
NeRF SYNTHESIS WITH SHADING GUIDANCE

Chenbin Li, Yu Xin, Gaoyi Liu, Xiang Zeng, Ligang Liu

University of Science and Technology of China

Hefei

China

{lichenbin, xy0731, Zengx56, liugaoysi}@mail.ustc.edu.cn

lgliu@ustc.edu.cn

ABSTRACT

The emerging Neural Radiance Field (NeRF) shows great potential in representing 3D scenes, which can render photo-realistic images from novel view with only sparse views given. However, utilizing NeRF to reconstruct real-world scenes requires images from different viewpoints, which limits its practical application. This problem can be even more pronounced for large scenes. In this paper, we introduce a new task called NeRF synthesis that utilizes the structural content of a NeRF patch exemplar to construct a new radiance field of large size. We propose a two-phase method for synthesizing new scenes that are continuous in geometry and appearance. We also propose a boundary constraint method to synthesize scenes of arbitrary size without artifacts. Specifically, we control the lighting effects of synthesized scenes using shading guidance instead of decoupling the scene. We have demonstrated that our method can generate high-quality results with consistent geometry and appearance, even for scenes with complex lighting. We can also synthesize new scenes on curved surface with arbitrary lighting effects, which enhances the practicality of our proposed NeRF synthesis approach.

Keywords NeRF, 3D Scene Synthesis, Texture Synthesis, Relighting

1 Introduction

Physically-based rendering (PBR) plays a critical role in a wide range of computer graphics applications, including virtual reality, augmented reality, and video games. However, it struggles to accurately represent real-world scenes that involve complex geometry and textures, such as fine hair and grass. Recently, NeRF [1] demonstrated a state-of-the-art approach to novel view synthesis using an implicit volumetric field that outputs density and radiance values. By training the field on a sparse set of input views, NeRF can efficiently generate photo-realistic images from any novel view using a ray-marching technique. Recent research has shown that NeRF is capable of high-fidelity modeling of complex natural scenes [2] and can be applied to large-scale scene modeling [3, 4].

While NeRF is adept at handling complex scenes, capturing multiple views for large-scale scenes can present challenges due to the large number of views that need to be captured. Although recent efforts have been made to improve the efficiency of NeRF training [5, 6], it remains time- and memory-intensive for large-scale scenes. The question arises: is it truly necessary to capture all real-world objects in a scene through cameras and model them by NeRFs in order to create virtual models of real scenes? For objects such as fur, lawns, and clouds, which possess clear repetitive structures. Modeling them on a large scale can be tedious and laborious. We observe that these objects are isotropic in the plane and are well-suited for block-by-block generation. Inspired by texture synthesis [7], we propose NeRF synthesis, which is the process of algorithmically constructing a large radiance field from a smaller exemplar by making use of its structural content.

This task presents several challenges. Firstly, unlike 2D images, NeRF is a function of viewpoint, which requires the synthesized results are rendered consistently from every viewpoint. Secondly, NeRF couples lighting information with material information, posing a significant challenge in ensuring plausible lighting in the synthesized results. A common method to achieve plausible lighting is to completely decouple the scene [8, 9, 10], obtaining information

such as geometry, material, and lighting, and then using traditional rendering pipelines to relight the scene. However, decoupling scenes with complex geometric details, particularly those with grass, fur, etc., poses a significant challenge. Similary, BTF [11] depends on lighting and viewpoints. In contrast to NeRF, the rendering result of BTF from different viewpoints and lighting conditions can be aligned to the same 2D image, enabling the extraction of features that incorporate both the viewpoint and lighting. However, since NeRF uses volume rendering to obtain images from different viewpoints, these images do not have a corresponding relationship at the same position, making it impossible to directly apply BTF synthesis method to NeRF synthesis.

To address the aforementioned issues, first, we utilize shading vector to control the lighting of the synthesized scene. The shading vector refers to the shading value of a voxel column in the radiance field at M different viewpoints. By comparing the distance between two shading vectors, we can quantify their lighting differences. All shading vectors in a scene together form a shading map. To relight the scene, the shading vector of the exemplar is obtained by intrinsic decomposition-based method, and then a ray-tracing-based method is proposed to convert lighting information into a shading map guider. With the aid of the shading map and the shading map guider, we can guide the synthesis process by selecting patches with the correct lighting information. Second, to ensure both geometric and appearance continuity in the synthesized results, especially when dealing with complex texture and geometry, we propose a two-phase method. For the boundaries of the scene, we introduce a boundary-constrained synthesis method to prevent artifacts at the boundaries when synthesizing large scenes.

To summarize, this paper makes the following contributions:

- Proposing a two-phase method to ensure both geometry and appearance continuity, and a boundary-constrained synthesis approach to avoid artifacts resulting from obscured parts of the radiance field.
- Proposing a shading-guided synthesis method to synthesize large-scale scene with reasonable lighting. This method allows for relighting scenes without the need for decoupling the scene.

2 Related work

2.1 Planar texture synthesis

The texture is a kind of pattern with repetition, such repetition does not need to be spatially periodic, but there is some similarity between any two regions. This special repetition makes it possible to synthesize large-scale textures from a local texture sample.

Methods of texture synthesis can be sorted into two categories: parametric methods and non-parametric methods. Parametric methods treat texture as a parametric math model. By adjusting the pre-defined parameters of a pure-noise image and matching them with those of the input texture sample, an output texture with similar parameters to the input texture will be produced. In many cases, these parameters are related to some statistics of features of the input texture, such as histograms of frequency bands, wavelet coefficients, or responses to a set of filters [12, 13, 14].

Correspondingly, non-parametric methods no longer care about those statistics of texture and establish some rules for directly constructing the output texture. One popular approach is modeling texture as a Markov Random Field (MRF) [15], which assumes that local patterns of texture have a stationary distribution. This allows us to infer the pattern in blank areas of output according to their synthesized neighborhood and induces a family of schematic methods called exemplar-based methods. These methods move one part from the input texture to output texture per step and try to maintain the stationary distribution for every move.

One branch of the exemplar-based method is the pixel-based method, which moves one pixel per step [15, 16, 17, 18]. The pixel from the input texture whose neighborhood has a minimum difference from that of the on-processing pixel from the output texture will be chosen. Another branch of the exemplar-based method is the patch-based method [19, 20, 21, 22], which moves a patch from the exemplar per step according to neighborhood-comparing criteria similar to that adopted by pixel-based method. Patch-based methods can be generally more efficient than pixel-based methods, and perform better in preserving large-scale features.

While the above works performed neighborhood comparisons on color channel information, Tong et al. [23] introduced a concept of ‘texton’ that integrates multi-orientation-lighting information together with responses of filters. By searching neighborhoods in texton space instead of pixel space, they successfully synthesized consistent BTF textures for arbitrary manifold surfaces by the exemplar-based method. This inspires the notion that traditional methods possess the capability to solve high-dimensional texture synthesis.

