NeRF-Texture: Synthesizing Neural Radiance Field Textures

Yi-Hua Huang, Yan-Pei Cao, Yu-Kun Lai, Ying Shan and Lin Gao*

Abstract—Texture synthesis is a fundamental problem in computer graphics that would benefit various applications. Existing methods are effective in handling 2D image textures. In contrast, many real-world textures contain meso-structure in the 3D geometry space, such as grass, leaves, and fabrics, which cannot be effectively modeled using only 2D image textures. We propose a novel texture synthesis method with Neural Radiance Fields (NeRF) to capture and synthesize textures from given multi-view images. In the proposed NeRF texture representation, a scene with fine geometric details is disentangled into the meso-structure textures and the underlying base shape. This allows textures with meso-structure to be effectively learned as latent features situated on the base shape, which are fed into a NeRF decoder trained simultaneously to represent the rich view-dependent appearance. Using this implicit representation, we can synthesize NeRF-based textures through patch matching of latent features. However, inconsistencies between the metrics of the reconstructed content space and the latent feature space may compromise the synthesis quality. To enhance matching performance, we further regularize the distribution of latent features by incorporating a clustering constraint. In addition to generating NeRF textures over a planar domain, our method can also synthesize NeRF textures over curved surfaces, which are practically useful. Experimental results and evaluations demonstrate the effectiveness of our approach.

Index Terms—Neural radiance fields, texture synthesis, meso-structure texture.

1 Introduction

Capturing, modeling, synthesizing, and rendering real-world textures are fundamental problems in computer graphics and computer vision. In the real world, textures with high-frequency geometry are ubiquitous, like grass, leaves, fabrics, and cobblestones. Unfortunately, it is intractable to directly model such meso-structure with polygons, curves, or voxels [1], like flowers shown in Fig. 1. While a conventional texture map can represent a range of surface properties, such as color, reflection, transparency, and displacement, it remains impractical to accurately portray view-dependent appearance and meso-structure [2].

Thanks to recently proposed neural implicit rendering approaches such as NeRF (Neural Radiance Fields) [3], textures in complex real scenes could be reconstructed from multi-view images. The vanilla NeRF mixes the representation of geometry and appearance, which limits the freedom to manipulate the reconstructed textures. To support texture swapping and editing, NeuMesh [4] and NeuTex [5] make an attempt to disentangle the texture and geometry. NeuMesh [4] supports geometry and appearance editing but is incapable of modeling and synthesizing meso-structure textures; NeuTex [5] parameterizes the scene content in 3D Euclidean space over 2D UV space, which is suitable for modeling smooth surfaces rather than high-frequency meso-structure textures.

In computer graphics applications, once texture samples are captured, texture synthesis is an essential step to produce similar (but not repetitive) larger textures to decorate a target surface. Although there is extensive research on 2D image texture synthesis, little attention has been paid to the synthesis of NeRF-based textures.

In this paper, we propose a novel NeRF-based approach for capturing, modeling, synthesizing, and applying textures with meso-structure and view-dependent appearance, leveraging multi-view images obtained from real-world scenes. Our method only requires a set of multi-view images of the texture to acquire as input, which can be easily obtained by shooting a short video using a mobile phone. Our approach then learns the representations of the texture and synthesizes it to the desired size over a UV parameter space, typically in several minutes. Ultimately, the synthesized NeRF texture can be applied to any given shape, enabling real-time rendering.

More specifically, we propose the following key techniques to facilitate the modeling and synthesis of the NeRF textures with detailed geometry and view-dependent effects:

Firstly, to learn the meso-structure of textures, we disentangle the scene with fine geometric details into the meso-structure and the underlying base shape. We then learn the meso-structure as a NeRF texture through a latent feature field defined on the base shape. To achieve this goal, we first extract the base shape and explicitly represent it as a coarse mesh using Instant-NGP [6] and Co-ACD [7]. We then propose to map each point in the 3D Euclidean space to the Cartesian product of the signed distance and its foot

* *Corresponding Author is Lin Gao (gaolin@ict.ac.cn).*
- *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. Yi-Hua Huang and Lin Gao are also with University of Chinese Academy of Sciences, Beijing, China. E-mail: {huangyihua20g, gaolin}@ict.ac.cn*
- *Yan-Pei Cao is with VAST. E-mail: caoyanpei@gmail.com*
- *Ying Shan is with ARC Lab, Tencent PCG, China. E-mail: yingsshan@tencent.com*
- *Yu-Kun Lai is with School of Computer Science & Informatics, Cardiff University, UK. E-mail: LaiY4@cardiff.ac.uk*

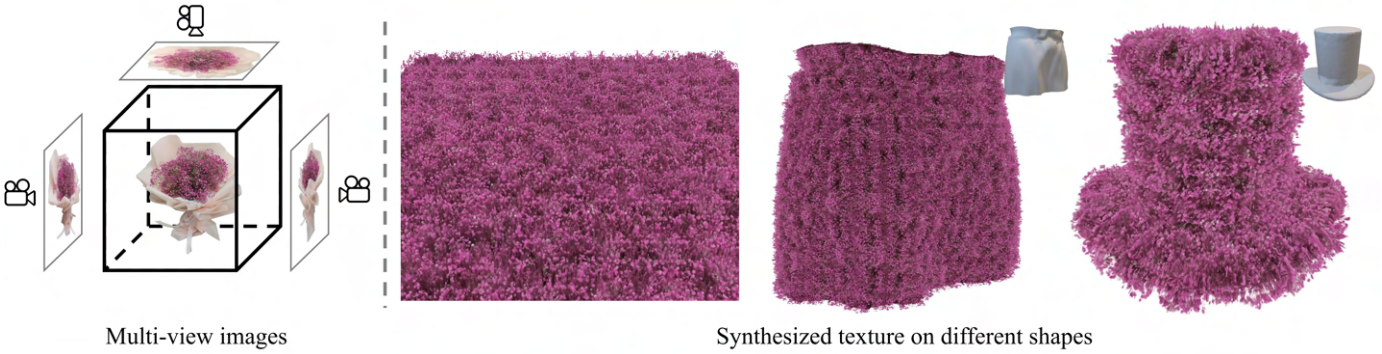


Fig. 1. Given a set of multi-view images of the target texture with meso-structure, our model synthesizes Neural Radiance Field (NeRF) textures, which can then be applied to novel shapes, such as the skirt and hat in the figure, with rich geometric and appearance details.

point when projected onto the base mesh. Latent features are defined on the base shape and fetched by the foot point. However, directly fetching latent codes from mesh vertices, like in [4], requires high-resolution meshes, which leads to a slowdown in latent code lookup and requires distillation from a well-trained NeRF. Instead, we fetch the latent representation for the texture with hash grids [6] to support real-time rendering and training from scratch.

Secondly, to apply captured NeRF-based textures to new shapes, it is crucial to synthesize textures at sufficient resolutions. We propose a novel NeRF-based texture synthesis method based on the coarse-fine disentanglement representation. Initially, we extract implicit patches from the base shape, on which latent features are defined, to create a patch collection. Subsequently, we implement an implicit patch matching algorithm to synthesize NeRF-based textures with collected patches. During this process, patches of latent features are sampled, matched and quilted to generate a texture space with the desired spatial resolution. Furthermore, we introduce an unsupervised metric learning approach to cluster the features of similar textures, thereby enhancing the quality of the synthesized results. In this paper, we significantly extended our previous conference paper [8] by extending the method to synthesize NeRF textures over curved surfaces as well as including more extensive evaluation, including results on newly captured datasets and more ablation study.

In summary, our main contributions are as follows:

- We propose a method to capture, model, synthesize and render NeRF textures with meso-structure from real-world multi-view images.
- We propose a coarse-fine disentanglement representation that learns the meso-structure and reflection coefficients as NeRF textures, which are separated from the underlying coarse surface.
- We adopt a patch matching algorithm in the latent space to synthesize NeRF textures. A clustering constraint is introduced to regularize the latent distribution for better matching. The method is further extended to NeRF texture synthesis on curved surfaces. To the best of our knowledge, this is the first work for NeRF texture synthesis.

2 Related Work

As our work is related to neural rendering and texture synthesis, we review papers related to these topics.

2.1 Neural Rendering

Various neural rendering approaches have been proposed to synthesize novel views of a scene with a given set of photographs. NeRF [3] models the scene as a radiance field with particles emitting and blocking lights. Inspired by NeRF, follow-up works extend it to achieve faster inference [6], [9], [10], handle large-scale scenes [11], [12], [13], [14] and dynamic scenes [15], [16], [17], and attain reflection decomposition [18], [19], [20], [21] and stylization [22], [23], [24]. Some other neural representations have been proposed to model meso-scale textures. Kuznetsov et al. [2] utilize neural bidirectional texture functions (BTFs) to model known texture with meso-structure. Wang et al. [25] propose to learn a complex shape as a combination of a smooth low-frequency signed distance function (SDF) and a continuous high-frequency signed distance function. Concurrent work [26] synthesizes DVGO (Direct Voxel Grid Optimization) [27]-based 3D scenes with the guidance of shading maps.

