps. For all other contents, we used maps generated from Metashape [29].

Compared methods: The AdaMPI [11] serves as a baseline method that uses adaptive disparity distances optimized for the scene. Through comparison with AdaMPI, we demonstrate the disocclusion capability of our proposed method. We also compare with LLFF [14] for comparisons on methods that leverage multiple-view information differently. The proposed method integrates multi-view information into a single MPI, whereas LLFF selects and fuses the five closest MPIs based on the provided camera pose. AdaMPI and LLFF use the pre-trained model on large datasets. The proposed OGRMPI is a per-scene modeling method. Thus, it trains on the scene to learn accurate scene geometry and textures. When training on videos, we found that a single set of weights is sufficient to generate MPIs for multiple consecutive frames without a scene change, as demonstrated through the results on 30 frame videos in our study. There is no need to train separate weights for each frame. For every iteration, we train on random four target views across randomly chosen frames. We train our method for 1000 iterations with SGD optimizer with an initial learning rate is 5×10^{-4}, with momentum set as 0.9. AdaMPI and OGRMPI both require disparity maps as inputs, and we used the same ones for both methods.

Table 1: Multiview Video dataset - Source view reconstruction result

	# layers	# fused	Data size	Guitarman PSNR↑	Guitarman SSIM↑	Guitarman LPIPS↓	Breakfast PSNR↑	Breakfast SSIM↑	Breakfast LPIPS↓	Barn PSNR↑	Barn SSIM↑	Barn LPIPS↓
LLFF	16	1	× 1	22.31	0.836	0.169	27.97	0.927	0.047	28.58	0.953	0.039
	16	5	× 5	22.48	0.837	0.169	28.07	0.925	0.048	28.48	0.952	0.040
	32	5	× 10	22.55	0.816	0.214	32.52	0.964	0.024	29.94	0.965	0.029
AdaMPI	16	1	× 1	43.50	0.997	0.004	40.10	0.995	0.005	38.04	0.995	0.007
	32	1	× 2	43.81	0.996	0.003	41.30	0.996	0.004	38.28	0.995	0.007
OGRMPI	16	1	-	**46.58**	**0.998**	**0.001**	**46.86**	**0.999**	**0.001**	**44.40**	**0.999**	**0.001**

Table 2: Multiview Video dataset - View interpolation results on intermediate poses.

	# layers	# fused	Data size	FID↓ Guitarman	FID↓ Breakfast	FID↓ Barn
LLFF	16	1	× 1	183.51	72.01	72.51
	16	5	× 5	121.90	63.02	70.34
	32	5	× 10	102.07	**54.76**	63.17
AdaMPI	16	1	× 1	62.55	69.20	67.70
	32	1	× 2	61.89	63.43	66.69
OGRMPI	16	1	-	**58.27**	61.02	**62.14**

4.2. Evaluation Results

RGBA representation: Fig. 9 displays a selection of layers from the 16-layer MPI generated by different methods. Fig. 9(a) clearly reveals that a single LLFF MPI exhibits a lot of alpha holes, indicating the necessity for multiple MPI fusion to accurately represent a scene. Fig. 9(b) reveals AdaMPI's limitation with a single view approach, resulting in incorrect opaque connections from foreground objects to the background wall, as delineated by red circles. Such inaccurate opacities may impair the clarity of the textures behind them. Fig. 9(c) presents the layers of the proposed method. Our method recognizes the presence of textures on occluded regions through OGR and removes unnecessary opacities from alpha layers, accordingly. Thus, it accurately exposes background textures when viewed from other camera poses leading to high-quality novel view renders.

Source view reconstruction: While MPI representation can facilitate novel view rendering, at its base functionality, it should also be able to faithfully restore the source view. Thus, we evaluated the source view reconstruction performances of each MPI method using PSNR, SSIM, and LPIPS. Table 1 presents results averaged over 30 frames of the video contents, where the proposed method achieved the best performances among all methods.

View interpolation: We evaluate the view interpolation performances in various aspects. The first is on the intermediate positions between the available camera views. We take the eight adjacent neighbor camera positions with respect to the source camera (s) and denote them as $t_0 \sim t_7$. For the intermediate positions, we take the midpoint between the source camera position s and each of t_v, $v \in [0,7]$ and denote them as n_v. The derivation of intermediate position can also involve rotation interpolation for non-planar camera configuration, which

