NEURAL PLENOPTIC SAMPLING: CAPTURE LIGHT-FIELD FROM IMAGINARY EYES

Anonymous authors
Paper under double-blind review

ABSTRACT

The Plenoptic function (Adelson et al., 1991) describes the light rays observed from any given position in every viewing direction. It is often parameterized as a 5-D function $L(x, y, z, \theta, \phi)$ for a static scene. Capturing all the plenoptic functions in the space of interest is paramount for Image-Based Rendering (IBR) and Novel View Synthesis (NVS). It encodes a complete light-field (*i.e.*, lumigraph) therefore allows one to freely roam in the space and view the scene from any location in any direction. However, achieving this goal by conventional light-field capture technique is expensive, requiring densely sampling the ray space using arrays of cameras or lenses. This paper proposes a much simpler solution to address this challenge by using only a small number of sparsely configured camera views as input. Specifically, we adopt a simple Multi-Layer Perceptron (MLP) network as a universal function approximator to learn the plenoptic function at every position in the space of interest. By placing virtual viewpoints (dubbed 'imaginary eyes') at thousands of randomly sampled locations and leveraging multi-view geometric relationship, we train the MLP to regress the plenoptic function for the space. Our network is trained on a per-scene basis, and the training time is relatively short (in the order of tens of minutes). When the model is converged, we can freely render novel images. Extensive experiments demonstrate that our method well approximates the complete plenoptic function and generates high-quality results.

1 INTRODUCTION

Image-Based Rendering (IBR) for view synthesis is a long-standing problem in the field of computer vision and graphics. It has a wide range of important applications, *e.g.*, robot navigation, film industry, AR/VR applications.

The plenoptic function, introduced by Adelson et al. (1991), offers an ultimate solution to this novel view synthesis problem. The plenoptic function captures the visual appearances of a scene viewed from any viewing direction (θ, ϕ) and at any location (x, y, z). Once a complete plenoptic function (i.e. the light-field) for the entire space is available, one can roam around the space and synthesize free-viewpoint images simply by sub-sampling the plenoptic light-field.

To model the plenoptic function, the best-known methods in the literature are the light field rendering and the lumigraph (Levoy & Hanrahan, 1996; Gortler et al., 1996). These approaches interpolate rays instead of scene points to synthesize novel views. However, they require the given camera positions to be densely or regularly sampled or restrict the target image to be a linear combination of source images. Unstructured light-field/lumigraph methods (Buehler et al., 2001; Davis et al., 2012) were proposed to address this limitation; they do so by incorporating geometric reconstruction with light ray interpolation.

This paper introduces a novel solution for plenoptic field sampling from a few and often sparse and unstructured multi-view input images. Since a plenoptic function is often parameterized by a 5D function map, we use a simple Multi-Layer Perceptron (MLP) network to learn such functional map: the MLP takes a 5D vector as input and outputs an RGB color measurement, *i.e.*, $\mathbb{R}^5 \rightarrow \mathbb{R}^3$.

However, capturing the complete plenoptic function for a scene remains a major challenge in practice. It requires densely placing many physical cameras or moving a camera (or even a commercial light-camera) to scan *every point and in every direction*.

To address this challenge, this paper uses an MLP to *approximate* the plenoptic function (*i.e.*, the entire light field), by placing thousands of virtual cameras (*i.e.*, imaginary eyes) during the network training. We use the available physical camera views, however a few and sparsely organized, to provide multi-view geometry constraints as the self-supervision signal to supervise the training of the MLP networks.

We introduce *proxy-depth* as a bridge to ensure that the multi-view geometry relationship is well respected during the training process. Those "imaginary eyes" is sampled randomly throughout the space following a uniform distribution. We use proxy-depth to describe the estimated depth by the visual similarity among input images. Once the proxy-depth of a virtual ray is determined, we can retrieve candidate colors from input images and pass them to a color blending network to determine the real color.

2 Related Work

Conventional view synthesis. Novel view synthesis is a long-standing problem in the field of computer vision and graphics (Chen & Williams, 1993; Debevec et al., 1996; Seitz & Dyer, 1996). Conventional methods use image colors or handcrafted features to construct correspondences between the views (Fitzgibbon et al., 2005; Penner & Zhang, 2017). With the advance of deep networks, recent approaches employ neural networks to learn the transformation between input and target views implicitly (Eslami et al., 2018; Nguyen-Ha et al., 2020; Park et al., 2017; Sun et al., 2018; Zhou et al., 2016). In order to explicitly encode the geometry guidance, several specific scene representations are proposed, such as Multi-Plane Images (MPI) (Zhou et al., 2018; Mildenhall et al., 2019; Flynn et al., 2019; Srinivasan et al., 2019; Tucker & Snavely, 2020), and Layered Depth Images (LDI) (Shade et al., 1998; Shih et al., 2020; Tulsiani et al., 2018). Some Image-Based Rendering (IBR) techniques (Choi et al., 2019; Hedman et al., 2018; Penner & Zhang, 2017; Riegler & Koltun, 2020a; Thies et al., 2019b; Riegler & Koltun, 2020b; Shi et al., 2021) warp input view images to a target viewpoint according to the estimated proxy geometry, and then blend the warped pixels to synthesize a novel view.

Panorama synthesis. Zheng et al. (2007) propose a layered depth panorama (LDP) to create a layered representation with a full field of view from a sparse set of images taken by a hand-held camera. Bertel et al. (2019) investigate two blending methods for interpolating novel views from two nearby views, one is a linear blending, and the other is a view-dependent flow-based blending. Serrano et al. (2019) propose to synthesize new views from a fixed viewpoint 360° video. Huang et al. (2017) employ a typical depth-warp-refine procedure in synthesizing new views. They estimate the depth map for each input image and reconstruct the 3D point cloud by finding correspondences between input images using handcrafted features. They then synthesize new views from the reconstructed point cloud.

Plenoptic modeling. Early light-field/lumigraph methods (Levoy & Hanrahan, 1996; Gortler et al., 1996) reduce the 5D representation (position (x, y, z) and direction (θ, ϕ)) of the plenoptic function to 4D ((u, v, s, t), intersection between two image planes). They do not require scene geometry information, but either require the camera grid is densely and regularly sampled, or the target viewing ray is a linear combination of the source views (Chai et al., 2000; Lin & Shum, 2000). For unstructured settings, a proxy 3D scene geometry is required to be combined with light-field/lumigraph methods for view synthesis (Buehler et al., 2001; Davis et al., 2012). Recent methods (Yoon et al., 2015; Kalantari et al., 2016; Srinivasan et al., 2017; Wu et al., 2017) applied learning methods to improve light field rendering.

