Neural Radiance Fields from Sparse RGB-D Images for High-Quality View Synthesis

Yu-Jie Yuan, Yu-Kun Lai, Yi-Hua Huang, Leif Kobbelt, and Lin Gao*

Abstract—The recently proposed neural radiance fields (NeRF) use a continuous function formulated as a multi-layer perceptron (MLP) to model the appearance and geometry of a 3D scene. This enables realistic synthesis of novel views, even for scenes with view dependent appearance. Many follow-up works have since extended NeRFs in different ways. However, a fundamental restriction of the method remains that it requires a large number of images captured from densely placed viewpoints for high-quality synthesis and the quality of the results quickly degrades when the number of captured views is insufficient. To address this problem, we propose a novel NeRF-based framework capable of high-quality view synthesis using only a sparse set of RGB-D images, which can be easily captured using cameras and LiDAR sensors on current consumer devices. First, a geometric proxy of the scene is reconstructed from the captured RGB-D images. Renderings of the reconstructed scene along with precise camera parameters can then be used to pre-train a network. Finally, the network is fine-tuned with a small number of real captured images. We further introduce a patch discriminator to supervise the network under novel views during fine-tuning, as well as a 3D color prior to improve synthesis quality. We demonstrate that our method can generate arbitrary novel views of a 3D scene from as few as 6 RGB-D images. Extensive experiments show the improvements of our method compared with the existing NeRF-based methods, including approaches that also aim to reduce the number of input images.

Index Terms—Novel View Synthesis, Neural Rendering, Neural Radiance Fields.

1 Introduction

Novel view synthesis (NVS) is a major research topic in computer vision and computer graphics. It has been widely applied in the digital and entertainment industry, from movie production to games, as well as the booming virtual and augmented reality (VR/AR) applications. However, generating highly realistic images under arbitrary views based on only a small number of input images is still an urgent problem to be solved. If we have the accurate geometry of an object or scene, the problem can be solved through rendering, which also requires additional information such as the surface material and lighting environment. The rendering process solves the integral of the rendering equation to obtain highly-realistic results. However, there are still full of challenges for real-world objects or scenes. On the one hand, it is difficult to obtain accurate geometry, surface normal and material information of objects or scenes in the real world. On the other hand, the lighting in the real environment is very complicated, and multi light source environment is difficult to model. Therefore, to avoid geometry reconstruction, traditional image-based rendering (IBR) methods [1], [2] directly interpolate between existing images to obtain images in novel views. However, the novel view results can only be within the range of existing views, and images from densely captured views are required. Moreover, these methods cannot handle occlusions well.

Recently, neural rendering [3] has shown a promising future. Neural-based methods have also emerged for the NVS task. A large group of these methods is based on the volume-based representation to model the appearance of the entire space [4]. Among these volume-based methods, NeRF (Neural Radiance Fields) [5] becomes a very popular method, due to the simple formulation and appealing performance. NeRF only needs to input a set of RGB images (the camera parameters are known or estimated by methods such as COLMAP [6], [7]) to generate highly-realistic images under novel views. The key idea of NeRF is to represent the entire space by a continuous function, which is approximated by a multi-layer perceptron (MLP). The network inputs the spatial coordinates and view direction of the sampled point, a 5-dimensional vector in total, then predicts the radiance emitted by the point and density of the location, and uses the classic volume rendering [8] to render the image at last.

Although the results of NeRF are fascinating, it still requires a fairly large number of images as the training samples, which puts a very high requirement on practical use. Some variants have been proposed to address this problem, but they still suffer from reduced performance. NeRF and its variants are still far from actual use by average users. These works only use the RGB images as inputs to generate novel view images of the scenes or objects. In this paper, we adopt a consumer product equipped with both an RGB camera and a LiDAR camera to capture RGB-D images in a limited number of views. The key idea of our

- * *Corresponding Author is Lin Gao (gaolin@ict.ac.cn).*
- *Yu-Jie Yuan, Yi-Hua Huang and Lin Gao are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences. Yu-Jie Yuan, Yi-Hua Huang and Lin Gao are also with University of Chinese Academy of Sciences, Beijing, China.*
  E-mail: {yuanyujie, huangyihua20g, gaolin}@ict.ac.cn
- *Yu-Kun Lai is with School of Computer Science & Informatics, Cardiff University, UK.*
  E-mail: LaiY4@cardiff.ac.uk
- *Leif Kobbelt is with Institute for Computer Graphics and Multimedia, RWTH Aachen University, Germany.*
  E-mail: kobbelt@cs.rwth-aachen.de

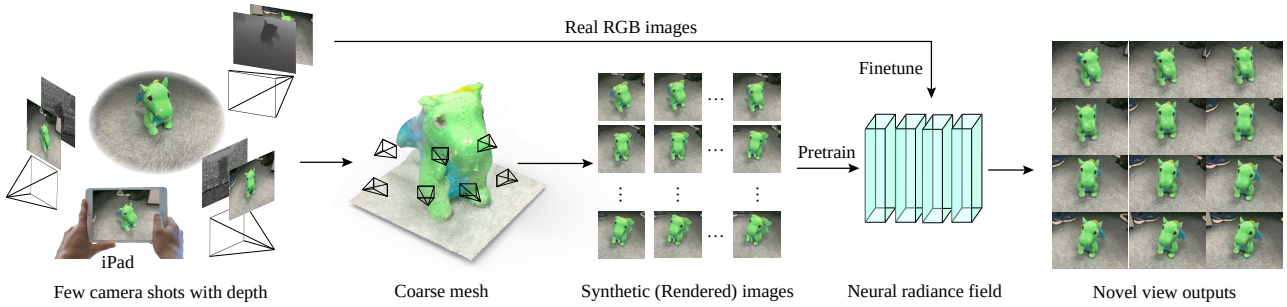


Figure 1. We propose a novel NeRF (Neural Radiance Field) framework that uses sparse RGB-D images to synthesize novel view images. These RGB-D images are acquired using consumer devices such as an iPad Pro. We first use the rendered images of the reconstructed mesh to pre-train the NeRF network, and then use the real captured images to fine-tune the network to synthesize realistic images from novel views.

method is to significantly reduce the camera view number by exploiting reliable depth information. The captured RGB-D images can provide a rough point cloud geometry for the scene. Although this point cloud is relatively coarse and cannot fully reflect the accurate geometry of the scene, and some other information such as the normals and the lighting cannot be directly obtained, we can still get lots of rendered images which depicts the approximate appearance of the scene, through rendering the mesh reconstructed from the point cloud. A large number of such images can be rendered with precise camera parameters, and they can be in arbitrary views. These characteristics are very important and helpful for the NeRF training. Our idea is to pre-train a neural radiance field network using these rendered images as the (pseudo) ground truth. Because the network is trained with precise camera parameters and a sufficient number of images, the novel view images generated by the trained network can fully retain the quality of those rendered images. Based on this pre-training, we fine-tune the network using the small number of captured real RGB images. With the basis provided by the rendered images, this strategy can obtain better results than training from scratch with the small amount of the real captured RGB images. Furthermore, we introduce a fine-tuning framework suitable for few-shot training, making full use of real captured RGB images. We also use depth information to provide more accurate RGB prior information from the point cloud of the scene for the network to better predict RGB values. We also initialize a sparse voxel octree to organize the scene given the reconstructed rough geometry. Although some recent works [9] adopt depth loss, we only sample within the voxel, which means that we only sample near the known depth. This also gives density prediction guidance while allowing the initial depth to have some deviation. This is more flexible than directly using depth loss. The contributions of our work are summarized as follows:

- We propose a NeRF system that only requires users to capture a small number of RGB-D images (as few as 6) with a consumer-grade product to render realistic images in novel views.
- We propose a training strategy for the NeRF network, from the pre-training by the rendered images with precise camera parameters and sufficient views to the fine-tuning by a small set of real captured RGB images.
- We further propose a fine-tuning method, with a patch discriminator enhancing supervision and colored point cloud providing voxel color priors.

Experimental results show that our method can produce high-quality NeRF-based novel view synthesis with sparse RGB-D input images captured using a consumer product (an iPad Pro), outperforming existing methods which only utilize RGB images.

2 Related Work

Image-based rendering. Early novel view synthesis work focused on image-based rendering (IBR) [1], [2], which interpolates between input images of different views to obtain novel view images. Some NVS works rely on explicit 3D scene representations [10], [11], [12], [13], and also use RGB-D sensors to enable fast rendering [14]. Choi et al. [15] obtain the depth probability volume (DPV) of the novel view by blending the DPVs of the input views, and further synthesize the novel view image from the obtained DPV. The synthesized image is refined by a patch-based refinement network with a U-Net architecture. Following a similar idea, Riegler and Koltun [16] warp the features of the input images to the target view with the help of the reconstructed rough geometry. And a recurrent neural network is used to synthesize novel view images from the warped features of nearby views. They further combine the reconstructed mesh with the input images to complete NVS [17]. However, the method still follows the idea of image translation. Based on the point cloud obtained from the RGB-D data, Aliev et al. [18] define learnable descriptors on each point. The point cloud with the descriptors is rasterized into images at different resolutions, which are translated to the realistic image by a U-net structure. The reconstruction-based approaches often exhibit artifacts due to the imprecise geometry. One way to improve this is to fill the missing content of 3D geometry with the learnable 2D texture [19], [20]. Some deep-learning-based methods [21], [22], [23] jointly estimate proxy shape and radiance field from 2D input views. On the other hand, the implicit geometry coming from dense input images could also help synthesize novel view images, such as some light field methods [24], [25]. Multiplane image (MPI) [26], [27] is a kind of light field representation, which can model some complex scenes. But it can only work within a small viewing angle [28]. Mildenhall et al. [29] extend

the synthesis views by requiring the input images to be at some specified views. Recently, Li et al. [30] propose a new DeepMPI representation, which appends learnable latent vectors to the voxels of MPI, modeling the same attributes among different view images, such as the scene geometry. By adopting an Adaptive Instance Normalization (AdaIN) layer [31] to input the appearance vector to the rendering network, novel view synthesis with different lighting conditions can be realized.

