NeuTex: Neural Texture Mapping for Volumetric Neural Rendering

Fanbo Xiang[1], Zexiang Xu[2], Miloš Hašan[2], Yannick Hold-Geoffroy[2], Kalyan Sunkavalli[2], Hao Su[1]

[1] University of California, San Diego
[2] Adobe Research

Abstract

Recent work [28, 5] has demonstrated that volumetric scene representations combined with differentiable volume rendering can enable photo-realistic rendering for challenging scenes that mesh reconstruction fails on. However, these methods entangle geometry and appearance in a "black-box" volume that cannot be edited. Instead, we present an approach that explicitly disentangles geometry—represented as a continuous 3D volume—from appearance—represented as a continuous 2D texture map. We achieve this by introducing a 3D-to-2D texture mapping (or surface parameterization) network into volumetric representations. We constrain this texture mapping network using an additional 2D-to-3D inverse mapping network and a novel cycle consistency loss to make 3D surface points map to 2D texture points that map back to the original 3D points. We demonstrate that this representation can be reconstructed using only multi-view image supervision and generates high-quality rendering results. More importantly, by separating geometry and texture, we allow users to edit appearance by simply editing 2D texture maps.

1. Introduction

Capturing and modeling real scenes from image inputs is an extensively studied problem in vision and graphics. One crucial goal of this task is to avoid the tedious manual 3D modeling process and directly provide a renderable and editable 3D model that can be used for realistic rendering in applications, like e-commerce, VR and AR. Traditional 3D reconstruction methods [38, 39, 20] usually reconstruct objects as meshes. Meshes are widely used in rendering pipelines; they are typically combined with mapped textures for appearance editing in 3D modeling pipelines.

However, mesh-based reconstruction is particularly challenging and often cannot synthesize highly realistic images for complex objects. Recently, various neural scene representations have been presented to address this scene acquisition task. Arguably the best visual quality is obtained by approaches like NeRF [28] and Deep Reflectance Volumes [5] that leverage differentiable volume rendering (ray marching). However, these volume-based methods do not (explicitly) reason about the object's surface and entangle both geometry and appearance in a volume-encoding neural network. This does not allow for easy editing—as is possible with a texture mapped mesh—and significantly limits the practicality of these neural rendering approaches.

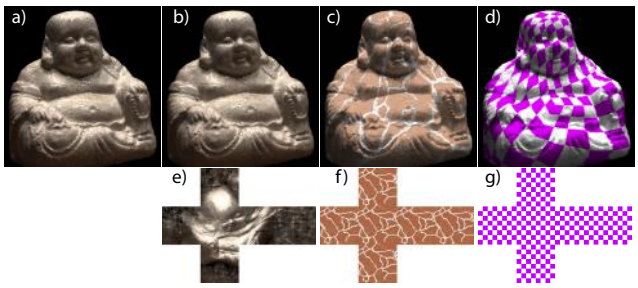


Figure 1. NeuTex is a neural scene representation that represents geometry as a 3D volume but appearance as a 2D neural texture in an automatically discovered texture UV space, shown as a cube-map in (e). NeuTex can synthesize highly realistic images (b) that are very close to the ground-truth (a). Moreover, it enables intuitive surface appearance editing directly in the 2D texture space; we show an example of this in (c), by using a new texture (f) to modulate the reconstructed texture. Our discovered texture mapping covers the object surface uniformly, as illustrated in (d), by rendering the object using a uniform checkerboard texture (g).

Our goal is to make volumetric neural reconstruction more practical by enabling both realistic image synthesis and flexible surface appearance editing. To this end, we present NeuTex—an approach that explicitly disentangles scene geometry from appearance. NeuTex represents geometry with a volumetric representation (similar to NeRF) but *represents surface appearance using 2D texture maps*. This allows us to leverage differentiable volume rendering to reconstruct the scene from multi-view images, while allowing

Research partially done When F. Xiang was an intern at Adobe Research.

for conventional texture-editing operations (see Fig. 1).

As in NeRF [28], we march a ray through each pixel, regress volume density and radiance (using fully connected MLPs) at sampled 3D shading points on the ray, accumulate the per-point radiance values to compute the final pixel color. NeRF uses a single MLP to regress both density and radiance in a 3D volume. While we retain this volumetric density-based representation for geometry, NeuTex represents radiance in a 2D (UV) texture space. In particular, we train a *texture mapping* MLP to regress a 2D UV coordinate at every 3D point in the scene, and use another MLP to regress radiance in the 2D texture space for any UV location. Thus, given any 3D shading point in ray marching, our network can obtain its radiance by sampling the reconstructed neural texture at its mapped UV location.