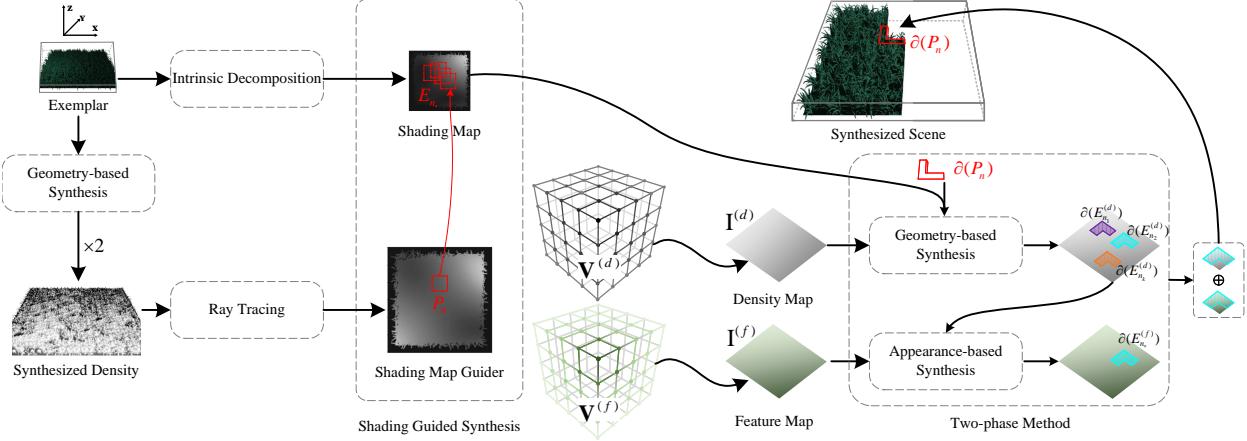


Figure 1: Pipeline of our proposed method. A density field ($\mathbf{V}^{(d)}$) and a color feature field ($\mathbf{V}^{(f)}$) are provided to represent the exemplar. The shading map extracted from the exemplar and shading map guider are also provided for synthesis with lighting constrained. For the patch P_n to be synthesized, shading guided synthesis first finds the closest k_s patches by the shading distance between P_n and patches in the exemplar. Then, the two-phase method is applied to these patches to filter the patches with high response values in terms of density and color. In the first phase, k_g candidate patches with geometric continuity are found based on $\mathbf{V}^{(d)}$. In the second phase, one patch is selected for color continuity in the overlap area based on $\mathbf{V}^{(f)}$. The found patch’s density field and radiance field are stitched onto the target area to complete one step of NeRF synthesis.

2.2 Geometric texture synthesis

The concept of texture also applies to three-dimensional (3D) objects. 3D objects with repetitive geometric patterns can be spotted everywhere. From the surfaces of natural objects like vegetation, terrain, waves, and fur, or from artificial objects, such as knitting, bricks wall, and buildings. Textures like fur and hair consist of rather complex surfaces. The volume integrates their average surface properties and color information that should be a suitable representation for this kind of textures [24, 25]. Other textures like terrain and wave have determined boundaries, which can be precisely represented by meshes. Zhou et al. [26] extended the patch-based method to handle mesh texture synthesis tasks and employed mesh deformation techniques to blend overlapping areas. Their results can be seamlessly applied to manifold surfaces, resulting in objects with intricate surface geometries.

2.3 NeRF-based generative models

Due to the similarity between NeRF and images in representation, successful generation techniques on images can also be applied to NeRF. For example, one can directly treat the radiance field as the parameter to optimize and define an adversarial loss based on images obtained through volumetric rendering and real images to train a generative adversarial network (GAN) [27]. This enables the learning of GAN-based NeRF generation models [28, 29, 30]. Alternatively, one can perform denoising of a probabilistic diffusion process applied to 3D radiance fields to learn a diffusion-based NeRF generation models in 3D space [31, 32]. However, these methods differ from NeRF synthesis in several key aspects. Firstly, these methods generate objects within the same distribution as the original data, limiting their ability to create objects of varying sizes. Secondly, these generative methods often require large amounts of training data, whereas NeRF synthesis only requires a single exemplar. While current works such as [33, 34] focus on scene generation from a single example represented by NeRF, our work focuses on synthesizing tiled scenes such as grass and carpets, achieving continuity in geometry and texture while maintaining control over lighting.

3 Preliminaries

3.1 Direct voxel grid optimization

Direct voxel grid optimization (DVGO) [35] is an explicit-implicit hybrid representation, which uses a voxel-grid model to represent a 3D scene for novel view synthesis. As shown in Fig. 1, the implicit modalities are stored explicitly in

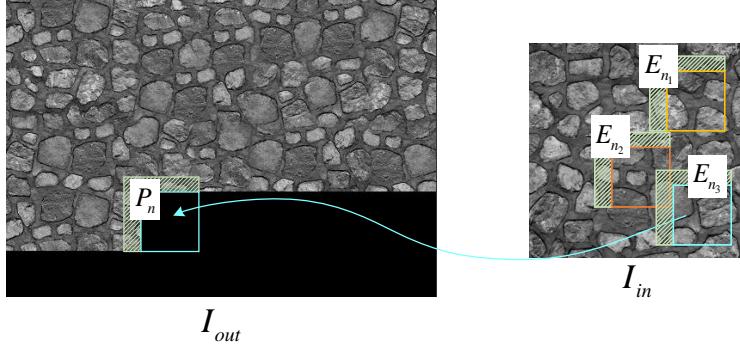


Figure 2: I_{in} is an example texture image. A new, larger texture image I_{out} is waiting to be synthesized based on I_{in} . A patch P_n to be synthesized is matched with E_{n_i} ($i = 1, 2, 3$), and E_{n_3} is selected to synthesize the texture patch P_n . It should be emphasized that patches in various locations may have different types of overlap. This figure only illustrates the most common case.

two voxel grids: a density grid $\mathbf{V}^{(d)}$ and a color feature grid $\mathbf{V}^{(f)}$. For a pixel-rendering ray r , DVGO samples n points between the pre-defined near and far planes and calculates the feature of each sampled point x_i using trilinear interpolation:

$$interp(x_i, \mathbf{V}) : (\mathbb{R}^3, \mathbb{R}^{C \times N_x \times N_y \times N_z}) \rightarrow \mathbb{R}^C. \quad (1)$$

DVGO uses the shifted softplus mentioned in Mip-NeRF [36] as density activation (*i.e.*, a mapping from \mathbb{R} to $\mathbb{R}_{\geq 0}$):

$$\sigma_i = softplus(\tilde{\sigma}_i) = \log(1 + \exp(\tilde{\sigma}_i + b)), \quad (2)$$

where $\tilde{\sigma}_i$ represents the value stored in $\mathbf{V}^{(d)}$, and the shift b is a hyperparameter. The color feature will be mapped to a view-dependent color emission \mathbf{c}_i by a shallow multilayer neural network parameterized by Θ :

$$\mathbf{c}_i = MLP_{\Theta}^{(rgb)}(interp(x_i, \mathbf{V}^{(f)}), x_i, d), \quad (3)$$

where d represents the direction of the ray r . Positional encoding is applied to encode x_i and d for the $MLP_{\Theta}^{(rgb)}$. Finally, the n results $\{\sigma_i, \mathbf{c}_i\}_{i=1}^n$ are accumulated into a single color $\hat{C}(\mathbf{r})$ using the volume rendering quadrature:

$$\hat{C}(\mathbf{r}) = \left(\sum_{i=1}^n T_i \alpha_i \mathbf{c}_i \right) + T_{n+1} \mathbf{c}_{bg}, \quad (4)$$

$$\alpha_i = alpha(\sigma_i, \delta_i) = 1 - \exp(-\sigma_i \delta_i), \quad (5)$$

$$T_i = \prod_{j=1}^{i-1} 1 - \alpha_j, \quad (6)$$

where α_i represents the probability of termination at the point x_i ; T_i denotes the accumulated transmittance from the near plane to point x_i ; δ_i indicates the distance to the adjacent sampled point, and \mathbf{c}_{bg} represents a pre-defined background color. In addition, DVGO shows that the post-activation for density: $\alpha^{(post)} = alpha(softplus(interp(x, \mathbf{V}^{(d)})))$, *i.e.* applying all the non-linear activation functions after the trilinear interpolation, is capable of producing sharp surfaces, thus requiring fewer cells for surface detail.

