Local Implicit Ray Function for Generalizable Radiance Field Representation

Xin Huang[1]*, Qi Zhang[2]†, Ying Feng[2], Xiaoyu Li[2], Xuan Wang[2], Qing Wang[1]†

[1] School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China
[2] Tencent AI Lab

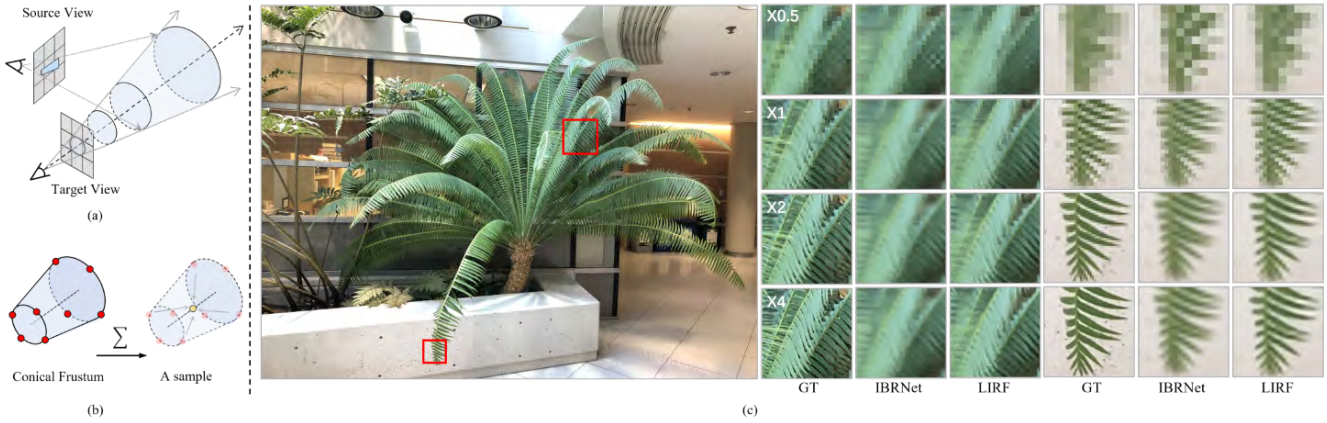


Figure 1. We propose LIRF to reconstruct radiance fields of unseen scenes for novel view synthesis. Given that current generalizable NeRF-like methods cast an infinitesimal ray to render a pixel at different scales, it causes excessive blurring and aliasing. Our method instead reasons about 3D conical frustums defined by the neighbor rays through the neighbor pixels (as shown in (a)). Our LIRF outputs the feature of any sample within the conical frustum in a continuous manner (as shown in (b)), which supports NeRF reconstruction at arbitrary scales. Compared with the previous method, our method can be generalized to represent the same unseen scene at multiple levels of details (as shown in (c)). Specifically, given a set of input views at a consistent image scale $\times 1$, LIRF enables our method to both preserve sharp details in close-up shots (anti-blurring as shown in $\times 2$ and $\times 4$ results) and correctly render the zoomed-out images (anti-aliasing as shown in $\times 0.5$ results).

Abstract

*We propose **LIRF** (**L**ocal **I**mplicit **R**ay **F**unction), a generalizable neural rendering approach for novel view rendering. Current generalizable neural radiance fields (NeRF) methods sample a scene with a single ray per pixel and may therefore render blurred or aliased views when the input views and rendered views capture scene content with different resolutions. To solve this problem, we propose LIRF to aggregate the information from conical frustums to construct a ray. Given 3D positions within conical frustums, LIRF takes 3D coordinates and the features of conical frustums as inputs and predicts a local volumetric radiance field. Since the coordinates are continuous, LIRF renders high-quality novel views at a continuously-valued scale via volume rendering. Besides, we predict the visible weights for each input view via transformer-based feature matching to improve the performance in occluded areas. Experimental results on real-world scenes validate that our method outperforms state-of-the-art methods on novel view rendering of unseen scenes at arbitrary scales.*

1. Introduction

Novel view synthesis has garnered recent attention with compelling applications of neural rendering in virtual and augmented reality. Different from image-based rendering [6, 20, 33, 43, 75], Neural Radiance Fields (NeRF) [44] implicitly represents the 3D scenes within multilayer perceptrons (MLPs) by mapping coordinates to color and geometry of scenes. To render a pixel, the ray projected to that pixel is traced and the color of each sampled point along the ray is accumulated based on volume rendering.

Despite NeRF and its variants having demonstrated remarkable performance in providing immersive experiences in various view synthesis tasks, their practical applications

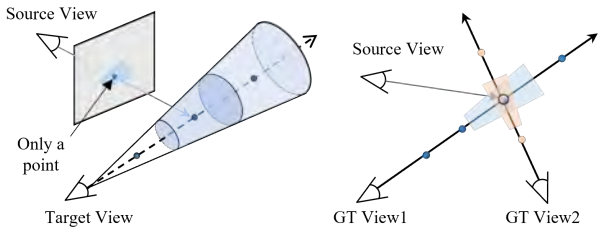
*Work was done during an internship at Tencent AI Lab.
†Corresponding authors.

are constrained by the requirement of training from scratch on each new scene, which is time-consuming. To overcome this problem, many researches [10, 13, 30, 34, 38, 59, 65, 71] introduce image-based rendering techniques to NeRF, which achieves generalization on unseen scenes. They take into consideration the image features (from nearby views) of a 3D point. The common motivation is to predict the density and color of this point by matching the multi-view features, which is similar to the stereo matching methods [21, 55, 69] that find a surface point by checking the consistency of multi-view features.

While these methods generalize well on new scenes when the distance of input and testing views are roughly constant from the scene (as in NeRF), they cannot properly deal with the less constrained settings such as different resolutions or varying focal length and produce results with blurring or aliasing artifacts. Since a single ray is cast through each pixel whose size and shape are ignored, it's challenging to query the accurate feature of the target ray from input images as shown in Fig. 2(a), and the model learns an ambiguous result as shown in Fig. 2(b). Mip-NeRF [3], a NeRF variant of per-scene optimization, proposes an anti-aliasing design that models the ray through a pixel as a cone and uses a 3D Gaussian to approximate the sampled *conical frustum* (a cone cut perpendicular to its axis) for volumetric representation. However, directly extending Mip-NeRF to a generalizable method is also challenging to extract the accurate features of the ray from input images due to the subpixel precision. Consequently, an efficient solution is to supersample each pixel by marching multiple rays according to its footprint, similar to the strategy used in offline raytracing.

Our key insight is the *local implicit ray function* (LIRF) that represents the feature aggregation of samples within ray conical frustum in a continuous manner, as shown in Fig. 1. Specifically, given any 3D sampled position within a conical frustum, our LIRF outputs the aggregated feature by taking the 3D coordinate of this position and the features of vertices within the conical frustum as inputs (the vertices of a conical frustum are defined with eight points (red points) as shown in Fig. 1). The continuous sampled position allows our method to arbitrarily upsample the rendered rays and thus synthesize novel views of the same unseen scene at multiple levels of detail (anti-blurring and anti-aliasing). Furthermore, recent generalizable NeRF methods [30, 38] introduce multi-view depth estimation to reduce the artifacts caused by occlusions, but it is computationally expensive to construct the cost volume for each view. We instead match local multi-view feature patches to estimate the visibility weights of each sample for anti-occlusion. Overall, our main contributions are:

1. A new generalizable approach that renders pixels by casting cones and outperforms existing methods on novel view synthesis at multiple scales.
2. A local implicit ray function that simplifies the representation of conical frustums and enables continuous supersampling of rays.
3. A transformer-based visibility weight estimation module that alleviates the occlusion problem.



(a) Inaccurate Features (b) Ambiguous Supervisions

Figure 2. Most generalizable variants of NeRF represent a ray as a set of infinitesimal samples (shown here as dots) along that ray and map these samples into input views to query image features for volumetric representation prediction. However, this results in two drawbacks when training on multi-scale images with less constrained settings: (a) Inaccurate features. The sampling strategy which ignores the shape and size of each ray is difficult to query accurate image features. (b) Ambiguous supervisions. The same 3D position captured by cameras under different scales results in different colors because these pixels are the integral of regions with different shapes and sizes (shown here as trapezoids). During the training, the network learns to map the same image features (from the source view) to these different colors, which causes ambiguous results.