NeuTex [5] explicitly represents the texture in a neural representation through UV parameterization to support texture editing and mapping. However, such 2D parameterization assumes the target object can be smoothly mapped to a 2D parameter space, which is not suitable for most textures with meso-structure. NeuMesh [4] proposed a mesh-based neural implicit representation to disentangle the shape and appearance. With geometry and texture features defined on vertices, it achieves the geometry and texture editing of the neural implicit field. Nevertheless, NeuMesh utilizes predicted SDF rather than densities in volume rendering, which cannot be defined on non-watertight meso-structure. Besides, the mesh storing encodings closely fits the target surface, and as a result the meso-structure is not learned as texture properties. NeRF-Tex [1] firstly investigated the possibility to model the texture with meso-structures through NeRF. The model is trained on synthetic datasets with rendering results of patches in a bounding box on a plane under known lighting conditions. Textures are mapped to the shapes by repeatedly placing the reconstructed bounding box on surfaces. In contrast, our approach targets NeRF texture synthesis, which simultaneously learns the Phong

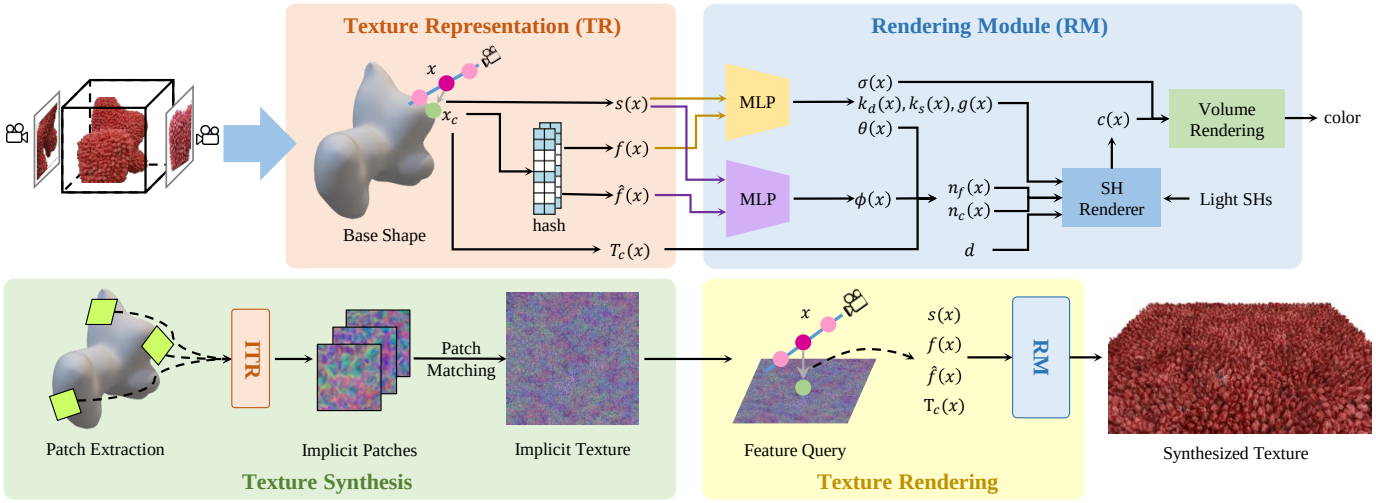


Fig. 2. **Overview of our method.** Given a set of multi-view images, we first estimate its base shape. Based on it, we model the scene with a disentangled representation of the base shape and NeRF texture with meso-structure. The query point x is projected onto the base shape as footpoint x_c. Latent features $f(x), \hat{f}(x)$ representing textures are fetched by feeding x_c to hash grids. Along with matrices of local tangent space $T_c(x)$, latent features $f(x), \hat{f}(x)$, and SDF value $s(x)$ are fed into the rendering module (RM). The density $\sigma(x)$, coefficients of Phong shading model $k_d(x), k_s(x), g(x)$, elevation and azimuth angles of the fine normal $\theta(x), \phi(x)$ are predicted based on the input features and SDF. The color $c(x)$ of the query point x is calculated by Spherical Harmonic (SH) rendering based on the coarse and fine normals $n_c(x), n_f(x)$, viewing direction d, shading coefficients $k_d(x), k_s(x), g(x)$ and lighting SHs. Based on the implicit texture representation (ITR), we extract implicit patches from the base shape and synthesize texture by an implicit patch matching algorithm. By querying $f(x), \hat{f}(x)$ and $T_c(x)$ from the synthesized implicit textures, we are able to render the appearance of the synthesized texture.

reflection coefficients, meso-structure and lighting conditions from real-world objects with textures.

2.2 Texture Synthesis

The goal of texture synthesis is to synthesize a new texture that appears to be generated by the same underlying process [28]. The pioneering work by [29] gradually grows the synthesized region by assigning pixels one by one. The assignment is determined by neighborhood similarity. Following this idea, a fixed neighborhood is used in [30] to avoid non-uniform pattern distribution. Patch-based method [31] proposes to blend the overlapped regions between patches. The works [32], [33] cut through the overlapped regions via dynamic programming and graph cut, respectively. PatchNet [34] searches an image library to locate ideal regions adhering to the synthesis constraints. Kwatra et al. [35] proposed an alternative approach by texture optimization.

In addition to traditional matching and optimization methods, neural networks are also introduced in texture synthesis. Gatys et al. [36] present a data-driven approach to generating texture through optimizing the Gram matrix of latent features extracted by VGG network [37]. Follow-up works [38], [39] train feed-forward convolutional networks to replace the time-consuming optimization process. Generative adversarial networks (GANs) are also widely used for texture synthesis [40], [41]. Zhou et al. [42] train a GAN to double the spatial extent of texture blocks, enabling the model to synthesize non-stationary texture. Portenier et al. [43] use the Gram matrix produced by the discriminator in adversarial loss to improve the quality of synthesized texture. Hertz et al. [44] propose a Mesh-CNN [45] based GAN architecture to synthesize geometric textures. PSGAN [46] proves that periodic encoding can improve the quality of GAN results. Inspired by it, Chen et al. [47] utilize periodic embedding as input and replace the convolution layer with a Multi-Layer Perceptron (MLP) to model implicit fields.

3 Method

We present a method to capture, model, synthesize and apply NeRF textures with meso-structure from real-world multi-view images. The overview of our pipeline is shown in Fig. 2. Given segmented multi-view images of the scene, our model learns to disentangle meso-structure textures and the underlying base shape. By sampling the implicit patches of latent features on the base shape and utilizing them to synthesize a larger texture map, we are able to decorate an arbitrary given mesh with the synthesized result. In the following, we will introduce texture representation in Sec. 3.1, texture synthesis in Sec. 3.2, and model optimization in Sec. 3.3.

3.1 Texture Representation

3.1.1 *Base Shape Extraction*

To model the base shape explicitly as a coarse mesh, we firstly adopt Instant-NGP [6] to reconstruct the coarse mesh by executing Marching Cubes [48] on the estimated density field with camera parameters estimated by COLMAP [49], [50]. To remove the meso-structure and make the coarse mesh smoother, the coarse mesh is transferred into the union of approximately decomposed convex hulls by CoACD [7]. The shape is then re-meshed [51], [52] to a mesh with vertices uniformly distributed on the surface. Fig. 3 illustrates the process of base shape extraction.

3.1.2 *Base Shape Projection*

We treat all attributes other than the base shape as texture attributes to learn, including meso-structure, normal and

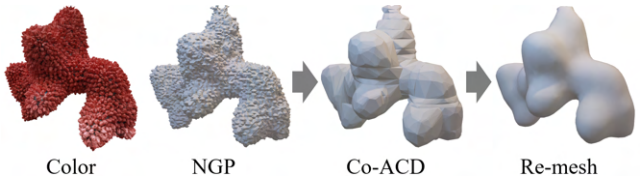


Fig. 3. **Base Shape Extraction.** We show the intermediate outputs during the base shape extraction, including NGP [6], Co-ACD [7], and re-meshing [51], [52].

appearance. To disentangle these attributes and the base shape, we utilize the coarse mesh mentioned above to re-parameterize 3D Euclidean space and learn the attributes into the latent features defined on the coarse mesh. In our approach, the coordinates of query point x are mapped to the coarse mesh to get the projected coordinates x_c and the signed distance $s(x)$, as depicted in Fig. 4. For each query point x, we look up its $K(=8)$ nearest neighbor points $\{v_k\}$ among the coarse mesh vertices. We interpolate the vertex normal n_v of neighbors together with the normalized vector from nearest neighbor v_1 to x using weighted KNN [53] to get the coarse mesh normal n_c of x:

$$\begin{aligned} \tilde{n}_c(x) &= \sum_{k=1}^{K} \frac{1}{W} \left(\frac{n_v(v_k)}{||x - v_k||_2} + \frac{x - v_1}{w||x - v_1||_2} \right), \\ n_c &= \frac{\tilde{n}_c}{||\tilde{n}_c||_2}, \; W = \sum_{k=1}^{K} \frac{1}{||x - v_k||_2} + \frac{1}{w}, \end{aligned} \tag{1}$$

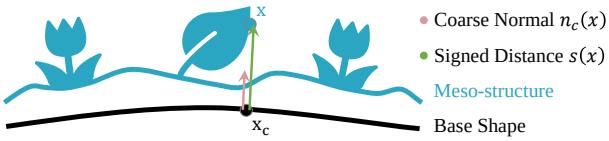


Fig. 4. **Illustration of Base Shape Projection in 2D**. Point x in Euclidean space is parameterized as the signed distance $s(x)$ and the projected footpoint x_c.

3.1.3 Differentiable Projection Layer

The step of ray casting makes the projected coordinates x_c non-differentiable with respect to the input coordinates x. However, the gradient is essential to approximate the normal [18], [19] or supervise the normal estimation [20], [54] for physically based rendering. In addition, back-propagating gradients to the camera parameters via coordinates x is crucial for camera pose modification [20], [55], [56] to improve the reconstruction quality. For these reasons, we construct a differentiable projection layer by specifying the following derivation rule:

$$\frac{\mathrm{d}x_c}{\mathrm{d}x} = (I - n_c(x)^T n_c(x)), \; \frac{\mathrm{d}s(x)}{\mathrm{d}x} = n_c(x) \tag{2}$$

where I is the identity matrix. The rule transfers the component of the gradient of x_c on the plane, which is perpendicular to $n_c(x)$, to x. It also passes the gradient of $s(x)$ to x after projection onto $n_c(x)$. The rule is consistent with parameterizing 3D coordinates as the footpoint and projected signed distance on a base shape.