Neural rendering. Deep networks have also demonstrated their capability of modeling specific scenes as implicit functions (Mildenhall et al., 2020; Niemeyer et al., 2020; Sitzmann et al., 2019a;b; Thies et al., 2019a; Zhang et al., 2020; Martin-Brualla et al., 2020; Liu et al., 2020; Yu et al., 2020; Srinivasan et al., 2020; Park et al., 2020). They encapsulate both the geometry and appearance of a scene as network parameters. They take input as sampled points along viewing rays and output the corresponding color and density values during the inference stage. The target image is then rendered from the sampled points by volume rendering techniques (Max, 1995). The denser the sampled points, the higher quality of rendered images. However, densely sampling points along viewing rays would significantly increase the rendering time, prohibiting interactive applications to real-world scenarios.

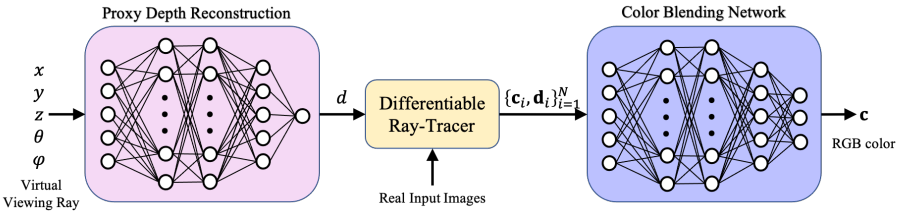


Figure 1: The overall pipeline of the proposed framework. Our framework includes a proxy depth reconstruction (PDR) model to determine the depth of a virtual viewing ray, a differentiable ray tracer to retrieve corresponding colors from real input images, and a color blending network (CBNet) to recover the RGB color information.

Our idea of using MLP to learn light-field is similar to that of NeRF (Mildenhall et al., 2020); however, there are key differences. NeRF focused on estimating the lights emitted at every location, in any direction, within a bounded volumetric region, often enclosing the 3D scene or 3D object of interest. In contrast, our method focuses on estimating all the light rays observed at any point in space, coming in any direction. In essence, our formulation is not only close to, but precisely is, the plenoptic function that Adelson and Bergen had contrived. In fact, in principle, our formulation can be extended to the original 7D plenoptic function by adding time and wavelength as new dimensions (Li et al., 2020; Bemana et al., 2020; Li et al., 2021). Our method also offers a computational advantage over NeRF. Namely, when the model has been well-approximated, we can directly display the network output as rendered images without sampling points along viewing rays and then rendering them in a back-to-front order. Our model will significantly accelerate the rendering speed and facilitate interactive applications.

3 Neural Plenoptic Sampling

A complete plenoptic function corresponds to the holographic representation of the visual world. It is originally defined as a 7D function $L(x, y, z, \theta, \phi, \lambda, t)$ which allows reconstruction of every possible view (θ, ϕ) from any position (x, y, z), at any time t and every wavelength λ. McMillan & Bishop (1995) reduce the dimensionality from 7D to 5D by ignoring the time and wavelength for the purpose of static scene view synthesis. By restricting the viewpoints or the object inside a box, light field (Levoy & Hanrahan, 1996) and lumigraph (Gortler et al., 1996) approaches reduce the dimensionality to four. Without loss of generality, this paper uses original 5D representations $L(x, y, z, \theta, \phi)$ for plenoptic function and focuses on the scene representation at a fixed time.

We model the plenoptic function as a Multi-Perceptron Layer (MLP). However, a brute-force training of a network mapping from *viewing* position and direction to RGB colors is infeasible. The observed images only have a partial coverage of the input space. By using the above training method, the model may fit well on the observed viewpoints, but also generates highly-blurred images on the non-observed regions. Our experiments in Fig. 5 demonstrate this situation.

To address this problem, we introduce an Imaginary Eye Sampling (IES) method to fully sample the target domain. We evaluate a proxy depth to provide self-supervision by leveraging photo-consistency among input images. Our method firstly outputs a proxy depth for a virtual viewing ray from the imaginary eye we randomly placed in the scene. Then, we retrieve colors from input views by a differentiable ray-tracer using this depth. Lastly, the colors pass through a color blending network to generate the real color. Figure. 1 depicts the overall pipeline of our framework.

3.1 Proxy Depth Reconstruction

We model the Proxy Depth Reconstruction (PDR) network by an MLP network F_Θ. It takes input as the camera position $\mathbf{x} = [x, y, z]^T \in \mathbb{R}^3$ and 2D camera viewing rays $\mathbf{v} = [\theta, \phi]^T \in \mathbb{R}^2$. The network estimates the distance value $d \in \mathbb{R}_+$ between the location $\mathbf{x}$ of the virtual camera and its observing scene in viewing direction,

$$d = F_\Theta(\mathbf{x}, \mathbf{v}), \tag{1}$$

where Θ represents the trainable network parameters.

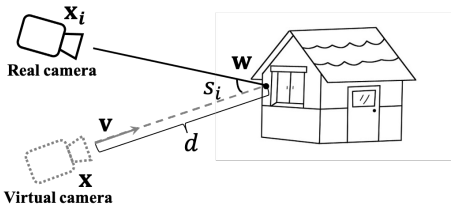


Figure 2: For a virtual camera position $\mathbf{x}$ and viewing direction $\mathbf{v}$, we estimate a depth d between the scene point $\mathbf{w}$ and the camera location $\mathbf{x}$. By reprojecting the scene point to real cameras, we retrieve the color $\mathbf{c}_i$ and high-level feature $\mathbf{f}_i$ from the observed images. The cosine distance (angle) s_i between the virtual viewing direction and real viewing direction determine the influence of corresponding real cameras when calculating the photometric consistency.

We use a similar MLP structure from NeRF (Mildenhall et al., 2020) to parameterize the neural plenoptic modeling. The difference is that NeRF approximates the emitting colors and transparency on the objects location, while our PDR model estimates the distance between the scene objects and observing cameras along the viewing direction.

3.2 Imaginary Eye Sampling

Since our purpose is to move around the scene and synthesize new views continuously, we need to sample the input space for the network training densely. However, in general, the camera locations of input images are sparsely sampled. The observed images only cover partial regions of such an input space.

To address this problem, we propose an Imaginary Eye Sampling (IES) strategy. We place thousands of imaginary eyes (virtual cameras) in the space of interest. Those imaginary eyes are randomly generated in the space to allow dense sampling of the plenoptic input space. By doing so, we are able to approximate a whole complete plenoptic function.

Here, note that we do not have ground-truth depths for supervision, even for real-observed images (viewing rays). In order to provide training signals, we propose a self-supervision method by leveraging photo-consistency among real input images.

3.3 Self-Supervision via Photo-Consistency

Given a virtual camera at a random location $\mathbf{x}$ and a viewing direction $\mathbf{v}$, our network predicts a depth d between the observed scene point and the input camera location. The world coordinate $\mathbf{w}$ of the scene point is then computed as

$$\mathbf{w} = \mathbf{x} + d\mathbf{v}. \tag{2}$$

When the estimated depth d is at the correct value, the colors of its projected pixels on real observed images should be consistent with each other. Hence, we then use a differentiable ray-tracer $T(\cdot)$ to find the projected pixel of the scene point at real camera image planes. Denote $\mathbf{P}_i$ as the projection matrix of real camera i. The projected image coordinate of a 3D point $\mathbf{w}$ is computed as $[u_i, v_i, 1]^T = \mathbf{P}_i\mathbf{w}$. Our ray-tracer then uses bilinear interpolation to fetch information (*e.g.*, color) from the corresponding real images.