To evaluate our method, we construct extensive series of experiments on real forward-facing scenes. Our experiments show that trained on large amounts of multi-view data, LIRF outperforms state-of-the-art generalizable NeRF-like methods on novel views synthesis for unseen scenes.

2. Related Work

Image-based rendering. When the images for a scene are captured densely, earlier lines of work directly interpolate novel views from input views [20, 33] or render views by weighted blending [6, 14]. More recently, researchers have been focused on the issue of novel views synthesis from sparse inputs. To map and blend the input images in a novel target view, some methods [7, 45] use the proxy geometry obtained via structure-from-motion (SfM) or multi-view stereo (MVS). Moreover, to improve the accuracy of mapping and blending, some improved methods are proposed for proxy geometry estimation [9, 25], optical flow correction [16, 17] and deep blending [24, 50, 51]. Differently, some other methods directly reconstruct the texture of mesh [15,26,61] or points cloud [1,42,48,53] for novel view synthesis. However, the rendering quality by IBR methods is directly affected by geometric inaccuracies of 3D recon-

struction methods [28, 54], especially in low-textured or reflective regions.

Explicit volumetric representations. The other methods render high-quality novel views by constructing an explicit scene representation, such as voxel grids [31, 63] and layered-depth images [47, 56], from captured images. For example, Zhou *et al* [75]. represent a scene with a set of depth-dependent images with transparency, which is called multi-plane images (MPIs). Novel views can be directly rendered from MPIs using alpha compositing. Recently, MPI representation has been applied in many scenarios, such as rendering novel views from multiple MPIs [43], novel view synthesis from a single image [35, 62], variants as multi-sphere images [2, 5], and 3D image generation [68, 74]. Trained on a large multi-view dataset, MPI-based methods can generalize well to unseen scenes and render photo-realistic novel views fast. However, those methods always render views within limited viewing volumes. Besides, the volumetric representations explicitly decompose a scene into extensive samples, which requires large memory to store them and limits the resolution of novel views.

Neural scene representations. Representing the geometry and appearance of a scene with neural networks has been a surge. Traditional methods explicitly represent scenes with point clouds [53, 66], meshes [26, 61], or voxels [39, 57], while neural scene representations implicitly encode the scene with continuous coordinate-based functions such as radiance fields [3, 44], signed distance fields [8, 19, 70], or occupancy fields [41, 46]. NeRF [44] approximates a 3D scene with an implicit function and has shown high-quality view synthesis results. Mip-NeRF [3] reduces objectionable aliasing artifacts and improves NeRF's ability to represent scene details by casting a cone instead of a ray. While NeRF has been expanded to many new scenarios [4, 11, 22, 27, 36, 40, 49, 58, 67, 72], most NeRF-like models take coordinate as inputs, which restricts their ability to generalize to unseen scenes. Recently, some generalizable NeRF-like methods have been proposed [10, 13, 30, 34, 38, 65, 71]. PixelNeRF [71], SRF [13], MINE [34] and MVSNeRF [10] try to construct radiance fields on-the-fly from sparse input views, while they struggle with complex scenes. To solve occlusions, NeuRay [38] and GeoNeRF [30] estimate depth using MVS methods [21, 69] and calculate the occlusion masks from the estimated depth maps. However, it's expensive to construct cost volumes per source view to estimate depth. While these generalizable methods have been capable of rendering novel views for unseen scenes, rendering novel views by casting a single ray may produce renderings that are aliased or blurred.

3. Method

Our goal is to predict volumetric radiance fields from a set of multi-view images captured at a consistent image scale ($\times 1$), and output novel views at continuous scales ($\times 0.5 \sim \times 4$). Our proposed framework is presented in Fig. 3, which is composed of five parts: 1) extracting 2D feature maps from source images (Sec. 3.2), 2) obtaining the image feature for the samples on target rays via local implicit ray function (Sec. 3.3), 3) predicting the visibility weights of each source view by matching feature patches (Sec. 3.4), 4) aggregating local image features from different source views and mapping them into colors and densities (Sec. 3.5), 5) rendering a target pixel via volume rendering (Sec. 3.1).

3.1. Volume Rendering

We first briefly review the volume rendering in NeRF [44]. NeRF implicitly encodes a scene as a continuous radiance field. Given a ray $\mathbf{r}$ passing through the scene, NeRF maps the 5D coordinates (3D for position, 2D for direction) of each sampled location into view-dependent color and volume density via an MLP. To determine the pixel color of the ray $\mathbf{r}$, NeRF first obtains the colors and densities of all samples on the ray $\mathbf{r}$ and then accumulates the colors according to densities. The volume rendering thus is defined by:

$$\begin{aligned} \widehat{C}(\mathbf{r}) &= \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i, \\ T_i &= \exp\left(- \sum_{l=1}^{i-1} \sigma_l \delta_l \right), \end{aligned} \tag{1}$$

where N is the number of samples along a ray with ascending depth values. The $\mathbf{c}$ and σ denote colors and densities, respectively. T denotes the accumulated transmittance. The δ is the distance between adjacent samples. Following IBRNet [65], we remove δ in volume rendering for a better generalization. To render a novel view of unseen scenes, we will introduce our proposed method to obtain colors and densities from input images instead of coordinates.

3.2. Image Feature Extraction

Given a target viewpoint, we select a set of V nearby source images and their corresponding camera poses as inputs. Our method relies on the local image features from source images to produce target images. Usually, most generalizable methods [38, 65] extract dense image features from each source view by a U-Net [52] architecture with ResNet [23]. Although U-Net and ResNet are widely adopted in high-level computer vision tasks such as image recognition and image segmentation, our rendering is based on pixel-wise operations. The feature extraction network is supposed to pay more attention to image details, so we use an EDSR network [37] to extract image features. The EDSR network removes unnecessary modules in conventional residual networks, which makes the network more