3.1.4 Attributes Prediction

Directly querying latent codes from mesh vertices, like in [4], is difficult to train from scratch and demands high-resolution meshes, which results in high query overhead and difficulty in real-time rendering. Hence, we fetch the latent texture representation $f(x)$ in $O(1)$ time complexity by feeding the projected coordinates x_c to hash grids storing latent features [6]. Through the tiny-cuda-nn framework [57], we map the concatenated texture feature $f(x)$ and Fourier embedded [58] SDF value $s(x)$ to the density $\sigma(x)$ and reflection coefficients. The estimation of the fine normal $n_f(x)$ on x is done in two parts: estimating elevation angle $\theta(x)$ and azimuth angle $\phi(x)$, respectively. Both angles are represented in the local tangent frame of x_c, denoted as $T_c(x_c) = (t(x_c), b(x_c), n(x_c))$, meaning tangent, bitangent, and normal at x_c. Notice that $T_c(x_c)$ is determined by the tangent, bitangent, and normal of x_c's locating triangle face, which is pre-computed and fixed. Since $\theta(x)$ is the angle between the coarse mesh normal $n_c(x)$ and the fine (meso-structure) normal $n_f(x)$, it is an attribute independent of the definition of the local tangent frame. Instead, $\phi(x)$ depends on the direction of $t(x_c)$, which can be flexibly pre-chosen. Hence we predict θ with $s(x)$ and $f(x)$, which is further used for patch matching, to encourage similar texture contents to have latent features f close to each other regardless of different local tangent definitions. We then learn a different feature $\hat{f}(x)$ stored in another hash grid table for predicting $\phi(x)$.

3.1.5 Shading Model

Unlike vanilla NeRF, which mixes the representation of materials and lighting, we decompose these elements to enable the rendering of textures mapped to novel locations. To ensure real-time rendering speed and stable convergence, we utilize Spherical Harmonics (SHs) [59] to represent illumination and materials in our rendering pipeline. We adopt Phong shading [60] to model the material reflection with three parameters: diffuse coefficient k_d, specular coefficient k_s, and glossiness g. Following the approach outlined in [59], we employ the convolution of SHs to compute the texture color $c(x)$. The decomposition is illustrated in Fig. 5.

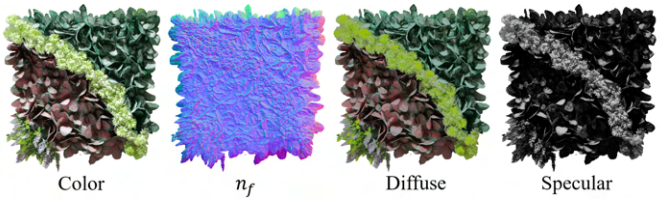


Fig. 5. **Shading Decomposition**. Our model predicts the fine normal n_f and decomposes the radiance into diffuse and specular components.

3.2 Texture Synthesis

3.2.1 Texture Patch Extraction

Since we have leveraged latent features on the base shape for representing texture attributes, the next step is to extract the implicit patches from the base shape for subsequent texture synthesis, as depicted in Fig. 6. In our approach, we firstly sample centers of patches on the entire base shape or user-specified regions via Poisson disk sampling [61] to get evenly distributed points. Next, we place square scan arrays of 128×128 resolution on each tangent plane of the coarse mesh to obtain the intersections of the scanning rays with the mesh. We discard patches with too long distance of ray casting to filter out those with excessive curvature. We then query the hash grids with these intersections to fetch latent features and obtain implicit patches. The obtained patches are denoted as $\{F_i \in \mathbb{R}^{128\times128\times C}\}$, where C is the latent dimension. We also denote the rotation of the sampling local frame to the world coordinate system as $T_s \in \mathbb{R}^{3\times3}$. We similarly define the rotation of the coarse mesh local frame to the world system as $T_c \in \mathbb{R}^{3\times3}$. For subsequent texture mapping, we also record T_c and T_s of each patch. The sampled patches are augmented by horizontal and vertical flipping for better synthesis. The transformation matrices of the sampling tangent space of augmenting patches are also flipped accordingly.

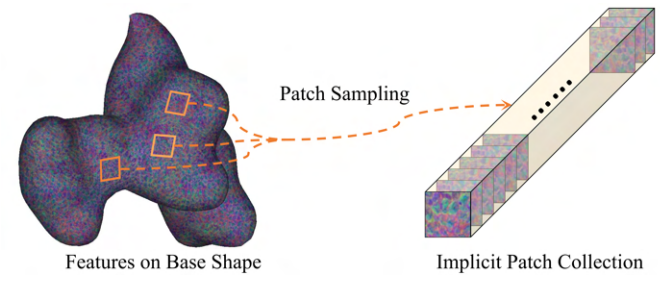


Fig. 6. **Texture Patch Extraction**. We extract implicit texture patches by sampling them on the base shape, where latent features are defined.

3.2.2 Patch-based Synthesis

We synthesize textures of arbitrary sizes based on the sampled exemplars using patch matching and quilting [32]. The output is initialized by copying a seed patch, and the synthesized region is gradually grown from the initial state by iteratively copying the picked patch onto it. The patch is randomly selected from 4 candidates with the most similar overlapping regions. The matching process is accelerated by pre-building a kd-tree of patches' overlapping regions. With a picked patch, the next step is to quilt it with the synthesized texture in the overlapping regions. Denoting the overlapping region of synthesized output and candidate patches as $F_1^{ov} \in \mathbb{R}^{H\times W\times C}$ and $F_2^{ov} \in \mathbb{R}^{H\times W\times C}$, where H and W are the height and width of overlapping regions. The error map is defined as $e_{i,j} = ||F_1^{ov}(i,j) - F_2^{ov}(i,j)||_2^2$, where i, j are indices in the error map. In each iteration, the choice of the patch is determined by the conditional distribution that measures the similarity of the overlapping region of the synthesized output and the candidate patch. With a picked patch, the minimum cost path along the overlapping region gives the boundary, and the patch is pasted onto the output texture. To determine the optimal vertical cut from bottom to top, we leverage dynamic programming to compute the minimum cumulative error E associated with each possible cut position:

$$E_{i,j} = e_{i,j} + \min\left(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1}\right) \quad (3)$$

We then utilize the values of E to backtrack and identify the position of the optimal cut, as illustrated in Fig. 7. The process of implicit patch matching for NeRF-based texture synthesis is demonstrated in Alg. 1.

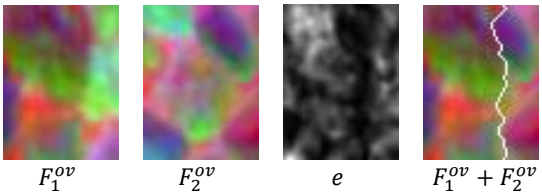


Fig. 7. **Patch Quilting.** The overlapping regions F_1^{ov} and F_2^{ov} are stitched together as $F_1^{ov} + F_2^{ov}$ based on the minimum cost path along the error map e. The cutting path is marked in white.

Algorithm 1 Implicit Patch Matching

Input: $\{F_i\}$: implicit exemplars
Output: $\hat{F}$: synthesized implicit texture
Procedure:

Randomly paste a seed patch from $\{F_i\}$ to the top left of $\hat{F}$
repeat
  Determine synthesizing and overlapping regions $\hat{F}_1, F_1^{ov}$
  Pick K' patches $\{F_k'\}$ with lowest average error $\overline{e}$
  Calculate the probability $\{p_k'\}$ of patches based on $\overline{e}$
  Sample a patch from the distribution $\{p_k'\}$
  Compute the cutting edge with minimum cost
  Paste the patch to the output texture
until Finish
Return $\hat{F}$

3.2.3 Synthesis on Curved Surface

By parameterizing local regions on a mesh surface as small rectangles [62], [63], the synthesis algorithm of our NeRF-based texture representation can also be extended to arbitrary curved surface just like in image texture [64], [65].

However, our NeRF-based texture synthesis on arbitrary surfaces poses several challenges. Firstly, unlike 2D textures, NeRF-based textures exhibit higher-frequency appearance and geometry, necessitating a higher synthesis resolution. To address this issue, we take measures at both the source and target ends of texture synthesis. Specifically, we perform more (8000) patch samplings from the source scene, each at a higher resolution (128×128) than those used in 2D texture synthesis. This ensures patch matching accuracy and enhances texture representation quality. On the target end, we increase the resolution of the target domain for synthesis. Previous works [62], [63] synthesized 2D textures on mesh vertices; however, to preserve the details of NeRF-based textures, these methods would require a mesh of very high resolution to preserve the details of NeRF-based textures, making the real-time rendering of NeRF-based texture intractable. To address this, instead of synthesizing texture on mesh vertices, we first parameterize the mesh as an atlas

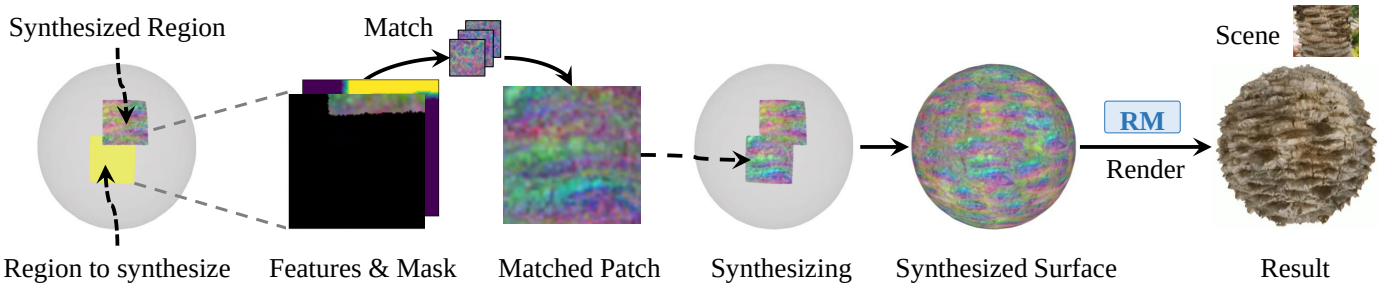


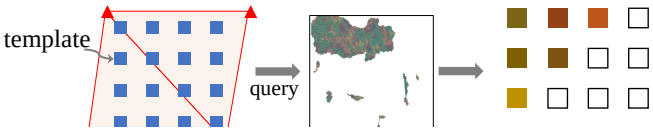
Fig. 8. **Synthesis on Arbitrary Surfaces.** The patch-matching synthesis algorithm of our NeRF-based texture can also be extended to arbitrary surfaces. Synthesized implicit texture is passed to the Rendering Module (RM) to produce final images of textured surface.

on a high resolution UV map (2048×2048) using existing approaches [66], [67]. Next, we back-map the grid points of the UV map to 3D space and synthesize textures on th obtaining a UV feature map. During the rendering proc query points are projected to the base mesh to obtain tl footprints. The UV coordinates of these footprints are cal lated through barycentric weighting of underlying triang and the latent feature is fetched via bilinear interpolatio the UV feature map.