By computing the photo-consistency (similarity) among the retrieved colors, we can measure the correctness of the estimated depth. In practice, we argue that only using the colors of the retrieved pixels is not accurate enough for this measurement because it cannot handle textureless and reflective regions. To increase the representative and discriminative ability, we propose to retrieve colors as well as high-level features from real input images for the photo-consistency measure.

Denote $\mathbf{f}_i$ and $\mathbf{c}_i$ as the retrieved features and colors from input camera i, respectively. The photo-consistency among all input cameras is defined as

$$\mathcal{L}_d = \sum_{i=1}^{N} s_i \left(\|\mathbf{c}^{\text{top}_k} - \mathbf{c}_i\|_1 + \lambda \|\mathbf{f}^{\text{top}_k} - \mathbf{f}_i\|_1 \right), \tag{3}$$

where $\|\cdot\|_1$ denotes the L_1 distance, N is the number of real input cameras, λ is the balance of the influence between color difference and the feature difference, s_i is the normalized weight assigned to each real camera i, and it is determined by the angle difference (cosine distance) between the virtual camera viewing ray $(\mathbf{w}-\mathbf{x})$ and the real camera viewing ray $(\mathbf{w}-\mathbf{x_i})$. Figure. 2 illustrates the situation. Mathematically, it is expressed as

$$s_i = \frac{\cos\left(\mathbf{w}-\mathbf{x}, \mathbf{w}-\mathbf{x_i}\right)}{\sum_{j=1}^{N}\cos\left(\mathbf{w}-\mathbf{x}, \mathbf{w}-\mathbf{x_j}\right)}, \tag{4}$$

where $\cos(\cdot,\cdot)$ is the cosine of the angle spanned by the two vectors. The smaller of the angle between the virtual camera viewing ray and the real camera viewing ray, the larger s_i is. Given the weight for each input camera, the reference color $\mathbf{c}^{\text{top}_k}$ and feature $\mathbf{f}^{\text{top}_k}$ in Eq. 3 is computed as the average of top k retrieved colors and features

$$\mathbf{c}^{\text{top}_k} = \sum_{i\in\text{top}_k}\mathbf{c}_i/k, \qquad \mathbf{f}^{\text{top}_k} = \sum_{i\in\text{top}_k}\mathbf{f}_i/k. \tag{5}$$

We use Eq. 3 as the supervision for our PDR model and the network is trained to minimize this objective function.

3.4 Color Blending for View Synthesis

Given an estimated depth d for a virtual viewing ray, we can retrieve colors from real input images for the virtual camera view synthesis. However, a naive aggregation of the retrieved colors would cause severe tearing or ghosting artifacts in the synthesized images. Hence, we propose a Color Blending Network (CBNet) to blend the retrieved colors and tolerate the errors caused by inaccurate depths to synthesize realistic images.

In order to provide sufficient clues, we feed the direction differences between the reprojected real viewing rays (solid line in Fig. 2) and the virtual (target) viewing ray (dash line in Fig. 2) along with the retrieved colors to the color blending network. Formally, our CBNet is expressed as

$$\mathbf{c} = F_\Phi\left(\{\mathbf{c}_i, \mathbf{d}_i\}_{i=1}^{N}\right), \tag{6}$$

where Φ is the trainable parameter of the CBNet, $\mathbf{c}_i$ is the RGB color retrieved from the real camera i and $\mathbf{d}_i$ is the projection of the real viewing ray on the target virtual viewing ray, $\mathbf{c}$ is the estimated color of the virtual viewing ray. We employ a Pointnet network architecture for our CBNet. The supervision of our CBNet is the colors observed from real cameras, denoted as

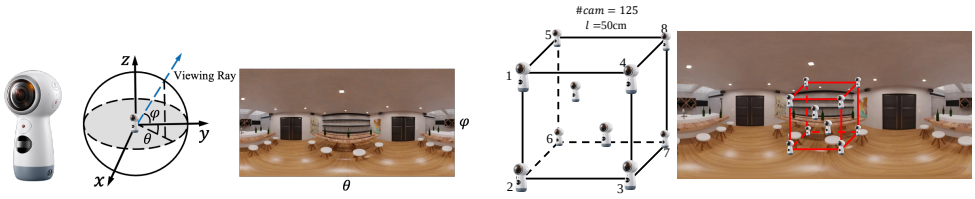
$$\mathcal{L}_\mathbf{c} = \|\mathbf{c}^* - \mathbf{c}\|_1, \tag{7}$$

where $\mathbf{c}^*$ is the ground truth colors.

Unlike our PDR model, the CBNet is trained only on the observed images (viewing rays) since it needs the ground-truth color as supervision. Instead of remembering the color of each training ray, the CBNet is trained to learn a sensible rule for blending retrieved colors from real input views. Thus it is able to be generalized to unseen viewing rays. The PDR and the CBNet in our framework are trained separately. During the training of CBNet, we fix the model parameters of PDR to not destroy the learned patterns for the whole plenoptic space. For inference, a query viewing ray first passes through our PDR model to compute a depth value; its corresponding colors on real input views are then retrieved and fed into the CBNet to estimate the color information. Since it is a single feed-forward pass through the network, the rendering speed is rapid (less than one second when rendering a 1024×512 image).

4 Experiments

In this section, we conduct comprehensive experiments to demonstrate the effectiveness of the proposed algorithm. We use 360° panoramas captured by an omnidirectional camera for the plenoptic modeling, since its representation well aligns with the plenoptic function. We show an omnidirectional camera, its imaging geometry, and an example image in Fig. 3. The pixel coordinates of a 360° panorama correspond to the azimuth angle θ and the elevation angle ϕ of the viewing rays.



(a) Omni-directional camera (b) Camera arrangement

Figure 3: (a) An illustration of an omni-directional camera and its captured light-field and a sample image. (b) An illustration of our camera arrangement for dataset generation. For each scene, we capture 125 omnidirectional images at different locations for evaluation. The cameras are positioned in a $50 \times 50 \times 50$ centimeter volume (roughly) at the center of each scene.

Table 1: Quantitative comparison of our method and others given eight input views. Here, **bold** indicates the best results and underline denotes the second best results.