3.2 Patch-based texture synthesis

As shown in Fig. 2, to synthesize a new texture patch P_n of new texture image I_{out} from the input texture image I_{in} , the patch-based algorithm pastes an example patch E_n of the input sample texture I_{in} into the synthesized texture I_{out} step by step. The selection of E_n is based on the patches already pasted in I_{out} to avoid mismatching features across patch boundaries. Specifically, we say that P_n and E_n match if

$$d(\partial(P_n), \partial(E_n)) < \eta, \quad (7)$$

where $\partial(\cdot)$ represents the boundary zone of a texture patch, and $d(\cdot, \cdot)$ represents the L2 distance between two texture patches. Here, η is a prescribed constant, and the overlap size is a pre-defined constant. All the matched example patches are directly searched in all example patches in I_{in} that have the known $\partial(P_n)$ as their boundary zones.

Liang et al. [37] firstly used the overlap region to initial optimized kd-trees for accelerating the approximate nearest neighbors (ANN) search. In each step, the ANN search is used to find the k most similar patches $\{E_{n_i}\}_{i=1}^k$. Then, a probability density function (PDF) is computed based on $d(\partial(P_n), \partial(E_{n_i}))$ for $i = 1$ to k , and a patch is randomly chosen according to the computed PDF. Finally, the chosen patch is pasted into the synthesized texture I_{out} , and the overlap area is blended to provide a smooth transition between adjacent texture patches.

4 Method

NeRF uses an implicit method to represent the radiance field by an MLP, which is not suitable be direct used in patch-based synthesis. Therefore, we utilize DVGO, which is an explicit-implicit hybrid representation. Since our method is not limited to DVGO, we continue to refer to it as NeRF synthesis. For both the density grid $\mathbf{V}^{(d)}$ and the feature grid $\mathbf{V}^{(f)}$ of DVGO, we consider them as 2D grids. This is because only two dimensions exhibit clear repeatability, even we want to synthesize a three-dimensional object, as shown in Fig 1. Taking grass as an example, it would be inefficient and unnecessary to synthesize the roots of the grass first and then search for matching leaves. Treating grass as a unified entity would be more reasonable. Therefore, we flatten the 3D grid into a 2D grid along the Z-axis, resulting in $\mathbf{I}^{(d)}$ and $\mathbf{I}^{(f)}$. Consequently, we can directly employ 2D texture synthesis methods for 3D NeRF synthesis. The only distinction is that, in NeRF synthesis, each unit stores density and feature values instead of RGB.

4.1 Two-phase method

Unlike 2D texture synthesis, NeRF synthesis requires ensuring both geometry and appearance continuity simultaneously. Therefore, it is necessary to consider them together during synthesis. We directly concatenate the two grids ($\mathbf{V}^{(d)}$ and $\mathbf{V}^{(f)}$) and treat them as a combined 2D image, denoted as $\mathbf{I}^{(d \oplus f)}$. The distance between different patches is measured using the L2 distance. The patch-based synthesis method is applied directly on this grid to obtain the synthesized results. We take this method as a baseline. However, a significant challenge arises in determining the weighting between density and color. Different scenes often require different suitable weights. As depicted in Fig. 7, using a higher weight for density leads to more pronounced color inconsistencies in the synthesized results, whereas employing a higher weight for color features results in more severe geometric inconsistencies. Additionally, in order to capture viewpoint-dependent color variations, $\mathbf{I}^{(d \oplus f)}$ tends to have high dimensionality, leading to increased search time during the process.

To address the aforementioned challenges, a two-phase method is proposed, as illustrated in Fig 1, including geometry-based synthesis and appearance-based synthesis. In the first phase, for a voxel patch P_n to be synthesized, the k_g best density voxel patches $E_{n_i}^{(d)} (i = 1, 2, \dots, k_g)$ are selected based on the similarity of $d(\partial(P^{(d)}), \partial(E^{(d)}))$. The assumption is that due to their relatively high similarity in the overlapping region of the density field, these patches also exhibit relatively high geometric continuity at the position to be synthesized. To ensure the continuity of appearance in the synthesized results, the similarity of $d(\partial(P_n^{(f)}), \partial(E_{n_i}^{(f)})) (i = 1, 2, \dots, k_g)$ is calculated for the selected k_g best patches. These similarities are then utilized as the likelihood of each example patch being selected during the random selection process, ensuring randomness in the synthesis procedure. Finally, the overlapping regions of the selected patch E_{n_*} and P_n are blended using inverse distance weighting.

Why prioritize density over color in the synthesis process? The synthesis process operates at the voxel column level, where the color feature inherently incorporates geometric information to some extent. By prioritizing density first, we ensure the continuity of geometric, while subsequent considerations of appearance are less affected by density.

4.2 Boundary-constrained synthesis method

As shown in Fig. 8, the two-phase method faces challenges when handling synthesis on the boundary. This is because the occluded interior parts are synthesized onto the boundary. Due to the lack of pixel value supervision for these occluded parts during the DVGO training process, they exhibit artifacts.

To address this issue, as shown in Fig. 3, we divide the entire radiance field into five parts: the upper boundary \mathbb{B}_u , the lower boundary \mathbb{B}_d , the left boundary \mathbb{B}_l , the right boundary \mathbb{B}_r , and the interior area \mathbb{I} . The length of each boundary is

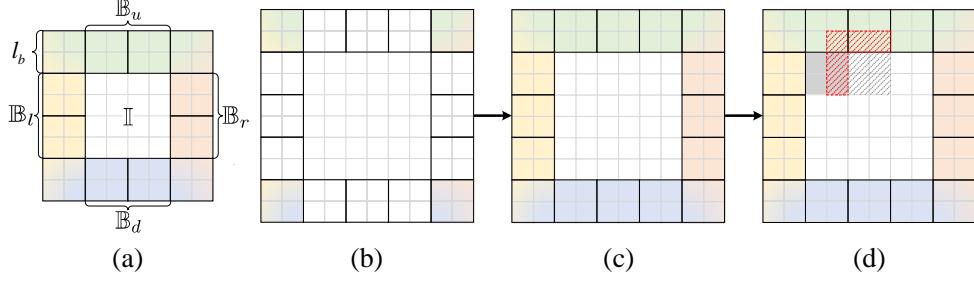


Figure 3: (a) Original exemplars for synthesis. (b) Four corner patches are set to initial the canvas. (c) The four boundaries are synthesized using the boundary-constrained synthesis method. The patches in I_{out} with color come from from the corresponding patches in I_{in} . (d) The boundary serves as a constraint for further synthesizing the inner region. The gray area represents the synthesized part. The gray slash part is the area to be synthesized. The white part indicates the unsynthesized area. The red slash represents the overlap area of the regions to be synthesized.