Neural Radiance Fields. Undoubtedly, the Neural Radiance Fields (NeRF) method [5] has led to a series of research work using MLP and volume rendering [8] to solve the novel view synthesis task. Prior to this, similar ideas that use MLP to model the appearance of the object or scene have been explored in the volume-based representation [4] and surface-based representation [32] of space. With the simple formulation and stunning effects of NeRF, people have high expectations on the combination of MLP and volume rendering. Taking a step forward, NSVF [33] proposes to use a sparse voxel octree to represent the scene. During the training process, they gradually prune the octree and subdivide the voxels, making them closer to the real geometry of the scene. At inference time, the sampled points only exist in voxels, so the rendering time is significantly reduced compared to NeRF, and the rendering quality has also improved. Our method also adopts a sparse voxel octree to organize the scene. Thanks to the captured depth, we can initialize voxels to be close to the real scene geometry at the beginning of training, which can help our training converge, even with a few captured images. There are also some works that introduce depth into NeRF. For example, imposing constraints between close frames by depth in the continuous dynamic scenes [34], [35], and accelerating inference time by only sampling near the predicted depth [36]. In particular, NSFF [34] has a sort of pre-training process that relies on the geometric prior and depth prior. Since these priors come from inaccurate predictions, the weight of the corresponding loss will be gradually reduced to zero through a linear mapping during the training. Although the pre-training process is similar to ours, the aimed task, the specific methods used, and the implementation of pre-training are different. In addition to the above extensions, NeRF has been extended for dynamic scenes [37], [38], better rendering effects [39], [40], generalization on multiple scenes [41], [42], [43], [44], [45], faster training or inference speed [46], [47], [48], [49], [50], [51], [52], re-lighting rendering [53], [54], [55], geometry or appearance editing [56], [57], [58], [59], [60] and specifically for processing human bodies [61], [62], [63] and faces [64], [65]. Some works are briefly summarized in [66].

Although NeRF only needs a relatively small set of images, the actual number of required images is still large for practical use. Our method aims to reduce the number of input images. Some implicit-field-based methods [67], [68] are able to perform single-image view extrapolation. PixelNeRF [69] realizes novel view synthesis from only one or few input images. It extracts convolutional features from the input images and inputs to the network. Although it reduces the demand of input images, the generated results of novel views show obvious artifacts, which greatly affects the user experience in actual use. DietNeRF [70] also aims to reduce the number of input images. It introduces a semantic consistency loss which preserves the scene attributes to be unchanged under novel views. This loss is built on the semantics extracted by a pre-trained CLIP Vision Transformer [71]. More recently, ToRF [72] also enhances NeRF with additional sensors to capture more information, but it adopts raw time-of-flight images instead of depth maps. DS-NeRF [9] utilizes the depth map reconstructed by COLMAP and adds depth loss to achieve few-shot NeRF training. However, it mainly considers the enhancement of the density aspect. RegNeRF [73] proposes to add constraints on both density and color to reduce the number of input images required, where the color constraint is patch-based supervision. Our method also introduces depth and patch-based supervision, but we propose a pre-training process using mesh rendering as pseudo ground truth and introduce the 3D voxel color prior, which provides more prior knowledge.

Few shot learning. Few shot learning has been widely applied in video colorization [74], image classification [75], [76], [77] and image generation tasks. The image generation task is more relevant to our work. The work of this task usually adopts adaptation methods. They typically first use a large number of samples of a source domain to train a basic model, and then adapt it to the target data domain with only a small number of samples. This could be done either by adding additional parameters [62], [78] or using some regularization to directly fine-tune on the source model [79]. There are also methods that use data augmentation to avoid over-fitting [80], [81], but the effect is not good for a very small number of images. Our method adopts a patch discriminator to enhance supervision, and at the same time uses the 3D color priors brought in by RGB-D data to achieve few shot learning on realistic images. Pang et al. [82] also rely on the RGB-D data and patch supervision to realize few-shot free-view rendering. However, it focuses on neural human rendering and is based on an image translation network, while our method focuses on the static scene modeling and is based on NeRF, leading to improved consistency across views. Recently, there is also the idea of using patch discriminator in few shot image generation [83] and the combination with NeRF [41], [84], but we also use 3D color priors to complete few shot learning from both prior and supervision. The similar idea of starting from a small amount of labeled data, gradually generating a larger dataset to complete network training is also applied to rendering in computer graphics, such as the inverse rendering problem [85].

3 PRELIMINARY

Our method is based on the recent work, Neural Radiance Fields (NeRF) [5] and Neural Sparse Voxel Fields (NSVF) [33], we will first briefly introduce these two methods.

Given a set of input images, NeRF represents the scene geometry and appearance using a simple fully-connected network. The multi-layer perceptron (MLP) network takes 3-dimentional spatial coordinates $\mathbf{p} = (x, y, z)$ and 2-dimentional viewing direction $\mathbf{d} = (\theta, \phi)$ as inputs and outputs the volume density σ and RGB values $\mathbf{c}$: $F(\Theta) : (\mathbf{p}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, where Θ represents the network weights.

NeRF assumes that the camera intrinsics and extrinsics for each input image are known. Those camera parameters are used to generate rays that start from the camera location and go through image pixels in the world coordinate system. Several points are sampled along the rays. The color $C(\mathbf{r})$ of each ray $\mathbf{r}(t)$ is calculated by the classical volume rendering method [8], and the continuous integral is approximated by the quadrature,

$$\begin{aligned} \hat{C}(\mathbf{r}) &= \sum_{i=1}^{N} T_i(1-\exp(-\sigma_i\delta_i))\mathbf{c}_i, \\ T_i &= \exp(-\sum_{j=1}^{i-1}\sigma_j\delta_j), \end{aligned} \tag{1}$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples. NeRF also adopts a stratified sampling method which samples uniformly in evenly-spaced bins. For more details, we refer the readers to [5].

Neural Sparse Voxel Fields (NSVF) [33], extends NeRF by introducing voxels to divide the scene. It assigns embedding vectors at each voxel vertex, and the spatial position $\mathbf{p}$ of the network input is replaced by the interpolated feature obtained from the aggregation of eight embedding features of the voxel vertices. The interpolated feature $g(\mathbf{p})$ and the viewing direction $\mathbf{d}$ will then go through positional encoding $\zeta_p(\cdot)$ and $\zeta_d(\cdot)$ respectively, which is also used in NeRF [5]. So the MLP network predicting density σ and color $\mathbf{c}$ in NSVF turns to:

$$F_\Theta : (\zeta_p(g(\mathbf{p})), \zeta_d(\mathbf{d})) \rightarrow (\mathbf{c}, \sigma), \forall \mathbf{p} \in V_i. \tag{2}$$

The initial voxels of the scene are obtained from the subdivision of the bounding box of the scene. During the training process, NSVF will gradually prune non-essential voxels according to the predicted density. On the other hand, the remaining voxels will be further subdivided. After the pruning and subdivision, the voxels will gradually approach the real geometry of the scene. This not only helps with the convergence of the training, but also reduces the number of sampled points during test and reduces the rendering time. In our method, we also represent the scene using the voxels. The difference is that from the collected RGB-D data, the scene geometry (point cloud or triangle mesh) can be directly obtained, and the voxels can be initialized according to the known geometry. This can provide a geometry prior at the very beginning of the training, but also helps us provide a color prior which we will introduce in Sec. 4.5. The octree structure can also be used to accelerate the rendering. We will introduce our method in the next section.

4 METHOD

4.1 Overview

The proposed method aims to reduce the demand on the number of user shots, with the help of the depth, while ensuring satisfactory rendering results from novel views. Our method uses the LiDAR camera and RGB camera equipped on consumer products to capture the RGB-D images. The point cloud of the scene can be reconstructed through the captured RGB-D images and the camera parameters exported from the ARKit. We further reconstruct the triangle mesh from the point cloud, and render it from any views we want. The rendered images from this step are not as realistic as the captured images but depict the approximate appearance of the scene. Previous NeRF-based methods need to provide a large number of real captured images to train the network. When the number of those images cannot reach the required number, we propose to introduce the depth, and provide a large number rendered images to pre-train the network. Since the rendered images have precise camera parameters and come from many view directions, the pre-training process can provide a strong prior for the network. On this basis, we further propose a fine-tuning method which uses a small number of real captured images so that the network after fine-tuning can generate realistic novel view images. The whole pipeline of our method is illustrated in Fig. 1.

4.2 Mesh Reconstruction

We first reconstruct the rough geometry, including the point cloud and the triangle mesh of the scene. We make the full use of the depth images captured through the consumer devices such as the LiDAR camera of Apple iPad Pro and Microsoft Kinect. As mentioned before, other NeRF-based methods including NeRF [5] and NSVF [33], require a lot of images to achieve satisfactory novel view synthesized images, which puts a burden on the users in practical use. While with our method, we only need to capture images in several scattered views, which cover the most appearances of the interested area of a scene.

Given the RGB images $\{I_i, i = 1, \cdots, n\}$ and the corresponding depth images $\{I_i^d, i = 1, \cdots, n\}$ captured by the RGB camera and the LiDAR camera, we first reconstruct the raw point cloud $\{P_i, i = 1, \cdots, n\}$ in each captured views, with the help of camera parameters exported from the ARKit. Due to the different resolutions, we scale the RGB images to the same resolution as the depth images and assign the color value for each point of the reconstructed point cloud in each view. The point cloud of each view is then denoised using the outlier removal algorithm of Open3D [86], aligned with the colored iterative closest point algorithm (ICP) to eliminate the pose error, and finally merged into a complete point cloud using a box grid filter. The complete point cloud will be transformed to a colored triangle mesh by Poisson surface reconstruction [87], [88]. The whole process from RGB-D images to the triangle mesh is shown in Fig. 2.

4.3 Pre-training by rendering

The core of our approach is the pre-training strategy using the rendered images as the pseudo ground truth. Once the reconstructed mesh is obtained, we can render the mesh from any view that we want to obtain a rendered image. To cover the most views of the scene, we interpolate between the known camera views $\{\mathbf{d}_i, i = 1, \cdots, n\}$ and perturb each known camera view to generate a large number of novel camera views. We denote those views as the sampled views $\{\mathbf{d}_i^{'}, i = 1, \cdots, k\}$, where k is the number of sampled views. Under these sampled views, we use OpenGL to render the reconstructed colored mesh to produce plenty of rendered images $\{I_i^{'}, i = 1, \cdots, k\}$, which are further

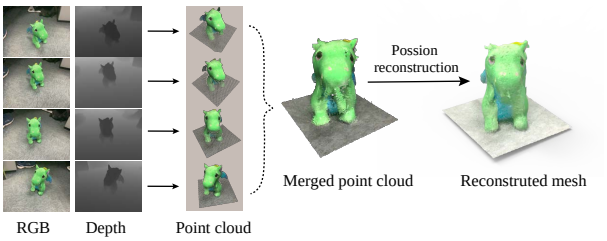


Figure 2. The mesh reconstruction process in our method. We first obtain the point cloud of each view from the RGB-D image, and then align and merge them into a single scene point cloud. The triangle mesh is reconstructed from the scene point cloud through Poisson reconstruction [87], [88].

used to pre-train the network. The number k of these rendered images can be large enough to train a satisfactory neural radiance field network. And the corresponding camera parameters are accurate, which is more conducive to the training of the network than the camera parameters reconstructed by the methods such as COLMAP directly from the RGB images.