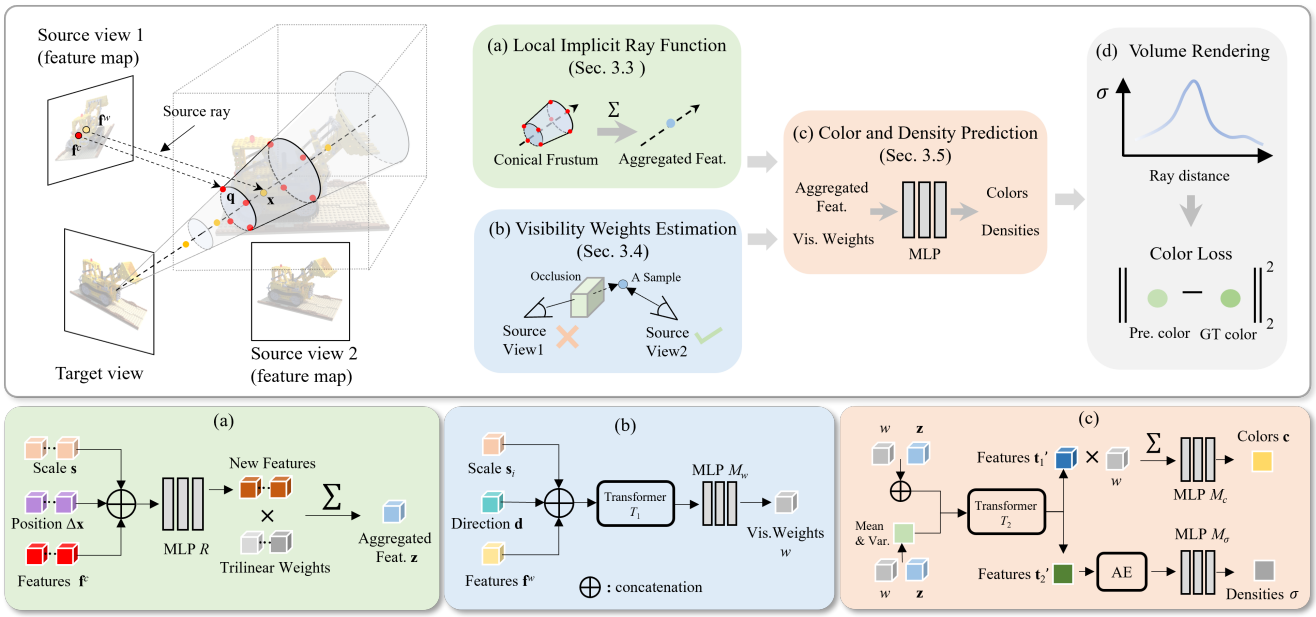


Figure 3. The overview of LIRF. To render a target image, a set of V neighboring source views are selected and the 2D feature map of each source view is extracted via an EDSR network [37]. N points (yellow points) are sampled along a target ray, and the corresponding conical frustum is represented with M vertices (red points). (a) For a sample on target ray at 3D position $\mathbf{x}$, we obtain its feature $\mathbf{z}$ by aggregating the features of the vertices and two latent codes (relative position $\Delta\mathbf{x} = \mathbf{x} - \mathbf{q}$ and target scale $\mathbf{s}$). (b) To deal with occlusions, we also estimate the visibility weights of each source view. Visibility weights can be considered the probability of a sample being observed by each source view. Taking the image features $\mathbf{f}^w$, source ray direction $\mathbf{d}$ and target scale $\mathbf{s}$ as inputs, our model outputs the visibility weights in source views of sample $\mathbf{x}$. (c) After obtaining the features $\mathbf{z}$ and corresponding visibility weights w, we aggregate them by a transformer [64] module $\mathcal{T}_2$ which outputs two parts of features: one for the colors and the other for densities. Next, we explicitly blend features $\mathbf{t}_1{}'$ and map them into colors $\mathbf{c}$ by an MLP $\mathcal{M}_c$. As for features $\mathbf{t}_2{}'$, an AE network [30] aggregates their information along a ray and an MLP $\mathcal{M}_\sigma$ maps the features to densities σ. (d) Finally, the color of target ray is rendered by volume rendering from the colors $\mathbf{c}$ and densities σ. The mean squared error between the predicted color and ground truth color is calculated for optimization.

focused on detailed content. We modify the last layer in the EDSR network to output two image features for the following modules, with one image feature for the estimation of visibility weights, and the other for the prediction of colors and densities. Given a source image I, the image feature extraction is formulated as:

$$(\mathbf{F}^w, \mathbf{F}^c) = \text{EDSR}(I), \tag{2}$$

where $\mathbf{F}^w$ denotes the feature map for visibility weights and $\mathbf{F}^c$ denotes the feature map for colors and densities.

3.3. Local Implicit Ray Function

In our method, a target ray is defined as the integral of a conical volume. To consider the shape and size of the volume viewed by each ray, we define the feature of each sample along the ray by a continuous integration within the corresponding conical frustum. Given the 3D position $\mathbf{x}$ within the conical frustum, it's feature $\mathbf{z}$ is defined as:

$$\mathbf{z} = \int W(||\Delta\mathbf{x}_\mathbf{q}||_2)\mathcal{R}(\mathbf{f}_q, \Delta\mathbf{x}_\mathbf{q})\, \mathrm{d}\mathbf{q}, \tag{3}$$

where $\mathbf{q}$ denotes a 3D position within the conical frustum, $\mathbf{f}_\mathbf{q}$ is the feature at position $\mathbf{q}$, $\Delta\mathbf{x}_\mathbf{q} = \mathbf{x} - \mathbf{q}$ denotes the relative position between two points, and $||\cdot||_2$ denotes the Euclidean distance. W is a weights function that outputs weights according to the distance $||\Delta\mathbf{x}_\mathbf{q}||_2$. $\mathcal{R}$ is an MLP-based function for the aggregation of features and positions.

Discrete representation. Considering the computation and memory consumption, we cast four rays from the cone's footprint to discretely approximate the continuous conical volume. Inspired by Plenoxels [18], a voxel-based representation, each conical frustum along the cone is represented with 8 vertices, and the target ray passes through all conical frustums, as shown in Fig. 4(b). Any samples within the conical frustum can be obtained by trilinear interpolation, which is similar to the sampling strategy of voxel-based representation. Specifically, we project the vertices on the four rays into the source views and query corresponding features $\mathbf{f}^c$ from feature maps $\mathbf{F}^c$. For a sample at position $\mathbf{x}$ within the corresponding conical frustum, its feature is defined as:

$$\begin{aligned} \mathbf{z} &= \sum_{j=1}^{M} W(||\Delta\mathbf{x}_j||_2)\mathcal{R}(\mathbf{f}_j^c, \Delta\mathbf{x}_j), \\ W(||\Delta\mathbf{x}_j||_2) &= \frac{||\Delta\mathbf{x}_j||_2}{\sum_{l=1}^{M} ||\Delta\mathbf{x}_l||_2}, \end{aligned} \tag{4}$$

where $M = 8$ is the number of vertices used to represent a conical frustum. $\mathbf{f}^c$ is the image feature of the vertices on the conical frustum. $\Delta\mathbf{x}$ denotes the relative position between the sample and the vertices within the conical frustum.

Cone radius. A cone cast from the view origin and passing through the image plane, the radius of the cone at the image plane is called cone radius and parameterized as r. The cone radius is affected by image resolutions and observation distances (being closer to objects means a larger scale). However, it is hard to know the observation distances beforehand. Besides, since our testing scenes are captured using a hand-held camera, the observation distance of novel views and source views are slightly different. To adjust the cone size accurately and conveniently, we don't directly modify the cone size only according to the relative image resolution. Instead, we set a maximum cone radius r_m which is the pixel width of the target image with the minimum resolution, and a latent code is introduced to implicitly control the size of a cone. Specifically, we first cast four rays passing through the four red points in the target image plane, as shown in Fig. 4(a). For any target ray projecting within the circle with radius r_m, its features are aggregated from the features of the four neighboring rays. Moreover, a scale-dependent input $\mathbf{s}$ is additionally introduced to modify the cone size, where $\mathbf{s}$ is the relative scale of a target image. We reformulate Eq. (4) with the form:

$$\mathbf{z} = \sum_{j=1}^{M} W(||\Delta\mathbf{x}_j||_2)\mathcal{R}(\mathbf{f}_j^c, \Delta\mathbf{x}_j, \mathbf{s}). \tag{5}$$

In practice, the positional encoding strategy [44] is applied on the relative position $\Delta\mathbf{x}$ but not on relative scale $\mathbf{s}$, since we want the implicit function to learn more high-frequency information of image feature and adjust the scale of renderings smoothly.

3.4. Visibility Weights Estimation

In this stage, we estimate the visibility weight that reveals the occlusions in each source view. Without any geometric priors, it's challenging to solve the occlusions thoroughly. As a compromise, we assume that the occluded content appears in most source views. Our model estimates the visibility weights of one source view by matching its feature patch with the feature patches from other source views. For the samples on a target ray, we obtain their feature patches $\mathbf{f}^w$ from the image features $\mathbf{F}^w$ as shown in Fig. 4(c). Apart from the feature patches, two latent codes are introduced to improve the performance. To consider the direction for each source ray, their directions $\mathbf{d}$ in target camera coordinates are input. We do not use the global directions of source rays in world coordinates, since different scenes have different world coordinates and local directions are more suitable for the generalization to unseen scenes.

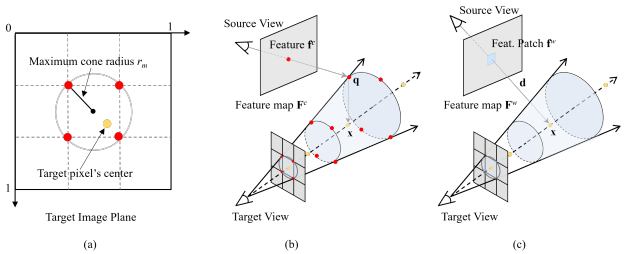


Figure 4. Method details of LIRF. The $\mathbf{x}$ denotes the position of a sample, $\mathbf{q}$ denotes the position of a vertex, and $\mathbf{d}$ denotes the direction of a source ray. (a) The rays projecting at target image plane. The size of target image is normalized. (b) A cone is represented with four rays (solid lines, only two are drawn) and a conical frustum is represented with eight vertices (red points). A target ray passes through the cone (dotted line). (c) The feature patches for visibility weights are obtained by warping samples on the target ray to feature maps.

To consider the scale of the target image, we also introduce the target scale $\mathbf{s}$. We define the feature concatenation of all inputs as:

$$\mathbf{t}_0 = \text{MLP}(\mathbf{f}^w \parallel \mathbf{d} \parallel \mathbf{s}), \tag{6}$$

where $\parallel$ denotes the concatenation operation and MLP denotes a two-layer MLP used to reduce the channels of features. Positional encoding [44] is applied on directions $\mathbf{d}$.