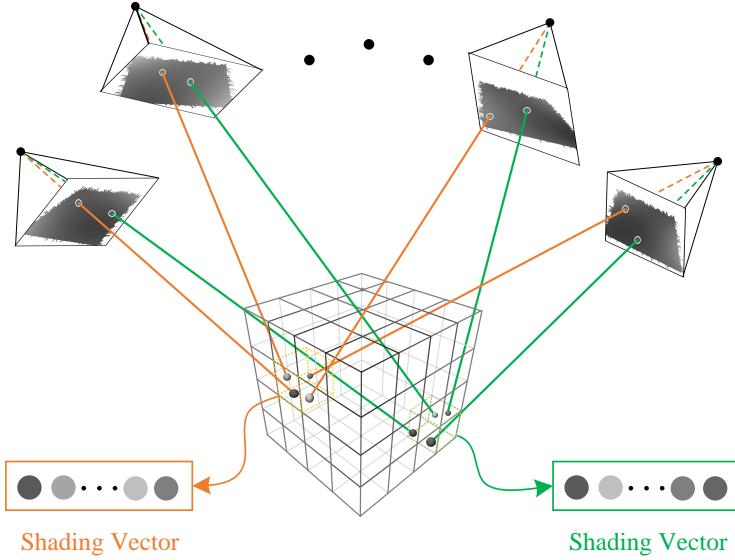


Figure 4: Shading vector from intrinsic decomposition.

denoted as l_b . Initially, we place the four corner patches from the original field directly onto the corresponding corners of the synthesized result. These corner patches are essential as they have boundaries in two directions and cannot be replaced. Next, we synthesize each of the four boundaries in the output by utilizing the corresponding set of boundary patches from the input samples. This synthesis process can be viewed as synthesizing individual strips. Finally, we employ the synthesized boundaries as constraints for synthesizing the interior region. We accomplish this by utilizing example patches from the interior area \mathbb{I} . It is important to note that the physical boundaries of the scene may span a certain length in the field. Therefore, it is necessary to choose an appropriate length to handle the boundaries separately. For the sake of clarity, we assume that the lengths of the four boundaries are set to the same parameters.

4.3 Shading guided NeRF synthesis

Under natural lighting conditions, the method described above can easily synthesize new scenes of any size. However, when the scene lighting becomes more complex, relying solely on feature grid matching without explicit lighting guidance leads to discontinuous shading changes in the synthesized scenes, as depicted in Fig. 9. To address this issue, we introduce the shading vector, which describes the lighting information of a basic unit in the scene. This vector represents the shading values of a point in the scene from different viewpoints. By incorporating the shading vector, we can ensure continuous lighting in each area of the synthesized scene, resulting in a visually coherent lighting across the entire scene.

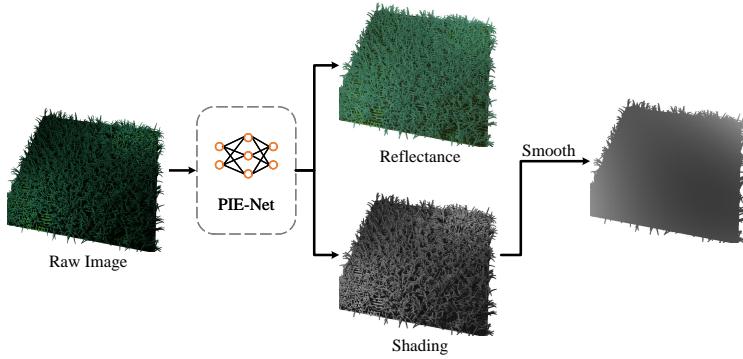


Figure 5: Preprocessing of the shading image obtained from the PIE-Net.

To incorporate the shading vector into the NeRF synthesis task, we introduce two simplifications. Firstly, since the voxel column serves as the fundamental unit of NeRF synthesis, we simplify the shading vector to represent the shading values of a voxel column from different viewpoints. Secondly, accurately fitting the shading values of a point from numerous viewpoints requires substantial computational resources. Therefore, we only approximate shading vector by sampling from M viewpoints. This approximation is reasonable as the primary role of the shading vector is to differentiate the brightness information among voxel columns. Through experimentation, we have determined that utilizing 50 viewpoints is sufficient for most scenes. By collecting shading information of a voxel column from multiple viewpoints within the scene, we can utilize the shading maps of the source and target scenes to synthesize a new scene that effectively captures the desired lighting effect.

To acquire the shading vectors, we project the shading value of each pixel in the training images onto the voxel grid based on depth. In this study, we utilize PIE-Net [38] as the method for intrinsic decomposition to derive the shading and albedo of the images. However, as shown in Fig. 5, the shading obtained from PIE-Net still retains geometric information, which can interfere with the matching of shading vectors. To tackle this challenge, we apply Gaussian blurring and polynomial fitting techniques to smooth the shading image. Subsequently, we project the shading from each pixel in the smoothed shading image along the camera ray to the corresponding depth position. The depth value is obtained from the density grid using the following formula:

$$d = \sum_k T_i (1 - \exp(-\sigma_i \delta_i)) t_i. \quad (8)$$

The shading values of each voxel column in the scene from multiple viewpoints can be obtained by projecting the 2D shading information from M views into 3D voxel grid. The vector formed by these values in sequential order is referred to as the shading vector, which encodes the lighting information of a voxel column. However, due to occlusions between voxels and the limited number of pixels in an image, certain voxel columns may lack shading in certain viewpoints. As a result, holes can appear in each channel image of the shading vector. And inaccurate depth values can introduce noisy points in the channel image. To address this issue, we employ the k-nearest neighbors (KNN) method to interpolate these holes, and we apply inverse distance weighting to weight the nearby points. Additionally, for noisy points, we use median filtering to repair the channel images of the shading map. If a particular channel exhibits a high number of missing shading values, the interpolation error will increase. Therefore, we only retained the n_c channels with the lowest rate of missing values.

Extracting the shading vector of a scene using intrinsic decomposition requires images from multiple views of the scene. However, this poses a limitation as it becomes impossible to synthesize a new scene with arbitrary lighting effects if multiple views are not available. To overcome this constraint, we propose a ray-tracing-based approach to obtain the shading vector in radiance field. This approach allows us to determine the shading information of the scene for any lighting condition by accurately tracking the path of light rays.

Specifically, the problem can be formulated as determining the shading value for each position in the scene from n_c viewpoints, using the NeRF representation of the scene and information regarding the light source. The shading of a point is determined by both the intensity of its reflected light and the probability of capturing its emitted light with the camera. Additionally, the brightness of a point is influenced by the brightness of the light source and the probability of the emitted light from the source reaching the point.

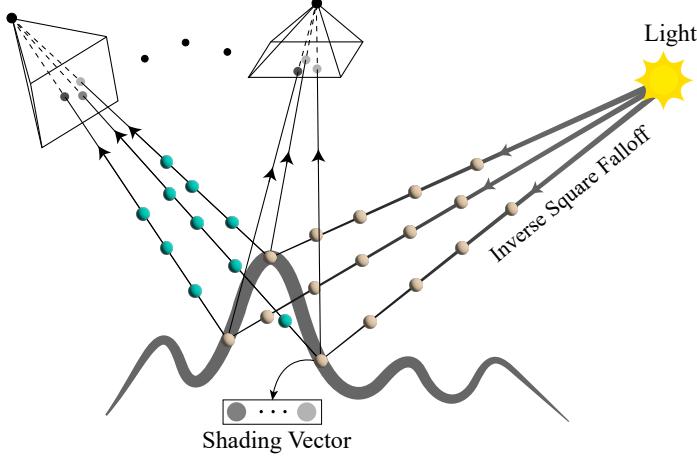


Figure 6: Shading vector from ray tracing.