For the neural radiance field network, we adapt the same architecture of NSVF [33]. Since we have the raw geometry of the scene, we can initialize the voxels of the scene from the point cloud or the triangle mesh. The initial voxel size depends on the scene size. We also use the octree structure [89] to organize those voxels of the scene, and apply the Axis Aligned Bounding Box intersection test [90] for each ray. This test is very efficient for the voxels in sparse octree structure. We define a learnable embedding feature $\tilde{g}$ at each voxel grid node. For each intersected voxel, we uniformly sample points on the intersecting ray segments, and use trilinear interpolation to obtain the input feature $g(\mathbf{p})$ of each sampled point $\mathbf{p}$. The inputs to our network are the interpolated feature $g(\mathbf{p})$ of the sampled point and the ray direction $\mathbf{d}$ of the ray $\mathbf{r}$. The input feature $g(\mathbf{p})$ and ray direction $\mathbf{d}$ will go through encoding process $\zeta_p(\cdot)$ and $\zeta_d(\cdot)$ respectively, which is the same as NeRF [5], to capture high-frequency details. The encoded input feature $\zeta_p(g(\mathbf{p}))$ will be input to an MLP network F_σ to predict the volume density σ of the sampled point,

$$\sigma = F_\sigma(\zeta_p(g(\mathbf{p}))), \tag{3}$$

and another MLP network F_c will concatenate the encoded input feature $\zeta_p(g(\mathbf{p}))$ and the encoded ray direction $\zeta_d(\mathbf{d})$ as the input and predict the color value $\mathbf{c}$ at the sampled point,

$$\mathbf{c} = F_c(\zeta_p(g(\mathbf{p})), \zeta_d(\mathbf{d})). \tag{4}$$

The whole network architecture is illustrated in Fig. 3. We adopt Eq. 1 to calculate the pixel color $\hat{C}(\mathbf{r})$ of the generated image and use those rendered images $\{I_i^{'}, i = 1, \cdots, k\}$ as the ground truth. The pre-training process is supervised by the RGB loss function $\mathcal{L}_{RGB}$, which is formulated as:

$$\mathcal{L}_{RGB} = \sum_{\mathbf{r} \in R} \|\hat{C}(\mathbf{r}) - C(\mathbf{r})\|_2^2, \tag{5}$$

where R is the set of sampled rays in the mini-batch, $C(\mathbf{r})$ is the ground truth color of the ray $\mathbf{r}$. It should be noted that, during the pre-training process, in order to ensure the completeness of the scene, we do not perform pruning operations on the voxels. But when necessary, we will subdivide the voxels, that is, the size of the voxels will be half of the original.

4.4 Few-shot training

After the pre-training of the network, since the rendered images are used as the (pseudo) ground truth, the novel view images generated by the network are also in the rendering style, which is different from the real captured images. In order to enable the network to generate realistic images in novel views with the help of a few captured images, we introduce a fine-tune process to finish the few-shot training of the network.

During the few-shot training process with few realistic images, we simply replace the rendered images $\{I_i^{'}, i = 1, \cdots, k\}$ with those real captured images $\{I_i, i = 1, \cdots, n\}$ as the ground truth and continue training. Although the operation of directly replacing the ground truth is very simple, we have found that this simple strategy has been able to help the network to generate novel view images of realistic style. However, there are still deficiencies in the details, especially when the novel view is quite different from the views of known real images. Therefore, we introduce a discriminator D and regard the whole MLP network, including F_σ and F_c, as the generator G, to improve the network training and the details of the generated images using a GAN architecture. During the training, we not only sample rays from the known camera views $\{d_i, i = 1, \cdots, n\}$ and perform the same RGB loss supervision as pre-training process, but also generate the images $\{I_i^{fake}, i = 1, \cdots, k\}$ from the sampled views $\{d_i^{'}, i = 1, \cdots, k\}$. Those generated images $\{I_i^{fake}, i = 1, \cdots, k\}$ are grouped with the real captured images $\{I_i, i = 1, \cdots, n\}$ to train our GAN architecture. As we do not acquire a large number of captured images, the number of "real" images is less than the number of "fake" (generated) images. In order to solve the problem of the inadequate "real" samples, we adopt a patch-based discriminator [91]. We randomly sample a certain size of image patch from the real image and the generated fake image respectively, and feed them into the discriminator to determine whether it is real or fake. The patch size is typically 32×32. The patch-based discriminator can not only help us increase the number of real samples in the training, but also reduce memory consumption. The whole framework is illustrated in Fig. 3.

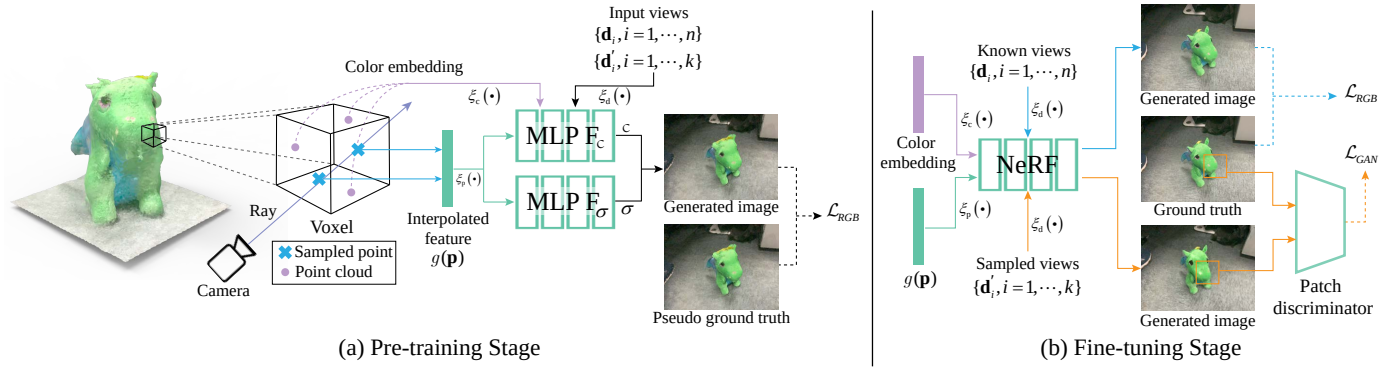
The training objective of the fine-tuning process consists of the RGB loss function $\mathcal{L}_{RGB}$ and the GAN loss function $\mathcal{L}_{GAN}$. The definition of the RGB loss $\mathcal{L}_{RGB}$ is the same as Eq. 5. The GAN loss function $\mathcal{L}_{GAN}$ is formulated as:

$$\begin{aligned} \mathcal{L}_{GAN} = \max_G \min_D (&E_{p \sim P_{fake}}[\log(D(p))] \\ &+ E_{\hat{p} \sim P_{real}}[\log(1 - D(\hat{p}))]), \end{aligned} \tag{6}$$

where P_{fake} is the set of fake image patches and P_{real} is the set of real image patches. The total training loss is the sum of $\mathcal{L}_{RGB}$ and $\mathcal{L}_{GAN}$:

$$\mathcal{L}_{fine-tune} = \mathcal{L}_{RGB} + \alpha \mathcal{L}_{GAN}, \tag{7}$$

where α is the adjustable weight. It should be noted that the RGB loss function $\mathcal{L}_{RGB}$ only works for the generator step.



(a) Pre-training Stage (b) Fine-tuning Stage

Figure 3. The training network of our method. Our training is divided into two stages. The first stage (a) is to pre-train the network with the rendered images, using both the known views and sampled views, and $\mathcal{L}_{RGB}$ as supervision. The second stage (b) uses realistic images to fine-tune the network. At this stage, the known views are supervised by $\mathcal{L}_{RGB}$, which is indicated by the blue arrows, while the sampled views are supervised by $\mathcal{L}_{GAN}$, which is indicated by orange arrows. We also obtain the color embedding from the point cloud as one of the inputs to predict the color value.

4.5 Voxel color prior

The GAN architecture can provide our network with additional supervision under more sampled views, in addition to the supervision of the RGB loss function under known views. Besides increasing supervision, it is also a good idea to make better use of the small amount of the RGB images and the depths to provide priors to enhance the generated results.

As a complete colored point cloud of the scene is reconstructed, we can know which part of the point cloud is contained in a voxel and use the color information of the point cloud as voxel color. Since the number of points contained in each voxel is inconsistent, we take the average color value of the point cloud in a voxel as the voxel color $\mathbf{c}_{voxel}$ of that voxel. The RGB value $\mathbf{c}_{voxel}$ is low-dimensional, so we further use similar positional encoding [5] $\zeta_c(\cdot)$ of spatial location to encode the 3-dimensional color information to the high-dimensional feature. The encoded voxel color feature $\zeta_c(\mathbf{c}_{voxel})$ is input into the color prediction network F_c as a condition, as shown in Fig. 3. So the MLP network F_c will be re-formulated as:

$$\mathbf{c} = F_c(\zeta_p(g(\mathbf{p})), \zeta_d(\mathbf{d}), \zeta_c(\mathbf{c}_{voxel})). \tag{8}$$

The density prediction network F_σ is irrelevant to the voxel color, so F_σ is still formulated as Eq. 3 and the density is only determined by the encoded positional feature at the sampled point. $\zeta_c(\mathbf{c}_{voxel})$ provides a color prior for the sampled points in the corresponding voxel, and it is a prior from the real scene. Compared with those works [42], [92] that extract color features from the 2D images, our method extracts color information from the 3D point cloud which can resolve the ambiguity in depth.

5 Experiments and Evaluations

In this section, we will evaluate our method in the realistic scenes that are captured by a consumer product equipped with both an RGB camera and a LiDAR camera. We first report the implementation details. Then we show some results and compare them with the previous state-of-the-art methods, both qualitatively and quantitatively. Finally, we verify the role of each technical component in our method.