Next, a self-attention transformer [64] module $\mathcal{T}_1$ is used to fully match these aggregated feature $\mathbf{t}_0$ and output new features for visibility weights. Formally, this stage is written as:

$$\mathbf{t}_0{}' = \mathcal{T}_1(\mathbf{t}_0). \tag{7}$$

We then map the new feature $\mathbf{t}_0{}'$ into visibility weights w by an MLP $\mathcal{M}_w$,

$$w = \mathcal{M}_w(\mathbf{t}_0{}'). \tag{8}$$

3.5. Color and Density Prediction

After obtaining the local image features $\mathbf{z}$ and the visibility weights w, the colors and densities can be predicted. We aggregate the features and visibility weights to features $\mathbf{t}_1$. Besides, the weighted mean and weighted variance of features $\mathbf{z}$ over all source views are calculated based on visibility weights w. The mean and variance are then aggregated into features $\mathbf{t}_2$. This process is defined as:

$$\begin{aligned} \mathbf{t}_1 &= \text{MLP}(\mathbf{z} \parallel w), \\ \mathbf{t}_2 &= \text{MLP}(mean(\mathbf{z}, w) \parallel var(\mathbf{z}, w)). \end{aligned} \tag{9}$$

$\mathbf{t}_1$ could be considered as the color information from source views, while $\mathbf{t}_2$ could be considered as the measure of image feature consistency. The densities are predicted by checking the feature consistency, since local image features from different source views are usually consistent when the sample is on the object surface [38].

Next, the two features are fed into a self-attention transformer $\mathcal{T}_2$ for fully information aggregation,

$$(\mathbf{t}_1{}', \mathbf{t}_2{}') = \mathcal{T}_2(\mathbf{t}_1, \mathbf{t}_2). \tag{10}$$

The feature $\mathbf{t}_1{}'$ and $\mathbf{t}_2{}'$ now have combined the information from the the local image features and visibility weights. Similar to most generalizable NeRF-like methods [30,38,65], the density is estimated by an auto-encoder-style network [30] AE which aggregates information along a ray, and an MLP $\mathcal{M}_\sigma$ that maps features to densities,

$$\sigma = \mathcal{M}_\sigma \left(\mathrm{AE}(\mathbf{t}_2{}')\right). \tag{11}$$

As for colors, we further explicitly blend the features $\mathbf{t}_1{}'$ according to the visibility weights, and then an MLP $\mathcal{M}_c$ are used for color prediction,

$$\mathbf{c} = \mathcal{M}_c \left(\sum_{k=1}^{V} (\mathbf{t}_1{}'_{,k} w_k) \right). \tag{12}$$

Once the densities and colors of the samples on target rays are predicted, the final colors of target rays are rendered via volume rendering (Eq. (1)).

3.6. Loss Function

We minimize the mean squared error between our predicted colors and ground truth colors for optimization:

$$\mathcal{L} = \sum_{\mathbf{r} \in \Omega} \|\widehat{C}(\mathbf{r}) - C(\mathbf{r})\|_2^2, \tag{13}$$

where Ω is a set of camera rays at target position. $\widehat{C}$ and C are the predicted colors and ground truth colors, respectively.

4. Experiments

4.1. Implementation Details

To render a target view, we select nearby source views using $V = 8$ during both training and testing. $M = 8$ vertices are used to represent a conical frustum. The cone radius r_m is set to the pixel width of the target image with the minimum resolution. We directly sample $N = 128$ points on a ray instead of using the coarse-to-fine sampling [44]. The extracted image feature maps have 32 channels and their size is the same as the original image size. The size of feature patches used for visibility weights is set to 7×7. We train the generalizable LIRF for 250k iterations. The training batch size of rays is set to 512. We use Adam optimizer [32] with a learning rate of 5×10^{-4} that decays exponentially along with the optimization. We train our model on eight V100 GPUs, which takes about two days.

4.2. Datasets

Our model is trained on three real datasets: real DTU multi-view dataset [29] and two real forward-facing datasets from LLFF [43] and IBRNet [65]. All 190 scenes (35 scenes from LLFF, 67 scenes from IBRNet and 88 scenes from DTU dataset) are used for training. Eight unseen scenes from the LLFF dataset are used for testing. During the multi-scale training, the resolutions of all input views are consistent, while the resolution of each target view is randomly selected from 1 to 4 times the input resolution. When training our model on single-scale datasets, the image resolutions of input and target images are the same. During testing, the resolution of input views is 504×378. We evaluate our model by rendering novel views at multiple scales: $\times 0.5$, $\times 1$, $\times 2$ and $\times 4$ ($\times 0.5$ denotes 0.5 times the resolution of input views, and so on).

4.3. Results

We evaluate our model on LLFF testing scenes and compare it against three state-of-the-art generalizable methods: IBRNet [65], NeuRay [38] and GeoNeRF [30]. For quantitative evaluations, we report three error metrics, PSNR, SSIM and LPIPS [73]. Following prior work [3, 60], we also summarize all three error metrics into an "average" error metric by calculating the geometric mean of MSE$=10^{-\mathrm{PSNR}/10}$, $\sqrt{1-\mathrm{SSIM}}$, and LPIPS, which provides an easier comparison.

Multi-scale novel views. The performance of our method on rendering multi-scale novel views can be seen in Tab. 1. and Fig. 5. As shown in Tab. 1, our method outperforms baseline methods on all four scales. Though our model isn't trained on $\times 0.5$ scale, it also can render low-scale views by explicitly modifying the cone radius r_m according to the pixel size at $\times 0.5$ scale. In contrast, the baselines fail to render view at a lower scale due to the aliasing artifacts (shown as the $\times 0.5$ results by IBRNet in Fig. 1(c)). We notice that the baselines produce relatively better results on rendering novel views at an up-scale after training on multi-scale datasets. On the other hand, they produce worse results on low-scale ($\times 0.5$ and $\times 1$) compared with their released models, since baselines have difficulty converging well when supervisions have different image scales, as discussed in Fig. 2 (b). Besides, our dataset contains fewer training scenes (about 15% of the training scenes used by IBRNet or NeuRay). The qualitative comparisons are shown in Fig. 5. It can be seen that the improvements produced by our method are most visually apparent in challenging cases such as small or thin structures. Besides, our renderings have fewer blurring artifacts and are closer to the ground truth.

Single-scale novel views. We also train our model on the single-scale dataset to evaluate our ability on novel view synthesis. The quantitative results are shown in Tab. 2. One

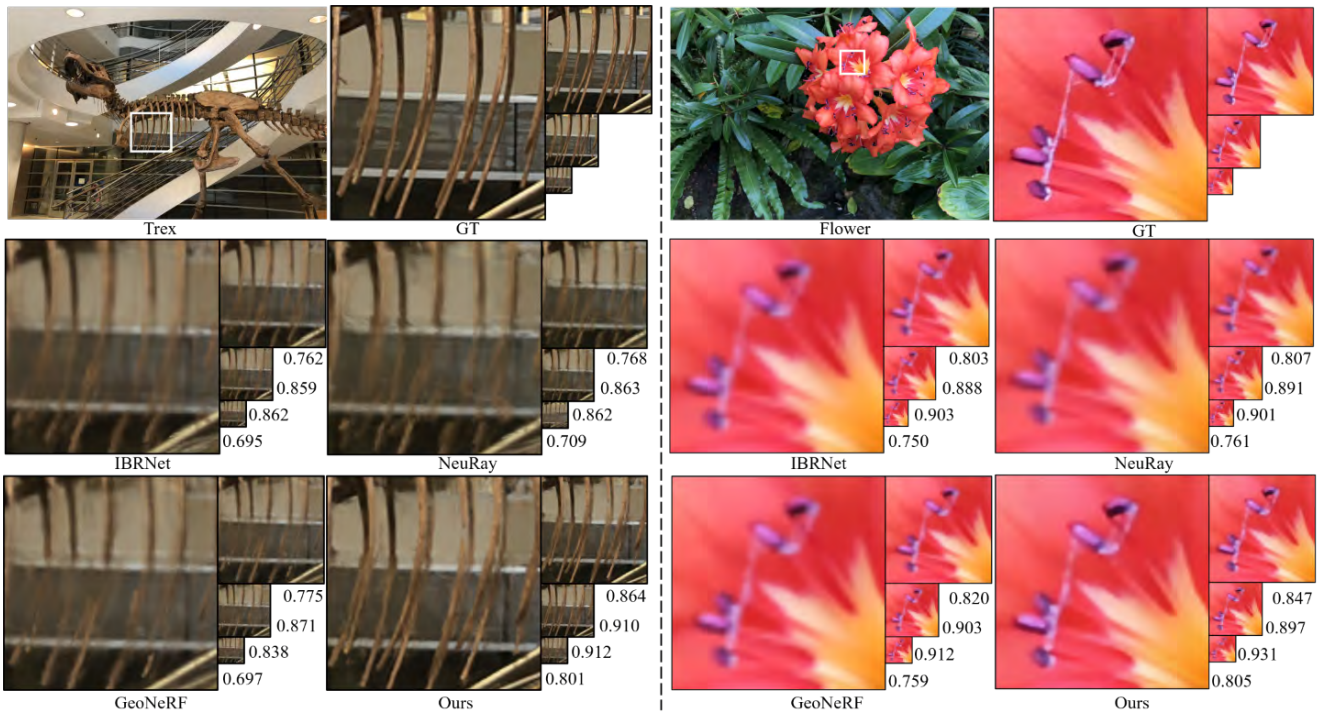


Figure 5. Qualitative comparisons on real forward-facing scenes. The cropped regions of two scenes are rendered at different scales ($\times 0.5$, $\times 1$,$\times 2$ and $\times 4$). SSIM averaged over all novel views for each scale are provided at the lower right of images.