Therefore, we begin by selecting n_p points from the scene with a high density. Subsequently, we calculate the probability of a light ray intersecting each of these points originating from the light source. To illustrate this, let's consider a point light source, as depicted in Fig. 6. For any point \mathbf{x} in space, we sample n_s points $\mathbf{x}_1, \dots, \mathbf{x}_{n_s}$ along the path between the light source \mathbf{x}_l and \mathbf{x} , where $\mathbf{x}_{n_s} = \mathbf{x}$. The brightness of a point can be represented by the following equation:

$$b = p_{n_s} \frac{1}{\|\mathbf{x} - \mathbf{x}_l\|_2^2} I. \quad (9)$$

Here, $p_{n_s} = T_{n_s}(1 - \exp(-\sigma_{n_s} \delta_{n_s}))$ represents the probability of the ray propagating to a point \mathbf{x}_{n_s} . The term $\frac{1}{\|\mathbf{x} - \mathbf{x}_l\|_2^2}$ corresponds to the light attenuation, and I denotes the intensity of the light source. Consequently, we can determine the shading values for these n_s points in the scene. Additionally, we leverage the reversibility of light paths to convert the probability of a point's brightness reaching the camera into the probability q_{n_s} of a camera ray intersecting with that point. By employing this method, we can calculate the shading of \mathbf{x} from a specific viewpoint using the following equation:

$$s = q_{n_s} b, \quad (10)$$

where $q_{n_s} = T_{n_s}(1 - \exp(-\sigma_{n_s} \delta_{n_s}))$. To compute q_{n_s} , we apply the same sampling procedure from the camera origin to \mathbf{x} , as described in the previous step. We utilize the same n_c viewpoints with the lowest rate of missing values to obtain the shading vector in the specific order. The n_s sampling points are obtained by projecting pixels from these viewpoints along the camera rays. The projection depth is also determined using formula 8. To address the presence of holes in the shading map caused by occlusions and the presence of the residual of geometry details, we apply the same post-processing method as described in the previous section.

The pipeline for guiding the synthesized result with shading is shown in Fig. 1. First, the shading map of the exemplar is used to define the multi-view shading of each voxel column, while the shading map guider is used to specify the lighting information of the target scene. The shading map is obtained using the intrinsic decomposition method, as multiple images of the exemplar are available. To obtain the shading map guider, we synthesize scene using geometry-based synthesis to create a geometrically continuous large scene with larger size (without appearance-based method, we directly find the patch with the closest density at the overlap). Then, the ray-tracing-based method is employed to generate the shading map guider for the target illumination. For the patch P_n to be synthesized, the L2 distance is initially utilized to identify the k_s shading patches in the shading map that are closest to the shading of P_n , which is accelerated by the kd-tree. As the shading map and shading map guider are obtained from different methods, it is necessary to normalize the values of them to the same scale. However, the selection of patches can be limited due to the requirements of shading, geometry, and texture. To address this limitation, we rotate the exemplar by 90, 180, and 270 degrees, respectively, to obtain additional available patches. Finally, a two-phase method is employed to select patches that satisfy both the geometric and texture continuity requirements from these k_s patches.

How to attach realistic surface structures onto mesh surface has been a long-standing problem, as the quality of modeling directly affects the visual effect. With the aid of shading vector, we can synthesize scenes with arbitrary lighting effects

on 3D mesh and apply the surface structure modeled by NeRFs to achieve mixed rendering of the two representation methods.

The attachment of realistic surface structures onto a mesh surface has long been a challenging problem, as the quality of modeling directly impacts the visual outcome. By leveraging shading, we can synthesize scenes with various lighting effects on a 3D mesh and incorporate the surface structure modeled by NeRFs.

To achieving this goal, first, we synthesize a scene on a plane with the desired lighting using a specific shading map guider. Then, we deform the corresponding voxel grid to match the 3D mesh using a deformation field. To obtain the shading map guider, we utilize the deformation field to bend the synthesized large-scale $V^{(d)}$ onto the 3D mesh. Subsequently, we employ the ray-tracing-based method to compute the shading map guider on the surface. Finally, we utlize the shading map guider to synthesize a new scene with desired lighting and deform it to match the target mesh surface. The deformation field is represented as $\Psi : \mathbf{x} \rightarrow \Delta\mathbf{x}$, which maps points in the deformed space to the canonical space. To obtain the deformation field, we employ the ARAP parametric method [39] to establish the mapping between the 3D mesh and the undersurface of the scene. Furthermore, we define the corresponding direction for each vertex on the mesh and its corresponding point on the plane along the normal direction. This correspondence allows us to collect a large number of discrete corresponding points in the deformation field. We employ a neural network to fit this deformation field and train the network Ψ using the coordinates of these corresponding points as supervision information.

5 Experiments

5.1 Datasets

We evaluated the effectiveness of our method mainly on synthetic data. For each synthetic scene, we assume the volume of interest is within a unit cube, and 100 observation images fully covering the scene are collected with cameras randomly distributed on a hemisphere.

To ensure the generalization ability of our method in real-world scenes, we also collected a significant amount of real-world data and tested our approach on them. Given the need for high geometric accuracy in NeRF synthesis and the inherent complexity of real-world scenes, we employed 300 images for real-world scenes. To eliminate background interference, we manually removed the background information from each image. Additionally, for each scene, we calibrated the images using colmap [40] to determine the pose of them.

5.2 Implementation details

We choose the same hyperparameters generally for all scenes. All hyperparameters of DVGO are consistent with those in their article. In particular, we initialize the shape of grid to $160 \times 160 \times 160$. The patch width and overlap area is set to $w_p = 15$ and $w_o = 5$ respectively. To generate the template patch within the original grid, we utilized a sliding window with a step size of 3. For the patch search process, we utilized $k_s = 20$ and $k_g = 10$ to locate suitable patches. Determining the length of the boundary was performed individually for each sample, ensuring that the boundary region of the grid encompassed all the physical boundaries.

For preprocessing the shading map, we applied a 10-times Gaussian blur with a mean of 0 and a variance of 7. Additionally, we performed cubic polynomial fitting on all channels of the shading map, which was obtained using two different methods, for all scenes.

We use a fully-connected network architecture to represent the deformation fields Ψ . The input position is passed through 3 fully-connected layers with ReLU activations, each with 16 channels. We utilize the MSE loss function to estimate the loss and employ the Adam [41] optimizer to optimize the network. For the Adam optimizer, we keep all the parameters at their default values.

5.3 Ablation

The exemplar and the results obtained from different synthesis methods are depicted in Fig. 7. It demonstrates that our proposed two-phase method effectively achieves a balance between appearance and geometric continuity. Furthermore, the two-phase method significantly enhances the synthesis speed by more than 10 times compared to the baseline method, as demonstrated in Table 1, since we only need to search patches for density in the first phase.

Fig. 8 illustrates that artifacts are generated at the boundary using the two-phase method. However, by incorporating the boundary constraint, we can obtain superior results at the boundary.

