NeRF-Texture: Texture Synthesis with Neural Radiance Fields

Yi-Hua Huang
Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
China
huangyihua16@mails.ucas.ac.cn

Yan-Pei Cao
ARC Lab, Tencent PCG
China
caoyanpei@gmail.com

Yu-Kun Lai
School of Computer Science and Informatics, Cardiff University
United Kingdom
LaiY4@cardiff.ac.uk

Ying Shan
ARC Lab, Tencent PCG
China
yingsshan@tencent.com

Lin Gao*
Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, CAS and UCAS
China
gaolin@ict.ac.cn

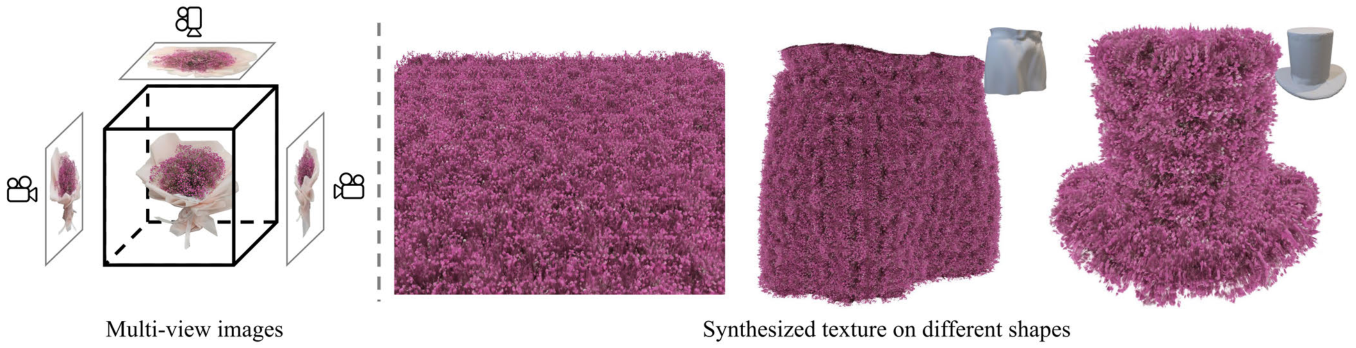

Figure 1: Given a set of multi-view images of the target texture with meso-structure, our model synthesizes Neural Radiance Field (NeRF) textures, which can then be applied to novel shapes, such as the skirt and hat in the figure, with rich geometric and appearance details.

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. Experimental results and evaluations demonstrate the effectiveness of our approach.

*Corresponding author is Lin Gao (gaolin@ict.ac.cn).

CCS CONCEPTS

• **Computing methodologies** → *Image-based rendering*.

KEYWORDS

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 [Baatz et al. 2022], 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 [Kuznetsov et al. 2022].

Thanks to recently proposed neural implicit rendering approaches such as NeRF (Neural Radiance Fields) [Mildenhall et al. 2021], 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 [Bao and Yang et al. 2022] and NeuTex [Xiang et al. 2021] make an attempt to disentangle the texture and geometry. NeuMesh [Bao and Yang et al. 2022] supports geometry and appearance editing but is incapable of modeling and synthesizing meso-structure textures; NeuTex [Xiang et al. 2021] 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 [Müller et al. 2022] and Co-ACD [Wei et al. 2022]. We then propose to map each point in the 3D Euclidean space to the Cartesian product of the signed distance and its foot 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 [Bao and Yang et al. 2022], 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 [Müller et al. 2022] 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 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. 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 [Mildenhall et al. 2021] 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 [Fridovich-Keil et al. 2022; Karnewar et al. 2022; Müller et al. 2022], handle large-scale scenes [Barron et al. 2022; Gao et al. 2023; Tancik et al. 2022; Zhang et al. 2020] and dynamic scenes [Liu et al. 2022; Qiao et al. 2022; Song et al. 2023], and attain reflection decomposition [Boss et al. 2021; Kuang et al. 2022; Munkberg et al. 2022; Srinivasan et al. 2021] and stylization [Fan et al. 2022; Huang et al. 2022; Zhang et al. 2022]. Some other neural representations have been proposed to model meso-scale textures. Kuznetsov et al. [2022] utilize neural bidirectional texture functions (BTFs) to model known texture with

