EfficientNeRF – Efficient Neural Radiance Fields

Tao Hu[1] Shu Liu [2] Yilun Chen [1] Tiancheng Shen [1] Jiaya Jia[1,2]
[1] The Chinese University of Hong Kong [2] SmartMore
{taohu,ylchen,tcshen,leojia}@cse.cuhk.edu.hk , sliu@smartmore.com

Abstract

Neural Radiance Fields (NeRF) has been wildly applied to various tasks for its high-quality representation of 3D scenes. It takes long per-scene training time and per-image testing time. In this paper, we present EfficientNeRF as an efficient NeRF-based method to represent 3D scene and synthesize novel-view images. Although several ways exist to accelerate the training or testing process, it is still difficult to much reduce time for both phases simultaneously. We analyze the density and weight distribution of the sampled points then propose valid and pivotal sampling at the coarse and fine stage, respectively, to significantly improve sampling efficiency. In addition, we design a novel data structure to cache the whole scene during testing to accelerate the rendering speed. Overall, our method can reduce over 88% of training time, reach rendering speed of over 200 FPS, while still achieving competitive accuracy. Experiments prove that our method promotes the practicality of NeRF in the real world and enables many applications. The code is available in https://github.com/dvlab-research/EfficientNeRF.

1. Introduction

Novel View Synthesis (NVS) aims to generate images at new views, given multiple camera-calibrated images. It is an effective line for realizing Visual or Augmented Reality. With Neural Radiance Fields (NeRF) [17] proposed, NVS tasks [20, 24], like large-scale or dynamic synthesis [21, 22, 25], were successfully dealt with in high quality. NeRF adopts implicit functions to directly map 3D-point spatial information, in terms of locations and directions, to the attributes of color and densities. To synthesize high-resolution images, NeRF needs to densely sample points over the whole scene, which consumes far more computation than traditional solutions [14, 16, 29]. For instance, for a scene containing 100 images with resolution 800×800, NeRF training time usually takes 1-2 days [17], and the per-image testing time is around 30 seconds. These two inefficiencies impede the fast practical applications of NeRF.

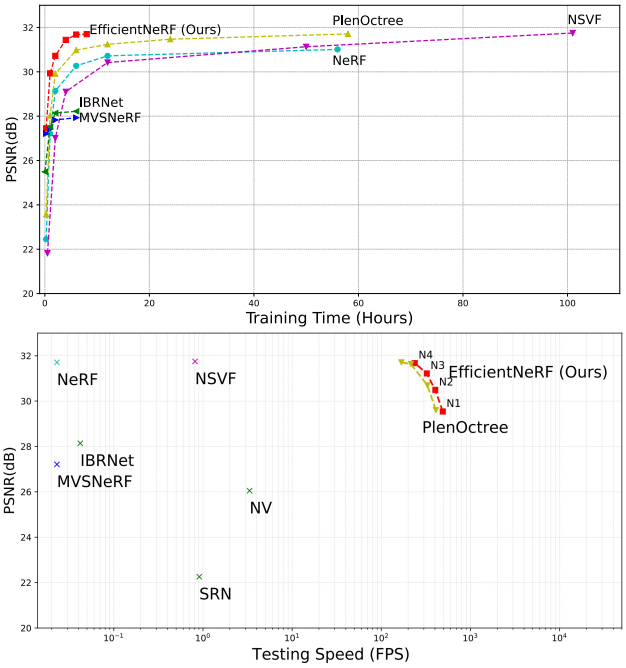

Figure 1. Training and testing efficiency on realistic synthetic dataset [17] on a single GPU. Our EfficientNeRF much improves efficiency in both training and testing phases.

Recently, methods [2, 4, 12, 18, 32, 35] were proposed to accelerate either the training process or the testing phase. On the one hand, during testing, NSVF [12] and DONeRF [18] decrease the number of samples by their generated sparse voxels or predicted depth. FastNeRF [4] and PlenOctree [35] discretely cache the target scene and synthesize novel-view images by fast query. Although these methods successfully reduce the per-image inference time, their training time is equivalent or even longer, as illustrated in Fig. 1.

On the other hand, during training, methods of [2, 32, 36] combine NeRF with image features extracted from ResNet [7] or MVSNet [34] to construct a generalized model, thus achieving fast training. Nevertheless, as the image prior comes from limited neighboring views, the synthesis accu-

racy tends to be lower than NeRF [2, 32]. Besides, obtaining features from multi-view images takes more time during testing. There is no work yet to significantly shorten both training and testing time simultaneously.

In this paper, we present the Efficient Neural Radiance Fields (EfficientNeRF) as the first attempt to accelerate both per-scene training and per-image testing. Apart from obtaining competitive accuracy, the training time can be reduced by more than 88%, and the rendering speed is accelerated to over 200 FPS, as illustrated in Fig. 1.

The pipeline of original NeRF [17] consists of the coarse and fine stages. During training, the coarse stage obtains the density distribution over the whole scene. It uniformly and densely samples points and calculates corresponding densities by a coarse MLP. However, as will be shown in Table 1, for common scenes with uniformly sampling, there are only around 10% - 20% of valid samples (in Eq. (5)) – 5% - 10% are pivotal samples (in Eq. (12)).

Also, since each point's density is shared by all rays, it is possible to memorize the global density by Voxels. Although NSVF [12] also marks this fact, its solution is to gradually delete invalid voxels, which may cause adverse effects when removal is wrong. Differently, we propose Valid Sampling, which maintains dense voxels and updates density in an online way with momentum. The coarse MLP only infers valid samples whose queried densities are greater than zeros, thus saving most of the time at the coarse stage.