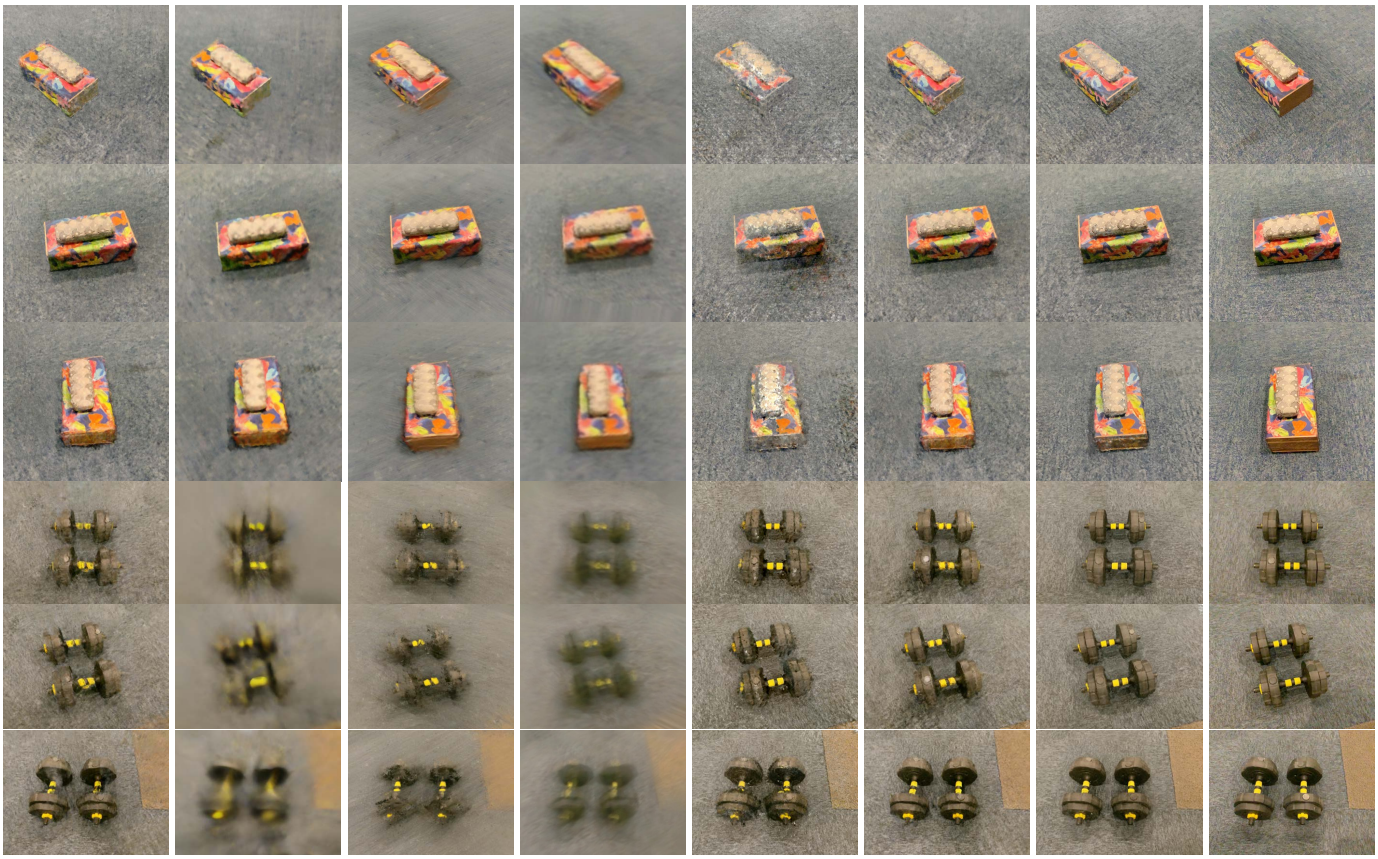
5.1 Implementation Details

We use a single Nvidia GeForce RTX3090 GPU for all of our experiments. Different scene sizes require different training time, which is ranging from 24 hours to 36 hours. Typically, the pre-training process requires about 9000 epochs, and the fine-tuning process requires 20000 epochs. The resolution of the voxel grid depends on the size of the captured scene. We generally take the length of a single voxel as $1/50$ of the longest edge of the scene bounding box. The patch size we used in the patch discriminator is 32×32. The parameter α in the loss function is set to 0.1 for all datasets. The Adam optimizer [93] is used for training, and the learning rate is set to 0.001.

We choose Apple iPad Pro which is more portable as the device to collect all of our datasets. The dataset we collected includes five indoor scenes ("crocodile toy", "plant", "box", "dumbbells" and "character toys") and two scenes containing relatively big objects ("stone bench" and "stone art"). All datasets contain 1920×1440 RGB images with 256×192 depth images and camera parameters exported from iPad Pro. We hold the iPad Pro while walking around the object and shoot RGB-D images continuously. A brief introduction to each scene and the RGB-D images selected for reconstruction and training are presented in the supplementary material. Note that in all these examples, a small number of RGB-D images are used for training and the rest are used for evaluation.

5.2 Metrics

In order to compare our method with the state-of-the-art methods quantitatively, we adopt three commonly used metrics: Peak Signal-to-Noise Ratio (PSNR), Structure Similarity Image Metric (SSIM) [94] and Learned Perceptual Image Patch Similarity (LPIPS) [95]. The three metrics are calculated between the synthesized images under the test views and the corresponding ground truth images. Note that for PSNR and SSIM, the larger the better, while for LPIPS the smaller the better.



(a) NeRF (b) NeRF++ (c) IBRNet (d) pixelNeRF (e) NSVF (f) NeRF w/ depth (g) Ours (h) Ground Truth

Figure 4. Comparisons of novel view synthesis on "box" dataset and "dumbbells" dataset. It can be clearly seen that our results are the clearest and the closest to the ground truth images.

5.3 Novel View Synthesis Results

For comparison, we first consider some current state-of-the-art NeRF-based methods, including the original NeRF [5], NeRF++ [96], NSVF [33] and IBRNet [42]. As the main goal of our method is to reduce the number of shots needed, we also compare with pixelNeRF [69], which also reduces the number of input images. It should be noted that the above methods do not use depth as input, but we emphasize that with consumer-grade products, users can also obtain a depth image while taking an RGB image, which will not bring extra burden. As we obtain the reconstructed mesh of the scene, the rough depth of each ray is also known. So we also add the depth loss to NeRF for comparison, denoted as 'NeRF w/ depth'. Both IBRNet and pixelNeRF are general models, which can be generalized in multiple scenes. Based on the official checkpoint, we fine-tune their networks on our dataset. The images used in fine-tuning are the same as those used in our method. For other methods, we re-train their networks on our own dataset, and the images used for training are the same as our method. NSVF also uses known voxel initialization during training. We also compare our method with some recent NeRF-based methods which also introduce depth information, including ToRF [72], DS-NeRF [9] and NerfingMVS [97]. Moreover, we propose a baseline method which combines the RGB loss with depth loss and patch GAN loss, denoted as "NeRF+3loss". A mesh-based reconstruction method, Intrinsics3d [98] is also compared. In addition to NeRF-based methods, recently there were also some novel view synthesis methods based on image translation. Some of these also adopt geometric priors, such as Stable View Synthesis (SVS) [17] and Neural Point Based Graphics (NPBG) [18]. We also show the comparisons with these two methods. At last, we show some NVS results on the public dataset, BlendedMVS [99] and NeRF dataset [5].

Comparisons with NeRF-based methods. We evaluate our method and other methods on the test set with ground truth images. We perform comparisons on the "box" dataset and the "dumbbell" dataset in Fig. 4, on the "plant" dataset in Fig. 5, on the "crocodile toy" dataset and the "character toy" dataset in Fig. 6, and on "stone bench" dataset and "stone art" dataset in Fig. 7. It can be seen that the results of other methods are blurry or lose some details. In particular, although NeRF with depth loss performs better than the original NeRF, it still suffers from obvious artifacts. The "plant" dataset has fine details, like the leaves of the plant. The leaves of IBRNet results are missing and the leaf color of NSVF results is significantly affected by the floor color. In contrast our method maintains the leaf geometry and color thanks to the patch discriminator and 3D voxel color prior. The "character" scene is also a forward-facing scene. Compared with other methods, the results of our method

are clearer and have more details. We can see that our results can reconstruct view-dependent effects, such as the reflections on the surface of the stone bench and handling the reflection of the water surface in novel view synthesis, while other methods cannot even maintain the basic shape and produce artifacts on the main object of the synthesized images. We show more novel view synthesis results on other scenes in Fig. 8. In addition to the visualization results, we also compare the quantitative results in Table 1. We show the results on four scene datasets, "plant", "crocodile", "box" and "dumbbells". The results on other scene datasets are shown in the supplementary file. Our method outperforms other methods in all three metrics, which demonstrates the ability of our method to use sparse views to synthesize high-quality novel view images.

Comparisons with NeRF-based methods using depth information and a mesh-based reconstruction method. We show the comparisons with some NeRF-based methods using depth information, including ToRF [72], DS-NeRF [9], NerfingMVS [97] and a baseline method "NeRF+3loss" in Fig. 9. We also show the comparisons with a mesh-based reconstruction method, Intrinsic3d [98]. It can be seen that our method still achieves better results. The quantitative comparisons are shown in Table 2. Our method achieves the best performance in all three metrics.

Comparisons with other geometry-based NVS methods. We show the comparisons with two recent geometry-based novel view synthesis methods SVS [17] and NPBG [18] in Fig. 10 and Table 3. We also show the point cloud rendering under the corresponding views in Fig. 10. For NPBG, we use our reconstructed scene point cloud as input and re-train the network. For SVS, we use our mesh reconstruction as input, and train the network with self-collected datasets other than the two shown in the results. Overall, these two methods can ensure the completeness of the scene under novel views. However, NPBG has some artifacts and instability across different views, which can also be seen in the supplementary video. The results of SVS are blurry and lack the details of the object surface. In contrary, our method can maintain the veins on the leaf and the texture of dumbbells.

Results on BlendedMVS. In addition to the self-collected datasets, we also test our method on the public dataset, BlendedMVS [99], which contains RGB-D images with camera parameters. The novel view synthesis results are shown in Fig. 11. For the first three scenes we show, we select 8 views using the farthest point sampling method from the provided RGB-D images. The last scene is a forward-facing case, and we select 6 views in the same way. The scene point cloud is then obtained from the selected RGB-D images and corresponding camera parameters. It can be seen that our method can still guarantee satisfactory results. The NVS results can maintain the details on the bread and sculptures. The forward-facing scene is also handled well. It is worth noting that the depth image resolution of this dataset is higher than that of our collection, which improves the quality of the scene point cloud. The calibrated camera parameters are also more accurate, which is conducive to training. We also show view-dependent effect on bread and pottery scenes in the supplementary video.