Table 1. Quantitative comparisons of LIRF and its ablations against IBRNet [65], NeuRay [38] and GeoNeRF [30] on LLFF multi-scale testing dataset. Metrics are averaged over all scenes. $*$ denotes training on the same multi-scale training set as our method.

	PSNR↑				SSIM↑				LPIPS↓				Avg.↓
	$\times 0.5$	$\times 1$	$\times 2$	$\times 4$	$\times 0.5$	$\times 1$	$\times 2$	$\times 4$	$\times 0.5$	$\times 1$	$\times 2$	$\times 4$	
IBRNet	25.06	25.28	23.18	22.15	0.866	0.840	0.731	0.669	0.108	0.160	0.299	0.442	0.079
NeuRay	24.80	25.17	23.10	22.08	0.859	0.837	0.729	0.667	0.107	0.157	0.294	0.434	0.080
GeoNeRF	24.89	25.74	23.43	22.27	0.864	0.864	0.753	0.679	0.108	0.136	0.274	0.421	0.076
IBRNet*	22.96	23.62	22.33	21.51	0.816	0.813	0.723	0.665	0.140	0.178	0.307	0.444	0.090
NeuRay*	22.79	22.40	21.23	20.61	0.794	0.733	0.646	0.622	0.172	0.262	0.382	0.493	0.107
GeoNeRF*	23.39	25.08	23.81	22.69	0.821	0.859	0.784	0.708	0.138	0.134	0.255	0.401	0.077
Ours	26.75	25.93	24.58	23.79	0.905	0.877	0.816	0.760	0.100	0.124	0.227	0.373	0.063
Ours (single ray)	25.38	25.72	24.27	23.43	0.881	0.871	0.794	0.723	0.135	0.132	0.254	0.392	0.069
Ours w/o scale	26.81	25.90	24.52	23.72	0.904	0.873	0.807	0.749	0.098	0.128	0.235	0.374	0.064
Ours w/o position	25.95	25.72	24.36	23.56	0.882	0.869	0.803	0.741	0.127	0.136	0.242	0.373	0.067
Ours w/o patch	26.85	25.95	24.56	23.76	0.906	0.877	0.813	0.756	0.097	0.127	0.231	0.367	0.064
Ours w/o direction	26.35	25.36	24.10	23.38	0.899	0.864	0.799	0.744	0.103	0.137	0.242	0.373	0.067
Ours w/o vis. weights	25.90	25.11	23.91	23.19	0.888	0.856	0.792	0.737	0.118	0.148	0.250	0.378	0.070
Ours (U-Net feat.)	26.00	25.10	23.43	22.77	0.887	0.853	0.761	0.706	0.116	0.151	0.298	0.427	0.075

can see that training on the single-scale dataset does not reduce our performance on rendering novel views at $\times 1$ scale, and even improves our results on PSNR and SSIM metrics. Figure 6 presents the visualized comparisons on cropped regions. Although training on the single-scale dataset, our method shows a visual improvement on the challenging objects, such as the thin leaves.

Visibility weights. To demonstrate the effectiveness of our method in dealing with occlusions, we visualize the estimated visibility weights. As shown in Fig. 7, region A is occluded in source view 1, so it has smaller visibility weights in region A. Conversely, region A is visible in source view 2, while region B is occluded. Therefore, the visibility weights of source view 2 have larger values at region A and smaller values at region B. With visibility weights estimation, our method correctly renders the scene content at

Figure 6. Qualitative comparisons on single-scale novel view synthesis. Though trained on the single-scale dataset, Ours′ still outperforms baselines. SSIM averaged over all novel views are provided at lower right.

Table 2. Quantitative comparisons of LIRF trained on single-scale dataset against baseline methods on rendering novel view at $\times 1$ scale. "Ours′" denotes our method training on single-scale dataset, while "Ours" denotes our method training on multi-scale dataset.

	IBRNet	NeuRay	GeoNeRF	Ours′	Ours
PSNR↑	25.28	25.17	25.74	26.30	25.93
SSIM↑	0.840	0.837	0.864	0.878	0.877
LPIPS↓	0.160	0.157	0.136	0.128	0.124

regions A and B.

4.4. Ablation Studies

Ablation studies are shown in Tab. 1 to investigate the individual contribution of key modules of our model. (a) For "single ray", we modify our model to render a pixel from a single ray instead of a cone and keep everything else the same. Compared with our full model, the performance by "single ray" is reduced on all four testing scales, especially on the $\times 0.5$ scale and $\times 4$ scale, which demonstrates the contribution of our local implicit ray function for rendering multi-scale novel views. (b) For "w/o scale", we remove the scale $\mathbf{s}$ in Eq. (5) and Eq. (6). It reduces our model's ability to modify our implicit ray function according to the scale of target views. (c) For "w/o position", the relative position $\Delta\mathbf{x}$ in Eq. (5) is removed, which prevents our implicit ray function from perceiving the relative 3D position between a sample and the vertices of conical frustum. (d) For "w/o patch", the size of the feature patches used to predict visibility weights is set to 1×1. This reduces our performance on rendering novel views with higher scales while improving our model on rendering $\times 0.5$ novel views. When testing on different scales, the patch size is set to 7×7, which may be too large to render views at $\times 0.5$ scale. (e) For "w/o direction", we remove the direction $\mathbf{d}$ in Eq. (6). Without considering the direction of source rays, our model produces worse results on all testing scales. (f) For "w/o vis. weights", we remove the visibility weights estimation module. Without visibility weights, our model fails to solve the occlusions, which significantly reduces our performance. (g) For "U-Net feat.", the image feature maps are extracted by the U-Net of IBRNet [65]. We can see that our performance is reduced a lot. Most recent generalizable methods focus on the framework design of predicting color and densities from image features. Actually, the image feature extraction network is also important, affecting the quality of novel views from the source.

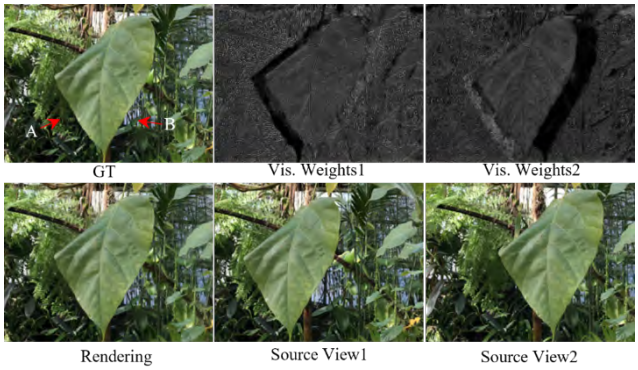


Figure 7. Visualizations of visibility weights (only two source views are presented). Our model accurately estimates the invisible regions for source views. For example, region A is occluded in source view 1, thus the visibility weights at this region are "dark".

5. Conclusion

We have proposed a novel method for novel view synthesis of unseen scenes. Our method not only renders novel views with fewer blurring artifacts, but also produces novel views at arbitrary scales, even at higher scales than input views. The core of our method is the local implicit ray function that constructs each target ray by aggregating the information of a cone. Besides, our method also estimates the visibility weights to mitigate the occlusion problem. Compared with prior works, our method performs better on novel view synthesis of unseen scenes. Our code and models will be released to the research community to facilitate reproducible research.