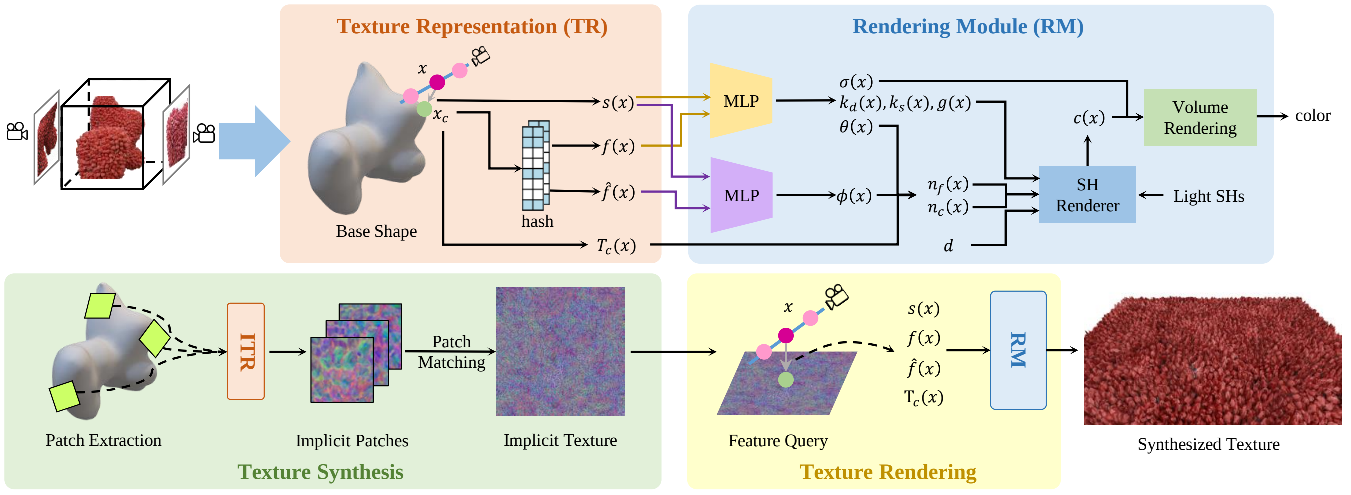

Figure 2: Given a set of multi-view images, we first estimate its base shape. Based on it, we model the scene with a disentangled representation of the base shape and NeRF texture with meso-structure. The query point x is projected onto the base shape as footpoint x_c. Latent features $f(x), \hat{f}(x)$ 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.

meso-structure. Wang et al. [2021a] 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.

NeuTex [Xiang et al. 2021] 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 [Bao and Yang et al. 2022] 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 [Baatz et al. 2022] 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 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 [Wei et al. 2009]. The pioneering work by [Efros and Leung 1999] 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 [Wei and Levoy 2000] to avoid non-uniform pattern distribution. Patch-based method [Liang et al. 2001] proposes to blend the overlapped regions between patches. The works [Efros and Freeman 2001; Kwatra et al. 2003] cut through the overlapped regions via dynamic programming and graph cut, respectively. PatchNet [Hu et al. 2013] searches an image library to locate ideal regions adhering to the synthesis constraints. Kwatra et al. [2005] 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. [2015] present a data-driven approach to generating texture through optimizing the Gram matrix of latent features extracted by VGG network [Simonyan and Zisserman 2015]. Follow-up works [Johnson et al. 2016; Ulyanov et al. 2016] train feed-forward convolutional networks to replace the time-consuming optimization process. Generative adversarial networks (GANs) are also widely used for texture synthesis [Jetchev et al. 2016; Li and Wand 2016]. Zhou et al. [2018] train a GAN to double the spatial extent of texture blocks, enabling the model to synthesize non-stationary texture. Portenier et al. [2020] use the Gram matrix produced by the discriminator in adversarial loss to improve the quality of synthesized texture. Hertz et al. [2020] propose a Mesh-CNN [Hanocka et al. 2019] based GAN architecture to synthesize geometric textures. PSGAN [Bergmann et al. 2017] proves that periodic encoding can improve the quality of GAN results. Inspired by it, Chen et al. [2022] 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 [Müller et al. 2022] to reconstruct the coarse mesh by executing Marching Cubes [Lorensen and Cline 1987] on the estimated density field with camera parameters estimated by COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016]. 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 Co-ACD [Wei et al. 2022]. The shape is then re-meshed [Huang et al. 2018; Vollmer et al. 1999] to a mesh with vertices uniformly distributed on the surface. Fig. 3 illustrates the process of base shape extraction.

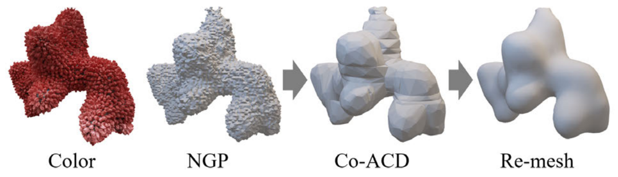

Figure 3: Base Shape Extraction. We show the intermediate outputs during the base shape extraction, including NGP [Müller et al. 2022], Co-ACD [Wei et al. 2022], and re-meshing [Huang et al. 2018; Vollmer et al. 1999].