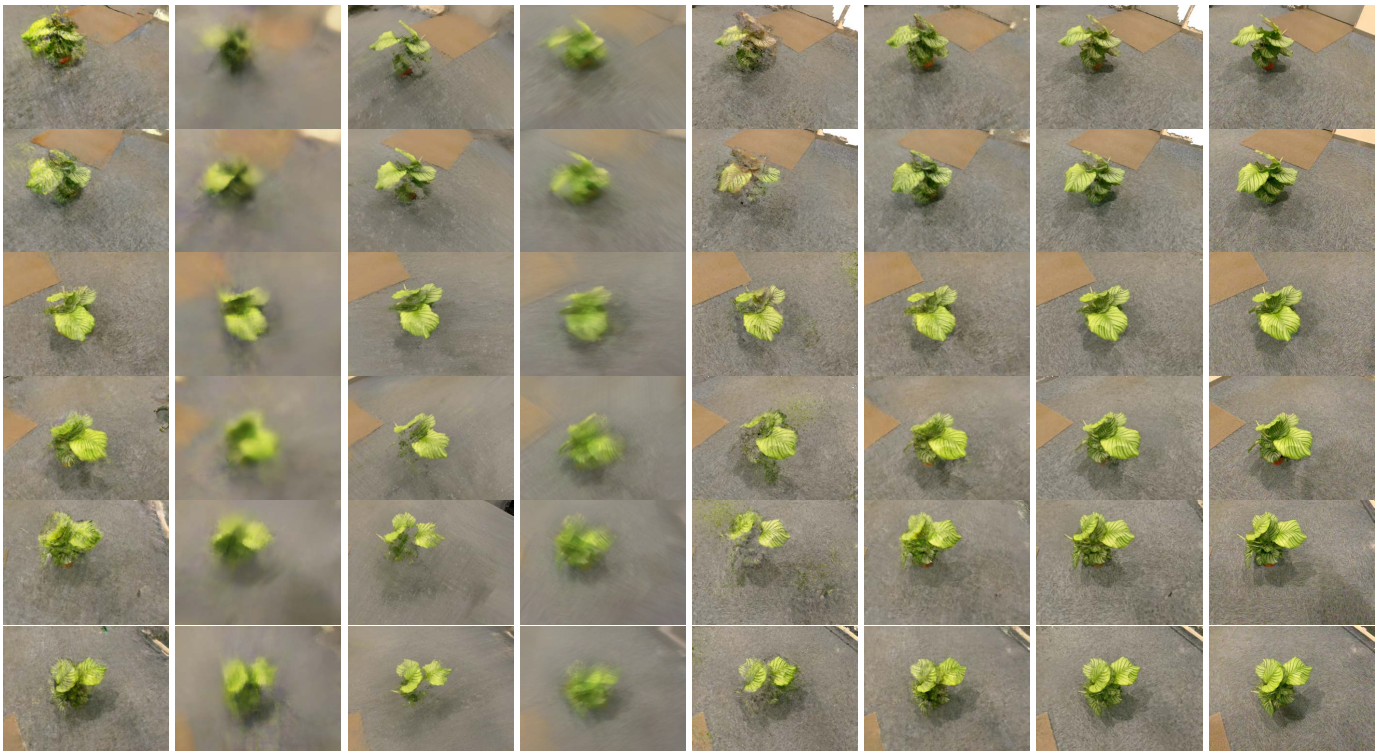
Results on NeRF dataset. In addition to using captured depth values, our method can also utilize depth values estimated from images. For the NeRF dataset, we use COLMAP to estimate the depth and camera parameters, and use our method to reconstruct the mesh and perform pre-train-fine-tune training. The training images (6 images) for each scene are selected by farthest point sampling. The result is shown in Fig. 12. It can be seen that our method still achieves satisfactory results.

5.4 Evaluation of Settings

Some settings in our method are based on empirical experiments. In this section, we evaluate such settings in our method to determine which choice is better to achieve satisfactory synthesis results. These evaluations include the number of real images and the distribution of input views. We also evaluate the quality of the reconstruction geometry, which is shown in the supplementary document.

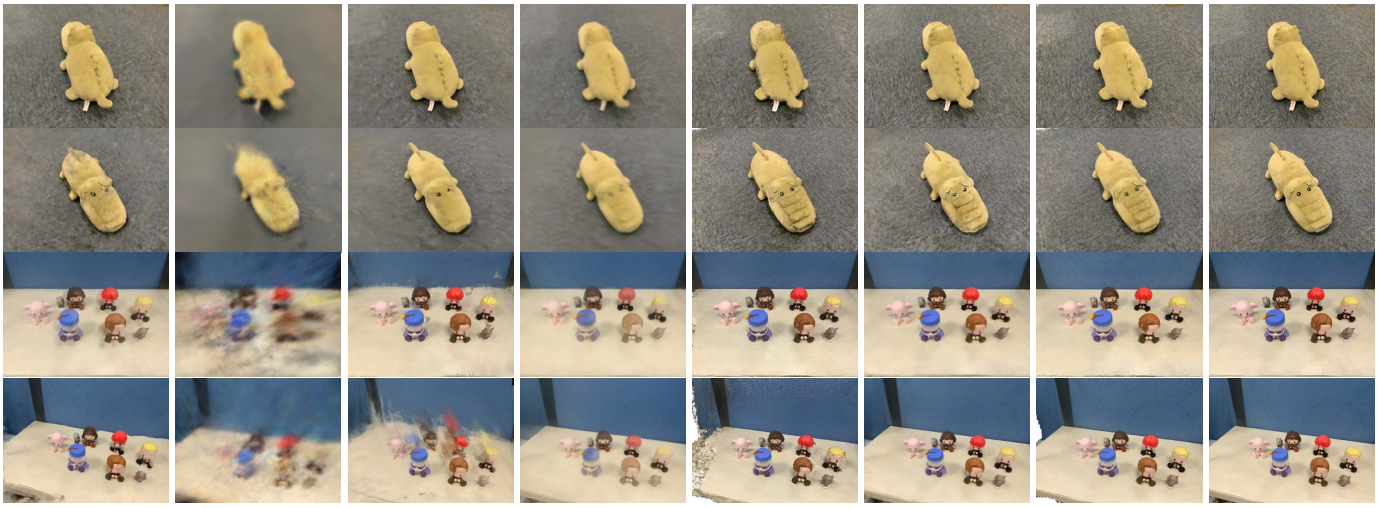
Number of real images. The number of real images we currently select on each dataset is manually determined. In order to clarify the effect of the number of real images in the fine-tuning stage on the final synthesis result, we compare the synthesis results of different real image numbers while ensuring that the reconstruction geometry and the pre-training stage are the same. When there is only one input image, we directly select the first image in the dataset for training in the fine-tuning stage. For other number settings, we adopt the farthest point sampling (FPS) method to select the training images. It should be noted that the RGB-D images used for reconstruction are fixed, and also selected by the FPS method (the same as the choices in previous section). The quantitative results are shown in Table 4. It can be seen that, in general, as the number of real images increases, the synthesis quality will also improve. But when the number of real images reaches a certain level, the improvement is no longer significant. This is because those images are able to train a satisfactory NeRF model. It is worthy to note that when using fewer real images than the comparison methods in Table 1, our method can achieve comparable or even better performance. We also compare with the well-performing baseline, NeRF w/ depth under different numbers of images. It can be seen that our method outperforms NeRF w/ depth when the number of images is smaller (up to 20 views), and can achieve comparable results (slightly worse PSNR but better SSIM and LPIPS which are more perceptual) when the number of images is large.

Distribution of input views. The distribution of input views is another factor that may affect the final synthesis results. As mentioned before, we adopt the farthest point sampling (FPS) method to select a specified number of input views. In order to evaluate the influence of the randomness of user capturing on the results, we set up some representative experimental groups. The first group (User Select) is hand-picking the views that are relatively far from each other. The second group (Uneven) is to first select the two most distant views, and select the remaining views near one of the picked views. And the last group is the FPS method. The first group simulates user selection, the second group is an extreme situation, and the last group is an ideal situation. The comparison results are shown in Table 5. It can be seen that the result of User Select is comparable to



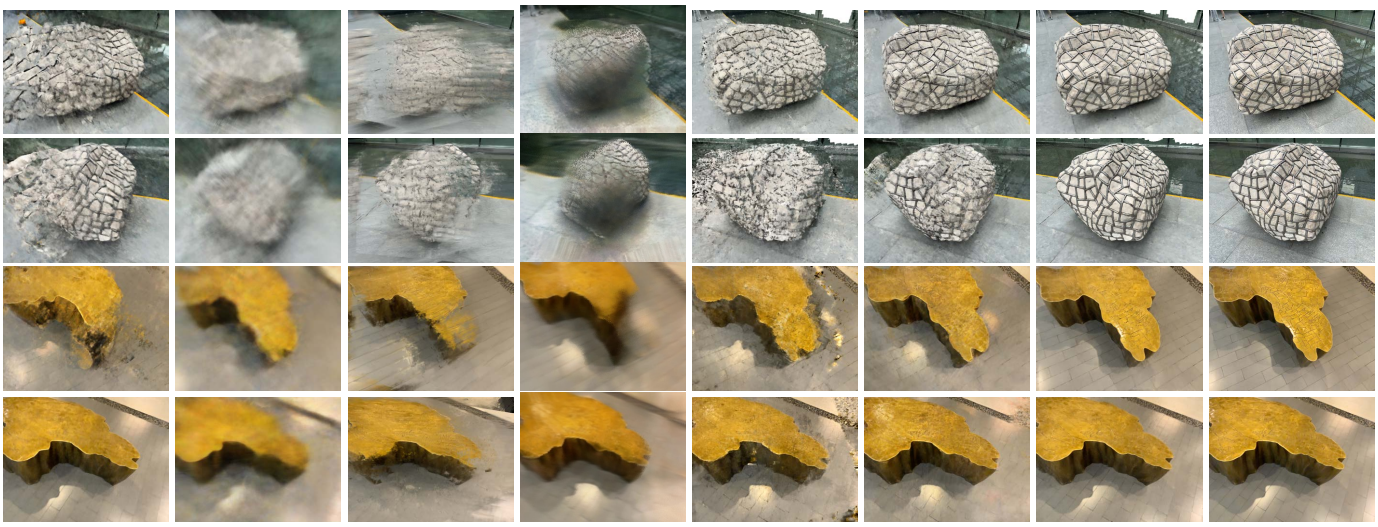
(a) NeRF (b) NeRF++ (c) IBRNet (d) pixelNeRF (e) NSVF (f) NeRF w/ depth (g) Ours (h) Ground Truth

Figure 5. Comparisons of novel view synthesis on "plant" dataset. Our results preserve the leaves of the plants to the greatest extent, and the results are the clearest.



(a) NeRF (b) NeRF++ (c) IBRNet (d) pixelNeRF (e) NSVF (f) NeRF w/ depth (g) Ours (h) Ground Truth

Figure 6. Comparisons of novel view synthesis on "crocodile" dataset and "character toy" dataset. The comparisons of these two datasets show that our method can handle the scene of plush objects and multiple small objects. The last one is also a forward-facing scene.



(a) NeRF (b) NeRF++ (c) IBRNet (d) pixelNeRF (e) NSVF (f) NeRF w/ depth (g) Ours (h) Ground Truth

Figure 7. Comparisons of novel view synthesis on "stone bench" dataset and "stone art" dataset. Our method can synthesize view-dependent effects, such as the reflection of the water surface (the second row) and the highlights of the bench surface (the third row).

Table 1
Quantitative comparison against several NeRF-based methods. Compared with NeRF, NeRF w/ depth, NeRF++, IBRNet, pixelNeRF and NSVF, our method achieves the best performance in all three metrics.