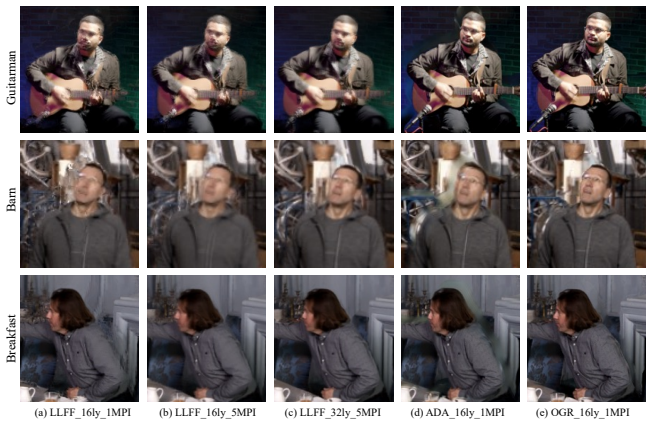


Figure 10: Visualization of views rendered at intermediate poses.

uses Spherical Linear Interpolation (SLERP) [30] in quaternion space. As we do not have associated ground truth images for these intermediate views, we evaluate the rendered quality using Fréchet Inception Distance (FID) [31], which is a representative Non-Reference (NR) model that compares the distributions of a real and generated image sets. In our experiment, we use nine ground-truth camera views ($t_0 \sim t_7$ and s) for the 'real' distribution and eight views ($n_0 \sim n_7$) for the 'generated' distribution. Since we have 30 frames, this translates into 270 real images and 240 generated images per content. Table 2 shows the FID scores on compared methods. Overall, the proposed method showed competitive or better performances compared to methods that use higher data size. Additionally, Fig. 10 presents the proposed method's ability to produce sharp images without occlusion artifacts.

We also evaluated the novel view rendering performance on a larger pose span, extending to other camera poses. Following [1] we hold out 1/8 of the camera views as a test set and evaluate the reconstruction performances on it. For LLFF, we ensure that these test views are not included as the fusion candidate. For OGRMPI, we omit the test views from the training procedure to prevent the model from extracting OGR pixels or deducing surface geometry from these views. Tables 3 and 4 present the view interpolation results for the Multiview Video and LLFF datasets, respectively. When compared to methods using the same data size, OGRMPI yielded the best results on all metrics. The LLFF fusion approach, which employs a larger data size, achieved higher PSNR and SSIM results possibly due to having access to multiple nearby MPIs. However, in LPIPS evaluation, OGRMPI outperformed all other methods across all contents, suggesting superior perceptual quality. Fig. 11 shows the rendered results on other camera views. The perceptual evaluations further confirm that our method produces sharp images with fewer artifacts.

Table 3: Multiview Video dataset - Results on other camera poses.

	# layers	# fused	Data size	Guitarman PSNR↑	SSIM↑	LPIPS↓	Breakfast PSNR↑	SSIM↑	LPIPS↓	Barn PSNR↑	SSIM↑	LPIPS↓
LLFF	16	1	× 1	20.57	0.653	0.262	23.53	0.800	0.113	22.50	0.838	0.131
	16	5	× 5	21.43	0.678	0.179	26.11	0.846	0.131	24.91	0.885	0.130
	32	5	× 10	**21.62**	**0.683**	0.179	**28.63**	**0.904**	0.104	**26.29**	**0.920**	0.111
AdaMPI	16	1	× 1	19.90	0.638	0.141	23.07	0.791	0.090	21.60	0.819	0.121
	32	1	× 2	20.10	0.665	0.129	24.11	0.805	0.085	20.82	0.799	0.123
OGRMPI	16	1	-	21.20	0.675	**0.094**	24.14	0.819	**0.070**	23.16	0.859	**0.084**

Table 4: LLFF dataset - Results on other camera poses.

	# layers	# fused	Data size	Flower PSNR↑	SSIM↑	LPIPS↓	Horns PSNR↑	SSIM↑	LPIPS↓	Orchids PSNR↑	SSIM↑	LPIPS↓
LLFF	16	1	× 1	25.29	0.925	0.087	19.71	0.720	0.181	16.70	0.617	0.234
	16	5	× 5	**28.71**	**0.954**	0.074	**20.99**	**0.753**	0.168	**18.66**	**0.708**	0.229
AdaMPI	16	1	× 1	18.58	0.776	0.150	13.35	0.452	0.437	12.58	0.393	0.316
	32	1	× 2	24.32	0.905	0.085	13.43	0.461	0.440	12.83	0.413	0.300
OGRMPI	16	1	-	26.84	0.942	**0.057**	20.27	0.739	**0.149**	16.83	0.648	**0.153**

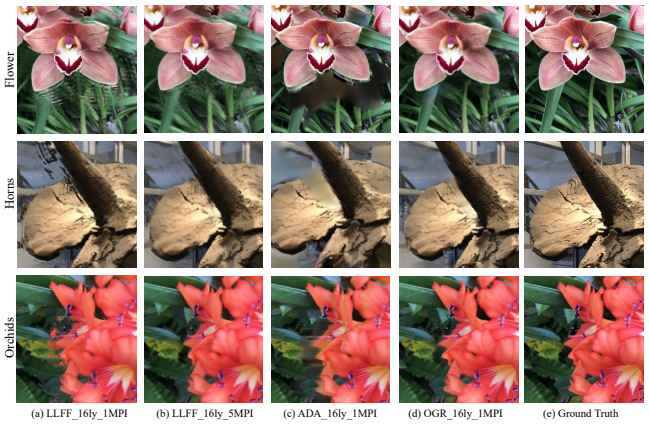


Figure 11: Visualization of views rendered at other camera poses.