For the fine stage, the original NeRF samples more points following previous coarse density distribution. We find that many rays even do not contain any valid and pivotal points because of the empty background. We instead propose Pivotal Sampling for the fine stage that focuses on the nearby area of pivotal samples to efficiently sample points. Our strategy substantially decreases the number of sampled points while achieving comparable accuracy.

During testing, inspired by [35] and [4] that replace MLP modules by caching the whole scene in voxels, we design a novel tree-based data structure, *i.e*. NerfTree, to more efficiently represent 3D scenes. Our NerfTree only has 2 layers. The first layer represents the coarse dense voxels extracted from the coarse MLP, and the second layer stores the fine sparse voxels obtained from the fine MLP. The combination of our dense and sparse voxels leads to fast inference speed.

Our main contributions are the following.

1. We propose EfficientNeRF, the first work to significantly accelerate both training and testing of NeRF-based methods while maintaining reasonable accuracy.

2. We propose Valid Sampling, which constructs dynamic Voxels to accelerate the sampling process at the coarse stage. Also, we propose Pivotal Sampling to accelerate the fine stage. They in total reduce over 88% of computation and training time.

3. We design a simple and yet efficient data structure, called NerfTree, for NeRF-based methods. It quickly caches and queries 3D scenes, thus improving the rendering speed by $4,000+$ times.

2. Related Work

Novel View Synthesis NVS is a long-standing problem in computer vision and computer graphics. Voxel grids [8, 10, 14, 28] can achieve real-time synthesis. But they are challenging to represent high-resolution images with large memory consumption. MPI-based methods [16, 23, 30, 31, 33, 38] can synthesize high-resolution images. They first synthesize multiple depth-wise images and then fuse them to the target views by α-compositing [15]. Large-view synthesis [32] is necessary.

NeRF-based Applications NeRF [17] resolves the resolution and memory issues, and can be easily expanded to various applications. Nerfies [21], NSVF [12], and D-NeRF [25] implicitly learn 3D spatial deformation functions for dynamic scenes where objects are moving in different frames. Neural Actor [13] and Animatable-NeRF [22] also adopt similar functions to synthesize human body with novel poses. GRAF [27], pi-GAN [1], and GIRAFFE [19] treat NeRF as a generator in GAN [5] and generate geometrically controllable images. Recently, StyleNeRF [6] succeeds in generating images at 1K resolution, which encourages development of NeRF generator.

Training Acceleration Training of NeRF [17] and its variants usually takes 1 to 2 days [12, 17, 35], which limits its efficiency-critical applications. Yu *et al*. proposed PixelNeRF [36] that introduces image features from ResNet [7] and skips training in novel scenes. But its synthesis accuracy reduces [2]. Wang *et al*. proposed IBRNet [32] that integrates multi-view features in the weighted sum, thus improving accuracy. Chen *et al*. proposed MVSNeRF [2], which employs MVSNet [34] to provide a feature volume for NeRF. It can synthesize high-quality images within 15-minute finetuning. However, the testing time of the above methods is as long as the original NeRF.

Testing Acceleration To accelerate per-image inference, NSVF [12] gives a hybrid scene representation that combines NeRF with sparse voxels structure. The generated sparse voxels guide and reduce sampling. It improves the inference speed to around 1 FPS. KiloNeRF [26] reduced the inference time by adopting around 1,000 tiny MLPs, where each MLP takes care of a specific 3D area. The running speed is over 10 FPS. PlenOctree [35] and FastNeRF [4] achieved inference speed over 168 FPS and 200