Naively adding a texture mapping network to NeRF (and supervising only with a rendering loss) leads to a degenerate texture mapping that does not unwrap the surface and cannot support texture editing (see Fig. 3). To ensure that the estimated texture space reasonably represents the object's 2D surface, we introduce a novel cycle consistency loss. Specifically, we consider the shading points that contribute predominantly to the pixel color along a given ray, and correspond to the points either on or close to the surface. We train an additional *inverse mapping* MLP to map the 2D UV coordinates of these high-contribution points *back* to their 3D locations. Introducing this inverse-mapping network forces our model to learn a consistent mapping (similar to a one-to-one correspondence) between the 2D UV coordinates and the 3D points on the object surface. This additionally regularizes the surface reasoning and texture space discovery process. As can be seen in Fig. 1, our full model recovers a reasonable texture space, that can support realistic rendering similar to previous work while also allowing for intuitive appearance editing.

Our technique can be incorporated into different volume rendering frameworks. In addition to NeRF, we show that it can be combined with Neural Reflectance Fields [4] to reconstruct BRDF parameters as 2D texture maps (see Fig. 6), enabling both view synthesis and relighting.

Naturally, NeuTex is more constrained than a fully-volumetric method; this leads to our final rendering quality to be on par or slightly worse than NeRF [28]. Nonetheless, we demonstrate that our approach can still synthesize photo-realistic images and significantly outperform both traditional mesh-based reconstruction methods [39] and previous neural rendering methods [41, 40]. Most importantly, our work is the first to recover a meaningful surface-aware texture parameterization of a scene and enable surface appearance editing applications (as in Fig. 1 and 5). This, we believe, is an important step towards making neural rendering methods useful in 3D design workflows.

2. Related Work

Scene representations. Deep learning based methods have explored various classical scene representations, including volumes [18, 47, 35, 40], point clouds [34, 2, 45], meshes [19, 46], depth maps [22, 17] and implicit functions [9, 27, 29, 51]. However, most of them focus on geometry reconstruction and understanding and do not aim to perform realistic image synthesis. We leverage volumetric neural rendering [28, 4] for realistic rendering; our method achieves higher rendering quality than other neural rendering methods [40, 41].

Mesh-based reconstruction and rendering. 3D polygonal meshes are one of the most popular geometry representations, widely used in 3D modeling and rendering pipelines. Numerous traditional 3D reconstruction techniques have been proposed to directly reconstruct a mesh from multiple captured images, including structure from motion [38], multi-view stereo [12, 21, 39], and surface extraction [25, 20]. Recently, many deep learning based methods [44, 50, 42, 7, 10] have also been proposed, improving the reconstruction quality in many of these techniques. In spite of these advances, it is still challenging to reconstruct a mesh that can directly be used to synthesize photo-realistic images. In fact, many image-based rendering techniques [6, 3, 15] have been presented to fix the rendering artifacts from mesh reconstruction; however, they often leverage view-dependent texture maps [11], which cannot be easily edited. We instead leverage volumetric neural rendering to achieve realistic image synthesis; our approach explicitly extracts surface appearance as view-independent textures, just like standard textures used with meshes, allowing for broad texture editing applications in 3D modeling and content generation.

Neural rendering. Recently, deep learning-based methods have proposed to ameliorate or completely bypass mesh reconstruction to achieve realistic neural renderings of real scenes for view synthesis [52, 48, 41, 40], relighting [49, 32, 8], and many other image synthesis tasks [24]. In particular, NeRF [28], Deep Reflectance Volumes [5] and other relevant works [4, 23] model a scene using neural volumetric representations (that encode geometry and appearance) and leverage differentiable volume rendering [26] to synthesize highly photo-realistic images. However, these volume representations do not explicitly reason about the 2D surface of a scene and are essentially "black-box" functions that cannot be easily modified after reconstruction. In contrast, we introduce a novel neural scene representation that offers direct access to the 2D surface appearance in volumetric neural rendering. Our representation has disentangled geometry and appearance components, and models appearance as a 2D neural texture in a auto-discovered texture