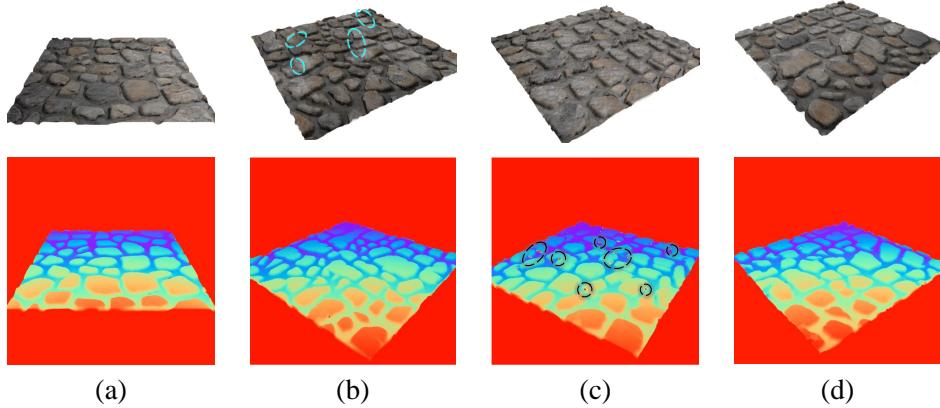


Figure 7: (a) Exemplar: rock [42]. RGB and depth images are shown. (b) Synthesis based on higher density weights. Clear color discontinuities can be observed in the RGB image (blue circles). (c) Synthesis based on higher feature weights. Geometric discontinuities can be observed in the depth image (black circles). (d) Results of the two-phase method. Both appearance and geometry exhibit continuity similar to the exemplar.

Table 1: Comparison of the two-phase method with the baseline in terms of speed when synthesizing results of different sizes. The experiment does not include the part of shading guided synthesis.

Size	100^2	200^2	300^2	400^2
Baseline (s)	163.47	270.70	454.41	714.64
Two-phase (s)	14.99	23.62	38.46	59.57
Times	10.90	11.46	11.82	12.00

To demonstrate the effectiveness of the shading vector, we performed an averaging operation on all channels of the shading vector, removing the luminance distribution of the shading vector across multiple views. We then used only the average value to describe the lighting information of a voxel. Fig. 9 shows that the average shading alone does not ensure luminance continuity in the synthesized scene from each viewpoint, whereas the shading vector produces better results.

5.4 Results

There are two approaches to synthesize new scenes with continuous illumination effects, differing only in the choice of the shading map guider.

The first approach obtains the shading map guider using a ray-tracing-based method. This method allows for the generation of new scenes with different lighting effects and varying sizes compared to the exemplar, as depicted in Fig. 10 and Fig. 11.

To synthesize a new scene with specified lighting effects on a curved surface, we synthesized a new shading map guider on a sphere using the ray-tracing-based method and then employed it on a plane to guide the synthesis of a large-scale

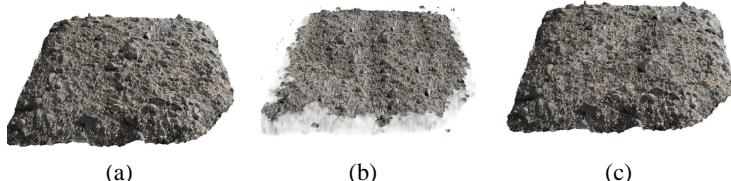


Figure 8: (a) Exemplar: ground [43]. (b) Synthesized result without boundary constraint: severe artifacts are observed at the boundaries. (c) Boundary-constrained synthesis method: the synthesized result is nearly indistinguishable from the exemplar.



Figure 9: Synthesized results of a grass [44] with complex lighting. Such scenes cannot be handled without shading vectors or with mean shading vectors. The shading vector can be used to synthesize results with continuous illumination. The shading map guider used in this experiment is obtained by doubling the size of the shading map from the exemplar.

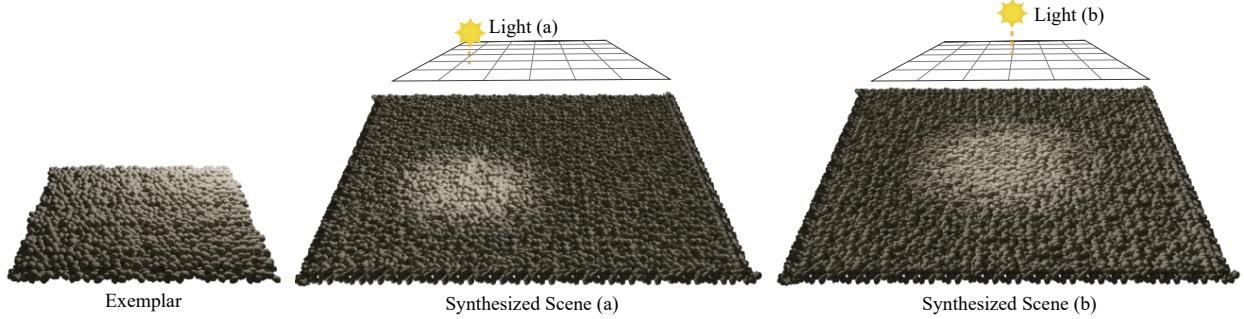


Figure 10: Relighting the scene of pebbles [45] using different lighting conditions.

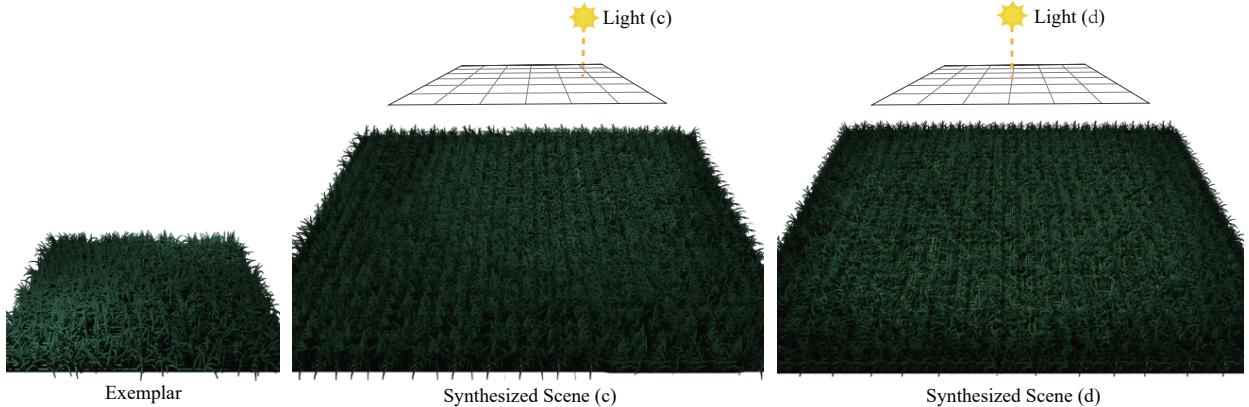


Figure 11: Relighting the scene of grass using different lighting conditions.



Figure 12: Relighting the scene of fur [46] on a 3D sphere. The synthesized scene on the plane is obtained by using the shading map extracted from the scene bent onto the surface of the 3D sphere as a guide for synthesis. The 3D synthesized scene is achieved by bending the synthesized scene on the plane onto the sphere.



Figure 13: The synthesized results of maintaining the lighting patterns. The shading map guider used in this experiment is obtained by doubling the size of the shading map from the exemplar.

scene with the desired lighting effects. The synthesized result was subsequently mapped onto the sphere using the deformation field, as illustrated in Fig. 12.

The second approach to obtaining the shading map guider involves directly scaling up the shading map of the exemplar. This method results in a new large-scale scene with the same lighting pattern as the exemplar, as demonstrated in Fig. 13.