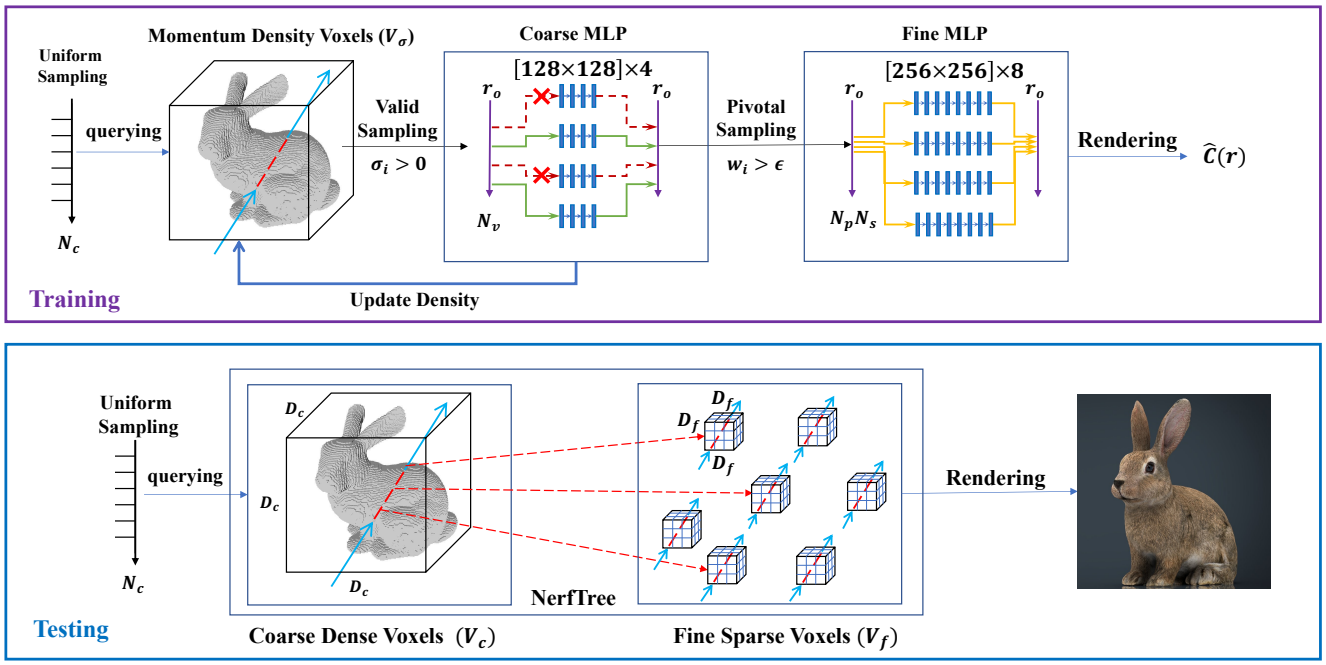

Figure 2. Overview of our proposed EfficientNeRF. **Training**: we first uniformly sample N_c points along each ray $\mathbf{r}$, and query the density from the Momentum Density Voxels V_σ. We calculate its coarse density for the valid samples whose density $\sigma_i > 0$, obtain weight to calculate the final ray color, and update V_σ by the coarse density. The pivotal samples with weights $w_i > \epsilon$ are taken care of. N_s nearby samples are linearly sampled along ray $\mathbf{r}$ at higher resolutions. Finally, we infer the fine density and color parameters by the fine MLP and predict the ray color by volume rendering. **Testing**: The Coarse Dense Voxels and Fine Sparse Voxels are respectively extracted from coarse and fine MLPs. The densities and colors are obtained by voxels query rather than MLPs.

FPS respectively by caching the whole 3D scenes. We note that their training is still heavy. In contrast, our EfficientNeRF achieves faster per-image inference speed along with far less training time.

3. Our Approach

Given M images $I_m(m = 1, 2, ..., M)$ with calibrated cameras parameters in multiple views of a scene, we aim to achieve accurate 3D scene representation and novel image synthesis regarding both fast training and testing. To begin with, we review the basic idea and pipeline of NeRF [17]. Then, we introduce our efficient strategies during training, including lightweight MLP, valid sampling at the coarse stage, and pivotal sampling at the fine stage. Finally, we represent the whole scene by our proposed NerfTree during testing to reach hundreds of FPS.

3.1. Background: Neural Radiance Fields

NeRF [17] is a new representation to 3D scenes. Different from 3D mesh, point clouds, and voxels, it introduces implicit functions to model scenes while adopting volume rendering to synthesize images. Compared with voxels-based representation, NeRF overcomes the limitation of resolution and storage to synthesize high-quality results.

Implicit Function NeRF employs implicit functions to inference the sampled points' 4D attributes when inputting 5D spatial information, formulated as

$$(r, g, b, \sigma) = f(x, y, z, \theta, \phi), \tag{1}$$

where $\mathbf{x} = (x, y, z)$ and $\mathbf{d} = (\theta, \phi)$ denote the point location and direction in the world coordinate. The color and density attributes are respectively represented by $\mathbf{c} = (r, g, b)$ and σ. f is a mapping function, usually implemented by a MLP network.

Volume Rendering For each pixel in the synthesized image, to calculate its color, NeRF first samples N points $\mathbf{x}_i(i = 1, 2, ..., N)$ along ray $\mathbf{r}$. It then calculates corresponding density σ_i and color $\mathbf{c}_i$ by Eq. (1). The final predicted color $\hat{C}(r)$ is rendered by α-compositing [15] as

$$\begin{aligned} \hat{C}(\mathbf{r}) &= \sum_{i=1}^{N} w_i c_i, \\ w_i &= T_i \alpha_i, \\ T_i &= exp(-\sum_{j=1}^{i-1} \alpha_j \delta_j), \\ \alpha_i &= 1 - exp(-\sigma_i \delta_i), \end{aligned} \tag{2}$$

Scene	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	**Mean**
Valid Samples (V, %)	9.58 %	7.00 %	3.85 %	9.35 %	15.43 %	19.47 %	8.44 %	11.32 %	**10.56** %
Pivotal Samples (P, %)	3.79 %	2.25%	1.68 %	3.59 %	5.81 %	7.42 %	3.14 %	4.62 %	**4.04** %
Scene	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex	**Mean**
Valid Samples (V, %)	24.28 %	13.68 %	23.45 %	21.34 %	15.09%	19.74 %	30.62 %	18.27 %	**20.81**%
Pivotal Samples (P, %)	15.63 %	7.49 %	4.48 %	10.45 %	8.49%	9.43 %	15.23 %	7.89 %	**9.89**%

Table 1. Proportions of valid and pivotal samples on the Realistic Synthetic dataset [17] and the Real Forward-Facing dataset [16].

where δ_i denotes interval of samples along ray $\mathbf{r}$.

Training Objective The training objective $\mathcal{L}$ of NeRF is the mean square error between each ground-truth pixel color $C(\mathbf{r})$ and the rendering color $\hat{C}(\mathbf{r})$ as

$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \| C(\mathbf{r}) - \hat{C}(\mathbf{r}) \|_2^2, \tag{3}$$

where $\mathcal{R}$ is the set of all rays shooting from the camera center to image pixels.