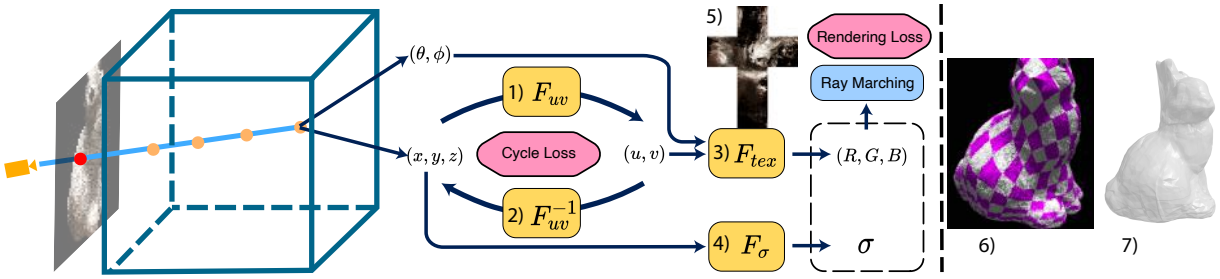


Figure 2. Overview. We present a disentangled neural representation consisting of multiple MLPs for neural volumetric rendering. As in NeRF [28], for geometry we use an MLP (4) F_σ to regress volume density σ at any 3D point $\mathbf{x} = (x, y, z)$. In contrast, for appearance, we use a texture mapping MLP (1) F_σ to map 3D points to 2D texture UVs, $\mathbf{u} = (u, v)$, and a texture network (3) F_{tex} to regress the 2D view-dependent radiance in the UV space given a UV $\mathbf{u}$ and a viewing direction $\mathbf{d} = (\theta, \phi)$. One regressed texture (for a fixed viewing direction) is shown in (5). We also train an inverse mapping MLP (2) F_{uv}^{-1} that maps UVs back to 3D points. We leverage a cycle loss (Eqn. 12) to ensure consistency between the 3D-to-2D mapping F_{uv} and the 2D-to-3D F_{uv}^{-1} mapping at points on the object surface. This enables meaningful surface reasoning and texture space discovery, as illustrated by (6, 7). We demonstrate the meaningfulness of the UV space learned by F_{uv} (6) by rendering the object with a uniform checkerboard texture. We also show the result of the inverse mapping network (7) by uniformly sampling UVs in the texture space and unprojecting them to 3D using F_{uv}^{-1}, resulting in a reasonable mesh.

space. Unlike previous volumetric neural rendering methods, this allows for easy texture/appearance editing.

Learning textures. Texture mapping is a standard technique widely used with meshes. Here, surface appearance is represented by a 2D texture image and a 3D-to-2D mapping from every mesh vertex to the texture space. Textures can be easily controlled and edited by artists as needed to create diversities of scene appearance. Recently, many deep learning based methods leverage texture-based techniques to model geometry or appearance in a scene [16, 14, 43]. Many works learn a 2D texture for a mesh by assuming a known mesh template [19, 37, 13], focusing on reconstruction problems for specific object categories. Our approach works for arbitrary shapes, and we instead learn a 2D texture in a volume rendering framework, discovering a 2D surface in the 3D volume space. Thies et al. [43] optimize neural textures to do rendering for a known fixed mesh with given UV mapping. In contrast, our approach simultaneously reconstructs the scene geometry as a volume, discovers a 2D texture UV space, and regresses a neural texture in the self-discovered texture space. Other methods learn appearance by regressing colors directly from 3D points [30, 31], which requires a known mesh and does not provide a 2D UV space necessary for texture editing.

AtlasNet [14] and follow-up work [33] train neural networks to map 2D UV coordinates into 3D locations (like an inverse texture mapping), modeling an object 2D surface as an unwrapped atlas. These works focus on learning generalized geometry representations, and cannot be directly applied to arbitrary shapes or used for realistic rendering. Our network instead discovers a cycle mapping between a 2D texture space and a 3D volume, learning both a texture mapping and an inverse mapping. We leverage differentiable volume rendering to model scene appearance from captured images for realistic rendering. We also show that our neural texture mapping, supervised using a rendering loss and a cycle mapping loss, can discover a more uniform surface than a simple AtlasNet, supervised by a noisy point cloud from COLMAP [39] (see Fig. 7).

3. Neural Texture Mapping

3.1. Overview

We now present the NeuTex scene representation and demonstrate how to use it in the context of volumetric neural rendering. While NeuTex can enable disentangled scene modeling and texture mapping for different acquisition and rendering tasks, in this section, we demonstrate its view synthesis capabilities with NeRF [28]. An extension to reflectance fields (with [4]) is discussed in Sec. 5.4.