6 Discussion

This article introduces a novel task called NeRF synthesis. It requires synthesizing a large-scale scene using a small exemplar represented by NeRF, while ensuring the continuity in terms of geometry and appearance of the synthesized scene. Applying traditional patch-based synthesis methods directly to the NeRF synthesis problem will result in some new issues. Firstly, in order to balance the weights of geometry and appearance, and address the boundary problem in NeRF synthesis, we propose a two-phase method and a boundary-constrained synthesis method, respectively. We evaluate the effectiveness of these methods on diverse scenes with complex geometry and rich textures, demonstrating their ability to generate high-quality synthesized results. Secondly, due to the coupling of lighting and material in NeRF, the lack of explicit lighting guidance can result in discontinuous lighting effects in the synthesized results. To overcome this, we propose the use of shading vectors to guide the synthesis process by representing the lighting information for each unit in the scene. For scene relighting, we develop two approaches for obtaining shading vectors and demonstrate through experiments that their combination can effectively achieve relighting without completely decoupling the scene. Our proposed methods provide a novel approach to synthesizing 3D objects, particularly well-suited for synthesizing natural environments such as lawns, ground, and stone surfaces in virtual worlds.

Limitations: Our method represents the scene directly as a grid, but scaling up the scene leads to higher memory consumption. To achieve optimal performance, our proposed approach requires a high-quality DVGO as input, which often requires a larger number of input images for real-world scenes. Since the relighting task requires continuity in geometry, texture, and lighting simultaneously, it is not suitable for scenes with complex geometry changes, as finding a suitable patch in the exemplar may not be possible. Furthermore, as capturing scenes with complex lighting effects in real-world settings is very challenging, we still lack experiments with re-lighting in real-world scenes.

Future work: Our future research plans involve exploring the application of learning-based approaches, such as GAN-based and diffusion model-based methods, to tackle NeRF synthesis problems. Additionally, although we have achieved successful scene synthesis on 3D meshes, these experiments have been limited to simple mesh structures. Therefore, our aim is to investigate how NeRF synthesis can be performed on arbitrary curved surfaces, which presents a substantial challenge and an opportunity for further research. This exploration would enable the seamless integration of NeRF with traditional rendering techniques.

References

- [1] 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. *Communications of the ACM*, 65:99–106, 2021.
- [2] Kyle Gao, Yina Gao, Hongjie He, Denning Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022.

- [3] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022.
- [4] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20374–20384, 2022.
- [5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Proceedings of the European Conference on Computer Vision*, pages 333–350, 2022.
- [6] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41:1–15, 2022.
- [7] Lara Raad, Axel Davy, Agnès Desolneux, and Jean-Michel Morel. A survey of exemplar-based texture synthesis. *Annals of Mathematical Sciences and Applications*, 3:89–148, 2018.
- [8] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (ToG)*, 40:1–18, 2021.
- [9] Viktor Rudnev, Mohamed Elgarib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Nerf for outdoor scene relighting. In *Proceedings of the European Conference on Computer Vision*, pages 615–631, 2022.
- [10] Wenqi Yang, Guanying Chen, Chaofeng Chen, Zhenfang Chen, and Kwan-Yee K Wong. Ps-nerf: Neural inverse rendering for multi-view photometric stereo. In *Proceedings of the European Conference on Computer Vision*, pages 266–284. Springer, 2022.
- [11] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (ToG)*, 21:665–672, 2002.
- [12] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238, 1995.
- [13] Jeremy S De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 361–368, 1997.
- [14] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40:49–70, 2000.
- [15] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038, 1999.
- [16] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, 2000.
- [17] Michael Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226, 2001.
- [18] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. 24:795–802, 2005.
- [19] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics (ToG)*, 23:364–367, 2004.
- [20] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001.
- [21] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics (ToG)*, 22:277–286, 2003.
- [22] Mudassir Rafi and Susanta Mukhopadhyay. Image quilting for texture synthesis of grayscale images using gray-level co-occurrence matrix and restricted cross-correlation. In *Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2017, Volume 1*, pages 37–47, 2019.
- [23] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (ToG)*, 21:665–672, 2002.
- [24] James T Kajiya and Timothy L Kay. Rendering fur with three dimensional textures. *ACM Siggraph Computer Graphics*, 23:271–280, 1989.

- [25] Fabrice Neyret. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics*, 4:55–70, 1998.
- [26] Kun Zhou, Xin Huang, Xi Wang, Yiyi Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. Mesh quilting for geometric texture synthesis. In *ACM SIGGRAPH 2006 Papers*, pages 690–697, 2006.
- [27] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63:139–144, 2020.
- [28] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021.
- [29] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166, 2020.
- [30] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021.
- [31] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Peter Kortschieder, and Matthias Nießner. Diffrrf: Rendering-guided 3d radiance field diffusion. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.
- [32] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. *arXiv preprint arXiv:2211.16677*, 2022.
- [33] Yujie Wang, Xuelin Chen, and Baoquan Chen. Singrav: Learning a generative radiance volume from a single natural scene. *arXiv preprint arXiv:2210.01202*, 2022.
- [34] Weiyu Li, Xuelin Chen, Jue Wang, and Baoquan Chen. Patch-based 3d natural scene generation from a single example. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [35] Cheng Sun, Min Sun, and Hwann-Tzong 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*, pages 5459–5469, 2022.
- [36] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.
- [37] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20:127–150, 2001.
- [38] Partha Das, Sezer Karaoglu, and Theo Gevers. Pie-net: Photometric invariant edge guided network for intrinsic image decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19790–19799, 2022.
- [39] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J Gortler. A local/global approach to mesh parameterization. In *Computer Graphics Forum*, volume 27, pages 1495–1504, 2008.
- [40] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.
- [41] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015.
- [42] yassin14000. Photoscanned patch of ground. <https://sketchfab.com/3d-models/photoscanned-patch-of-ground-c92e083da06d4295849c9f67e84c3664>, 2021.
- [43] chrisg4919. Cobblestones. <https://free3d.com/3d-model/cobblestones-2-41224.html>, 2019.
- [44] ROY. Green grass. <https://sketchfab.com/3d-models/train-wagon-42875c098c33456b84bcfcde4c7f1c58>, 2022.
- [45] 3dcruz. Pebbles model. <https://www.turbosquid.com/3d-models/pebbles-model-1222261>, 2017.
- [46] Tessa Grace. Old couch 3d. <https://www.turbosquid.com/3d-models/couch-old-3d-1171782>, 2017.
- [47] Miaad. carpet 3d model. <https://www.cgtrader.com/3d-models/furniture/other/carpet-8b996554-95ea-43a1-adb3-e89ae111d2f3>, 2019.
- [48] Javad Yavari. Green lawn free 3d model. <https://www.cgtrader.com/free-3d-models/plant/grass/green-lawn-8d4341d7-6281-40e9-8872-d429512a3b3b>, 2020.