3.2. Network

The original NeRF [17] adopts a coarse-to-fine pipeline to represent scenes. There are two Multi-Layer Perceptrons (MLPs) in the model with the same network size, while respectively operate the coarse and fine stages. We call them coarse and fine MLPs. Since the coarse MLP mainly infers a coarse density distribution, original coarse MLP is redundant and reducible.

For the sake of simplicity, we directly decrease both the depth and width of coarse MLP by half and keep the consistency of the fine MLP, as illustrated in Fig. 2. According to the experimental results in Table 5, our lightweight coarse MLP almost does not weaken performance while improving the overall inference speed. Taking advantage of our lightweight coarse MLP, we increase N_c to improve the synthesized quality.

Different from the original NeRF that employs an implicit mapping from direction to color, we adopt the Spherical Harmonics model in PlenOctree [35] to explicitly predict color parameters by the MLP network. It not only improves the accuracy but also is beneficial for offline caching during testing.

3.3. Valid Sampling at the Coarse Stage

Valid Samples We define the point with location $\mathbf{x}_v$ with density $\sigma_v > 0$ as a valid sample, as shown in Fig. 3. For the N points along ray $\mathbf{r}$, suppose a point with location $\mathbf{x}_i$ has density $\sigma_i = 0$. Because $T_i = exp(-\sum_{j=1}^{i-1} \alpha_j \delta_j)$ belongs to the range of $[0, 1]$ and $\alpha_i = 1 - exp(-\sigma_i \delta_i) = 0$, we calculate its contribution w_i to the ray color $\hat{C}(r)$ by

$$w_i = T_i \cdot \alpha_i = 0. \tag{4}$$

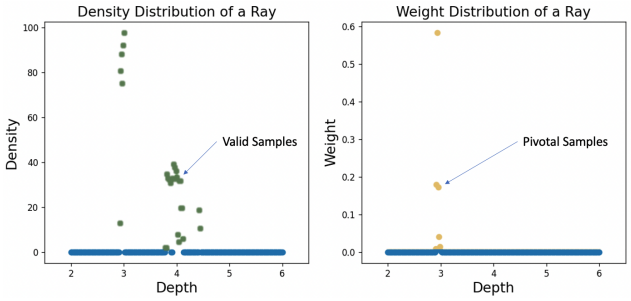

Figure 3. Density and weight distributions of a typical ray for NeRF-based methods. Green and yellow points indicate valid and pivotal samples, respectively.

It means the point is an invalid sample and makes no difference to the final rendering result. Therefore, it can be skipped once we know the locations. The proportion of valid samples is represented as

$$V = \frac{N_v}{N_c}. \tag{5}$$

We measure the percentage of area by trained NeRF that is valid over common scenes and show the numbers in Table 1. It is surprising to note that only a small portion (around 10% - 20%) of the samples are valid. From this analysis, we conclude that it is feasible and necessary to adopt sparse and valid sampling to achieve efficient scene representation.

Momentum Density Voxels For a specific scene, the density of any world coordinate $\mathbf{x} \in \mathbb{R}^3$ is shared by all rays. Thus, we construct momentum density voxels V_σ with resolution $D \times D \times D$ to memorize the latest global value of density over the target scene during training.

Initialization Since each point's density $\sigma \geq 0$, we initialize the default density value in V_σ as a positive number ε. It means that all points in V_σ are valid samples in the beginning.

Update For a sampled point with location $\mathbf{x} \in \mathbb{R}^3$, we infer its coarse density $\sigma_c(\mathbf{x})$ by the coarse MLP. Then we update the density voxels V_σ by $\sigma_c(\mathbf{x})$. We add a momentum to stabilize the values. Specifically, we first transfer $\mathbf{x}$

to 3D Voxels index $\mathbf{i} \in \mathbb{R}^3$ as

$$\mathbf{i} = \frac{\mathbf{x} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}} \cdot D, \tag{6}$$

where $\mathbf{x}_{min}, \mathbf{x}_{max} \in \mathbb{R}^3$ denote the minimal and maximal world coordinate borders of the scene.

Next, for every training iteration, we update the global density σ at index $\mathbf{i}$ of V_σ through

$$V_\sigma[\mathbf{i}] \leftarrow (1-\beta) \cdot V_\sigma[\mathbf{i}] + \beta \cdot \sigma_c(\mathbf{x}). \tag{7}$$

Where $\beta \in [0, 1]$ controls the updating rate.

Our momentum density Voxels V_σ reflect the latest density distribution over the whole scene. Thus, we directly obtain the density attribute at coordinate x through query rather than calculating through a MLP module. It primarily reduces the inference time and is utilized to guide a dynamic sampling process.

Valid Sampling During training, for each ray $\mathbf{r}$ whose starting point is $r_o \in \mathbb{R}^3$ and normalized direction is $r_d \in \mathbb{R}^3$, the original NeRF adopts a uniform sampling strategy to obtain the sampled points as

$$\mathbf{x}_i = r_o + i\delta_c r_d, \tag{8}$$

where $i \in Z$ and $\forall i \in [1, N_c]$. δ_c is the interval between the nearest coarse sampled points along ray $\mathbf{r}$.

We propose Valid Sampling to pay attention to valid samples. Specifically, instead of directly inferring all these samples, we first query the latest density from V_σ, and only input $\mathbf{x}_i$ with global densities

$$V_\sigma[\mathbf{i}] > 0 \tag{9}$$

to the coarse MLP.