Limitations. Rendering novel views of unseen scenes is a challenging task. Similar to most image-based rendering methods, our method renders novel views from selected local input views, which causes two problems. First, our method can't render the regions occluded in all input views. Second, since selected input views vary with the novel view position, compared with NeRF, our method is weak in view consistency. Moreover, our method renders a pixel by casting four rays to approximate a cone, which increases the computational cost. Finally, though our method can mitigate the artifacts caused by occlusions, it fails in some challenging scenes. To solve this, a possible choice is introducing geometric priors.

Acknowledgements. The work was supported by NSFC under Grant 62031023.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, pages 696–712, 2020. 2

[2] Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *ECCV*, 2020. 3

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5855–5864, 2021. 2, 3, 6

[4] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. NeRD: Neural reflectance decomposition from image collections. In *ICCV*, pages 12684–12694, 2021. 3

[5] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM TOG*, 39(4):86–1, 2020. 3

[6] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, pages 425–432, 2001. 1, 2

[7] Rodrigo Ortiz Cayon, Abdelaziz Djelouah, and George Drettakis. A bayesian approach for selective image-based rendering using superpixels. In *3DV*, pages 469–477, 2015. 2

[8] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *ECCV*, pages 608–625, 2020. 3

[9] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM TOG*, 32(3):1–12, 2013. 2

[10] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, pages 14124–14133, 2021. 2, 3

[11] Xingyu Chen, Qi Zhang, Xiaoyu Li, Yue Chen, Ying Feng, Xuan Wang, and Jue Wang. Hallucinated neural radiance fields in the wild. In *CVPR*, pages 12943–12952, 2022. 3

[12] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *CVPR*, pages 8628–8638, 2021. 12, 13

[13] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes. In *CVPR*, pages 7911–7920, 2021. 2, 3

[14] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. In *Comput. Graph. Forum*, volume 31, pages 305–314, 2012. 2

[15] Paul Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Workshop on Rendering Techniques*, pages 105–116. Springer, 1998. 2

[16] Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. Montage4d: interactive seamless fusion of multiview video textures. In *I3D*, 2018. 2

[17] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating textures. *Comput. Graph. Forum*, 27(2):409–418, 2008. 2

[18] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5501–5510, 2022. 4

[19] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *CVPR*, pages 4857–4866, 2020. 3

[20] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996. 1, 2

[21] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, pages 2495–2504, 2020. 2, 3

[22] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv preprint arXiv:2012.08503*, 2020. 3

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3

[24] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM TOG*, 37(6):1–15, 2018. 2

[25] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. Scalable inside-out image-based rendering. *ACM TOG*, 35(6):1–11, 2016. 2

[26] Jingwei Huang, Justus Thies, Angela Dai, Abhijit Kundu, Chiyu Jiang, Leonidas J Guibas, Matthias Nießner, Thomas Funkhouser, et al. Adversarial texture optimization from rgb-d scans. In *CVPR*, pages 1559–1568, 2020. 2, 3

[27] Xin Huang, Qi Zhang, Ying Feng, Hongdong Li, Xuan Wang, and Qing Wang. HDR-NeRF: High dynamic range neural radiance fields. In *CVPR*, pages 18398–18408, 2022. 3

[28] Michal Jancosek and Tomás Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*, pages 3121–3128, 2011. 3

[29] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, pages 406–413, 2014. 6, 12

[30] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. GeoNeRF: Generalizing nerf with geometry priors. In *CVPR*, pages 18365–18375, 2022. 2, 3, 4, 6, 7, 12, 16

[31] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *NeurIPS*, 30, 2017. 3

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[33] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, 1996. 1, 2

[34] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *ICCV*, pages 12578–12588, 2021. 2, 3

[35] Qinbo Li and Nima Khademi Kalantari. Synthesizing light field from a single image with variable mpi and two network fusion. *ACM TOG*, 39(6), 2020. 3

[36] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, pages 6498–6508, 2021. 3

[37] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, pages 136–144, 2017. 3, 4, 12

[38] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *CVPR*, pages 7824–7833, 2022. 2, 3, 5, 6, 7, 12, 16

[39] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM TOG*, 38(4):65:1–65:14, 2019. 3

[40] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V Sander. Deblur-NeRF: Neural radiance fields from blurry images. In *CVPR*, pages 12861–12870, 2022. 3

[41] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, pages 4460–4470, 2019. 3

[42] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *CVPR*, pages 6878–6887, 2019. 2

[43] 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 TOG*, 38(4):1–14, 2019. 1, 3, 6, 12, 16

[44] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, pages 405–421, 2020. 1, 3, 5, 6, 12

[45] Ryan S Overbeck, Daniel Erickson, Daniel Evangelakos, Matt Pharr, and Paul Debevec. A system for acquiring, processing, and rendering panoramic light field stills for virtual reality. *ACM TOG*, 37(6):1–15, 2018. 2

[46] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, pages 523–540, 2020. 3

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

[48] Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. Revealing scenes by inverting structure from motion reconstructions. In *CVPR*, pages 145–154, 2019. 2

[49] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, pages 10318–10327, 2021. 3

[50] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, pages 623–640, 2020. 2

[51] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, pages 12216–12225, 2021. 2

[52] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015. 3

[53] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM TOG*, 41(4):1–14, 2022. 2, 3

[54] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. 3

[55] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, pages 501–518, 2016. 2

[56] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH*, pages 231–242, 1998. 3

[57] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*, pages 2437–2446, 2019. 3

[58] 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. In *CVPR*, pages 7495–7504, 2021. 3

[59] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. *arXiv preprint arXiv:2207.10662*, 2022. 2

[60] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *CVPR*, pages 8269–8279, 2022. 6

[61] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM TOG*, 38(4):1–12, 2019. 2, 3

[62] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, pages 551–560, 2020. 3

[63] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, pages 2626–2634, 2017. 3

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017. 4, 5

[65] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *CVPR*, pages 4690–4699, 2021. 2, 3, 6, 7, 8, 12, 13, 14, 15, 16

[66] Minye Wu, Yuehao Wang, Qiang Hu, and Jingyi Yu. Multi-view neural human rendering. In *CVPR*, pages 1682–1691, 2020. 3

[67] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, pages 9421–9431, 2021. 3

[68] Jianfeng Xiang, Jiaolong Yang, Yu Deng, and Xin Tong. Gram-hd: 3d-consistent image generation at high resolution with generative radiance manifolds. *arXiv preprint arXiv:2206.07255*, 2022. 3

[69] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, pages 767–783, 2018. 2, 3

[70] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *NeurIPS*, 34:4805–4815, 2021. 3

[71] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, pages 4578–4587, 2021. 2, 3, 12

[72] Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. Editable free-viewpoint video using a layered neural representation. *ACM TOG*, 40(4):1–18, 2021. 3

[73] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 6

[74] Xiaoming Zhao, Fangchang Ma, David Güera, Zhile Ren, Alexander G Schwing, and Alex Colburn. Generative multiplane images: Making a 2d gan 3d-aware. *arXiv preprint arXiv:2207.10642*, 2022. 3

[75] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM TOG*, 37(4):1–12, 2018. 1, 3

Supplemental Materials

A. Additional Implementation Details

A.1. Network Details

Our feature extraction network (EDSR [37]) is based on the implementation from this URL (https://github.com/sanghyun-son/EDSR-PyTorch). The tail module of the EDSR network is removed, and two convolutional layers are added at the tail. The one outputs the feature maps for visibility weights, while the other one outputs the features maps for colors and densities. The implementation AE network is from GeoNeRF [30]. The ray function $\mathcal{R}$ is a three-layer MLP with 32 channels for each linear layer. Both networks $\mathcal{M}_w$ and $\mathcal{M}_\sigma$ are a two-layer MLP with 32 channels. The network $\mathcal{M}_c$ is a three-layer MLP with 32 channels. The transformer module $\mathcal{T}_1$ contains a single multi-head self-attention layer with the number of heads set to 4, while $\mathcal{T}_2$ contains four multi-head self-attention layers with the number of heads set to 4. The "MLP" used to reduce feature channels is a two-layer MLP and the number of channels is set to 32. For all MLP-based networks, ELU is used between each of two adjacent linear layers as the non-linear activation function. In our experiments, all networks are trained from scratch. Our code and model will be made available.