	Plant			Crocodile			Box			Dumbbells			Mean		
Models	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF	20.20	0.3465	0.6064	20.63	0.4134	0.6358	21.09	0.3644	0.5988	21.98	0.3531	0.5784	20.98	0.3694	0.6049
NeRF++	23.62	0.3906	0.7042	21.16	0.4173	0.7440	21.68	0.3867	0.6872	22.49	0.3692	0.6233	22.24	0.3910	0.7062
IBRNet	22.20	0.3755	0.4906	19.88	0.4250	0.5182	22.10	0.4005	0.4926	22.20	0.3754	0.5073	21.60	0.3941	0.5022
pixelNeRF	21.64	0.3442	0.7224	20.13	0.4003	0.6110	22.65	0.3609	0.7331	20.62	0.3072	0.7257	21.26	0.3532	0.6981
NSVF	19.95	0.4943	0.3933	20.02	0.5373	0.3250	22.17	0.5433	0.3304	23.33	0.5328	0.2901	21.36	0.5269	0.3347
NeRF w/ depth	23.60	0.4714	0.5356	22.16	0.4805	0.4738	23.29	0.4307	0.3884	22.77	0.3866	0.4489	22.96	0.4423	0.4617
Ours	**25.14**	**0.5886**	**0.3461**	**24.91**	**0.5408**	**0.2676**	**25.07**	**0.5555**	**0.2987**	**25.96**	**0.6313**	**0.2799**	**25.27**	**0.5791**	**0.2981**

Figure 8. Novel view synthesis results on more scenes. In each group, the first row is the synthesized result, and the second row is the ground truth.

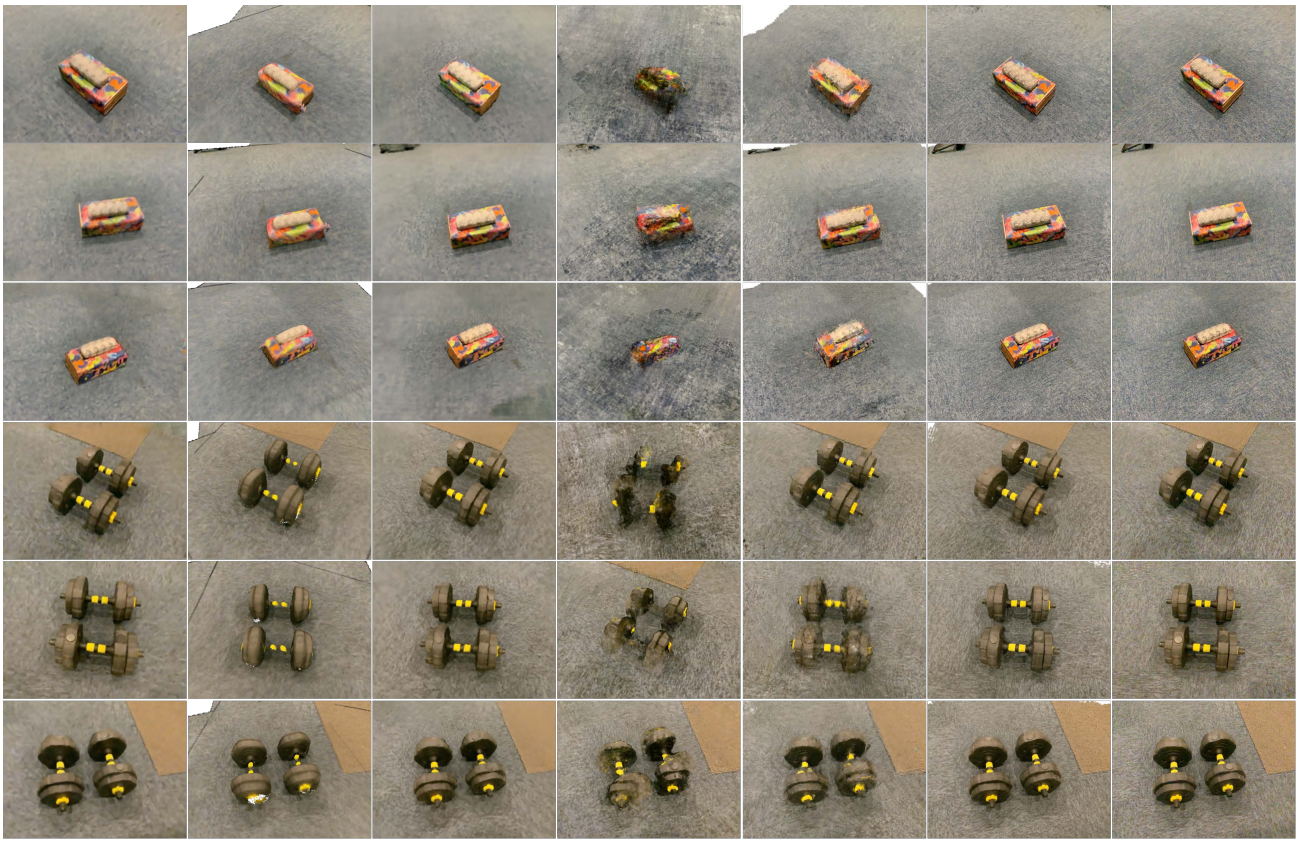
Table 2
Quantitative comparison against several NeRF-based methods using depth information and a mesh-based reconstruction method. Compared with these methods, our method achieves the best performance in all three metrics.

	Box			Dumbbells			Mean		
Models	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
ToRF	18.89	0.3179	0.5681	19.36	0.3037	0.5096	19.13	0.3108	0.5388
Intrinsics3d	19.49	0.2935	0.4936	19.59	0.2771	0.4890	19.54	0.2853	0.4913
DS-NeRF	24.05	0.4875	0.5332	25.59	0.6076	0.4000	24.82	0.5476	0.4666
NerfingMVS	18.13	0.2271	0.4668	20.85	0.5153	0.3137	19.49	0.3712	0.3903
NeRF+3loss	21.38	0.2967	0.3552	22.76	0.3641	0.3437	22.07	0.3304	0.3494
Ours	**25.07**	**0.5555**	**0.2987**	**25.96**	**0.6313**	**0.2799**	**25.52**	**0.5934**	**0.2893**

Table 3
Quantitative comparison against several geometry-based NVS methods. Compared with NPBG and SVS, our method achieves the best performance in all three metrics.

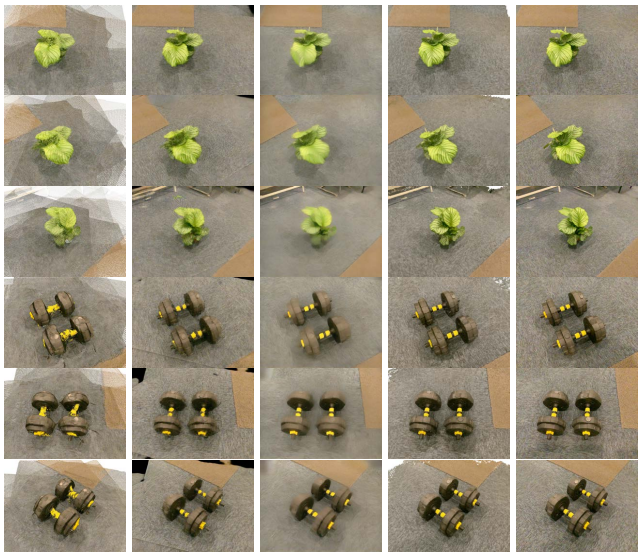
	Plant			Dumbbells			Mean		
Models	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NPBG	20.73	0.3954	**0.3454**	19.44	0.3116	0.3669	20.09	0.3535	0.3562
SVS	24.36	0.5502	0.3981	23.55	0.5111	0.4038	23.96	0.5306	0.4009
Ours	**25.14**	**0.5886**	0.3461	**25.96**	**0.6313**	**0.2799**	**25.55**	**0.6100**	**0.3130**

the result of FPS, but the result of Uneven is much worse, which is understandable because the information provided by real images is limited to two limited view ranges. We can add user guidance in practical use to ensure better results. This is also one of the future work. Moreover, we also add comparisons with NeRF w/ depth under different settings. It can be seen that our method still achieves better results under these three settings.



(a) ToRF (b) Intrinsic3d (c) DS-NeRF (d) NerfingMVS (e) NeRF+3loss (f) Ours (g) Ground Truth

Figure 9. The comparisons with ToRF [72], DS-NeRF [9], NerfingMVS [97], Intrinsics3d [98] and a baseline method "NeRF+3loss". Our method outperforms other methods in detail and clarity.



(a) PC Renderer (b) NPBG (c) SVS (d) Ours (e) Ground Truth

Figure 10. Comparisons of novel view synthesis on "plant" dataset and "dumbbells" dataset with NPBG and SVS. We also show the rendered images of the scene point cloud under the corresponding views. It can be seen that our method achieves clearer results without artifacts of other methods.

Figure 11. Novel view synthesis results on the public dataset, BlendedMVS.

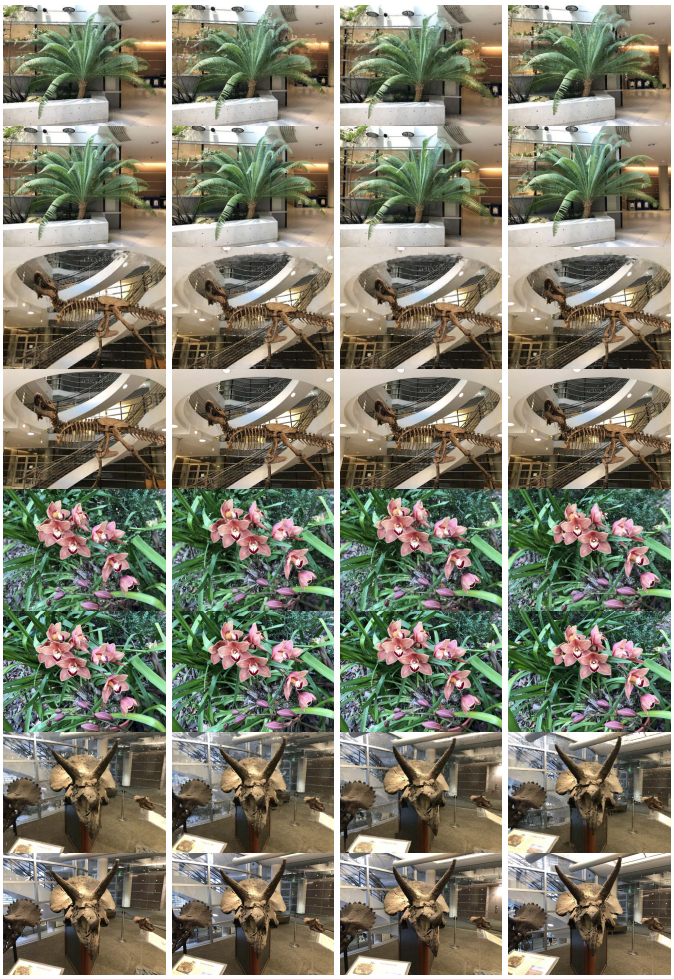


Figure 12. Novel view synthesis results on the NeRF dataset. In each group, the first row is the synthesized result, and the second row is the ground truth.

Table 4
Evaluation of NeRF reconstruction quality w.r.t. the number of real images. When the number reaches a certain level, the improvement is no longer significant. But when using fewer real images than the comparison methods in Table 1, our method can achieve comparable or even better performance. We also compare our method with NeRF w/ depth under different numbers of images. It can be seen that our method outperforms NeRF w/ depth when the number of images is smaller, and can achieve comparable results (slightly worse PSNR but better SSIM and LPIPS) when the number of images is large.