Inferring a single point by a coarse MLP takes times T_m, and querying a single point from voxels takes T_q. For all sampled points along ray $\mathbf{r}$, predicting their densities through a coarse MLP consumes time $N_c T_m$. Our method takes time $(N_v T_m + (N_c - N_v)T_q)$. Considering time of voxels query $T_q \ll T_m$ [4,35], we calculate the acceleration ratio A_c of the coarse stage by

$$A_c = \frac{N_c T_m}{N_v T_m + (N_c - N_v)T_q} \approx \frac{N_c}{N_v} = \frac{1}{V}. \tag{10}$$

As illustrated in Table 1, if the proportion of valid samples $V = 10\%$, the coarse stage can be accelerated by 10 times in theory.

3.4. Pivotal Sampling at the Fine Stage

During the fine stage, 3D points should be sampled in higher resolution for better quality. The original NeRF [17] first samples N_f points along each ray $\mathbf{r}$ that follows the coarse weight distribution. It then predicts densities and colors by the fine MLP. Since the number of points at the fine stages is usually 2 times of N_c, it requires more computation during running time. To achieve efficient sampling at the fine stage, we propose a Pivotal Sampling strategy.

Pivotal Samples We define the point with location $\mathbf{x}_p$ whose weight $w_p > \epsilon$ as a pivotal sample, where ϵ is a tiny threashold, as illustrated in Fig. 3.

Pivotal Sampling w_i represents the contribution of $\mathbf{x}_i$ to the ray $\mathbf{r}$'s color. The nearby area of the pivotal samples is focused to infer more detailed densities and colors. We uniformly sample N_s points near $\mathbf{x}_p$ along each ray $\mathbf{r}$ as

$$\mathbf{x}_{p,j} = \mathbf{x}_p + j\delta_f r_d, \tag{11}$$

where $j \in Z$ and $\forall j \in (-\frac{N_s}{2}, \frac{N_s}{2}]$. δ_f is the interval at the fine stage. Suppose there are N_p pivotal points, the proportion of the pivotal samples can be represented by

$$P = \frac{N_p}{N_c}. \tag{12}$$

Similar to the coarse stage, we calculate the acceleration ratio A_f of the fine stage as

$$A_f = \frac{N_f}{N_p N_s} = \frac{2N_c}{N_p N_s} = \frac{2}{P N_s}. \tag{13}$$

In our experiments with results listed in Table 1, if $N_s = 5$ and $P = 5\%$, our pivotal sampling strategy can accelerate the fine stage by 8 times.

3.5. Represent Scene by NerfTree

Although the training time has been significantly shortened through our valid and pivotal sampling, the system is still constrained by the inference time of MLP during testing. Inspired by [4,35] that cache the target scene in Voxels or Octrees, we design an efficient tree-based data structure, called NerfTree, for NeRF-based methods to accelerate the inference speed. NerfTree can store the whole scene offline, thus eliminating the coarse and fine MLP.

Different from the dense Voxels and Octrees in PlenOctree [35], our NerfTree $T = \{V_c, V_f\}$ is a 2-depth tree. The first depth caches the coarse dense Voxels V_c, and the second depth caches the fine sparse Voxels V_f, as illustrated in Fig. 2. $V_c \in \mathbb{R}^{D_c \times D_c \times D_c}$ only contains density attribute, which is extracted by inferring the density values of every voxel grid by the coarse MLP.

For the fine sparse voxels $V_f \in \mathbb{R}^{N_V \times D_f^3}$, the first dimension N_V represents the number of all valid samples, and the second dimension represents local voxels with size $D_f \times D_f \times D_f$. Each voxel in V_f stores the density and color parameters inferred from the fine MLP. As illustrated

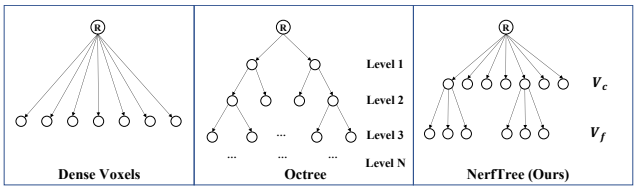

Figure 4. 2D graph representation of different 3D data structures. Left: Dense Voxels. Middle: Octrees. Right: NerfTree (Ours).

in Fig. 4, between these three representations, dense voxels only have one depth layer, thus achieving the minimal access time and maximum storage. Octree has the opposite characteristic of Voxels. Our NerfTree combines the advantages of both Voxels and Octrees. Thus it can be accessed at a fast speed while not consuming much storage.

4. Experiments

4.1. Experimental Setting

Datasets We first introduce the common high-resolution Novel View Synthesis datasets, including the Realistic Synthetic dataset [17] and the Real Forward-Facing dataset [16]. The Realistic Synthetic dataset [17] contains 8 synthetic scenes. Each scene contains 100 training images and 200 testing images, all at 800×800 resolution. The Real Forward-Facing dataset [16] consists of 8 complex and real-world scenes, each has 20 to 62 images at $1,008 \times 756$ resolution. We follow the same training and testing dataset split as the original NeRF [17].

Metrics We evaluate the accuracy of synthesized images via metrics including PSNR / SSIM (the higher the better), and LPIPS [37] (the lower the better), following recent methods [12, 14, 17, 35]. Moreover, we measure the training speed by total training time in terms of hour and the rendering speed by Frame Per Second (FPS). For fair comparison, we re-train their source code on the same machine and skip the evaluation time during training.