A.2. Dataset Details

We train our model on three real datasets: the real DTU multi-view dataset [29] and two real forward-facing datasets from LLFF [43] and IBRNet [65]. All 190 scenes (35 scenes from LLFF dataset, 67 scenes from IBRNet dataset and 88 scenes from DTU dataset) are used for training. We exclude the views with incorrect exposure from the DTU dataset as done in pixelNeRF [71]. Eight unseen scenes from LLFF dataset are used as our testing scenes. During the multi-scale training, the resolutions of all input views are consistent (252×189 for LLFF and IBRNet datasets, 200×150 for DTU dataset), while the resolution of each target view is randomly selected from 1 to 4 times the input resolution (from 252×189 to 1008×756 for LLFF and IBRNet datasets, from 200×150 to 800×600 for DTU dataset). When training our model on single-scale datasets, the image resolutions of input and target images are the same (504×378). During testing, the resolution of input views is 504×378. We evaluate our model on rendering novel views at multiple scales: $\times 0.5$, $\times 1$, $\times 2$ and $\times 4$ ($\times 0.5$ denotes 0.5 times the resolution of input views, and so on). During the dataset preprocessing, we use bicubic interpolation to downsample high resolution images.

Table 3. Fine-tuning results of our method and state-of-the-art methods. We fine-tune our pretrained model on each scene for 10k iterations with resolution of 1008×756. The resolution of testing views is also set to 1008×756.

	PSNR↑	SSIM↑	LPIPS↓
NeRF [44]	26.50	0.811	0.250
IBRNet [65]	26.73	0.851	0.175
NeuRay [38]	27.06	0.850	0.172
GeoNeRF [30]	26.58	0.856	0.162
Ours(10k)	26.85	0.865	0.159

B. Additional Experiments

B.1. Fine-tuning

Although our approach focuses on generalizations to unseen scenes, we also fine-turn our pre-trained model on each testing scene for comparison against previous methods. We follow the setting of IBRNet [65] and train our model on each of the eight testing scenes for 10k iterations. The resolution of images used for training and testing is set to 1008×756. Note that the multi-view images used for fine-tuning are single-scale. The results are reported in Tab. 3.

B.2. Comparisons with Two-stage Methods

To further evaluate our method on rendering novel views at high scales ($\times 2$ and $\times 4$), we try to compare our method with two-stage methods. We first render novel views at $\times 1$ scale via three baselines and then upsample the novel views via bicubic interpolation and a single-image super resolution method, LIIF [12]. The results are presented in Tab. 5. It shows the superiority of our model on rendering novel views at high scales with respect to the two-stage methods, though they introduce external data priors.

B.3. Number of Source Views

To investigate the robustness of our model to the number of source views, our model is tested on unseen scenes with different numbers of source views (4, 6, 8, and 10). The quantitative results are shown in Tab. 4. The results show that our model produces competitive results when the number of source views is set to 6, 8, and 10. The model produces the best results when setting the number of source views to 8, since our model is trained with 8 source views. However, the performance of our method reduces a lot when the source views are sparse (4 views), since it is challenging to estimate visibility weights by matching sparse local image features.

B.4. Number of Vertices

We represent a conical frustum using several vertices. The results with different numbers of vertices are shown in Tab. 5. Using more vertices, our performance improves, but the rendering time increases too. Considering computing

Table 4. Quantitative comparisons of varying the number of source views on LLFF real forward-facing scenes.

	PSNR↑				SSIM↑				LPIPS↓				Avg.↓
	×0.5	×1	×2	×4	×0.5	×1	×2	×4	×0.5	×1	×2	×4	
4 views	24.80	24.03	22.93	22.31	0.864	0.825	0.761	0.711	0.134	0.168	0.269	0.403	0.080
6 views	26.11	25.51	24.24	23.50	0.893	0.866	0.804	0.750	0.106	0.131	0.233	0.377	0.066
8 views	26.75	25.93	24.58	23.79	0.905	0.877	0.816	0.760	0.100	0.124	0.227	0.373	0.063
10 views	26.46	25.91	24.56	23.78	0.900	0.877	0.815	0.760	0.101	0.126	0.231	0.376	0.064

Table 5. Quantitative comparisons of our LIRF against two-staget methods on rendering novel view at higher scales (×2 and ×4). We upsample low resolution novel views via bicubic interpolation (BI) or LIIF [12]. "M" denotes the number of vertices used to represent a conical frustum.

	PSNR↑		SSIM↑		LPIPS↓		Avg. ↓
	×2	×4	×2	×4	×2	×4	
IBRNet-BI	23.50	22.85	0.740	0.691	0.307	0.438	0.099
IBRNet-LIIF	23.80	23.11	0.760	0.712	0.278	0.421	0.093
NeuRay-BI	23.44	22.79	0.738	0.689	0.305	0.437	0.099
NeuRay-LIIF	23.70	23.02	0.757	0.709	0.276	0.419	0.094
GeoNeRF-BI	23.89	23.19	0.765	0.708	0.282	0.420	0.093
GeoNeRF-LIIF	24.26	23.53	0.788	0.733	0.251	0.400	0.087
Ours (M=4)	23.91	23.15	0.789	0.741	0.248	0.398	0.089
Ours (M=8)	24.58	23.79	0.816	0.760	0.227	0.373	0.081
Ours (M=10)	24.93	23.95	0.838	0.784	0.218	0.366	0.077

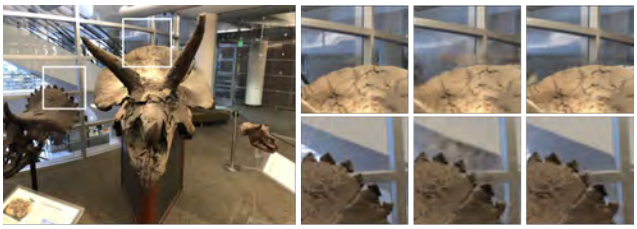


Figure 8. The qualitative results of our model without visibility weights. Ours denotes our full model.

burdens and inspired by the voxel-based volume rendering, we use M = 8 vertices to approximate a conical frustum. The samples within the conical frustum can be calculated by our implicit ray function.

B.5. Comparisons of Rendering Time

LIRF (45s for rendering an image with ×1 scale) is about three times slower than IBRnet (15s for rendering an image with ×1 scale). However, once the conical frustums are constructed, we directly infer rays from the conical frustums to render multi-scale views. Compared with baselines on rendering multi-scale views, we save the time of querying features from feature maps, especially on rendering high resolution views.

C. Additional Results

C.1. Qualitative Results for Ablation Studies

As shown in Tab. 6, three ablations (Ours(single ray), Ours w/o vis. weights and Ours(U-Net feat.)) mainly affect the performance of our LIRF. To further investigate their contributions to our model, the qualitative results are shown in Figs. 8, 10 and 11.

Ours w/o vis. weights. We remove the visibility weights estimation module to evaluate the impact of the visibility weights. Figure 8 shows the performance of our model without visibility weights. Our method produces renderings with ghosting artifacts on the boundary of objects due to occlusions.

Ours (single ray). To investigate the contribution of our local implicit ray function, we render a pixel from a single ray instead of conical frustums. The results are presented in Fig. 10. One can see that our model (single ray) produces renderings that are excessively aliased when rendering novel views at ×0.5 scale. Besides, our model (single ray) produces renderings containing artifacts at thin structures when rendering novel views at ×2 scale.

Ours (U-Net feat.). Moreover, the feature extraction network is also important to our method, especially on rendering novel views at high scales. We therefore extract 2D image features via the U-Net in IBRNet [65]. Our model with the U-Net is trained from scratch on our multi-scale dataset. The rendered testing views are presented in Fig. 11. Our model produces renderings with more blurred artifacts when the image features are extracted by the U-Net.

C.2. A Failure Case

As discussed in the limitations, though the visibility weights can mitigate the artifacts caused by occlusions, they fail in some challenging scenes such as the *orchids* scene. Figure 9 shows a failure example on the *orchids* scene. The multi-view images of this scene are captured sparsely, which is challenging for our model to estimate the accurate visibility weights. The baselines also struggle with this challenging scene, such as the renderings by IBRNet [65] with blurred artifacts. After fine-tuning on this scene for 10k iterations, our model produces results with fewer artifacts on the boundary of objects.