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 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 [Shepard 1968] to get the coarse mesh normal n_c of x:

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

where w is a constant set to 0.01. Next, we cast the ray from x along the opposite coarse normal direction $-n_c(x)$ to hit the coarse mesh on a projected point x_c. The first term in Eq. 1 is the weighted average of normals from the K nearest neighbors to improve robustness. When x is far from the coarse mesh, normals of K neighbors may be less reliable so the second term becomes dominant to ensure the ray-mesh collision. At this step, the signed distance of x projected onto the coarse mesh is also obtained and denoted as $s(x)$.

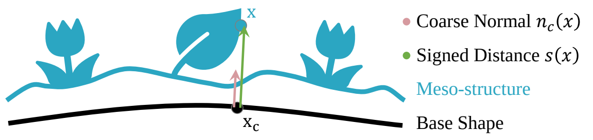

Figure 4: Illustration of Base Shape Projection in 2D. Point x in Euclidean space is parameterized as the signed distance $s(x)$ and the projected footpoint x_c.

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 [Boss et al. 2021; Srinivasan et al. 2021] or supervise the normal estimation [Kuang et al. 2022; Zhang et al. 2021] for physically based rendering. In addition, back-propagating gradients to the camera parameters via coordinates x is crucial for camera pose modification [Kuang et al. 2022; Lin et al. 2021; Wang et al. 2021b] 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 [Bao and Yang et al. 2022], 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 [Müller et al. 2022]. Through the tiny-cuda-nn framework [Müller 2021], we map the concatenated texture feature $f(x)$ and Fourier embedded [Tancik et al. 2020] 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) [Ramamoorthi and Hanrahan 2001] to represent illumination and materials in our rendering pipeline. We adopt Phong shading [Phong 1975] to model the material reflection with three parameters: diffuse coefficient k_d, specular coefficient k_s, and glossiness g. Following the approach outlined in [Ramamoorthi and Hanrahan 2001], we employ the convolution of SHs to compute the texture color $c(x)$. The decomposition is illustrated in Fig. 5.

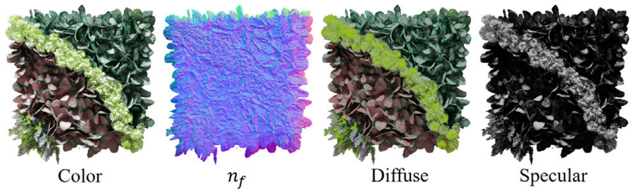

Figure 5: Shading Decomposition. Our model predicts the fine normal n_f and decomposes the radiance into diffuse and specular components.

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 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 then query the hash grids with these intersections to fetch latent features and obtain implicit patches. We 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.

3.2.2 Patch-based Synthesis. We synthesize textures of arbitrary sizes based on the sampled exemplars using patch matching and quilting [Efros and Freeman 2001]. 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

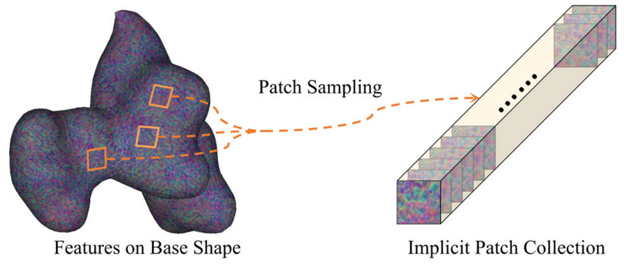

Figure 6: Texture Patch Extraction. We extract implicit texture patches by sampling them on the base shape, where latent features are defined.

it. 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. 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.

3.2.3 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 texture contents are represented by similar latent features. 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 [Xie et al. 2016] for latent features into the optimization objective. Student's τ-distribution is used as the kernel to measure the similarity [Van der Maaten and Hinton 2008] between latent features f_i and trainable cluster centers μ_j. The distribution Q and its hardened auxiliary distribution P are defined as:

$$q_{ij} = \frac{(1 + ||f_i - \mu_j||_2^2/\kappa)^{-\frac{\kappa+1}{2}}}{\sum_{j'}(1 + ||f_i - \mu_{j'}||_2^2/\kappa)^{-\frac{\kappa+1}{2}}},\ p_{ij} = \frac{q_{ij}^2/\sum_i q_{ij}}{\sum_{j'}(q_{ij'}^2/\sum_i q_{ij'})} \qquad (3)$$

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 [Kullback and Leibler 1951] between them: $L_{clu} = KL(P||Q)$. For hash grids at each resolution level, we cluster the embedding features with the clustering loss.