	Diningroom		Bar		Livingroom		Lounge		Average	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM ↑	PSNR↑	SSIM ↑
FVS* (Riegler & Koltun, 2020a)	26.09	0.770	24.54	0.800	25.61	0.780	21.23	0.690	24.37	0.760
NeRF (Mildenhall et al., 2020)	**37.54**	0.938	**33.95**	0.941	33.62	0.936	31.96	0.939	34.27	0.939
NeRF++ (Zhang et al., 2020)	36.29	0.931	32.87	0.936	33.72	0.929	34.05	0.947	34.23	0.936
Ours w/o Imaginary Eye	32.32	0.929	32.93	0.950	32.57	0.948	30.50	0.932	32.08	0.940
Ours w/o Feature	36.03	0.957	33.47	0.954	33.97	0.957	32.17	0.960	33.91	0.957
Ours w/o weighting	32.69	0.931	29.57	0.903	30.81	0.919	29.18	0.925	30.56	0.920
Ours	36.62	**0.959**	33.86	**0.961**	**34.33**	**0.965**	**34.31**	**0.968**	**34.78**	**0.963**

4.1 Datasets and Evaluations

Synthetic dataset. When the plenoptic function has been correctly (approximately) modeled, we want to freely move across the space to synthesize new views. For the purpose of performance evaluation, we need to sample evaluation viewpoints densely in the space and their corresponding ground truth data. Hence, we propose to synthesize a dataset for our evaluation. Following recent novel view synthesis methods, we use SSIM and PSNR for the performance evaluation.

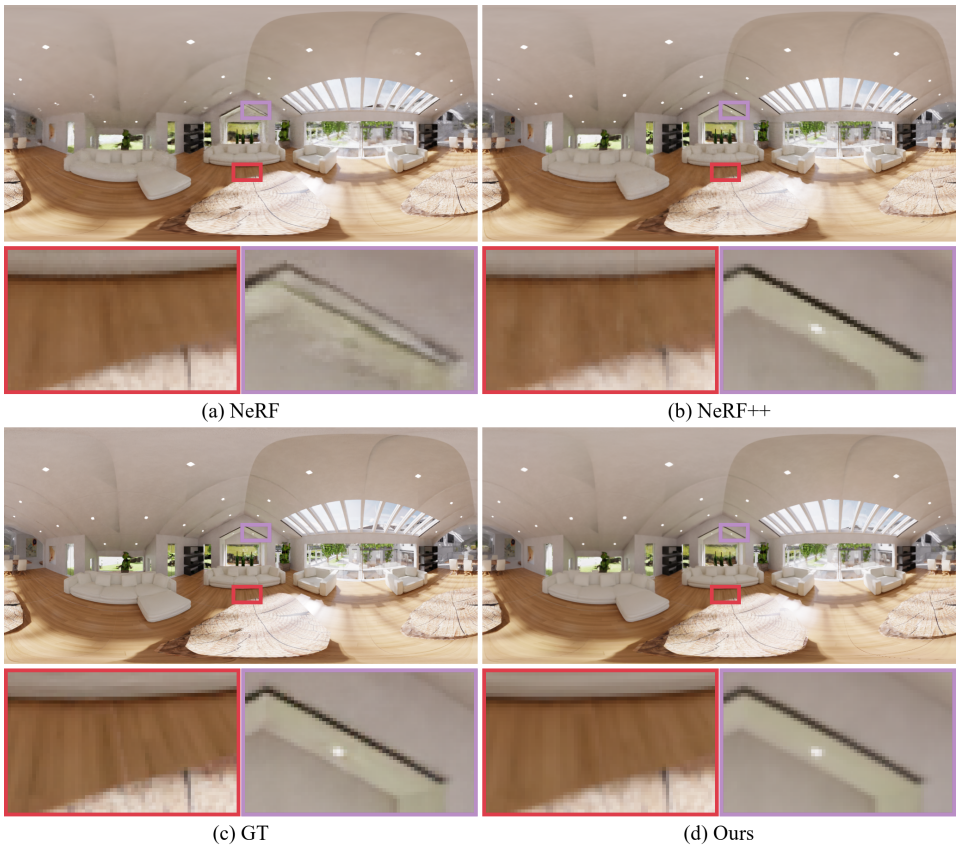
We use Blender (Community, 2020) to synthesize images with freely moving camera viewpoints. Figure. 3 shows the camera setting. Specifically, we randomly sample 125 points in a $50 \times 50 \times 50$ cm^3 volume within the space and synthesize corresponding omni-directional images. The images are generated from four scenes, *i.e.*, "Bar", "Livingroom", "Lounge" and "Diningroom". We refer the readers to our qualitative comparisons for the visualization of sampled images from the four scenes. This evaluation set is adopted for all the experiments in this paper, although the input views and training methods might be changed.

Real dataset. To fully demonstrate the effectiveness of the proposed method, we also conduct experiments on real-world data. The real-world data we use are from Wang et al. (2020), which sparsely captured two images per scene. We only provide qualitative results for visual evaluation, and interested readers are suggested to watch our supplementary video for more results.

4.2 Training Details

We train a separate plenoptic function for each scene. To approximate the sharp edge of real world objects and textures, our plenoptic function model usually has high frequency output in both the viewing rays and camera position. We encode the 5D input into Fourier features as the positional encoding (Tancik et al., 2020) before feeding it into the proxy-depth reconstruction network. The PDR network F_Θ is designed following the structure of NeRF. It consists of 8 fully-connected (fc) layers with 256 hidden channels, and a ReLU activation layer follows each fc layer.

When training the MLP, we randomly sample a virtual camera at location $\mathbf{x}$ and draw an arbitrary viewing direction $\mathbf{v}$. Given this 5D input, the MLP estimates a proxy-depth d in the output, which is then self-supervised by the photometric consistency loss $\mathcal{L}_d$. The above network is end-to-end differentiable. Once we have sampled and trained the virtual camera domain thoroughly, the MLP for proxy-depth reconstruction is then frozen for the training of the color blending network later.



(a) NeRF (b) NeRF++

(c) GT (d) Ours

Figure 4: Qualitative comparison with NeRF and NeRF++ on our generated scenes "Lounge". Our method generates sharper results than the comparison algorithms.

The CBNet takes a series color and direction $(\mathbf{c}_i, \mathbf{d}_i)$ to inference the output color of the plenoptic function. Its design follows the structure of the PointNet (Qi et al., 2017). The observations from real cameras are firstly processed separately by three fc layers. Next, a max-pooling layer is applied to select the most salient features from them. We then employ a prediction layer to generate the color values $\mathbf{c}$.

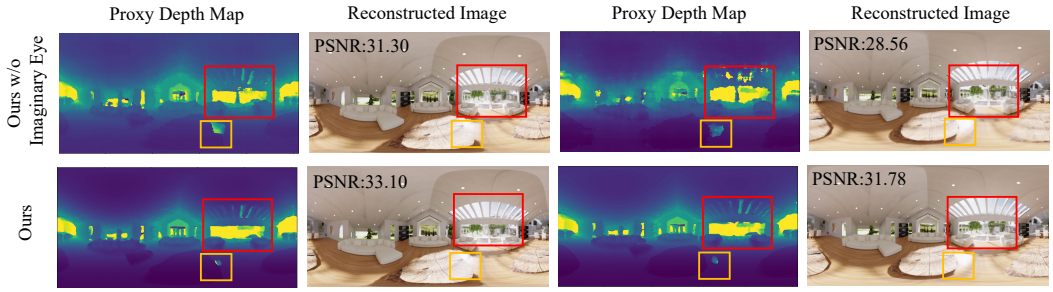
In our experiments, we use 200k rays per iteration for the PDR network training, and 100k rays for the CBNet training. Our model is trained from scratch with an Adam optimizer. The learning rate is set to 5×10^{-4}. The PDR network takes around 30k iterations to converge, while the CBNet only takes 10k iterations. The complete model takes around one hour to converge in a NVIDIA RTX 3090 GPU.