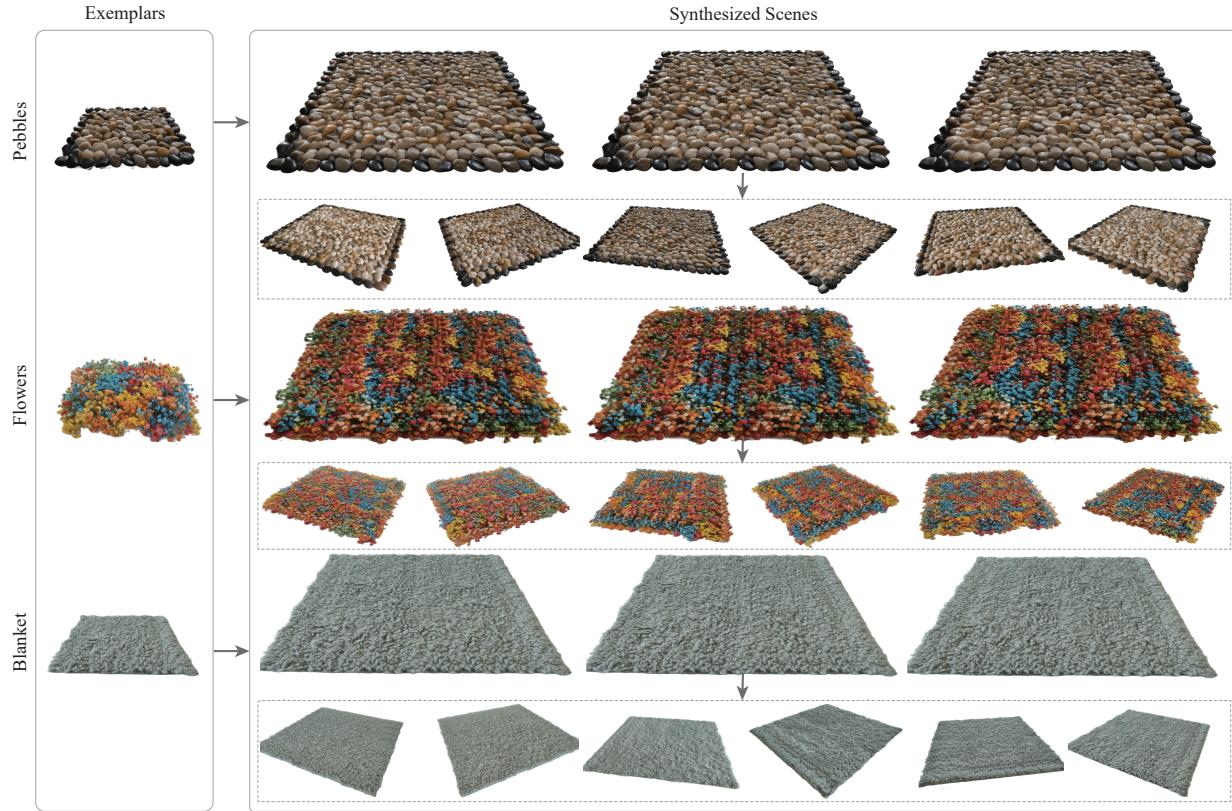


Figure 14: Synthesized results of real-world scenes with natural light. The odd rows display the various synthesized results, while the even rows show multi-view images of one of the synthesized results.

Supplementary Material

For natural light scenes, the consideration of lighting effects is not necessary, thus shading is not required to guide the NeRF synthesis process. We conducted extensive testing of our method on challenging synthetic and real-world data with complex geometry and vibrant colors under natural lighting conditions. The results, as illustrated in Fig. 15 and Fig. 14, demonstrate that our method achieves synthesized results that are nearly indistinguishable from the exemplar. It is important to note that our method is not limited to scenes with distinct geometry boundaries like rocks and ground, but can also handle scenes with intricate geometry such as grass and lawns.

Due to space limitations, only the results of synthesizing the exemplar at double the size are presented in the main text. However, it is worth mentioning that our method is capable of synthesizing results of any size and aspect ratio, as exemplified in Fig. 16.

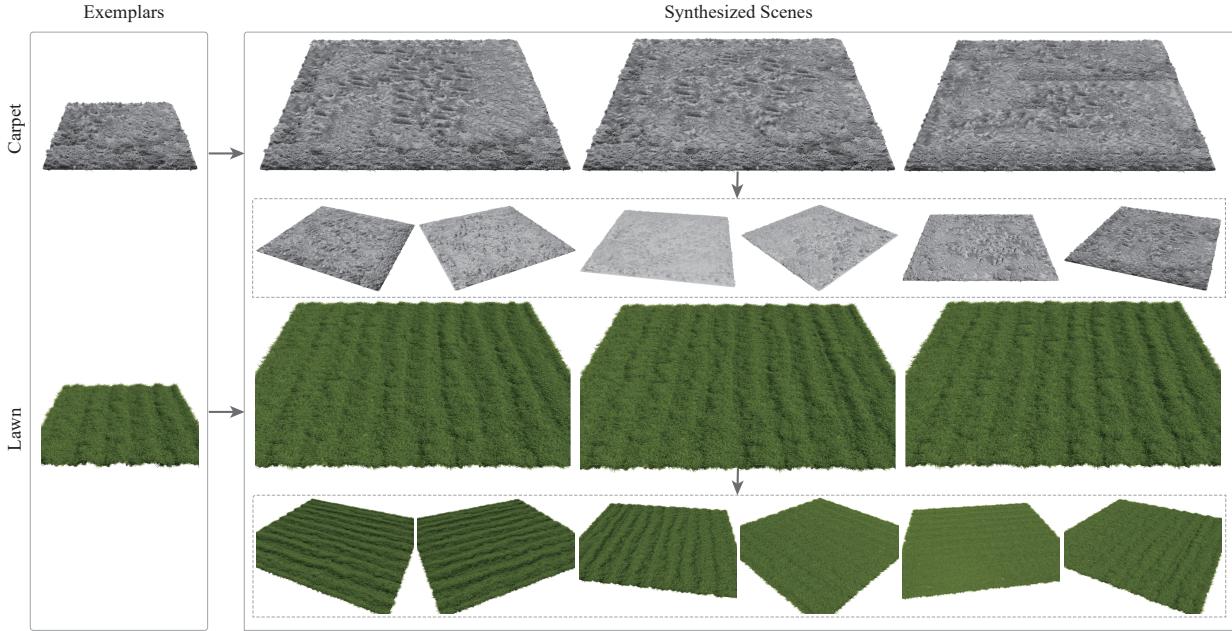


Figure 15: We present synthesized results under natural lighting conditions, using carpet [47] and lawn [48] as exemplars. The odd rows display the various synthesized results, while the even rows show multi-view images of one of the synthesized results.

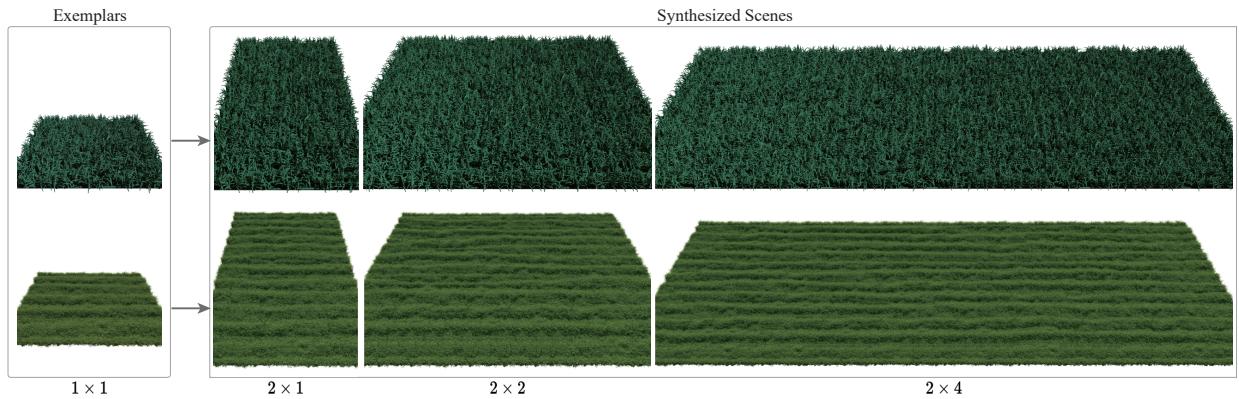


Figure 16: Synthesized results for larger sizes with natural light.