Secondly, texture synthesis on arbitrary surfaces can rely on a pre-built kd-tree as done in synthesis in space to accelerate the patch matching process, since overlapping regions are not fixed. Furthermore, the h resolution and large number of patches furt the matching process. To overcome this cha pose to build a multi-resolution pyramid for perform the matching process in a coarse- Specifically, given a patch region with syn we first build a pyramid for it and then id similar patches at the coarsest level while f remaining patches. The picked patches are at a finer level and re-filtered. At the fines resolution is used to select the optimal pa and pasting. This strategy allows us to av patch matching for the full resolution for t collection, significantly reducing the time c each patch matching from approximately 10 than 0.05 seconds.

To cope with anisotropic textures, given represented by a triangular mesh, we deter field on it by interpolating some user-defi several control vertices, as proposed in prev [63]. Notice that the vector field can also b other state-of-the-art methods [68], [69]. As s our synthesis algorithm follows the spirit of and fills the texture space by iterating the fo until all vertices are grown with textures:

1) **Region Picking.** Randomly pick an unsynthesized vertex with an appropriate distance from the synthesized region to grow texture.
2) **Feature Fetching.** With the picked vertex and its orientation vector, place a patch template on it and fetch synthesized features and mask. The fetching process is illustrated in Fig. 9.
3) **Feature Matching and Quilting.** Select a patch that has minimum overlapping error with synthesized regions from candidate patches. Quilt the selected patch and synthesized regions with Alpha blending [64] or minimum cut quilting [32].



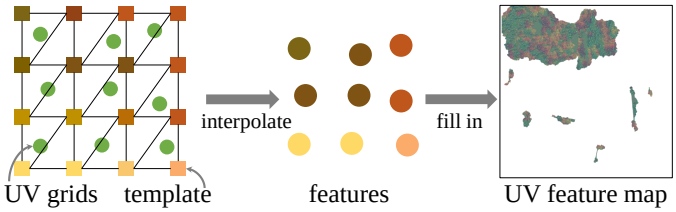
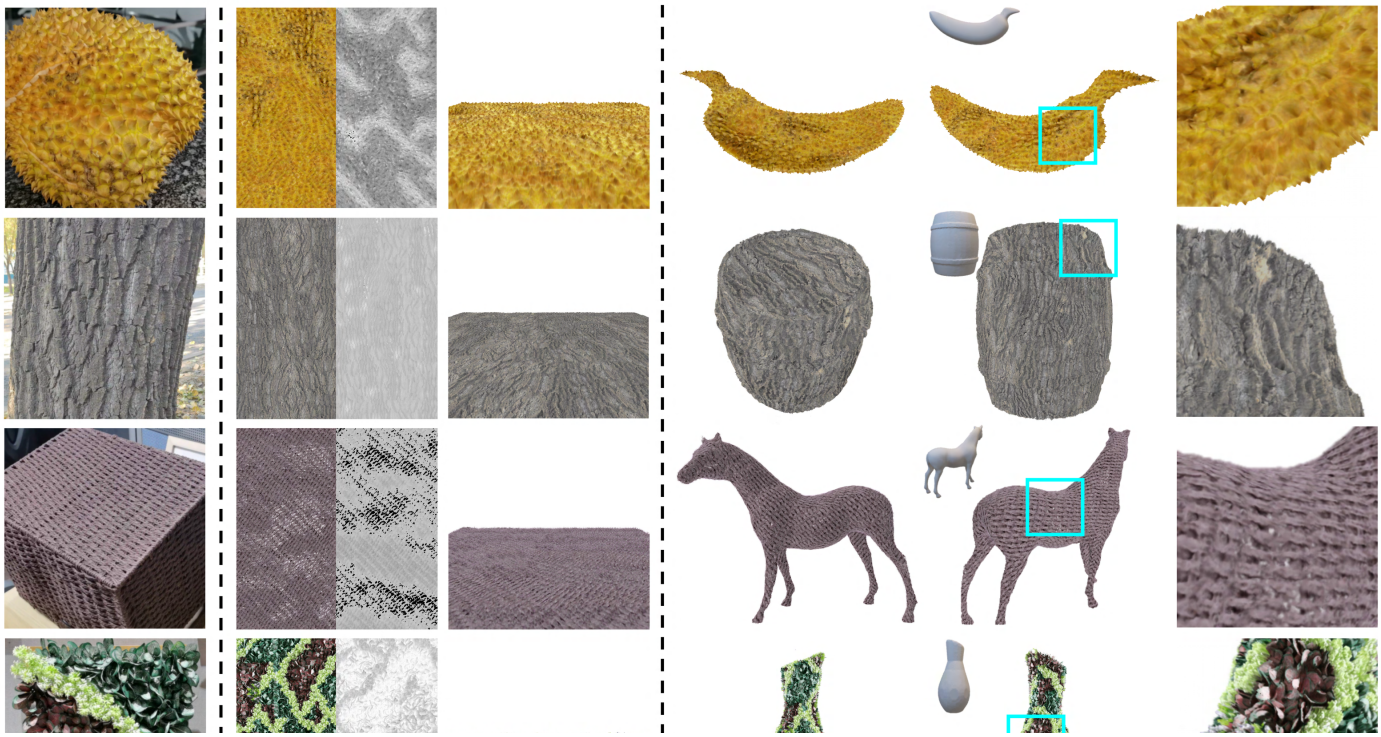


Fig. 10. **Patch Pasting.** The features of the UV grid are calculated via barycentric weighting on the template patch and the synthesized results are filled in the UV feature map.

3.2.4 *Latent Feature Clustering*

Ideally, the metric of latent space should be consistent with that of the reconstructed content space to ensure the plausibility of patch matching. Thanks to the continuity of neural networks, latent features close to each other reconstruct similar textures. However, it does not guarantee that similar latent features represent similar texture contents. To this end, we ensure the consistency of metrics in two aspects. First, latent features corresponding to similar texture contents have similar optimization targets (e.g. k_d, k_s, g and θ) during the training, which means that they have close optima when the training converges. Second, to avoid the latent features corresponding to similar textures falling into different optima during training, we introduce a clustering loss [70] for latent features into the optimization objective. Student's τ-distribution is used as the kernel to measure



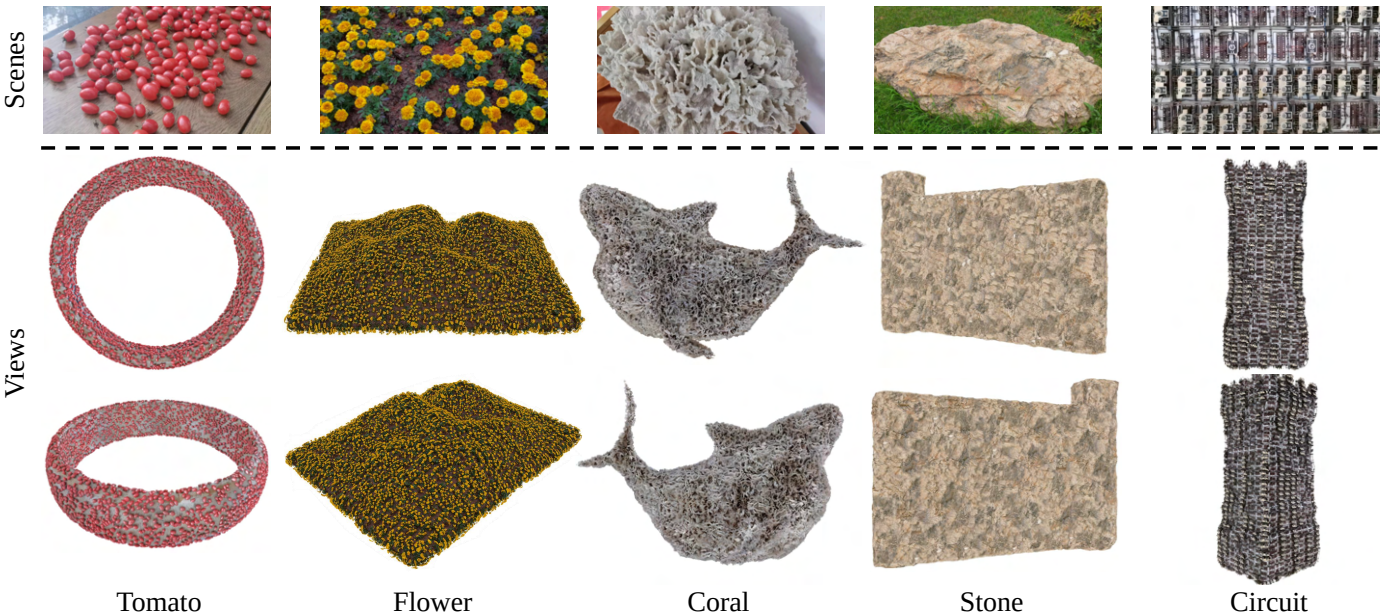


Fig. 12. **Texture Synthesis on Curved Surfaces.** Our synthesis algorithm can be extended to arbitrary curved surfaces, considering the continuity on the surface instead of UV space.