4.3 Comparison With Other Methods

Comparison with NeRF and NeRF++. We conduct experiments to compare with NeRF and its variant NeRF++ (Zhang et al., 2020). In this comparison, all of the methods take eight views as input. The quantitative evaluation results are presented in Table 1. Visual comparison is presented in Fig. 4. It can be seen that our method achieves better performance than NeRF and NeRF++ in most of the scenarios. NeRF and NeRF++ aim to estimate the radiance emitted by scene points at any position and direction, while our method is designed to recover the irradiance perceived by an observer from any point and direction. Since NeRF and Nerf++ need to sample points along viewing rays and render them in a back-to-front order, they require hundreds of network calls when synthesizing an image. Thus their rendering time is very long. In contrast, our method directly outputs the color information given a viewing ray. Thus, our training and testing time are relatively shorter, as shown in appendix Table 4.

Table 2: Quantitative comparison with volume-based method on two input views.

	Diningroom		Bar		Livingroom		Lounge	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
360SD-Net (Wang et al., 2020)	24.76	0.746	23.38	0.781	23.30	0.747	21.10	0.700
Ours (Vertical)	**27.54**	**0.910**	**27.29**	**0.918**	**28.20**	**0.907**	**26.13**	**0.888**
MatryODShka (Attal et al., 2020)	20.43	0.673	27.26	0.864	23.85	0.766	22.19	0.765
Ours (Horizontal)	**30.50**	**0.921**	**28.20**	**0.918**	**29.07**	**0.907**	**27.68**	**0.898**



(a) Close to a real camera (b) Far from any real camera

Figure 5: Qualitative comparisons of our method w or w/o Imaginary Eye Sampling (IES). Without using IES, the image synthesized at a position far from any real camera (top right) suffers much lower quality compared to the one closer to a real camera (top left) (2.74dB drop). When the IES is applied, the quality of both images (bottom left and bottom right) improves, and the PSNR gap decreases (1.32dB).

Comparison with FVS. To demostrate the effectiveness of our CBNet. We compare our CBNet with another image-based warping method FVS (Riegler & Koltun, 2020a).[1] The results are presented in the first row of Table 1. It is evident that our method achieves significantly better performance.

Comparison with conventional novel view synthesis approaches. We employ a deep-based method 360SD-Net (Wang et al., 2020) to estimate depth maps for input images. We then build a point cloud from the depth map and input images. The point cloud are warped and refined for novel view synthesis. Since 360SD-Net only takes two vertically aligned panoramas as input, we take the same vertical inputs in this comparison, denoted as "Our (Vertical)" in Tab. 2. We further compare with a multi-sphere-images-based method MatryODShka (Attal et al., 2020) on view synthesis. Note that MatryODShka only takes two horizontally aligned panoramas as input. For fair comparison, we take the same input and denoted as "Our (Horizontal)" in Tab. 2. The numerical evaluations in Tab. 2 demonstrate that our method significantly outperforms the conventional depth-warp-refine and multi-sphere-images procedure in synthesizing new views. Besides, the competing methods both requires a structured input (horizontally or vertically aligned). This limitation does not apply to our method.

4.4 Effectiveness of Imaginary Eye Sampling

We demonstrate the necessity and effectiveness of the imaginary eye sampling strategy. In doing so, we train our network only using real camera locations and directions, without any imaginary eye sampling, denoted as "Ours w/o Imaginary Eye". The quantitative results and qualitative evaluations are presented in Tab. 1 and Fig. 5 respectively. For better comparison, we select two images for visualization. One is close to a real camera, and the other is far from input cameras.

As illustrated by the results, the performance of "Ours w/o Imaginary Eye" is inferior to our whole pipeline. More importantly, the performance gap between images that are near and far from the real camera is significant. There is 2.75dB difference in terms of PSNR metric. This demonstrates that the model learns better for training data while does not have the ability to interpolate similar-quality test data.

A network is usually good at learning a continuous representation from discrete but uniformly distributed samples in a general case. In our plenoptic modeling, the values of the input parameters

[1]FVS is originally proposed on perspective images. We change its code so that it takes 360° panoramas and our depth estimation as input in comparison.

Table 3: Quantitative comparison of different imaginary eye sampling (IES) regions (large or small). Larger imaginary eye sampling space contributes to higher image quality.

Scene	Lounge				Livingroom			
N	2		4		2		4	
IES Range	PSNR↑	SSIM↑	PSNR	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Small	24.82	0.8484	27.98	0.8995	26.77	0.8690	29.93	0.9144
Large	**26.13**	**0.8883**	**29.13**	**0.9193**	**28.20**	**0.9068**	**31.27**	**0.9383**

(x, y, z, θ, ϕ) are continuous and always span in a large range, while the input images only cover small and sparsely sampled regions in the whole space. Hence, it is not surprising that the model can fit well in training data while interpolating low-quality images at camera locations far from real cameras. Using our imaginary eye sampling strategy, the performance gap between the two cases is reduced (1.32dB in terms of PSNR). Furthermore, the quality of synthesized images on the location that is near to a real camera is further improved. This is owed to our photometric consistency self-supervision loss for the virtual eye training. It helps the learned model to encode the geometry constraints across different viewpoint images.

We also conduct comparison experiments on the imaginary eye sampling area (large or small). The results are shown in Tab. 3. We found that sampling on a larger region will allow more freedom on the moving space of rendering cameras, while the downside is that it requires longer training time.

4.5 Proxy-Depth from Colors and Features

In what follows, we conduct experiments to demonstrate why the features are required in our photometric consistency loss. In doing so, we remove the feature item in Eq. 3 and train our model again, denoted as "Ours w/o Feature". The quantitative results are presented in the third last row of Tab. 1.

Compared to pixel-wise RGB colors, features have a larger reception field that makes textureless regions discriminative, and the encoded higher level information is more robust to illumination changes and other noises. Thus, the reconstructed proxy depths from both RGB colors and features are more accurate than those purely from RGB colors. Consequently, the quality of synthesized images is facilitated. We present the qualitative illustrations in the supplementary material.

4.6 With or Without View-direction Based Weighting

We also ablate the necessity of the view-direction based weighting in Eq. 3. In this experiment, we set the weighting term s_i to zero, denoted as "Ours w/o weighting". The results are presented in the second last row of Tab. 1. Not surprisingly, the performance drops. This demonstrates the effectiveness of our view-direction based weighting strategy.