Implementation Details During training, for the Momentum Density Voxels V_σ, its resolution is set as $384 \times 384 \times 384$, the initial density $\varepsilon = 10.0$, and the updating rate $\beta = 0.1$. The degree of Spherical Harmonics is 3, which means the output dimension of MLPs is 49 [35].The pivotal threshold ϵ is 1×10^{-4}. The learning rate is initialized to 5×10^{-4} with Adam [9] optimizer and exponentially decays by 0.1 for every 500K iteration when the batch size is 1024. Finally, the total number of training iteration is 1×10^6.

During testing, we set $D_c = 384$ and $D_f = 3$, which means the coarse voxels have resolution $384 \times 384 \times 384$, and local fine sparse voxels have resolution $3 \times 3 \times 3$. The quantification from MLP's continuous coordinates to discrete ones usually weakens the performance [35]. We avoid it by directly inputting the converted discrete coordinate to the MLP during training or using linear interpolation. All our experiments are performed on a server with one RTX-3090 GPU. Please refer to the supplementary material for more experimental results.

4.2. Quantitative Comparison

We compare the proposed EfficientNeRF with state-of-the-art methods [11, 12, 14, 17, 29, 35] in terms of both accuracy and speed. Results are listed in Table 2. Depending on whether image prior is introduced or not, state-of-the-art methods can be divided into two groups. The first [2, 14, 29, 32] is based on image prior. These methods can fine-tune a novel scene in short time while sacrificing testing accuracy. The second group [11, 12, 35] is by training from scratch. They were designed to improve the rendering speed. However, the training time is found even longer than that of the original NeRF [17].

In contrast, our EfficientNeRF achieves competitive accuracy on both datasets and demonstrates notable advantage in terms of training and testing efficiency. As shown in Fig. 1 and Table 2, even though our method does not introduce image prior, it still outperforms previous fast finetuning method *i.e.*, MVSNeRF [2], when training for 15 minutes or longer. In addition, our proposed NerfTree quickly queries the 3D attributes at the target locations, which contributes to our final 238.46 FPS during testing.

In summary, our EfficientNeRF adopts efficient strategies, including lightweight MLP, valid sampling, and pivotal sampling, thus accelerating both the training and testing while maintaining comparable accuracy.

Trade-off between Accuracy and Speed To balance the synthesized accuracy and inference speed, we provide four versions of EfficientNeRF ($N1$-$N4$) according to the number of coarse and fine sampling parameters N_c and N_s in Table 3 and plot the PSNR-Speed curves in Fig. 1. Our work achieves a better rendering speed than other state-of-the-art methods like PlenOctree [35].

4.3. Qualitative Comparison

We also demonstrate the performance of our method by visual comparison. As illustrated in Fig. 5, we intuitively show the training visualization of different methods at 0.25, 2, and 5 hours, and the final training time. In the timeline of training, ours already synthesizes detailed images within 1 hour, while other methods [12, 32, 35] need to train 5 hours or longer to achieve similar performance. Also, compared with IBRNet [32] based on image prior, our method is trained from scratch while outperforming it within 0.25-hour training.

Method	Realistic Synthetic [17]					Real Forward Facing [16, 17]				
	PSNR(↑)	SSIM (↑)	LPIPS (↓)	Training Time (Hours, ↓)	Rendering Speed (FPS, ↑)	PSNR (↑)	SSIM (↓)	LPIPS (↓)	Training Time (Hours, ↓)	Rendering Speed (FPS, ↑)
SRN [29]	22.26	0.846	0.170	-	0.909	22.84	0.668	0.378	-	-
NV [14]	26.05	0.893	0.160	-	3.330	-	-	-	-	3.052
MVSNeRF [2]	27.21	0.945	0.227	**0.25**	0.020	26.25	**0.907**	0.139	0.25	0.016
IBRNet [32]	28.14	0.942	0.072	**2.0**	0.042	26.73	0.851	0.175	**2.0**	0.036
NeRF [17]	31.01	0.947	0.081	56	0.023	26.50	0.811	0.250	20	0.018
NSVF [12]	**31.75**	0.953	0.047	100+	0.815	-	-	-	-	0.758
AutoInt [11]	25.55	0.911	0.170	-	0.380	24.13	0.820	0.176	-	-
KiloNeRF [11]	31.00	0.950	**0.030**	25+	10.64	-	-	-	-	-
FastNeRF [4]	-	-	-	-	∼200	26.04	0.856	**0.085**	-	∼200
Nex [33]	-	-	-	-	-	**27.26**	0.904	0.178	18+	**300**
PlenOctree [35]	**31.71**	**0.958**	0.053	58	**167.68**	-	-	-	-	-
EfficientNeRF	31.68	**0.954**	**0.028**	6	**238.46**	**27.39**	**0.912**	**0.082**	**4**	**218.83**

Table 2. Accuracy and time comparison on the Realistic Synthetic [17] and the Real Forward-Facing [16, 17] datasets. Ours achieves comparable PSNR/SSIM/LPIPS accuracy with state-of-the-art methods, while showing promising acceleration in both training and testing phases.

	Version	# Sampling		PSNR (↑)	Rendering Speed (FPS, ↑)
		N_c	N_s		
EfficientNeRF	$N1$	64	1	29.54	493.62
	$N2$	64	2	30.49	403.28
	$N3$	96	3	31.22	324.62
	$N4$	128	4	31.68	238.46

Table 3. Different versions of our EfficientNeRF with trade-off between synthesized accuracy and rendering speed.