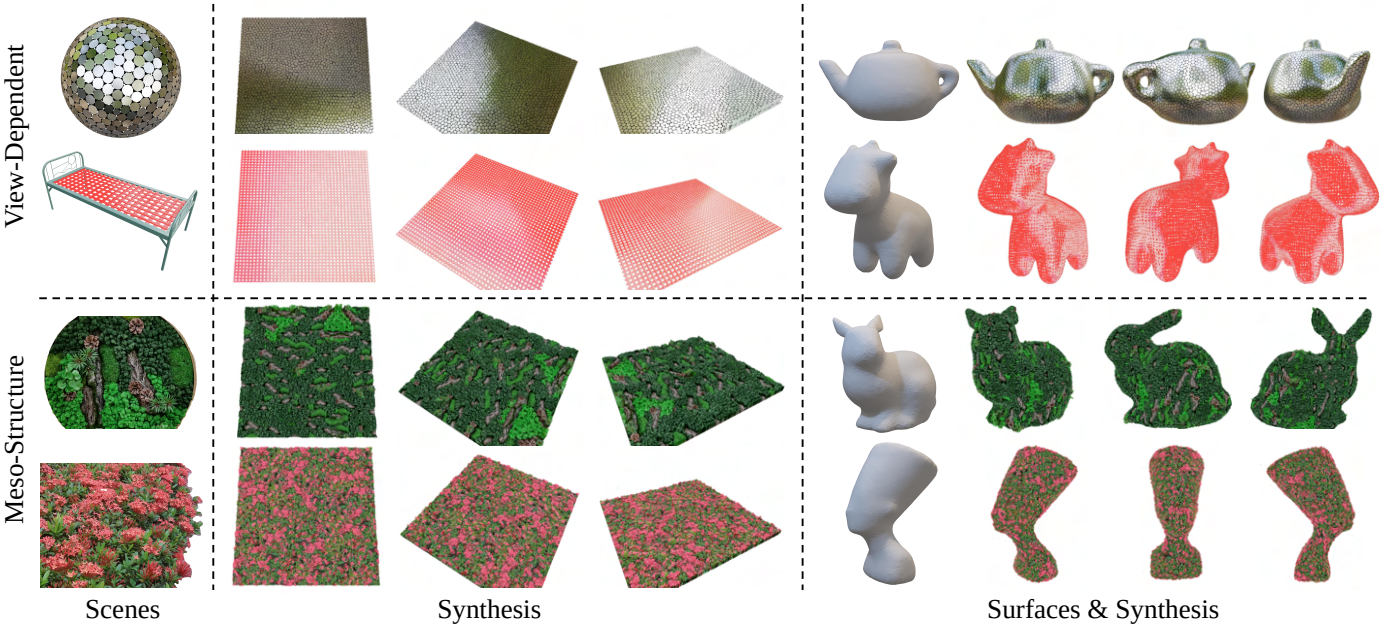


Fig. 13. **Synthesis of View-Dependent and Meso-Structure Textures.** Our NeRF-based approach synthesizes textures that capture both the view-dependent appearance (in the first two rows) and meso-structure details (in the last two rows) with accuracy and fidelity. Our method preserves these attributes of textures and achieves high-quality rendering results.

the similarity [71] between latent features f_i and trainable cluster centers μ_j. The distribution Q and its hardened auxiliary distribution P are defined as:

$$q_{ij} = \frac{(1+||f_i-\mu_j||_2^2/\kappa)^{-\frac{\kappa+1}{2}}}{\sum_{j'}(1+||f_i-\mu_{j'}||_2^2/\kappa)^{-\frac{\kappa+1}{2}}},\ p_{ij} = \frac{q_{ij}^2/\sum_i q_{ij}}{\sum_{j'}(q_{ij'}^2/\sum_i q_{ij'})} \tag{4}$$

where κ is the degree of freedom of the Student's τ-distribution. P is stricter than Q and closer to 0 or 1. The clustering loss is given by the KL divergence [72] between them: $L_{clu} = KL(P||Q)$. For hash grids at each resolution level, we cluster the embedding features with the clustering loss.

3.2.5 *Texture Mapping*

Given a new 3D shape with known UV coordinates, query point x is projected onto the surface, with the foot point denoted as $\tilde{x}_c$, as described in Sec. 3.1. The latent features $\tilde{f}(x)$ of the x is obtained by bilinear interpolation on the synthesized texture with UV coordinates of $\tilde{x}_c$. The residual transformation from the original coarse mesh local frame to the sampling local frame $T_s^{-1}(x)T_c(x)$ is also obtained by nearest-neighbor interpolation on synthesized T_s and T_c maps. Based on the feature and SDF value, the network predicts the appearance and geometry of the query point. With the transformation of the new tangent space on the target surface, denoted as $\tilde{T}_c(x)$, the predicted normal on the new shape is calculated as:

$$\begin{aligned} \tilde{n}_f(x) &= \tilde{T}_c(x)T_s^{-1}(x)T_c(x)R(\theta(x),\phi(x)), \\ R(\theta,\phi) &= (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)^T \end{aligned} \tag{5}$$

The density and reflection coefficients are also calculated by $\tilde{f}(x)$ and SDF value $\tilde{s}(x)$ relative to the new shape.

3.3 Optimization

Our model is trained with the Adam optimizer [73]. The optimization target of our method consists of four terms: $L = L_{rec} + \lambda_1 L_{clu} + \lambda_2 L_{dis} + \lambda_3 L_{nor}$. L_{rec} is the L1 RGB reconstruction loss. L_{dis} is the distortion loss [12] removing floating artifacts. L_{nor} supervises the prediction of (θ, ϕ) based on the negative gradients of density $\sigma(x)$ relative to x. Owing to the noise of density gradients, we employ the relaxed cosine distance to supervise the estimated normal:

$$L_{nor} = -\cos\{\min(\langle < -\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x}, n_f(x)\rangle >, \frac{\pi}{8})\} \tag{6}$$

In our experiments, λ_1, λ_2, and λ_3 are set to 10^{-5}, 10^{-2}, and 1.

4 Results

In this section, we perform several experiments to demonstrate the utility of our method. We will firstly show the results on texture synthesis and applications in Sec. 4.1. Then we quantitatively and qualitatively compare the novel view synthesis quality to show the rendering quality of our method in Sec. 4.2. We also compare the 2D texture and our representation in Sec. 4.3, which the advantage of our method in texture modeling. Finally, we compare with NeRF-Tex in Sec. 4.4 and perform an ablation study on the impact of latent feature constraint in Sec. 4.5.

4.1 Texture Synthesis and Applications

We demonstrate the utility of our method by acquiring and synthesizing textures from the real world captured by a mobile phone as shown in Fig. 11. The target texture includes durian, bark, fabric, leaves, and flowers. The synthesized results and depth visualization are shown in the 2nd and 3rd columns. We also applied captured textures to grow on the desired shape or pattern shown in 4th-6th columns. We

TABLE 1
Quantitative comparison of view synthesis. We show the average PSNR/SSIM/LPIPS for novel view synthesis on DTU.

Methods	Scan 55			Scan 83			Scan 105			Scan 122		
	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
NeRF	28.244	0.940	0.212	37.816	0.990	0.092	34.152	0.947	0.208	36.464	0.979	0.135
NGP	**34.108**	**0.991**	**0.086**	42.602	0.996	0.049	**38.247**	**0.991**	0.085	41.976	0.996	0.057
Ours	32.378	0.988	0.104	**43.842**	**0.998**	**0.027**	36.809	0.990	**0.067**	**42.704**	**0.998**	**0.031**

synthesize the durian's texture with thorns and transfe to a banana. The tree bark is usually covered with stri of ravines. We synthesize and apply such texture to a ba shape and obtain a wooden barrel. We capture fabric text on a woven basket and construct a woven horse. Leaves a grass are also typical textures in nature; we synthesize oc of leaves and grass and apply it to a vase. We also synthes colorful flowers guided by the shown text image, by cons ering the rendered color of patches during texture synthe (see supplementary for details). The zoomed-in view in column shows the effect of the material on oblique vie and object edges, where 2D textures appear unrealistic to the lack of meso-structure modeling, demonstrating advantages of our method over 2D textures.

We also show the results of our NeRF-based text synthesis on curved surfaces using various captured t tures in Fig. 12. We capture scenes of tomatoes on the de flowers on the ground, a coral, a stone, and circuits of old computer. These textures are synthesized on surfaces of a ring, a mountain-valley shape, a shark, a wall, and a tower respectively. The synthesis only considers the neighboring relationship in the 3D space rather than the UV space.

To better highlight the advantages of our NeRF-based method in modeling view-dependent appearance and meso-structure geometry, we present additional results in Fig. 13. The first two rows showcase the texture synthesis of mirror balls and a metal bed, which are made up of reflective mirror faces and metal specular materials, respectively. To improve the modeling of such specular materials and accurately reflect high-frequency environment lights, we use the lighting representation of Ref-NeRF [74] by introducing an extra MLP to predict the reflected color of a reflective direction during the shading step. We also synthesize textures on Utah Teapot [75] and Spot [76], resulting in high-fidelity view-dependent appearance of the textured meshes. In addition, we synthesize textures of various leaves, barks, pinecones, moss, and tiny flowers, as shown in the last two rows. We also evaluate our method on the Standord Bunny [77] and Nefertiti, demonstrating that the NeRF-based representation preserves the high fidelity of captured textures in both view-dependent appearance and meso-scale geometry.

4.2 View synthesis quality

We evaluate the view synthesis quality of our method on the published dataset DTU [78], in which the scenes are of objects suitable for our method to represent as they contain texture-like structure. We test on 4 scenes with masks provided by [79]. In each scene, 5 images are randomly picked as the test set. Qualitative comparison with NeRF [3], Instant-NGP (NGP) [6] and ours is shown in

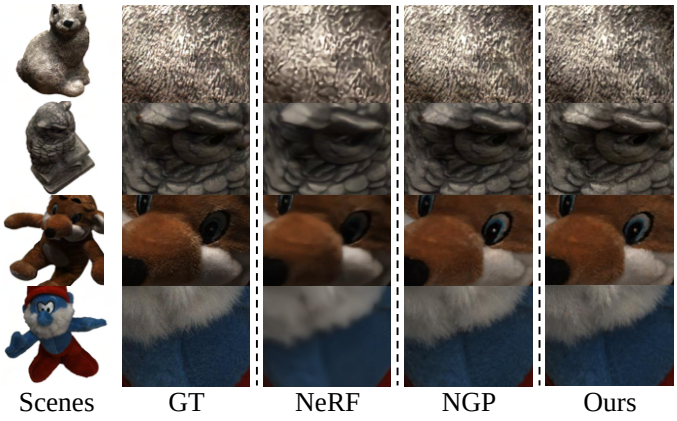


Fig. 14. **Qualitative Comparison of View Synthesis Results.** Note that our method supports texture capture, synthesis and application while visually close to the state of the arts.