5. Conclusion

In this paper, we proposed a novel framework to generate an efficient MPI representation that integrates information from multiple views. We showed how occlusion guided residuals can be used to jointly optimize both scene opacity (A) and textures (RGB), leading to an accurate MPI representation. Experimental results show that the proposed MPI excels in terms of source view fidelity and view interpolation capabilities. Furthermore, it demonstrates performance comparable to that of multiple MPI fusion methods with much higher data size, showcasing its data-efficient volumetric representation capability. Due to its high effectiveness and deployability, we envision the proposed method to be employed across a wide range of 3D applications.

Acknowledgements

We would like to express our gratitude to Vijay Kamarshi for his insightful discussions and assistance throughout the research.

References

[1] 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," *Commun. ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[2] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 1-15, 2022.

[3] R. Li, M. Tancik, and A. Kanazawa, "Nerfacc: A general nerf acceleration toolbox," arXiv preprint arXiv:2210.04847, 2022.

[4] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *Proc. Eur. Conf. Comput. Vis.*, Springer Nature Switzerland, pp. 333–350, Oct. 2022.

[5] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proc. IEEE/CVF. Int. Conf. Comput. Vis.*, pp. 5752–5761, 2021.

[6] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 5501–5510, 2022.

[7] C. Sun, M. Sun, and H. T. Chen, "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 5459–5469, 2022.

[8] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, "Nerv: Neural reflectance and visibility fields for relighting and view synthesis," *in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 7495–7504, 2021.

[9] R. Tucker and N. Snavely, "Single-view view synthesis with multiplane images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 551-560, 2020.

[10] Q. Li and N. K. Kalantari, "Synthesizing light field from a single image with variable mpi and two network fusion," *ACM Trans. Graph.*, vol. 39, no. 6, Article 229, 2020.

[11] Y. Han, R. Wang, and J. Yang, "Single-view view synthesis in the wild with learned adaptive multiplane images," in *ACM SIGGRAPH Conf. Proc.*, pp. 1-8, 2022.

[12] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *ACM Trans. Graph.*, vol. 37, no. 4, 2018.

[13] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely, "Pushing the boundaries of view extrapolation with multiplane images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 175–184, 2019.

[14] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar., "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–14, 2019.

[15] S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, and S. Suwajanakorn, "Nex: Real-time view synthesis with neural basis expansion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 8534–8543, 2021.

[16] J. Li, Z. Feng, Q. She, H. Ding, C. Wang, and G. H. Lee, "Mine: Towards continuous depth mpi with nerf for novel view synthesis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, pp. 12578–12588, 2021.

[17] M. Zhang, J. Wang, X. Li, Y. Huang, Y. Sato, and Y. Lu, "Structural multiplane image: Bridging neural view synthesis and 3D reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 16707–16716, 2023.

[18] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[19] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proc. German Conf. Pattern Recognit.*, Springer Int. Publ., vol. 36, pp.31-42, 2014.

[20] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 3260–3269, 2017.

[21] J. L. Schonberger and J. M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 4104–4113, 2016.

[22] J. L. Schönberger, E. Zheng, J. M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.*, Springer Int. Publ., vol. 14, pp. 501–518, 2016.

[23] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, pp. 3828–3838, 2019.

[24] C. H. Lin, W. C. Ma, A. Torralba, and S. Lucey, "Barf: Bundle-adjusting neural radiance fields," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, pp. 5741–5751, 2021.

[25] S. F. Chng, S. Ramasinghe, J. Sherrah, and S. Lucey, "Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, Springer Nature Switzerland, pp. 264–280, Oct. 2022.

[26] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[27] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 586–595, 2018.

[28] MPEG document MDS22394 WG 4 N 307, "Common test conditions for MPEG immersive video," Feb. 2023.

[29] Agisoft, "Metashape," available online: https://www.agisoft.com [Accessed on: Mar. 2024]

[30] K. Shoemake, "Animating rotation with quaternion curves," in *Proc. Annu. Conf. Comput. Graph. Interact. Tech.*, pp. 245–254, 1985.

[31] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.