As shown in Fig. 2, our method is composed of four learned components: F_σ, F_{tex}, F_{uv} and F_{uv}^{-1}. Unlike NeRF, which uses a single MLP, NeuTex uses a disentangled neural representation consisting of three sub-networks, which encode scene geometry (F_σ), a texture mapping function (F_{uv}), and a 2D texture (F_{tex}) respectively (Sec. 3.2). In addition, we use an inverse texture mapping network (F_{uv}^{-1}) to ensure that the discovered texture space reasonably explains the scene surfaces (Sec. 3.3). We introduce a cycle mapping loss to regularize the texture mapping and inverse mapping networks, and use a rendering loss to train our neural model end-to-end to regress realistic images (Sec. 3.4).

3.2. Disentangled neural scene representation

Volume rendering. Volume rendering requires volume density σ and radiance $\mathbf{c}$ at all 3D locations in a scene. A pixel's radiance value (RGB color) $\mathbf{I}$ is computed by marching a ray from the pixel and aggregating the radiance values

$\mathbf{c}_i$ of multiple shading points on the ray, as expressed by:

$$\mathbf{I} = \sum_i T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i, \quad (1)$$

$$T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j), \quad (2)$$

where $i = 1, ..., N$ denotes the index of a shading point on the ray, δ_i represents the distance between two consecutive points, T_i is known as the transmittance, and $\mathbf{c}_i$ and σ_i are the volume density (extinction coefficient) and radiance at shading point i. The above ray marching process is derived as a discretization of a continuous volume rendering integral; for more details, please see previous work [26].

Radiance field. In the context of view synthesis, a general volume scene representation can be seen as a 5D function (i.e. a radiance field, as referred to by [28]):

$$F_{\sigma,\mathbf{c}} : (\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathbf{c}), \quad (3)$$

which outputs volume density and radiance $(\sigma, \mathbf{c})$ given a 3D location $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (\theta, \phi)$. NeRF [28] proposes to use a single MLP network to represent $F_{\sigma,\mathbf{c}}$ as a neural radiance field and achieves photo-realistic rendering results. Their single network encapsulates the entire scene geometry and appearance as a whole; however, this "bakes" the scene content into the trained network, and does not allow for any applications (e.g., appearance editing) beyond pure view synthesis.

Disentangling $F_{\sigma,\mathbf{c}}$. In contrast, we propose explicitly decomposing the radiance field $F_{\sigma,\mathbf{c}}$ into two components, F_σ and $F_\mathbf{c}$, modeling geometry and appearance, respectively:

$$F_\sigma : \mathbf{x} \rightarrow \sigma, \qquad F_\mathbf{c} : (\mathbf{x}, \mathbf{d}) \rightarrow \mathbf{c}. \quad (4)$$

In particular, F_σ regresses volume density (i.e., scene geometry), and $F_\mathbf{c}$ regresses radiance (i.e., scene appearance). We model them as two independent networks.

Texture mapping. We further propose to model scene appearance in a 2D texture space that explains the object's 2D surface appearance. We explicitly map a 3D point $\mathbf{x} = (x, y, z)$ in a volume onto a 2D UV coordinate $\mathbf{u} = (u, v)$ in a texture, and regress the radiance in the texture space given 2D UV coordinates and a viewing direction $(\mathbf{u}, \mathbf{d})$. We describe the 3D-to-2D mapping as a texture mapping function F_{uv} and the radiance regression as a texture function F_{tex}:

$$F_{\text{uv}} : \mathbf{x} \rightarrow \mathbf{u}, \qquad F_{\text{tex}} : (\mathbf{u}, \mathbf{d}) \rightarrow \mathbf{c}. \quad (5)$$

Our appearance function $F_\mathbf{c}$ is thus a composition of the two functions:

$$F_\mathbf{c}(\mathbf{x}, \mathbf{d}) = F_{\text{tex}}(F_{\text{uv}}(\mathbf{x}), \mathbf{d}). \quad (6)$$

Neural representation. In summary, our full radiance field is a composition of three functions: a geometry function F_σ, a texture mapping function F_{uv}, and a texture function F_{tex}, given by:

$$(\sigma, \mathbf{c}) = F_{\sigma,\mathbf{c}}(\mathbf{x}, \mathbf{d}) = (F_\sigma(\mathbf{x}), F_{\text{tex}}(F_{\text{uv}}(\mathbf{x}), \mathbf{d})). \quad (7)$$

We use three separate MLP networks for F_σ, F_{uv} and F_{tex}. Unlike the black-box NeRF network, our representation has disentangled geometry and appearance modules, and models appearance in a 2D texture space.