3.2.4 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

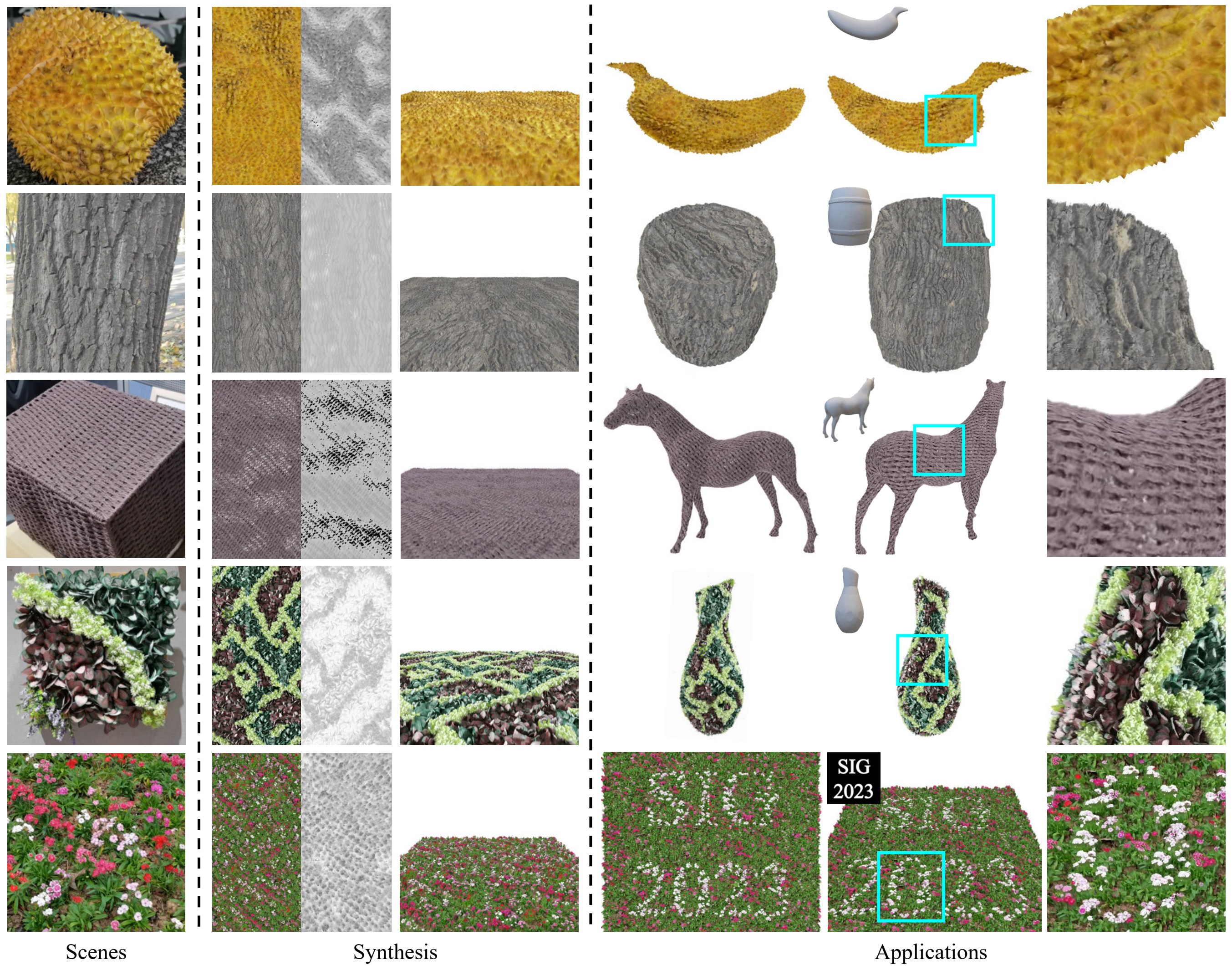

Figure 7: Texture synthesis and applications. We show the synthesized textures of durian, tree bark, woven basket, leaves, and flowers. The textures are also applied to different shapes. The last example is constrained synthesis guided by the text image.

$T_s^{-1}(x)T_c(x)$ 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:

$$\begin{aligned} \tilde{n}_f(x) &= \tilde{T}_c(x)T_s^{-1}(x)T_c(x)R(\theta(x), \phi(x)), \\ R(\theta, \phi) &= (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)^T \end{aligned} \tag{4}$$

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 [Kingma and Ba 2015]. 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 [Barron et al. 2022] 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:

$$L_{nor} = -\cos\{\min(\langle< -\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x}, n_f(x)\rangle >, \frac{\pi}{8})\} \tag{5}$$