Fig. 14. We report the PSNR, SSIM, and LPIPS in Tab. 1. Due to the specific design for disentangling meso-structure and materials, our approach is slightly worse than NGP in some quantitative comparisons. Despite this, our rendered results are still realistic in high-frequency details and perceptually close to NGP's results.

4.3 Comparison with 2D texture

To verify the advantages of our texture representation over 2D image textures, we conduct quantitative and qualitative experiments to demonstrate it. To obtain 2D texture patches, we simultaneously render the RGB patches when sampling patches as described in Sec. 3.2. Based on the RGB patches, we use the patch matching algorithm to synthesize a texture image the same size as our generated neural texture. We render both 2D and neural textures in different angles of elevation from 0° to 80° as samples for comparison.

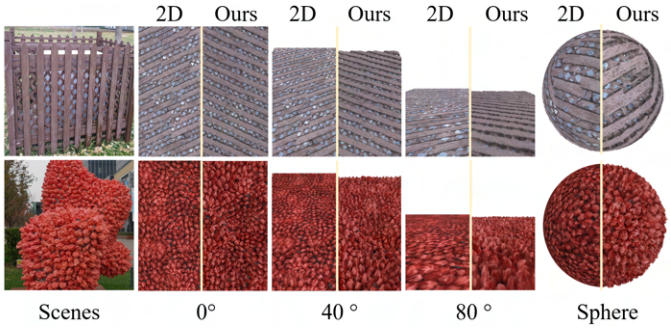


Fig. 15. **Qualitative Comparison with 2D Textures.** We show the rendering results of our synthesized textures and 2D textures. Our representation of maintains realism even at high-elevation viewing angles.

Single Image Fréchet Inception Distance (SIFID) SIFID introduced in [80] is a commonly used metric to assess the realism of generated images. We crop the regions, where the corresponding 3D shape approximate planes, from the captured images as ground truths. We then calculate the SIFIDs between rendered textures with ground truths of the closest viewing directions relative to planes. Average SIFIDs reported in Tab. 2 indicate that our texture representation is more realistic than 2D textures.

TABLE 2
Quantitative Comparison with 2D Texture. Our texture has lower SIFIDs in all elevation angles.

Degree	0°	20°	40°	60°	80°	**Average**
2D	0.73	0.75	0.82	1.21	1.75	1.05
Ours	**0.52**	**0.51**	**0.56**	**0.58**	**0.82**	**0.60**

Qualitative comparison We also show the qualitative comparison of 2D image texture with our representation in Fig. 15 in different viewing directions. The synthesized 2D texture of meso-structure will be unrealistic at high elevation angles while our representation can well represent the geometry occlusion of meso-structure.

4.4 Comparison with NeRF-Tex

We present a visual comparison between NeRF-Tex [1] and our proposed method, as demonstrated in Fig. 16. In contrast to our approach, NeRF-Tex does not perform texture synthesis; instead, it repeatedly places planar texture patches on anchor points of the mesh in an unstructured manner, leading to a loss of regularity for typical st textures. Besides, it is crucial to note that NeRF-Tex NeRF using synthetic data with known tightly boun geometry, which cannot be directly applied to re data. Thus, we utilize our coarse-fine disentanglem resentation to generate multi-view images of re meso-structure textures within a bounding box, se training data for NeRF-Tex.

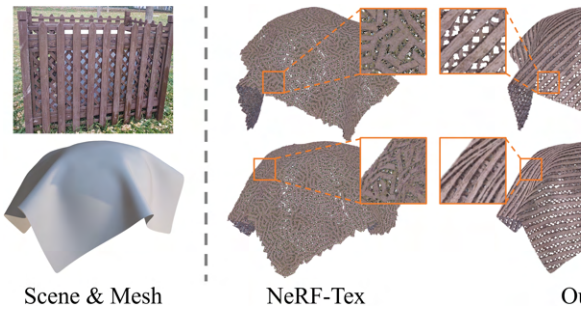


Fig. 16. **Comparison with NeRF-Tex [1]** Our method dem superior preservation of texture continuity and structure thai synthesis algorithm of NeRF texture.

4.5 Ablation on Clustering Constraint

The complexity and randomness of textures can easily lead to the disordered distribution of features, even if these features share the same reconstruction target. The clustering constraint regularizes the latent distribution by encouraging similar textures represented by close features. We visualize the synthesized feature with and without the constraint by Principal Component Analysis dimensionality reduction to 3D, which is further visualized as RGB channels in Fig. 17 (left). We found that the constraint makes the latent distribution more compact and reduces the variance. Results without the constraint tend to have more artifacts (Fig. 17 (right)).

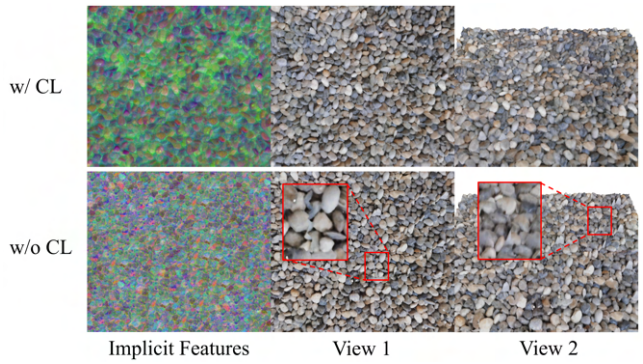


Fig. 17. **Impact of Clustering Constraint.** With the clustering loss (w/ CL), latent features are constrained to cluster, which reduces the distance of latent features corresponding to similar textures and further reduces artifacts in synthesized results.

4.6 Ablation on Patch Resolution

The resolution of patches is a pre-defined hyperparameter; therefore, we perform ablation experiments to analyze its impact. According to the work [32], patch size needs to be large enough to capture the structure of the texture

Fig. 18. **Ablation on Patch Resolution.** A resolution of 128×128 is sufficient to represent a patch area of the required size.

4.7 Ablation on the Number of Training Views

NeRF is built on the principle that when light rays intersect a surface at the same position and from the same view direction, they have the same color. During training, NeRF casts rays from training views and optimizes rendering loss for each pixel. Therefore, increasing the number of training images can reduce the ambiguity of ray intersection and improve the synthesis of new perspectives. Our method focuses on textured scenes with high-frequency features, where ambiguity is less pronounced compared to scenes that lack color and geometry richness.

We conducted an ablation study on a scene with a ground covered in rocks. The rendering results of our model, trained on different numbers of views, are shown in Fig. 19. With only a few views (e.g., 4 views shown in the first column), the rendering results exhibit blue tint and artifacts, as the training views are unable to capture every detail of the texture. With 16 training views (second column), most parts of the scene become much clearer, but a few floating pieces remain. When using 64 training views, the visual quality is very close to that of using 256 views. This experiment demonstrates that texture reconstruction

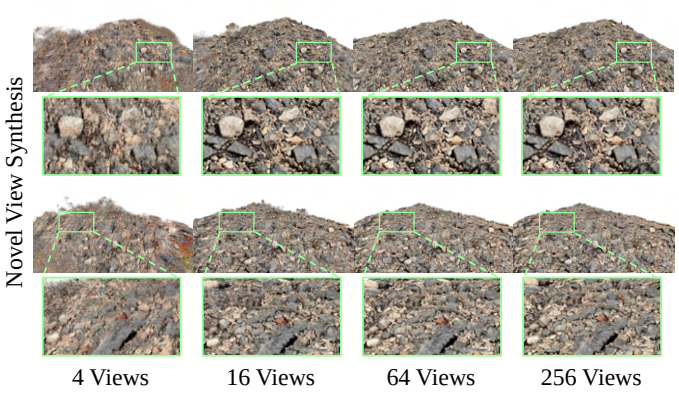


Fig. 19. **Ablation Study on the Number of Training Views.** Using fewer training views results in ambiguity in geometry and appearance. Increasing the number of training views improves the quality of rendered textures.

4.8 Rendering Speed Analysis

Our approach utilizes hash grids to efficiently retrieve latent features and employs tiny MLPs [57] for quick color and opacity querying. However, our method consumes additional time for the K-Nearest Neighbor (KNN) search during coarse normal calculation and ray tracing for footprint projection. To address this, we applied specific engineering strategies to enhance the speed of these two processes in our implementation: **1)** To expedite the KNN search, we evenly divide the space into bins [81] and insert each vertex of the base mesh into them. During the KNN search, we only need to search mesh vertices within neighboring bins of the query point within a specified radius. **2)** Additionally, we construct a Bounding Volume Hierarchy (BVH) [82] with nodes formatted in Axis-Aligned Bounding Boxes (AABBs) [83] to organize the triangles of the base mesh. The pre-built BVH avoids the unnecessary ray collision detection of those triangles within AABBs that do not intersect, thereby accelerating the ray-tracing process [84]. All these operations are implemented using CUDA and executed by hundreds of threads in parallel on the GPU, enabling fast calculations.

We evaluated the rendering speed of our method and compared it to Vanilla-NeRF [3] and Instant-NGP [6] as benchmarks. The rendering speed, measured in frames per second (FPS), is presented in Tab. 3. Although our method is slower than Instant-NGP due to the additional calculations, it still achieves real-time rendering speed and is much faster than Vanilla-NeRF.

TABLE 3
Rendering speed comparison. We compared the rendering speed of NeRF, Instant-NGP and our method, at a resolution of 512×512.

Method	NeRF [3]	Instant-NGP [6]	Ours
Speed (FPS) ↑	0.02	307.05	84.45

5 Challenge Analysis

As we have demonstrated, our method can easily capture, reconstruct, and synthesize common real-world textures on various objects. To better understand and explore the boundaries of the method's capabilities, we conducted several experiments and analyses on more challenging scenes in this section. We summarize the challenges into two aspects: texture capture and texture synthesis, and discuss them respectively in the following.