Coarse MLP		Fine MLP		Time (s / iter, ↓)	PSNR(↑)
Lightweight	Standard	Lightweight	Standard		
	✓		✓	0.184	**31.01**
✓		✓		0.121	29.28
	✓	✓		0.132	29.39
✓			✓	**0.138**	**30.96**

Table 4. Performance of different combinations between lightweight and standard MLPs at the coarse and fine stages of the original NeRF [17]. The batch size is 1024.

4.4. Ablation Studies

Networks We explore the influence of the different sizes of the coarse and fine networks, as shown in Table 4. The baseline combination is two identical standard MLPs, which come from the original NeRF [17]. The accuracy is the best under the longest running time. The combination of two lightweight MLPs yields opposite performance, which indicates the effect of standard MLPs.

It is found that lightweight coarse MLP plus standard fine MLP yields nearly the same accuracy and fast rendering speed. It reveals that the size of fine MLP mainly determines the final synthesis quality. We thus adopt the final combination as the network of our EfficientNeRF.

Efficient Modules As shown in Table 5, we evaluate performance of the proposed efficient modules. First, representing the color in different directions by Spherical Harmonic (SH) [35] is in favor of the performance. Second, our lightweight coarse MLP and coarse valid sampling accelerate the training process. Third, our fine pivotal sampling further reduces the training time and improves synthesis accuracy. It performs better than the original probabilistic sampling of original NeRF [17].

Method	PSNR (↑)	Time (↓)	Improvement (↑)
NeRF [17]	31.01	0.184 s / iter	-
+ SH [35]	31.57	0.183 s / iter	-
+ Lightweight Coarse MLP	31.52	0.137 s / iter	25.54%
+ Coarse Valid Sampling	31.49	0.085 s / iter	53.80%
+ Fine Pivotal Sampling	**31.68**	**0.021 s / iter**	**88.58%**

Table 5. Contributions of our proposed modules to the training time on the Realistic Sythetic dataset [17].

Method	Sampling Strategy			Training Time (Hours)	PSNR
	Uniform	NeRF [17]	Ours		
NeRF [17]	✓			41	30.06
		✓		56	31.01
			✓	**6**	**31.25**
IBRNet [32]	✓			2	25.49
			✓	**1**	**26.23**
MVSNeRF [2]	✓			0.25	27.21
			✓	**0.18**	**28.03**
PlenOctree [35]		✓		58	**31.71**
			✓	**6**	31.66

Table 6. Effect of our efficient sampling strategies when combined with different NeRF-based Methods.

Sampling Strategy We compare our efficient sampling with the common uniform sampling and original sampling of NeRF. The result is presented in Table 6. First, uniform sampling is adopted by IBRNet [32] and MVSNeRF [2]. It is faster than NeRF sampling while achieving lower accuracy. Second, NeRF [17] and PlenOctree [35] adopt NeRF sampling and achieve high performance. However, the training time is very long. Finally, our efficient sampling successfully accelerates all these methods while achieving comparable or even better accuracy.

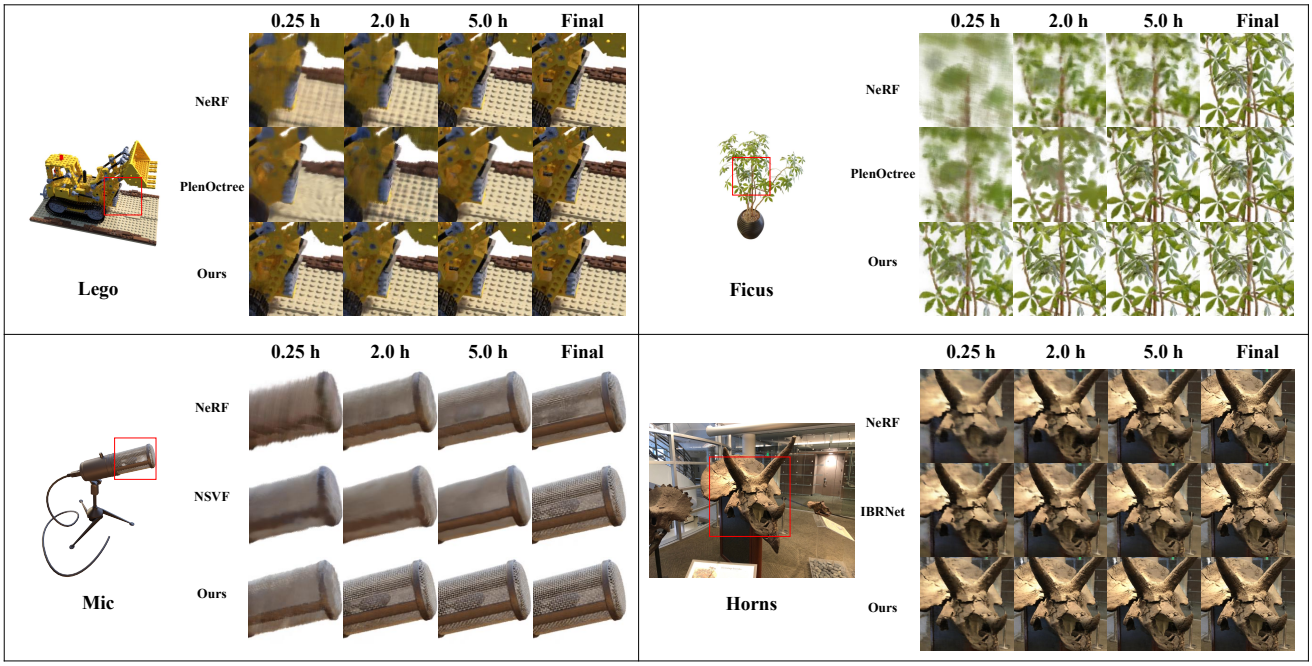

Figure 5. Qualitative comparison with state-of-the-art methods on the Realistic Synthetic dataset [17] and the Real Forward-Facing dataset [16, 17]. It is best viewed by zoom-in.