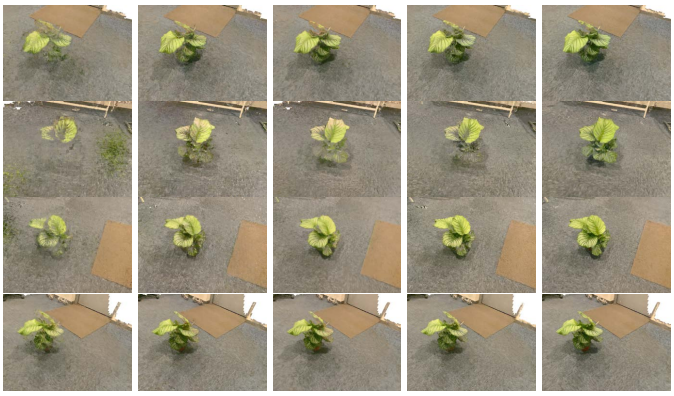
# of views	Methods	PSNR ↑	SSIM ↑	LPIPS ↓
1	NeRF w/ depth	19.38	0.3754	0.5547
	Ours	**21.75**	**0.3971**	**0.3639**
3	NeRF w/ depth	22.03	0.3646	0.4599
	Ours	**23.15**	**0.4759**	**0.3380**
5	NeRF w/ depth	22.37	0.4251	0.4699
	Ours	**23.25**	**0.5163**	**0.3562**
7	NeRF w/ depth	23.14	0.4549	0.4488
	Ours	**23.49**	**0.5234**	**0.3506**
9	NeRF w/ depth	23.60	0.4570	0.4714
	Ours	**25.14**	**0.5886**	**0.3461**
20	NeRF w/ depth	25.38	0.5356	0.4459
	Ours	**25.42**	**0.6155**	**0.3759**
40	NeRF w/ depth	**25.91**	0.5948	0.4684
	Ours	25.67	**0.6277**	**0.3878**
60	NeRF w/ depth	**26.05**	0.6063	0.4745
	Ours	25.71	**0.6285**	**0.3974**
80	NeRF w/ depth	**26.18**	0.6059	0.4746
	Ours	25.77	**0.6283**	**0.3940**

Table 5
Evaluation of the distribution of input views. The result of User Select is comparable to the result of FPS, but the result of Uneven is much worse. We can add some user guidance to avoid the extreme case, which is also one of the future work. We also add comparisons with NeRF w/ depth under different settings. It can be seen that our method still achieves better results under these three settings.

Settings	Methods	PSNR ↑	SSIM ↑	LPIPS ↓
User Select	NeRF w/ depth	23.60	0.4714	0.5356
	Ours	**24.73**	**0.6163**	**0.3462**
Uneven	NeRF w/ depth	21.72	0.4527	0.4943
	Ours	**23.37**	**0.5303**	**0.3785**
FPS	NeRF w/ depth	24.64	0.5492	0.4318
	Ours	**25.14**	**0.5886**	**0.3461**

5.5 Ablation Study

In this section, we perform qualitative and quantitative experiments to evaluate that each component of our method achieves the expected effects. We experiment five different settings. The first one removes the pre-training stage and directly trains the network with voxel color prior and patch discriminator using those real captured images, denoted as 'No pre-train'. The other four adopt the pre-training stage, while the second one does not use voxel color prior and patch discriminator, denoted as 'Directly fine-tune', the third one uses patch discriminator but does not use voxel color prior, denoted as 'w/o dis.', the forth one uses voxel color prior but does not use patch discriminator, denoted as 'w/o dis.', and the last one uses both voxel color prior and patch discriminator, during the fine-tuning stage, denoted as 'Ours'. The visual results of the comparisons are shown in Fig. 13. By comparing the results of 'No pre-train' with the results of 'Ours', we can conclude that the pre-training-fine-tuning strategy performs well. By comparing the results of 'Directly fine-tune' with the results of 'w/o dis.', we can find that due to the use of voxel color prior, the plant leaves in 'w/o dis.' will not be affected by the color of the floor, and the generated colors are more vivid and closer to the reality. However, due to the complex structure of the plant itself, the results of 'w/o dis.' still fail to show plant leaves in some views (the second row). Compared with 'w/o dis.', 'Ours' adds a patch discriminator, which can provide better supervision under novel views. From the comparison between the forth column and the last column, it can be seen that in the view where the leaves are missing in 'w/o dis.', the result of 'Ours' can maintain the appearance well and is visually better. The comparisons of the above five settings fully prove the effects of the pre-training-fine-tuning strategy and the necessity of voxel color prior and patch discriminator in our few-shot learning framework. They have achieved the expected effect. We further show the quantitative comparison results in Table 6, where our approach outperforms other variations in terms of all three metrics. We also show more ablation study results on some



(a) No pre-train (b) Directly fine-tune (c) w/o v.c. (d) w/o dis. (e) Ours

Figure 13. Ablation study of technical components. "No pre-train" represents that the network is directly trained with voxel color prior and patch discriminator using real captured images. "Directly fine-tune" represents that the network is fine-tuned without voxel color prior and patch discriminator. "w/o v.c." represents that the network is fine-tuned with the patch discriminator but without voxel color prior. "w/o dis." represents that the network is fine-tuned with voxel color prior but without patch discriminator. "Ours" represents that the network is fine-tuned with both voxel color prior and patch discriminator, which is also our choice. It can be clearly found that our choice achieves the best results.

other datasets in the supplementary file.

Table 6
Quantitative experiments of ablation study. The comparison shows that our choice which adopts the pre-training-fine-tuning strategy and combines voxel color prior and patch discriminator obtains the best performance.

Settings	PSNR ↑	SSIM ↑	LPIPS ↓
No pretrain	22.08	0.4794	0.3902
Directly finetune	24.26	0.5755	0.3744
w/o v.c.	24.61	0.5542	0.3552
w/o dis.	24.50	0.5844	0.3542
Ours	**25.14**	**0.5886**	**0.3461**

6 Conclusions and Limitations

In this paper, based on an off-the-shelf consumer product with an RGB camera and a LiDAR camera, we propose a novel view synthesis method that significantly reduces the number of captured views. We first reconstruct the rough geometry of the scene from the captured RGB-D images and render it to obtain sufficient rendered images with precise camera parameters. We then propose to use rendered images to pre-train the network which helps the network learn a prior of the scene. Then the captured few real images are used to fine-tune the network. In order to better use the information of real images to fine-tune the network, we introduce a patch discriminator and voxel color prior to enhance supervision and prior information respectively. The experiments show that our method outperforms the current state-of-the-art NeRF-based method including those that also aim to reduce the number of inputs.

Our method still has some limitations. First, our method is more suitable for indoor scenes and provide robust novel view synthesis under sparse input views. Our method is hard to synthesize novel view images of very large or complex outdoor scenes. This is firstly because it is difficult to model the whole details of large scenes or complex scenes under sparse captured views. On the other hand, it is also limited by the resolution of the LiDAR camera we use, which is only 256×192. However, the pre-training-fine-tuning framework we propose is not limited to iPad Pro, one can improve the effect by adopting a depth sensor with higher precision and slightly increasing the number of captured views. In order to show the usability of public users, we use iPad Pro, a consumer mobile device to perform experiments in this paper. At the same time, limited by the depth sensor and mesh reconstruction method, if the mesh has missing parts or the scene contains high specularity, thin structures or furry appearance, the pre-training stage may not provide the correct prior. Although this can be corrected by the fine-tuning stage, the desired result may not be obtained. We show a failure case of a furry bear in Fig. 14, where some high-frequency texture details fail to be reconstructed. The appearance issues could be solved by combining the idea of Mip-NeRF [39] and Ref-NeRF [40], and the geometry problem could be solved by dynamically expanding or deleting the scene point cloud [44]. Another limitation is that our method cannot synthesize images under the views that are not covered by the captured images. This is also the limitation of most current novel view synthesis method, including NeRF. The inference ability of a completely unknown view requires a large amount of data for training, and NeRF fitting on a single scene is difficult to deal with this situation.

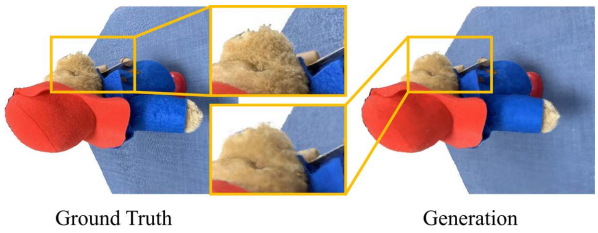


Figure 14. We show a failure case of a furry bear, where some high-frequency texture details fail to be reconstructed.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61872440 and No. 62061136007), the Beijing Municipal Natural Science Foundation for Distinguished Young Scholars (No. JQ21013), the Science and Technology Service Network Initiative, Chinese Academy of Sciences (No. KFJ-STS-QYZD-2021-11-001), the Youth Innovation Promotion Association CAS and Royal Society Newton Advanced Fellowship (No. NAF\R2\192151).

References

[1] H. Shum and S. B. Kang, "Review of image-based rendering techniques," in *Visual Communications and Image Processing*, 2000.
[2] C. Zhang and T. Chen, "A survey on image-based rendering - representation, sampling and compression," *Signal Process. Image Commun.*, vol. 19, pp. 1–28, 2004.

[3] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. M. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B. Goldman, and M. Zollhofer, "State of the art on neural rendering," *Computer Graphics Forum*, vol. 39, 2020.

[4] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: learning dynamic renderable volumes from images," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–14, 2019.

[5] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision*. Springer, 2020, pp. 405–421.

[6] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[7] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] J. T. Kajiya and B. P. Von Herzen, "Ray tracing volume densities," *ACM SIGGRAPH computer graphics*, vol. 18, no. 3, pp. 165–174, 1984.

[9] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised nerf: Fewer views and faster training for free," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 882–12 891.

[10] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 425–432.

[11] N. Snavely, S. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," in *SIGGRAPH 2006*, 2006.

[12] G. Chaurasia, S. Duchêne, O. Sorkine-Hornung, and G. Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *ACM Trans. Graph.*, vol. 32, pp. 30:1–30:12, 2013.

[13] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf, "Casual 3d photography," *ACM Transactions on Graphics (TOG)*, vol. 36, pp. 1 – 15, 2017.

[14] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow, "Scalable inside-out image-based rendering," *ACM Transactions on Graphics (TOG)*, vol. 35, pp. 1 – 11, 2016.

[15] I. Choi, O. Gallo, A. Troccoli, M. H. Kim, and J. Kautz, "Extreme view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7781–7790.

[16] G. Riegler and V. Koltun, "Free view synthesis," in *European Conference on Computer Vision*. Springer, 2020, pp. 623–640.