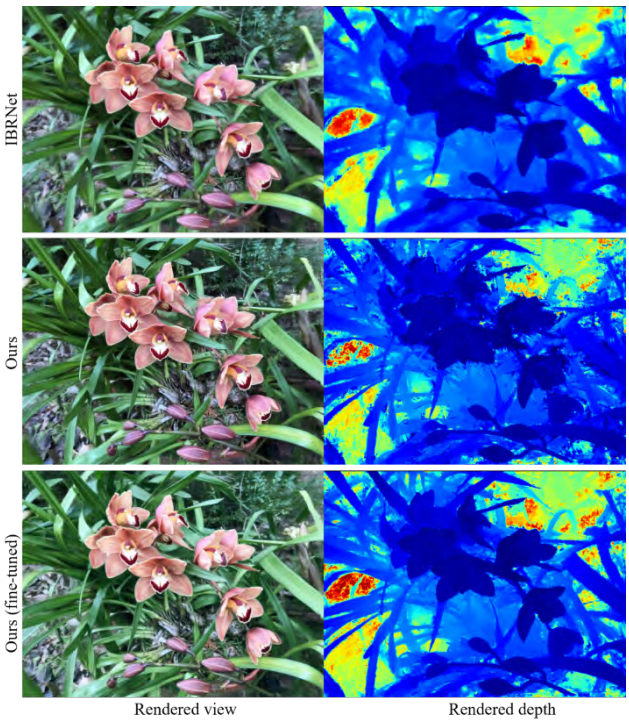


Figure 9. A failure example on *orchids* scene. Our model fails to predict the geometry on the boundaries of flowers due to occlusions. The renderings by IBRNet [65] also contain blurred artifacts. After fine-tuning on this scene for 10k iterations, our model produces results with fewer artifacts

C.3. Per-Scene Results

To evaluate our approach compared to previous methods on each individual scene, per-scene results on the eight testing scenes are presented in Tab. 6. We report the arithmetic mean of each metric averaged over the four testing scales used for testing. Our method yields a significant improvement in three error metrics across most scenes.

Figure 10. The qualitative results of our model that renders a pixel from a single ray. The top row shows the novel views rendered at $\times 0.5$ scale. Our model (single ray) produces aliased novel view. The bottom row shows the novel views rendered at $\times 2$ scale. Our model (single ray) produces novel view with artifacts at thin structures.

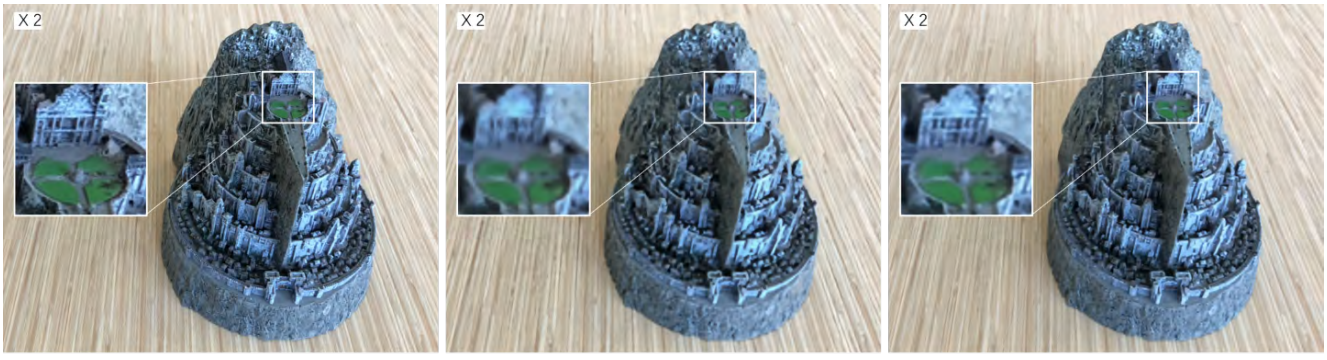


Figure 11. The qualitative results of our model that extracts image features via the U-Net in IBRNet [65]. Our model (U-Net feat.) produces novel views with more blurred artifacts when the image features are extracted by the U-Net.

Table 6. Per scene quantitative comparisons of our LIRF and its ablations against IBRNet [65], NeuRay [38] and GeoNeRF [30] on LLFF [43] multi-scale testing dataset. Metrics are averaged over four testing scales ($\times 0.5$, $\times 1$, $\times 2$ and $\times 4$). $*$ denotes training on the same multi-scale training set as our method.

	Average PSNR↑							
	fern	*flower*	*fortress*	*horns*	*leaves*	*orchids*	*room*	*trex*
IBRNet	23.40	25.80	28.12	24.78	19.69	19.20	27.49	22.89
NeuRay	23.21	25.89	28.18	24.78	19.48	18.87	27.09	22.81
GeoNeRF	23.73	26.35	28.88	25.19	19.75	19.81	27.02	21.91
IBRNet*	22.38	24.61	26.32	23.68	18.48	18.07	25.49	21.80
NeuRay*	21.26	23.56	25.43	22.45	17.89	17.50	25.12	20.84
GeoNeRF*	23.55	26.21	28.17	24.92	19.89	19.50	26.24	21.47
Ours	25.21	26.77	29.07	26.59	21.59	19.39	28.59	24.91
Ours w/o scale	25.05	26.63	29.25	26.56	21.33	19.28	28.75	25.06
Ours w/o patch	25.18	26.64	29.17	26.47	21.47	19.44	28.72	25.16
Ours w/o position	24.78	26.69	28.18	26.16	20.97	19.33	28.42	24.64
Ours w/o direction	24.60	26.31	28.34	25.65	20.86	19.22	28.66	24.73
Ours w/o vis. weights	24.72	25.95	27.92	25.50	20.66	18.98	27.75	24.75
Ours (U-Net feat.)	24.21	26.03	28.67	25.33	20.44	19.14	27.12	23.67
Ours (single ray)	24.49	26.60	28.11	25.78	20.83	19.28	28.07	24.41

	Average SSIM↑							
	fern	*flower*	*fortress*	*horns*	*leaves*	*orchids*	*room*	*trex*
IBRNet	0.741	0.836	0.832	0.805	0.678	0.629	0.899	0.794
NeuRay	0.739	0.836	0.833	0.808	0.668	0.617	0.896	0.790
GeoNeRF	0.768	0.847	0.844	0.825	0.683	0.659	0.897	0.795
IBRNet*	0.717	0.820	0.801	0.790	0.650	0.593	0.877	0.786
NeuRay*	0.675	0.761	0.723	0.733	0.568	0.531	0.862	0.735
GeoNeRF*	0.774	0.852	0.833	0.829	0.704	0.655	0.893	0.802
Ours	0.825	0.870	0.897	0.876	0.787	0.666	0.924	0.872
Ours w/o scale	0.817	0.865	0.896	0.870	0.776	0.656	0.921	0.866
Ours w/o patch	0.821	0.867	0.898	0.875	0.782	0.668	0.923	0.870
Ours w/o position	0.806	0.858	0.882	0.863	0.761	0.652	0.913	0.856
Ours w/o direction	0.806	0.861	0.891	0.864	0.756	0.651	0.920	0.864
Ours w/o vis. weights	0.807	0.849	0.886	0.853	0.748	0.635	0.910	0.860
Ours (U-Net feat.)	0.777	0.849	0.858	0.831	0.728	0.642	0.898	0.829
Ours (single ray)	0.799	0.856	0.873	0.849	0.757	0.651	0.904	0.851

	Average LPIPS↓							
	fern	*flower*	*fortress*	*horns*	*leaves*	*orchids*	*room*	*trex*
IBRNet	0.282	0.201	0.195	0.252	0.285	0.316	0.214	0.272
NeuRay	0.282	0.191	0.189	0.246	0.293	0.311	0.206	0.265
GeoNeRF	0.251	0.187	0.170	0.226	0.283	0.287	0.207	0.267
IBRNet*	0.297	0.208	0.221	0.263	0.297	0.339	0.235	0.279
NeuRay*	0.359	0.269	0.296	0.336	0.369	0.395	0.262	0.331
GeoNeRF*	0.245	0.176	0.181	0.224	0.264	0.288	0.212	0.265
Ours	0.217	0.174	0.152	0.191	0.219	0.288	0.190	0.219
Ours w/o scale	0.223	0.177	0.149	0.193	0.226	0.296	0.188	0.220
Ours w/o patch	0.221	0.175	0.149	0.187	0.222	0.289	0.185	0.216
Ours w/o position	0.234	0.181	0.165	0.199	0.254	0.311	0.188	0.223
Ours w/o direction	0.231	0.182	0.153	0.196	0.233	0.303	0.187	0.223
Ours w/o vis. weights	0.233	0.193	0.162	0.209	0.247	0.320	0.199	0.225
Ours (U-Net feat.)	0.266	0.199	0.200	0.246	0.269	0.313	0.227	0.263
Ours (single ray)	0.240	0.188	0.181	0.214	0.257	0.318	0.200	0.229