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

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**	<u>42.602</u>	<u>0.996</u>	<u>0.049</u>	**38.247**	**0.991**	<u>0.085</u>	<u>41.976</u>	<u>0.996</u>	<u>0.057</u>
Ours	<u>32.378</u>	<u>0.988</u>	<u>0.104</u>	**43.842**	**0.998**	**0.027**	<u>36.809</u>	<u>0.990</u>	**0.067**	**42.704**	**0.998**	**0.031**

2D texture and our representation in Sec. 4.3, which indicates 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. 7. The target texture includes durian, tree 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 the 4th-6th columns. We synthesize the durian's texture with thorns and transfer it to a banana. The tree bark is usually covered with stripes of ravines. We synthesize and apply such texture to a barrel shape and obtain a wooden barrel. We capture fabric texture on a woven basket and construct a woven horse. Leaves and grass are also typical textures in nature; we synthesize an ocean of leaves and grass and apply it to a vase. We also synthesize colorful flowers guided by the shown text image, by considering the rendered color of patches during texture synthesis (see supplementary for details). The zoomed-in view in the 6th column shows the effect of the material on oblique views and object edges, where 2D textures appear unrealistic due to the lack of meso-structure modeling, demonstrating the advantages of our method over 2D textures.

4.2 View synthesis quality

We evaluate the view synthesis quality of our method on the published dataset DTU [Aanæs et al. 2016], 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 [Yariv et al. 2020]. In each scene, 5 images are randomly picked as the test set. Qualitative comparison with NeRF [Mildenhall et al. 2021], Instant-NGP (NGP) [Müller et al. 2022] and ours is shown in Fig. 10. 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.

Single Image Fréchet Inception Distance (SIFID). SIFID introduced in [Shaham et al. 2019] 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 NeRF 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. 8 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.

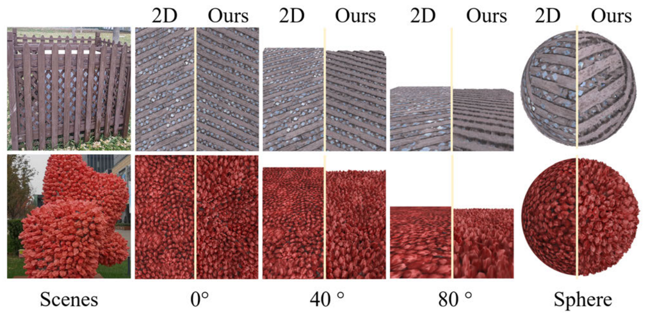

Figure 8: Qualitative comparison with 2D textures. We show the rendering results of our synthesized textures and 2D textures. Our representation of texture with meso-structure maintains realism even at high-elevation viewing angles.

4.4 Comparison with NeRF-Tex

We present a visual comparison between NeRF-Tex [Baatz et al. 2022] and our proposed method, as demonstrated in Fig. 9. 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 structural textures. Besides, it is crucial to note that NeRF-Tex trains a NeRF using synthetic data with known tightly-bounded planar geometry, which cannot be directly applied to real-world data. Thus, we utilize our coarse-fine disentanglement representation to generate multi-view images of real-world meso-structure textures within a bounding box, serving as training data for NeRF-Tex.

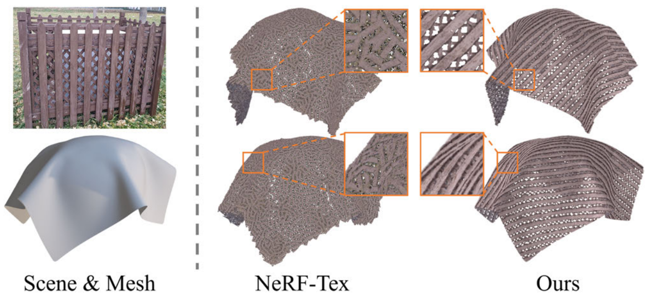

Figure 9: Compared with NeRF-Tex [Baatz et al. 2022], our method demonstrates superior preservation of texture continuity and structure.

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 to be represented by close features. We visualize the synthesized feature maps with and without the constraint by Principal Component Analysis dimensionality reduction to 3D, which is further visualized as RGB channels in Fig. 11 (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. 11 (right)).

5 LIMITATION

We analyze the performance of our method on different examples and summarize two aspects of limitations. **1)** Limitation on texture capture. Due to the requirement of base shape for texture representation, our approach is limited by the base shape extraction and fails to capture meso-structure textures from objects with complex geometry like the Lego shown in Fig. 12, where the detailed surface is difficult to approximate with a coarse mesh. It is also challenging to extract complete implicit patches on regions with limited spatial extents of the base surface, like the perforated structure on the Lego loader, for subsequent synthesis. On the contrary, our method can easily extract and synthesize the bump texture on the Lego base. **2)** Limitation on texture synthesis. Our patch-matching approach follows a greedy strategy to select patches for quilting, which may cause the break of structures for textures requiring strict matching with only limited patches, as shown in the 1st row of Fig. 13. Besides, our synthesis algorithm is semantically agnostic. Semantic contents like keycap shapes may be distorted, and synthesized characters may be incorrect, as shown in the 2nd row of Fig. 13. Our approach could potentially incorporate recent generative techniques, such as Diffusion Models [Ho et al. 2020], to address the limitation.