5 Conclusion

Capturing a complete and dense plenoptic function from every point and angle within a space has been the "holy grail" for IBR-based view synthesis applications. There is always a tension between how densely one samples the space using many real cameras and the total efforts and cost that one has to bear in doing this task. This paper proposes a simple yet effective solution to this challenge. By placing thousands of imaginary eyes (virtual cameras) at randomly sampled positions in the space of interest, this paper proposes a new neural-network-based method to learn (or to approximate) the underlying 5D plenoptic function. Real images captured by physical cameras are used as a teacher to train our neural network. Although those randomly placed imaginary eyes themselves do not provide new information, they are critical to the success of our method, as they provide a bridge to leverage the existing multi-view geometry relationship among all the views (of both real and virtual). Our experiments also validate this claim positively and convincingly. Our method produces accurate and high-quality novel views (on the validation set) and compelling visual results (on unseen testing images).

6 ETHICS STATEMENT

Our approach to capture and reconstruct the light-field from only a few input images has the immediate utility of many applications, such as augmented, virtual and mixed reality. Our 360° inputs also open up the ability to fully reconstruct and re-render the whole scene at a low cost. Such ability also enables the possibility to reconstruct humans in a scene. The acquisition of such personal information, if without their consent, may lead to privacy and security breaching. Appropriate privacy-preserving steps must be taken to mitigate the potential risk of abusing this technique.

7 REPRODUCIBILITY STATEMENT

The dataset we use in the paper is stated in Section 4.1. The detail design of our network architecture and training processes can be found in Section 4.2. We will release the code once the paper is public available.

REFERENCES

Edward H Adelson, James R Bergen, et al. *The plenoptic function and the elements of early vision*, volume 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of . . . , 1991.

Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *European Conference on Computer Vision*, pp. 441–459. Springer, 2020.

Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. X-fields: implicit neural view-, light-and time-image interpolation. *ACM Transactions on Graphics (TOG)*, 39(6): 1–15, 2020.

Tobias Bertel, Neill DF Campbell, and Christian Richardt. Megaparallax: Casual 360° panoramas with motion parallax. *IEEE transactions on visualization and computer graphics*, 25(5):1828–1835, 2019.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 425–432, 2001.

Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 307–318, 2000.

Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 279–288, 1993.

Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7781–7790, 2019.

Blender Online Community. Blender - a 3d modelling and rendering package, 2020. URL `http://www.blender.org`.

Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. In *Computer Graphics Forum*, volume 31, pp. 305–314. Wiley Online Library, 2012.

Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 11–20, 1996.

SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

Andrew Fitzgibbon, Yonatan Wexler, and Andrew Zisserman. Image-based rendering using image-based priors. *International Journal of Computer Vision*, 63(2):141–151, 2005.

John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2367–2376, 2019.

Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021.

Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 43–54, 1996.

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

Jingwei Huang, Zhili Chen, Duygu Ceylan, and Hailin Jin. 6-dof vr videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*, pp. 37–44. IEEE, 2017.

Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016.

Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 31–42, 1996.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis. *arXiv preprint arXiv:2103.02597*, 2021.

Zhengqi Li, Wenqi Xian, Abe Davis, and Noah Snavely. Crowdsampling the plenoptic function. In *European Conference on Computer Vision*, pp. 178–196. Springer, 2020.

Zhouchen Lin and Heung-Yeung Shum. On the number of samples needed in light field rendering with constant-depth assumption. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pp. 588–595. IEEE, 2000.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *arXiv preprint arXiv:2007.11571*, 2020.

Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *arXiv preprint arXiv:2008.02268*, 2020.

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

Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 39–46, 1995.

Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

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 *European Conference on Computer Vision*, pp. 405–421. Springer, 2020.

Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of neural radiance fields using depth oracle networks. *arXiv preprint arXiv:2103.03231*, 2021.

Phong Nguyen-Ha, Lam Huynh, Esa Rahtu, and Janne Heikkila. Sequential neural rendering with transformer. *arXiv preprint arXiv:2004.04548*, 2020.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3504–3515, 2020.

Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 3500–3509, 2017.

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Gernot Riegler and Vladlen Koltun. Free view synthesis. In *European Conference on Computer Vision*, pp. 623–640. Springer, 2020a.

Gernot Riegler and Vladlen Koltun. Stable view synthesis. *arXiv preprint arXiv:2011.07233*, 2020b.

Steven M Seitz and Charles R Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 21–30, 1996.

Ana Serrano, Incheol Kim, Zhili Chen, Stephen DiVerdi, Diego Gutierrez, Aaron Hertzmann, and Belen Masia. Motion parallax for 360 rgbd video. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1817–1827, 2019.

Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 231–242, 1998.

Yujiao Shi, Hongdong Li, and Xin Yu. Self-supervised visibility learning for novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9675–9684, 2021.

Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8028–8038, 2020.

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2437–2446, 2019a.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pp. 1121–1132, 2019b.

Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2243–2251, 2017.

Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 175–184, 2019.

Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. *arXiv preprint arXiv:2012.03927*, 2020.

Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. Multi-view to novel view: Synthesizing novel views with self-learned confidence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 155–171, 2018.

Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019a.

Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. Image-guided neural object rendering. In *International Conference on Learning Representations*, 2019b.

Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 551–560, 2020.

Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 302–317, 2018.

Ning-Hsu Wang, Bolivar Solarte, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. 360sd-net: 360° stereo depth estimation with learnable cost volume. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 582–588. IEEE, 2020.

Gaochang Wu, Mandan Zhao, Liangyong Wang, Qionghai Dai, Tianyou Chai, and Yebin Liu. Light field reconstruction using deep convolutional network on epi. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6319–6327, 2017.

Youngjin Yoon, Hae-Gon Jeon, Donggeun Yoo, Joon-Young Lee, and In So Kweon. Learning a deep convolutional network for light-field image super-resolution. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 24–32, 2015.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. *arXiv preprint arXiv:2012.02190*, 2020.

Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields, 2020.

Ke Colin Zheng, Sing Bing Kang, Michael F Cohen, and Richard Szeliski. Layered depth panoramas. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2007.

Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European conference on computer vision*, pp. 286–301. Springer, 2016.

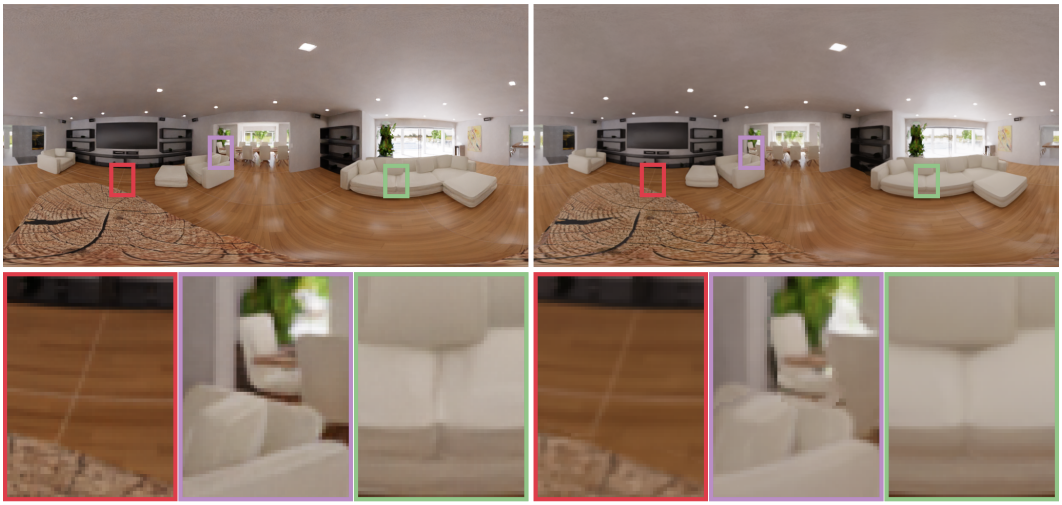
Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

APPENDIX

A ADDITIONAL COMPARISONS WITH NERF AND ITS VARIANTS

We also present addition results on comparison with NeRF (Mildenhall et al., 2020) and its variant NeRF++ (Zhang et al., 2020). In this comparison, all of the methods take eight unstructured views as input. The quantitative evaluation results are presented in the main paper. The additional qualitative comparison results are presented in Fig. 6. It can be seen that our method achieves better performance in most of the scenarios.