[17] ——, "Stable view synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[18] K.-A. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, and V. Lempitsky, "Neural point-based graphics," in *European Conference on Computer Vision*. Springer, 2020, pp. 696–712.

[19] S. Lombardi, J. Saragih, T. Simon, and Y. Sheikh, "Deep appearance models for face rendering," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–13, 2018.

[20] J. Thies, M. Zollhöfer, and M. Nießner, "Deferred neural rendering: Image synthesis using neural textures," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[21] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman, "Multiview neural surface reconstruction by disentangling geometry and appearance," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2492–2502, 2020.

[22] M. Niemeyer, L. M. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3501–3512, 2020.

[23] P. Kellnhofer, L. C. Jebe, A. Jones, R. Spicer, K. Pulli, and G. Wetzstein, "Neural lumigraph rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4287–4297.

[24] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[25] M. Levoy and P. Hanrahan, "Light field rendering," *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[26] R. Szeliski and P. Golland, "Stereo matching with transparency and matting," in *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE, 1998, pp. 517–524.

[27] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," in *SIGGRAPH*, 2018.

[28] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely, "Pushing the boundaries of view extrapolation with multiplane images," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 175–184, 2019.

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

[30] Z. Li, W. Xian, A. Davis, and N. Snavely, "Crowdsampling the plenoptic function," in *ECCV*, 2020.

[31] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1510–1519, 2017.

[32] A. Chen, M. Wu, Y. Zhang, N. Li, J. Lu, S. Gao, and J. Yu, "Deep surface light fields," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 1, pp. 1–17, 2018.

[33] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[34] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6498–6508.

[35] W. Xian, J.-B. Huang, J. Kopf, and C. Kim, "Space-time neural irradiance fields for free-viewpoint video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9421–9431.

[36] T. Neff, P. Stadlbauer, M. Parger, A. Kurz, J. H. Mueller, C. R. A. Chaitanya, A. S. Kaplanyan, and M. Steinberger, "DONeRF: Towards real-time rendering of compact neural radiance fields using depth oracle networks," *Computer Graphics Forum*, vol. 40, no. 4, 2021.

[37] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-NeRF: Neural radiance fields for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 318–10 327.

[38] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhofer, C. Lassner, and C. Theobalt, "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 959–12 970.

[39] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864.

[40] D. Verbin, P. Hedman, B. Mildenhall, T. Zickler, J. T. Barron, and P. P. Srinivasan, "Ref-nerf: Structured view-dependent appearance for neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5491–5500.

[41] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "Graf: Generative radiance fields for 3d-aware image synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[42] Q. Wang, Z. Wang, K. Genova, P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, "IBRNet: Learning multi-view image-based rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4690–4699.

[43] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 124–14 133.

[44] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-nerf: Point-based neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5438–5448.

[45] X. Zhang, S. Bi, K. Sunkavalli, H. Su, and Z. Xu, "Nerfusion: Fusing radiance fields for large-scale scene reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5449–5458.

[46] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin, "FastNeRF: High-fidelity neural rendering at 200fps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 346–14 355.

[47] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.

[48] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking neural radiance fields for real-time view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5875–5884.

[49] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 335–14 345.

[50] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127

[51] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *European Conference on Computer Vision (ECCV)*, 2022.

[52] C. Sun, M. Sun, and H.-T. Chen, "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5459–5469.

[53] S. Bi, Z. Xu, P. P. Srinivasan, B. Mildenhall, K. Sunkavalli, M. Havsan, Y. Hold-Geoffroy, D. Kriegman, and R. Ramamoorthi, "Neural reflectance fields for appearance acquisition," *ArXiv*, vol. abs/2008.03824, 2020.

[54] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch, "NeRD: Neural reflectance decomposition from image collections," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 684–12 694.

[55] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, "NeRV: Neural reflectance and visibility fields for relighting and view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7495–7504.

[56] S. Liu, X. Zhang, Z. Zhang, R. Zhang, J.-Y. Zhu, and B. Russell, "Editing conditional radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5773–5783.

[57] Y.-J. Yuan, Y.-T. Sun, Y.-K. Lai, Y. Ma, R. Jia, and L. Gao, "Nerf-editing: geometry editing of neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 353–18 364.

[58] Y.-H. Huang, Y. He, Y.-J. Yuan, Y.-K. Lai, and L. Gao, "StylizedNeRF: consistent 3D scene stylization as stylized NeRF via 2D-3D mutual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 342–18 352.

[59] Chong Bao and Bangbang Yang, Z. Junyi, B. Hujun, Z. Yinda, C. Zhaopeng, and Z. Guofeng, "NeuMesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing," in *European Conference on Computer Vision (ECCV)*, 2022.

[60] T. Xu and T. Harada, "Deforming radiance fields with cages," in *ECCV*, 2022.

[61] S. Peng, Y. Zhang, Y. Xu, Q. Wang, Q. Shuai, H. Bao, and X. Zhou, "Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9054–9063.

[62] A. Noguchi, X. Sun, S. Lin, and T. Harada, "Neural articulated radiance field," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5762–5772.

[63] J. Zhang, X. Liu, X. Ye, F. Zhao, Y. Zhang, M. Wu, Y. Zhang, L. Xu, and J. Yu, "Editable free-viewpoint video using a layered neural representation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–18, 2021.

[64] G. Gafni, J. Thies, M. Zollhofer, and M. Nießner, "Dynamic neural radiance fields for monocular 4D facial avatar reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8649–8658.

[65] A. Raj, M. Zollhofer, T. Simon, J. Saragih, S. Saito, J. Hays, and S. Lombardi, "Pixel-aligned volumetric avatars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 733–11 742.

[66] F. Dellaert and L. Yen-Chen, "Neural volume rendering: NeRF and beyond," 2021.

[67] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2304–2314, 2019.

[68] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[69] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural radiance fields from one or few images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4578–4587.

[70] A. Jain, M. Tancik, and P. Abbeel, "Putting NeRF on a diet: Semantically consistent few-shot view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5885–5894.

[71] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[72] B. Attal, E. Laidlaw, A. Gokaslan, C. Kim, C. Richardt, J. Tompkin, and M. O'Toole, "Törf: Time-of-flight radiance fields for dynamic scene view synthesis," *Advances in neural information processing systems*, vol. 34, 2021.

[73] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. Sajjadi, A. Geiger, and N. Radwan, "Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5480–5490.

[74] M. Shi, J.-Q. Zhang, S.-Y. Chen, L. Gao, Y.-K. Lai, and F. Zhang, "Deep line art video colorization with a few references," *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[75] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *NIPS*, 2016.

[76] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017.

[77] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[78] Y. Wang, A. Gonzalez-Garcia, D. Berga, L. Herranz, F. Khan, and J. van de Weijer, "Minegan: Effective knowledge transfer from gans to target domains with few images," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9329–9338, 2020.

[79] Y. Li, R. Zhang, J. C. Lu, and E. Shechtman, "Few-shot image generation with elastic weight consolidation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 885–15 896, 2020.

[80] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han, "Differentiable augmentation for data-efficient gan training," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7559–7570, 2020.

[81] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 104–12 114, 2020.

[82] P. Anqi, C. Xin, L. Haimin, W. Minye, Y. Jingyi, and X. Lan, "Few-shot neural human performance rendering from sparse rgbd videos," in *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021.

[83] U. Ojha, Y. Li, J. Lu, A. A. Efros, Y. J. Lee, E. Shechtman, and R. Zhang, "Few-shot image generation via cross-domain correspondence," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 743–10 752.

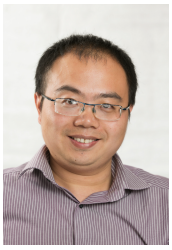
[84] Q. Meng, A. Chen, H. Luo, M. Wu, H. Su, L. Xu, X. He, and J. Yu, "Gnerf: Gan-based neural radiance field without posed camera," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6351–6361.

[85] X. Li, Y. Dong, P. Peers, and X. Tong, "Modeling surface appearance from a single photograph using self-augmented convolutional neural networks," *ACM Transactions on Graphics (TOG)*, vol. 36, pp. 1 – 11, 2017.

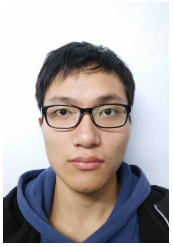
[86] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[87] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.
[88] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, pp. 1–13, 2013.
[89] S. Laine and T. Karras, "Efficient sparse voxel octrees–analysis, extensions, and implementation," *NVIDIA Corporation*, vol. 2, 2010.
[90] E. Haines, "Essential ray tracing algorithms," *An introduction to ray tracing*, pp. 33–77, 1989.
[91] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
[92] A. Trevithick and B. Yang, "GRF: Learning a general radiance field for 3D representation and rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 182–15 192.
[93] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.
[94] Z. Wang, A. Bovik, H. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, 2004.
[95] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.
[96] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and improving neural radiance fields," *ArXiv*, vol. abs/2010.07492, 2020.
[97] Y. Wei, S. Liu, Y. Rao, W. Zhao, J. Lu, and J. Zhou, "Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5610–5619.
[98] R. Maier, K. Kim, D. Cremers, J. Kautz, and M. Nießner, "Intrinsic3d: High-quality 3d reconstruction by joint appearance and geometry optimization with spatially-varying lighting," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3114–3122.
[99] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan, "Blendedmvs: A large-scale dataset for generalized multi-view stereo networks," *Computer Vision and Pattern Recognition (CVPR)*, 2020.

Yu-Jie Yuan received the bachelor's degree in mathematics from Xi'an Jiaotong University in 2018. He is currently a Ph.D. candidate in the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer graphics and neural rendering.



Yu-Kun Lai received his bachelor's degree and PhD degree in computer science from Tsinghua University in 2003 and 2008, respectively. He is currently a Professor in the School of Computer Science & Informatics, Cardiff University. His research interests include computer graphics, geometry processing, image processing and computer vision. He is on the editorial boards of *Computer Graphics Forum* and *The Visual Computer*.



Yi-Hua Huang obtained his bachelor degree from the University of Chinese Academy of Sciences. He is currently a graduate candidate in the Institute of Computation Technology, Chinese Academy of Sciences. His research interests include computer graphics and visions.

Leif Kobbelt is a full professor and the head of the Computer Graphics Group at the RWTH Aachen University, Germany. His research interests include all areas of Computer Graphics and Geometry Processing with a focus on multi resolution and free-form modeling as well as the efficient handling of polygonal mesh data. He received his master's degree in 1992 and Ph.D. in 1994 from the University of Karlsruhe, Germany.

Lin Gao received the bachelor's degree in mathematics from Sichuan University and the PhD degree in computer science from Tsinghua University. He is currently an Associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences. He has been awarded the Newton Advanced Fellowship from the Royal Society and the Asia Graphics Association young researcher award. His research interests include computer graphics and geometric processing.