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. A clustering constraint regularizes the distribution of latent features for better synthesis.

ACKNOWLEDGMENTS

This work was supported by grants from the Beijing Municipal Natural Science Foundation for Distinguished Young Scholars (No. JQ21013), the National Natural Science Foundation of China (No. 62061136007), the Royal Society Newton Advanced Fellowship (No. NAF\R2\192151) and the Youth Innovation Promotion Association CAS.

REFERENCES

Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. 2016. Large-Scale Data for Multiple-View Stereopsis. *IJCV* 120, 2 (2016), 153–168.

Hendrik Baatz, Jonathan Granskog, Marios Papas, Fabrice Rousselle, and Jan Novák. 2022. NeRF-Tex: Neural Reflectance Field Textures. In *Comput. Graph. Forum.* 287–301.

Bao and Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. 2022. NeuMesh: Learning Disentangled Neural Mesh-based Implicit Field for Geometry and Texture Editing. In *ECCV.* 597–614.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR.* 5470–5479.

Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. 2017. Learning Texture Manifolds with the Periodic Spatial GAN. In *ICML.* 469–477.

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

Haiwei Chen, Jiayi Liu, Weikai Chen, Shichen Liu, and Yajie Zhao. 2022. Exemplar-Based Pattern Synthesis With Implicit Periodic Field Network. In *CVPR.* 3708–3717.

Alexei A. Efros and William T. Freeman. 2001. Image Quilting for Texture Synthesis and Transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH.* 341–346.

Alexei A Efros and Thomas K Leung. 1999. Texture synthesis by non-parametric sampling. In *ICCV.* 1033–1038.

Zhiwen Fan, Yifan Jiang, Peihao Wang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. 2022. Unified Implicit Neural Stylization. In *ECCV.* 636–654.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields Without Neural Networks. In *CVPR.* 5501–5510.

Yiming Gao, Yan-Pei Cao, and Ying Shan. 2023. SurfelNeRF: Neural Surfel Radiance Fields for Online Photorealistic Reconstruction of Indoor Scenes. *arXiv preprint arXiv:2304.08971* (2023).

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. Texture Synthesis Using Convolutional Neural Networks. In *NeurIPS.* 262–270.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: A Network with an Edge. *ACM Trans. Graph.* 38, 4 (2019), 1–12.

Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. 2020. Deep Geometric Texture Synthesis. *ACM Trans. Graph.* 39, 4 (2020), 1–11.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *NeurIPS.* 6840–6851.

Shi-Min Hu, Fang-Lue Zhang, Miao Wang, Ralph R. Martin, and Jue Wang. 2013. PatchNet: A Patch-Based Image Representation for Interactive Library-Driven Image Editing. *ACM Trans. Graph.* 32, 6 (2013), 1–12.

Jingwei Huang, Hao Su, and Leonidas Guibas. 2018. Robust Watertight Manifold Surface Generation Method for ShapeNet Models. *arXiv preprint arXiv:1802.01698* (2018).

Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. 2022. StylizedNeRF: consistent 3D scene stylization as stylized NeRF via 2D-3D mutual learning. In *CVPR*. 18342–18352.

Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. 2016. Texture synthesis with spatial generative adversarial networks. In *Workshop on Adversarial Training, NeurIPS*.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*. 694–711.

Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. 2022. ReLU Fields: The Little Non-Linearity That Could. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Zhengfei Kuang, Kyle Olszewski, Menglei Chai, Zeng Huang, Panos Achlioptas, and Sergey Tulyakov. 2022. NeROIC: Neural Rendering of Objects from Online Image Collections. *ACM Trans. Graph.* 41, 4 (2022), 1–12.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

Alexandr Kuznetsov, Xuezheng Wang, Krishna Mullia, Fujun Luan, Zexiang Xu, Milos Hasan, and Ravi Ramamoorthi. 2022. Rendering Neural Materials on Curved Surfaces. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.

Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. 2005. Texture Optimization for Example-Based Synthesis. In *Proceedings of the 32th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*. 795–802.

Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. 2003. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. *ACM Trans. Graph.* 22, 3 (2003), 277–286.

Chuan Li and Michael Wand. 2016. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*. 702–716.

Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. 2001. Real-Time Texture Synthesis by Patch-Based Sampling. *ACM Trans. Graph.* 20, 3 (2001), 127–150.

Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. 2021. Barf: Bundle-adjusting neural radiance fields. In *ICCV*. 5741–5751.

Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. 2022. DeVRF: Fast Deformable Voxel Radiance Fields for Dynamic Scenes. In *NeurIPS*. 36762–36775.

William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*. 163–169.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Thomas Müller. 2021. Tiny CUDA Neural Network Framework. https://github.com/nvlabs/tiny-cuda-nn.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4 (2022), 1–15.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *CVPR*. 8280–8290.

Bui Tuong Phong. 1975. Illumination for computer generated pictures. *Commun. ACM* 18, 6 (1975), 311–317.

Tiziano Portenier, Siavash Arjomand Bigdeli, and Orcun Goksel. 2020. GramGAN: Deep 3D Texture Synthesis From 2D Exemplars. In *NeurIPS*. 6994–7004.

Yi-Ling Qiao, Alexander Gao, and Ming Lin. 2022. NeuPhysics: Editable Neural Geometry and Physics from Monocular Videos. In *NeurIPS*. 12841–12854.

Ravi Ramamoorthi and Pat Hanrahan. 2001. A Signal-Processing Framework for Inverse Rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*. 117–128.

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *CVPR*. 4104–4113.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *ECCV*. 501–518.

Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. 2019. SinGAN: Learning a generative model from a single natural image. In *ICCV*. 4570–4580.

Donald Shepard. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*. 517–524.

Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2023. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE TVCG* 29, 5 (2023), 2732–2742.

Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. 2021. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*. 7495–7504.

Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. 2022. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8248–8258.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *NeurIPS*. 7537–7547.

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. 2016. Texture Networks: Feed-forward Synthesis of Textures and Stylized Images. In *ICML*, Vol. 48. 1349–1357.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.

Jörg Vollmer, Robert Mencl, and Heinrich Mueller. 1999. Improved Laplacian smoothing of noisy surface meshes. In *Comput. Graph. Forum*, Vol. 18. 131–138.

Yifan Wang, Lukas Rahmann, and Olga Sorkine-hornung. 2021a. Geometry-Consistent Neural Shape Representation with Implicit Displacement Fields. In *ICLR*.

Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. 2021b. NeRF−−: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064* (2021).

Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. 2009. State of the Art in Example-based Texture Synthesis. In *Eurographics*. 93–117.

Li-Yi Wei and Marc Levoy. 2000. Fast Texture Synthesis Using Tree-Structured Vector Quantization. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*. 479–488.

Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. 2022. Approximate Convex Decomposition for 3D Meshes with Collision-Aware Concavity and Tree Search. *ACM Trans. Graph.* 41, 4 (2022), 1–18.

Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. 2021. NeuTex: Neural texture mapping for volumetric neural rendering. In *CVPR*. 7119–7128.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *ICML*. 478–487.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. 2020. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In *NeurIPS*. 2492–2502.

Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. 2022. ARF: Artistic Radiance Fields. In *ECCV*. 717–733.

Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492* (2020).

Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. 2021. NeRFactor: Neural Factorization of Shape and Reflectance under an Unknown Illumination. *ACM Trans. Graph.* 40, 6 (2021), 1–18.

Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2018. Non-Stationary Texture Synthesis by Adversarial Expansion. *ACM Trans. Graph.* 37, 4 (2018), 1–13.

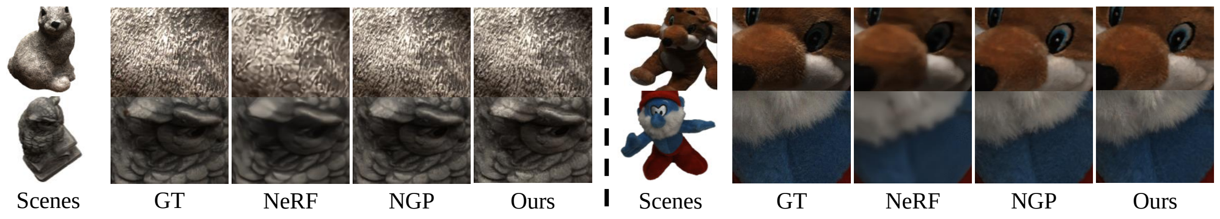

Figure 10: Qualitative comparison of view synthesis. Note that our method supports texture capture, synthesis and application while visually close to the state of the arts.