(a) NeRF (b) NeRF++



(c) GT (d) Ours

Figure 6: Qualitative comparison with NeRF and NeRF++ on the scene "Livingroom". Our method generates sharper results than the comparison algorithms.

NeRF and NeRF++ aim to estimate the radiance emitted by scene points at any position and direction, while our method is designed to recover the irradiance perceived by an observer from any point and direction. In essence, our formulation is more close to the plenoptic sampling invented by Adelson and Bergen (Adelson et al., 1991) Since NeRF and Nerf++ need to sample points along viewing rays and render them in a back-to-front order, they require hundreds of network calls when synthesizing an image. Thus their rendering time is very long, as shown in Tab. 4. In contrast, our method directly outputs the color information given a viewing ray. Thus, our training and testing time are relatively shorter.

A recent work, DoNeRF (Neff et al., 2021), shares some similarity with ours. Both DoNeRF and our method first regress the depth for a target viewing ray. The difference is that DoNeRF has the ground-truth depth map for each viewing ray during training, while our approach offers a self-supervision for the target view depth regression. FastNeRF (Garbin et al., 2021) is another recent work that is proposed to accelerate the rendering speed during inference. Their approach changes the network architecture and explores a way to cache a number of pre-sampled scene points (with colors and densities) for testing when the model has been trained. By doing so, they successfully reduce the testing time for view synthesis. However, the training time remains the same as the original

Table 4: Training and testing time comparison given eight input views. The testing time is for rendering images with resolution of 512×1024.

	Training (hours)	Testing (seconds)
NeRF	10	30
NeRF++	20	110
Ours	1	0.14

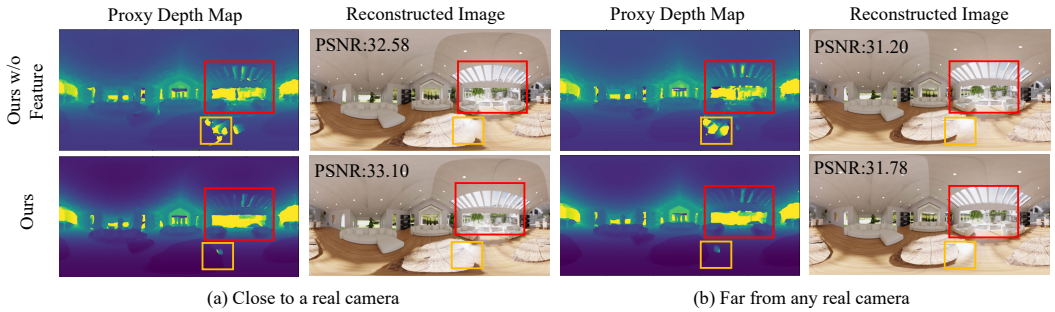


Figure 7: Qualitative comparison of our method with or without features in proxy depth estimation.

NeRF. Compared to FastNeRF, our method manages to achieve a shorter time for both training and testing.

B Experiments on Real Dataset and Supplementary Video

As mentioned in the paper, the performance on the real dataset, 360SD-Net (Wang et al., 2020), is hard to be quantitatively evaluated due to the lack of ground truth data. Hence, we qualitatively visualize the synthesized images by our method in the supplementary video. Furthermore, we synthesize continuously generated images by our method on the four synthetic scenes when roaming around the space. Please refer to our supplementary video for the results.

C Qualitative comparison for "w vs. w/o features"

We provide the qualitative comparisons for our method with vs. without features in Fig. 7. It can be seen that the generated depths by our whole method are better than those generated by our w/o feature similarity, and the image quality synthesize by our whole method is better.

D Different Number of Input Views

Below, we conduct experiments on a different number of input views. For this experiment setting, we aim to investigate the performance difference when the input views are located on a line, a flat plane, and a cube, corresponding to 2, 4, 8, and 25 input views, respectively. Tab. 5 and Fig. 8 provide the quantitative and qualitative results, respectively. As shown by the results, when the input view number is reduced to 2, our method still generates acceptable quality novel view images. As the number of input views increases, the quality of the view synthesis improves rapidly. For 8 and 25 input views in this experiment, the input cameras are randomly sampled within the region (cube) of interest. This demonstrates that our method is not limited to structured settings and can synthesize free-viewpoint images from unstructured input images.

Table 5: Quantitative evaluations on different input view numbers.

N	2			4			8			25		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Lounge	26.13	0.8883	0.1947	29.13	0.9193	0.1546	34.31	0.9684	0.1086	37.27	0.9775	0.0916
Livingroom	28.20	0.9068	0.2572	31.27	0.9383	0.2298	34.33	0.9648	0.1590	36.72	0.9746	0.1331