Scene Representation	Memory	Caching Time	Querying Time
Dense Voxels	16 GB	16.55 ms	13.64 ms
Sparse Tensor (Minkowski Engine [3])	2.1 GB	24.72 s	121.21 ms
Octree (PlenOctree [35])	2.6 GB	14.51 s	18.84 ms
NerfTree (Ours)	2.8 GB	22.43 ms	15.39 ms

Table 7. Comparison of different data structures when caching and querying all points in a common 3D scene with size $1,024 \times 1,024 \times 1,024$ and 20% valid samples.

Scene Representation Before applying NerfTree, other data structures, such as dense voxels, sparse tensor [3], and Octree [35], can also be used to cache the trained scene for fast synthesis of novel views. To compare their efficiency, we calculate memory consumption and running time when storing and querying a whole scene with resolution $1,024 \times 1,024 \times 1,024$ and 20% of valid space. Each voxel has a 4D feature (r, g, b, σ) with the same data type.

The results are shown in Table 7, Dense voxels representation spends the least time in caching and querying while requiring 16.0 GB memory. The required memory by Sparse Tensor (Minkowski Engine [3]) is the smallest. But as it adopts hash structure making caching and query longer process. PlenOctree [35] balances memory consumption and query time. However, its caching time is long because of its internal optimization. Our NerfTree representation performs the best with all these three aspects. It does not require much storage memory, and cache and query 3D points at a very fast speed.

5. Conclusion

In this paper, we have presented Efficient Neural Radiance Fields (EfficientNeRF) to accomplish accurate representation of 3D scenes and synthesis of novel view images at a fast speed. We studied the distribution of density and weight and proposed valid sampling at the coarse stage and pivotal sampling at the fine stage. These two sampling strategies are efficiently handle the important samples, thus saving a great amount of computation. Also, we designed NerfTree for NeRF-based methods to cache 3D scenes. It yields faster speed than state-of-the-art methods [12, 26, 35] during testing.

Limitations and Future Work Our EfficientNeRF achieves fast and accurate 3D scene representation and view synthesis. It still needs to train from scratch when handling novel scenes. This is also a common issue in other state-of-the-art NeRF-based methods [35]. Although we combined images prior with our efficient sampling in Table 6, the synthesis accuracy is limited. In future work, we will improve generalization of EfficientNeRF and aim to achieve competitive accuracy when there is no finetuning in novel scenes.

References

[1] Eric Chan, Marco Monteiro, Peter Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis. *https://arxiv.org/abs/2012.00926*, 2020. 2

[2] 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. *arXiv preprint arXiv:2103.15595*, 2021. 1, 2, 6, 7

[3] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 8

[4] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien P. C. Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *CoRR*, abs/2103.10380, 2021. 1, 2, 5, 7

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2

[6] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis, 2021. 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*. IEEE Computer Society, 2016. 1, 2

[8] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NeurIPS*, pages 365–376, 2017. 2

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015. 6

[10] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *Int. J. Comput. Vis.*, 2000. 2

[11] David B. Lindell, Julien N. P. Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *CVPR*. Computer Vision Foundation / IEEE, 2021. 6, 7

[12] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 1, 2, 6, 7, 8

[13] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM SIGGRAPH Asia*, 2021. 2

[14] Stephen Lombardi, Tomas Simon, Jason M. Saragih, Gabriel Schwartz, Andreas M. Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. *ACM TOG*, 38(4):65:1–65:14, 2019. 1, 2, 6, 7

[15] Nelson L. Max. Optical models for direct volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 1(2):99–108, 1995. 2, 3

[16] 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 Trans. Graph.*, 2019. 1, 2, 4, 6, 7, 8

[17] 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 Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, volume 12346, pages 405–421, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[18] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021. 1

[19] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. *https://arxiv.org/abs/2011.12100*, 2020. 2

[20] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan Goldman, Steven Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *https://arxiv.org/abs/2011.12948*, 2020. 1

[21] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 1, 2

[22] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *ICCV*, 2021. 1, 2

[23] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Trans. Graph.*, 2017. 2

[24] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. *https://arxiv.org/abs/2011.13961*, 2020. 1

[25] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. DeRF: Decomposed radiance fields. *https://arxiv.org/abs/2011.12490*, 2020. 1, 2

[26] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *CoRR*, 2021. 2, 8

[27] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3D-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 2

[28] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. *Int. J. Comput. Vis.*, pages 151–173, 1999. 2

[29] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. pages 1119–1130, 2019. 1, 6, 7

[30] 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 *CVPR*, 2019. 2

[31] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. *Int. J. Comput. Vis.*, 1999. 2

[32] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. *arXiv preprint arXiv:2102.13090*, 2021. 1, 2, 6, 7

[33] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, June 2021. 2, 7

[34] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *ECCV*, volume 11212 of *Lecture Notes in Computer Science*, pages 785–801, 2018. 1, 2

[35] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *arXiv*, 2021. 1, 2, 4, 5, 6, 7, 8

[36] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. *https://arxiv.org/abs/2012.02190*, 2020. 1, 2

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

[38] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Trans. Graph.*, 2018. 2