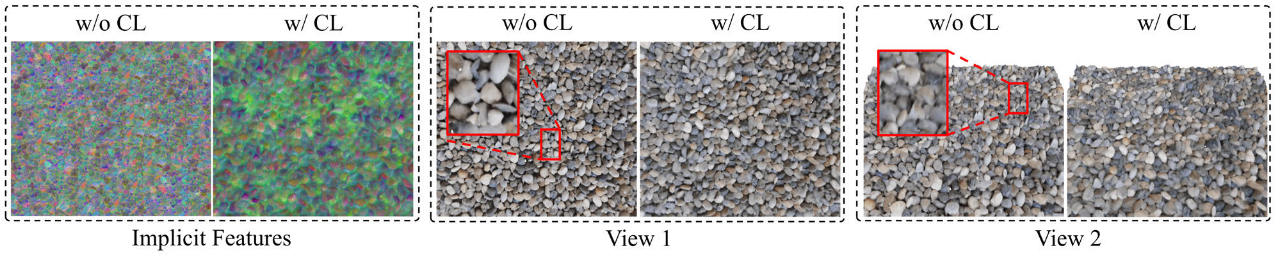

Figure 11: Impact of clustering constraint. With the clustering loss (w/ CL), latent features are constrained to cluster, which reduces the distance of latent features corresponding to similar textures and further reduces artifacts in synthesized results.

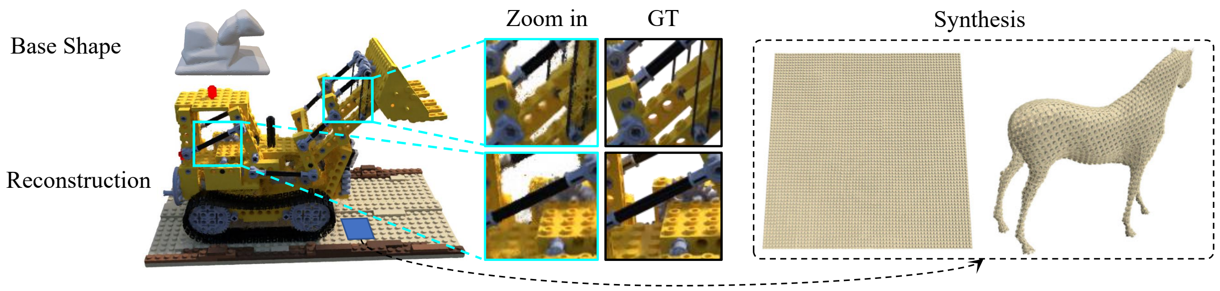

Figure 12: Limitation on texture capture. Our approach fails to reconstruct and capture textures on shapes with complex coarse geometry due to the difficulty in base shape estimation and patch sampling on regions with limited spatial extents of the base surface. On the contrary, the bump texture on the Lego base can be easily acquired and synthesized.

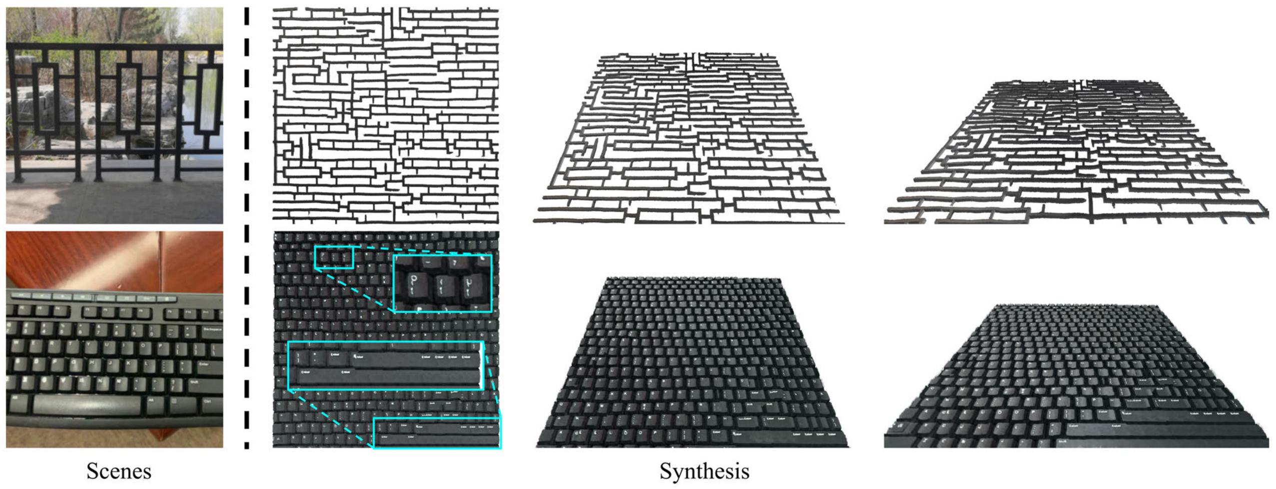

Figure 13: Limitation on texture synthesis. Our synthesis approach based on patch matching struggles to synthesize highly structured textures requiring strict matching with a few captured exemplars while maintaining the semantic contents of textures.