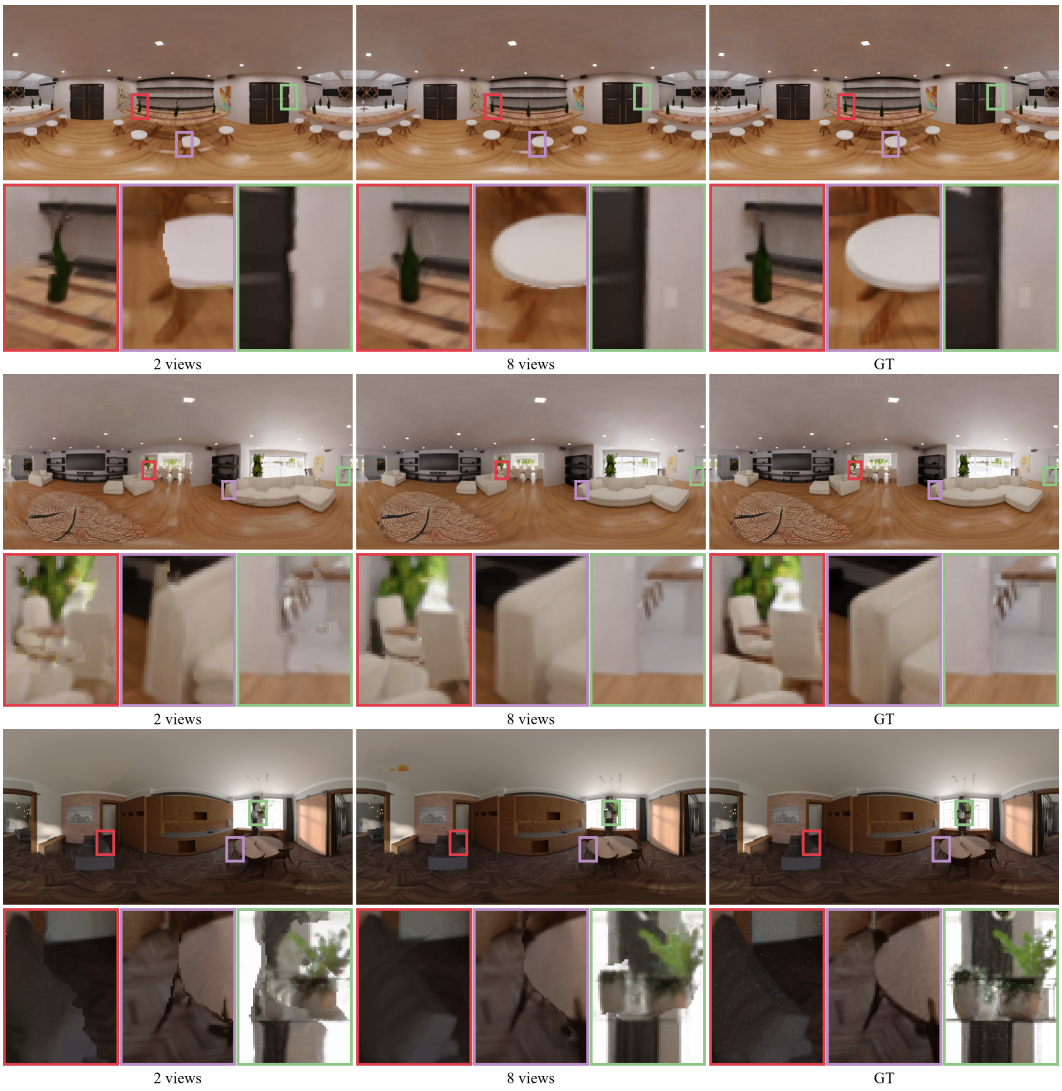
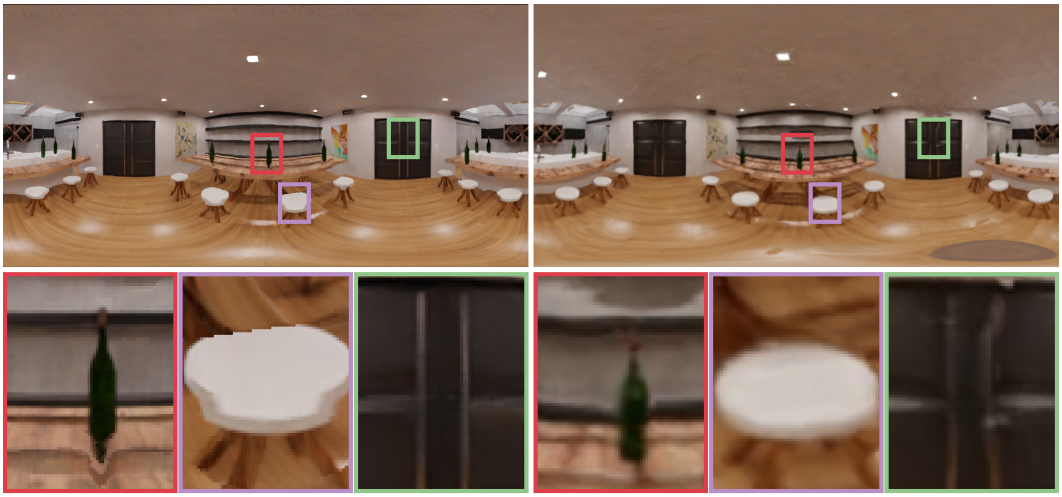


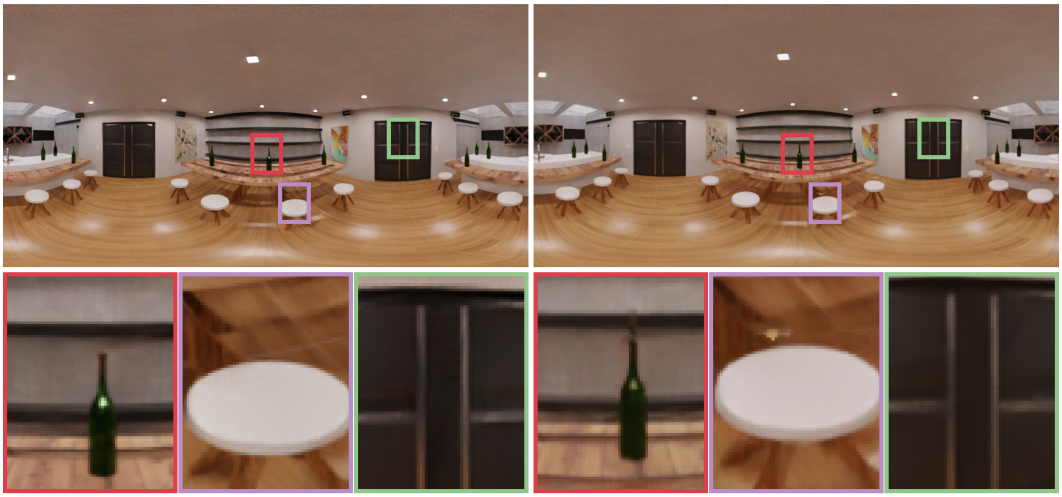
Figure 8: Qualitative visualization on different input view number. The three images are from our generated scenes "Bar", "Livingroom" and "Diningroom" respectively. Here we compare the reconstruction results by only 2 input views and 8 input views.

E Qualitative Comparison with Conventional NVS Approaches on 360° Panorama Synthesis

Here we present the qualitative comparison results with conventional NVS approaches in Fig. 9 and Fig. 10. (Quantitative results can be found in the main paper Table 2.) From the figure, it can be seen that there are severe distortions in the synthesized images by typical depth-warp-refine (*i.e.*, 360SD-Net (Wang et al., 2020)) strategy, while the synthesized images by our method are much similar to the ground truth. The numerical evaluations in Table 2 also demonstrate that our method significantly outperforms the conventional depth-warp-refine and multi-sphere-images procedure in synthesizing new views.



(a) 360SD (b) MatryODShka



(c) GT (d) Ours

Figure 9: Qualitative comparison of our algorithm with 360SD-Net (Wang et al., 2020) and MatryODShka (Attal et al., 2020).

(a) 360SD (b) MatryODShka

(c) GT (d) Ours

Figure 10: Qualitative comparison of our algorithm with 360SD-Net (Wang et al., 2020) and MatryODShka (Attal et al., 2020).