5.1 Challenging Texture Capture

Our approach faces certain challenges when it comes to capturing textures from objects with **1)** limited coarse extents or **2)** complex topology.

Limited Coarse Extents For scenes where the coarse shape has limited spatial extents, the size of sampled patches is also limited. In extreme cases, the patch size will be too small to capture texture patterns, resulting in synthesized textures containing numerous artifacts. As shown in the second row of Fig. 21, the truss's underlying base shape is too narrow to sample sufficiently large and diverse patches, resulting in poor synthesis results with artifacts and irregular patterns. Conversely, when multiple trusses are placed together to construct a larger roof (first row), the underlying base shape becomes large enough to capture patches that maintain the structural patterns, resulting in more satisfactory synthesized textures. Additionally, the perforated structure on the Lego loader shown in Fig. 20 is too narrow to sample any patches, posing a significant challenge for subsequent synthesis. On the other hand, our method is capable of easily extracting and synthesizing the bump texture on the Lego base.

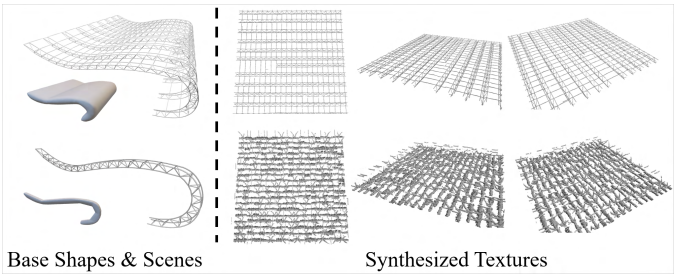


Fig. 21. **Challenges in Limited Coarse Extents.** The repeated structures of the truss pattern are distributed on a long but narrow coarse surface, which makes sampled patches too small to synthesize high-quality textures. The roof composed of trusses, on the contrary, has enough coarse extents to capture and synthesize satisfactory results.

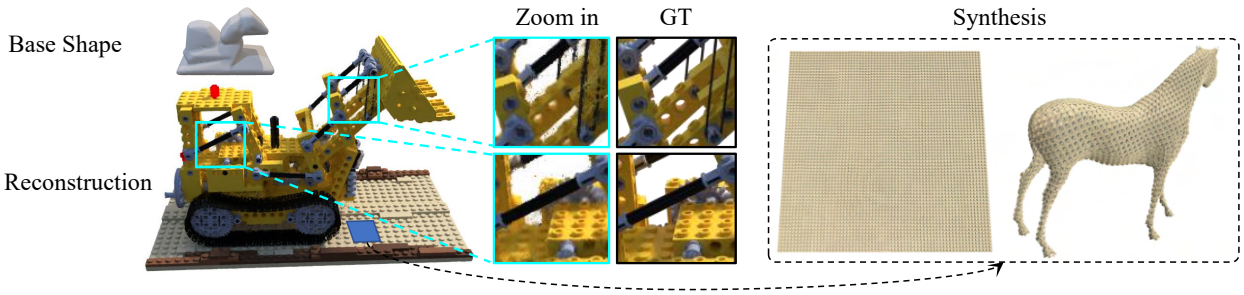


Fig. 20. **Challenges on Texture Capture.** Our approach fails to reconstruct and capture textures on shapes with complex coarse geometry due to the difficulty in base shape estimation and patch sampling on regions with limited spatial extents of the base surface. On the contrary, the bump texture on the Lego base can be easily acquired and synthesized.

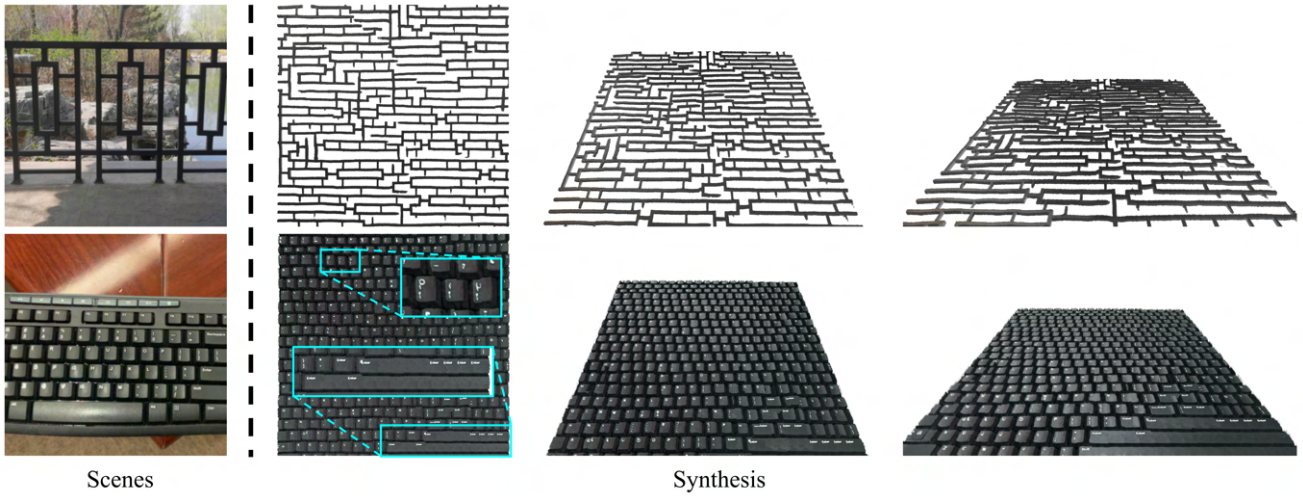


Fig. 22. **Challenges on Texture Synthesis.** Our synthesis approach based on patch matching struggles to exactly preserve the continuity of highly structured textures (first row) requiring strict matching with a few captured exemplars. Our method is also agnostic to the semantic contents of textures like the keycap texture of a keyboard (second row).

Complex Topology For scenes with complex topology, such as the Lego example shown in Fig. 20, accurately representing the detailed surface with a coarse mesh becomes challenging. Consequently, capturing the desired bump texture on the inner surface of the Lego cockpit becomes difficult. Additionally, the rendering quality of NeRF-Texture may degrade in such scenarios, leading to artifacts as seen in the zoomed-in section of Fig. 20. This degradation is due to the coarse shape approximation, which fails to provide an optimal parameterization for NeRF.

5.2 Challenging Texture Synthesis

The challenges in texture synthesis arise when textures **1)** require strict matching or **2)** have semantic contents.

Strict Matching Our patch-matching approach selects patches from the best-matched candidates based on the matching errors between patches and synthesized regions. However, this approach is a local optimal and not necessarily a global one since the selected candidate may cause significant matching errors in subsequent iterations. Therefore, it is a greedy strategy and may cause the breaking of structures for textures requiring strict matching, especially when only limited patch exemplars are available. As shown in the first row of Fig. 22, although the overall structure of the synthesized railing texture is preserved, there are still some breaking parts in the connection of rails.

Semantic Content Our synthesis algorithm is semantically agnostic, which can distort semantic content such as keycap shapes and incorrect synthesis of characters, as shown in the second row of Fig. 22. The letter P and bracket symbol are printed on the same keycap in the zoom-in window. The space key is extended to an unreasonable length. To address this limitation, our approach could potentially incorporate recent generative techniques, such as Diffusion Models [85].

6 Conclusion

We present NeRF-Texture, a novel approach that captures, models, synthesizes, and renders real-world textures with rich geometric and appearance details. A coarse-fine decomposition representation is introduced to disentangle the meso-structure texture and base shape. Based on the representation, we adopt a latent patch-matching algorithm to synthesize acquired textures on the UV plane or arbitrary surfaces. A clustering constraint regularizes the distribution of latent features for better synthesis.

7 Acknowledgement

This work was supported by Beijing Municipal Natural Science Foundation for Distinguished Young Scholars (No. JQ21013), National Natural Science Foundation of China (No. 62322210) and Beijing Municipal Science and Technology Commission (No. Z231100005923031).

REFERENCES

[1] H. Baatz, J. Granskog, M. Papas, F. Rousselle, and J. Novák, "NeRF-Tex: Neural reflectance field textures," in *Comput. Graph. Forum*, no. 6, 2022, pp. 287–301.

[2] A. Kuznetsov, X. Wang, K. Mullia, F. Luan, Z. Xu, M. Hasan, and R. Ramamoorthi, "Rendering neural materials on curved surfaces," in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022, pp. 1–9.

[3] 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," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[4] Bao and 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 *ECCV*, 2022, pp. 597–614.

[5] F. Xiang, Z. Xu, M. Hasan, Y. Hold-Geoffroy, K. Sunkavalli, and H. Su, "NeuTex: Neural texture mapping for volumetric neural rendering," in *CVPR*, 2021, pp. 7119–7128.

[6] 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. 1–15, 2022.

[7] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3D meshes with collision-aware concavity and tree search," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 1–18, 2022.

[8] Y.-H. Huang, Y.-P. Cao, Y.-K. Lai, Y. Shan, and L. Gao, "NeRF-Texture: Texture synthesis with neural radiance fields," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023, pp. 1–10.

[9] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *CVPR*, 2022, pp. 5501–5510.

[10] A. Karnewar, T. Ritschel, O. Wang, and N. Mitra, "Relu fields: The little non-linearity that could," in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022, pp. 1–9.

[11] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and improving neural radiance fields," *arXiv preprint arXiv:2010.07492*, 2020.

[12] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded anti-aliased neural radiance fields," in *CVPR*, 2022, pp. 5470–5479.

[13] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-NeRF: Scalable large scene neural view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8248–8258.

[14] Y. Gao, Y.-P. Cao, and Y. Shan, "SurfelNeRF: Neural surfel radiance fields for online photorealistic reconstruction of indoor scenes," *arXiv preprint arXiv:2304.08971*, 2023.

[15] J.-W. Liu, Y.-P. Cao, W. Mao, W. Zhang, D. J. Zhang, J. Keppo, Y. Shan, X. Qie, and M. Z. Shou, "DeVRF: Fast deformable voxel radiance fields for dynamic scenes," in *NeurIPS*, 2022, pp. 36 762–36 775.

[16] Y.-L. Qiao, A. Gao, and M. Lin, "NeuPhysics: Editable neural geometry and physics from monocular videos," in *NeurIPS*, 2022, pp. 12 841–12 854.

[17] L. Song, A. Chen, Z. Li, Z. Chen, L. Chen, J. Yuan, Y. Xu, and A. Geiger, "NeRFPlayer: A streamable dynamic scene representation with decomposed neural radiance fields," *IEEE TVCG*, vol. 29, no. 5, pp. 2732–2742, 2023.

[18] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch, "NeRD: Neural reflectance decomposition from image collections," in *ICCV*, 2021, pp. 12 684–12 694.

[19] 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 *CVPR*, 2021, pp. 7495–7504.

[20] Z. Kuang, K. Olszewski, M. Chai, Z. Huang, P. Achlioptas, and S. Tulyakov, "NeROIC: Neural rendering of objects from online image collections," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 1–12, 2022.

[21] J. Munkberg, J. Hasselgren, T. Shen, J. Gao, W. Chen, A. Evans, T. Müller, and S. Fidler, "Extracting triangular 3D models, materials, and lighting from images," in *CVPR*, 2022, pp. 8280–8290.

[22] K. Zhang, N. Kolkin, S. Bi, F. Luan, Z. Xu, E. Shechtman, and N. Snavely, "ARF: Artistic radiance fields," in *ECCV*, 2022, pp. 717–733.

[23] 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 *CVPR*, 2022, pp. 18 342–18 352.

[24] Z. Fan, Y. Jiang, P. Wang, X. Gong, D. Xu, and Z. Wang, "Unified implicit neural stylization," in *ECCV*, 2022, pp. 636–654.

[25] Y. Wang, L. Rahmann, and O. Sorkine-hornung, "Geometry-consistent neural shape representation with implicit displacement fields," in *ICLR*, 2021.

[26] C. Li, Y. Xin, G. Liu, X. Zeng, and L. Liu, "NeRF synthesis with shading guidance," *arXiv preprint arXiv:2306.11556*, 2023.

[27] C. Sun, M. Sun, and H.-T. Chen, "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction," in *CVPR*, 2022, pp. 5459–5469.

[28] L. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," in *Eurographics*, 2009, pp. 93–117.

[29] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *ICCV*, 1999, pp. 1033–1038.

[30] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 2000, p. 479–488.

[31] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Trans. Graph.*, vol. 20, no. 3, p. 127–150, 2001.

[32] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 2001, p. 341–346.

[33] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Trans. Graph.*, vol. 22, no. 3, p. 277–286, 2003.

[34] S.-M. Hu, F.-L. Zhang, M. Wang, R. R. Martin, and J. Wang, "PatchNet: A patch-based image representation for interactive library-driven image editing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–12, 2013.

[35] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," in *Proceedings of the 32th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 2005, p. 795–802.

[36] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *NeurIPS*, 2015, pp. 262–270.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[38] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images," in *ICML*, vol. 48, 2016, pp. 1349–1357.

[39] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016, pp. 694–711.

[40] N. Jetchev, U. Bergmann, and R. Vollgraf, "Texture synthesis with spatial generative adversarial networks," in *Workshop on Adversarial Training, NeurIPS*, 2016.

[41] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *ECCV*, 2016, pp. 702–716.

[42] Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang, "Non-stationary texture synthesis by adversarial expansion," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–13, 2018.

[43] T. Portenier, S. A. Bigdeli, and O. Goksel, "GramGAN: Deep 3d texture synthesis from 2d exemplars," in *NeurIPS*, 2020, pp. 6994–7004.

[44] A. Hertz, R. Hanocka, R. Giryes, and D. Cohen-Or, "Deep geometric texture synthesis," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 1–11, 2020.

[45] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "MeshCNN: A network with an edge," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, 2019.

[46] U. Bergmann, N. Jetchev, and R. Vollgraf, "Learning texture manifolds with the periodic spatial GAN," in *ICML*, 2017, p. 469–477.

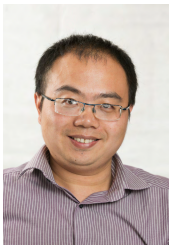
[47] H. Chen, J. Liu, W. Chen, S. Liu, and Y. Zhao, "Exemplar-based pattern synthesis with implicit periodic field network," in *CVPR*, 2022, pp. 3708–3717.

[48] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 1987, p. 163–169.

[49] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *CVPR*, 2016, pp. 4104–4113.
[50] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *ECCV*, 2016, pp. 501–518.
[51] J. Huang, H. Su, and L. Guibas, "Robust watertight manifold surface generation method for ShapeNet models," *arXiv preprint arXiv:1802.01698*, 2018.
[52] J. Vollmer, R. Mencl, and H. Mueller, "Improved Laplacian smoothing of noisy surface meshes," in *Comput. Graph. Forum*, vol. 18, no. 3, 1999, pp. 131–138.
[53] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM national conference*, 1968, pp. 517–524.
[54] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron, "NeRFactor: Neural factorization of shape and reflectance under an unknown illumination," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 1–18, 2021.
[55] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "BARF: Bundle-adjusting neural radiance fields," in *ICCV*, 2021, pp. 5741–5751.
[56] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "NeRF−−: Neural radiance fields without known camera parameters," *arXiv preprint arXiv:2102.07064*, 2021.
[57] T. Müller, "Tiny CUDA neural network framework," 2021, https://github.com/nvlabs/tiny-cuda-nn.
[58] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *NeurIPS*, 2020, pp. 7537–7547.
[59] R. Ramamoorthi and P. Hanrahan, "A signal-processing framework for inverse rendering," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 2001, p. 117–128.
[60] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, 1975.
[61] R. L. Cook, "Stochastic sampling in computer graphics," *ACM Trans. Graph.*, vol. 5, no. 1, p. 51–72, 1986.
[62] G. Turk, "Texture synthesis on surfaces," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 2001, p. 347–354.
[63] L.-Y. Wei and M. Levoy, "Texture synthesis over arbitrary manifold surfaces," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 2001, p. 355–360.
[64] E. Praun, A. Finkelstein, and H. Hoppe, "Lapped textures," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 2000, p. 465–470.
[65] C. Soler, M.-P. Cani, and A. Angelidis, "Hierarchical pattern mapping," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 2002, p. 673–680.
[66] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM Trans. Graph.*, vol. 21, no. 3, p. 362–371, 2002.
[67] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe, "Texture mapping progressive meshes," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 2001, pp. 409–416.
[68] E. Zhang, K. Mischaikow, and G. Turk, "Vector field design on surfaces," *ACM Trans. Graph.*, vol. 25, no. 4, p. 1294–1326, 2006.
[69] N. Sharp, Y. Soliman, and K. Crane, "The vector heat method," *ACM Trans. Graph.*, vol. 38, no. 3, pp. 1–19, 2019.
[70] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *ICML*, 2016, pp. 478–487.
[71] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.
[72] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.
[74] 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 *CVPR*, 2022, pp. 5481–5490.
[75] A. Torrence, "Martin newell's original teapot," in *ACM SIGGRAPH*, 2006, pp. 29–es.
[76] K. Crane, U. Pinkall, and P. Schröder, "Robust fairing via conformal curvature flow," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–10, 2013.
[77] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *ACM SIGGRAPH*, 1994, pp. 311–318.
[78] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, "Large-scale data for multiple-view stereopsis," *IJCV*, vol. 120, no. 2, pp. 153–168, 2016.
[79] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, R. Basri, and Y. Lipman, "Multiview neural surface reconstruction by disentangling geometry and appearance," in *NeurIPS*, 2020, pp. 2492–2502.
[80] T. R. Shaham, T. Dekel, and T. Michaeli, "SinGAN: Learning a generative model from a single natural image," in *ICCV*, 2019, pp. 4570–4580.
[81] R. Hoetzlein, "Fast fixed-radius nearest neighbors: Interactive million-particle fluids," in *GPU Technology Conference (GTC)*, 2014, pp. 1–25.
[82] H. J. Haverkort, "Introduction to bounding volume hierarchies," in *Part of the PhD Thesis, Utrecht University*, 2004.
[83] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The r*-tree: An efficient and robust access method for points and rectangles," in *ACM SIGMOD*, 1990, pp. 322–331.
[84] D. Meister, S. Ogaki, C. Benthin, M. J. Doyle, M. Guthe, and J. Bittner, "A survey on bounding volume hierarchies for ray tracing," in *Comput. Graph. Forum*, vol. 40, no. 2, 2021, pp. 683–712.
[85] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *NeurIPS*, 2020, pp. 6840–6851.

Yi-Hua Huang obtained his bachelor's and master's degrees from the University of Chinese Academy of Sciences. He is currently a Ph.D. student at the University of Hong Kong. His research interests include computer graphics and visions.

Yan-Pei Cao Yan-Pei Cao received his bachelor's and Ph.D. degrees in computer science from Tsinghua University in 2013 and 2018, respectively. He is currently the Head of Research and co-founder at VAST. His research interests include computer graphics and 3D computer vision.



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 *IEEE Transactions on Visualization and Computer Graphics* and *The Visual Computer*.

Ying Shan received the bachelor's degree from Zhejiang University in 1990, and the MS and Ph.D. degrees in computer science from Shanghai Jiaotong University in 1993 and 1997, respectively. Ying Shan is a Distinguished Scientist at Tencent, and the Director of the Tencent ARC Lab and Tencent AI Lab CVC. Before joining Tencent, he worked at Microsoft Research as a post-doc researcher, SRI International (Sarnoff Subsidiary) as a Senior MTS, and Microsoft Bing Ads as a Principal Scientist Manager. He has published over 70 papers in top conferences and journals in the areas of computer vision, machine learning, and data mining, served as ACs of CVPR and senior PC of KDD, and holds a number of US/International patents. He is currently leading R&D efforts in web search, and content AI for a suite of social media and content distribution products.

Lin Gao received the PhD degree in computer science from Tsinghua University. He is currently a Professor at the University of 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.