3.3. Texture space and inverse texture mapping

As described in Eqn. 5, our texture space is parameterized by a 2D UV coordinate $\mathbf{u} = (u, v)$. While any continuous 2D topology can be used for the UV space in our network, we use a 2D unit sphere for most results, where $\mathbf{u}$ is interpreted as a point on the unit sphere.

Directly training the representation networks (F_σ, F_{uv}, F_{tex}) with pure rendering supervision often leads to a highly distorted texture space and degenerate cases where multiple points map to the same UV coordinate, which is undesirable. The ideal goal is instead to uniformly map the 2D surface onto the texture space and occupy the entire texture space. To achieve this, we propose to jointly train an "inverse" texture mapping network F_{uv}^{-1} that maps a 2D UV coordinate $\mathbf{u}$ on the texture to a 3D point $\mathbf{x}$ in the volume:

$$F_{\text{uv}}^{-1} : \mathbf{u} \rightarrow \mathbf{x}. \quad (8)$$

F_{uv}^{-1} projects the 2D texture space onto a 2D manifold (in 3D space). This inverse texture mapping allows us to reason about the 2D surface of the scene (corresponding to the inferred texture) and regularize the texture mapping process. We leverage our texture mapping and inverse mapping networks to build a cycle mapping (a one-to-one correspondence) between the 2D object surface and the texture space, leading to high-quality texture mapping.

3.4. Training neural texture mapping

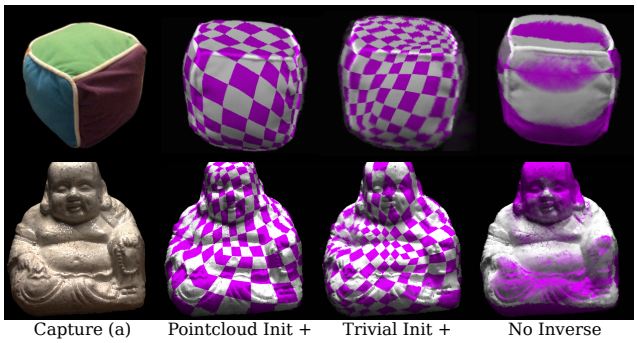
We train our full network, consisting of F_σ, F_{tex}, F_{uv}, and F_{uv}^{-1}, from end to end, to simultaneously achieve surface discovery, space mapping, and scene geometry and appearance inference.

Rendering loss. We directly use the ground truth pixel radiance value $\mathbf{I}_{\text{gt}}$ in the captured images to supervise our rendered pixel radiance value $\mathbf{I}$ from ray marching (Eqn. 1). The rendering loss for a pixel ray is given by:

$$L_{\text{render}} = \|\mathbf{I}_{\text{gt}} - \mathbf{I}\|_2^2. \quad (9)$$

This the main source of supervision in our system.

Cycle loss. Given any sampled shading point $\mathbf{x}_i$ on a ray in ray marching, our texture mapping network finds its UV $\mathbf{u}_i$



Capture (a) | Pointcloud Init + Cycle Loss (b) | Trivial Init + Cycle Loss (c) | No Inverse Network (d)

Figure 3. A checkerboard texture applied to scenes (a). When trained with or without initialization using coarse point cloud (b,c), the learned texture space is relatively uniform compared to trained without F_{uv}^{-1} and cycle loss (d).

in texture space for radiance regression. We use the inverse mapping network to map this UV $\mathbf{u}_i$ back to the 3D space:

$$\mathbf{x}_i' = F_{\text{uv}}^{-1}(F_{\text{uv}}(\mathbf{x}_i)). \tag{10}$$

We propose to minimize the difference between $\mathbf{x}_i'$ and $\mathbf{x}_i$ to enforce a cycle mapping between the texture and world spaces (and force F_{uv}^{-1} to learn the inverse of F_{uv}).

However, it is unnecessary and unreasonable to enforce a cycle mapping at any 3D point. We only expect a correspondence between the texture space and points on the 2D *surface* of the scene; enforcing the cycle mapping in the empty space away from the surface is meaningless. We expect 3D points near the scene surface to have high contributions to the radiance. Therefore, we leverage the radiance contribution weights per shading point to weigh our cycle loss. Specifically, we consider the weight:

$$w_i = T_i(1 - \exp(-\sigma_i \delta_i)), \tag{11}$$

which determines the contribution to the final pixel color for each shading point i in the ray marching equation 1. Equation 1 can be simply written as $\mathbf{I} = \sum_i w_i \mathbf{c}_i$. This contribution weight w_i naturally expresses how close a point is to the surface, and has been previously used for depth inference [28]. Our cycle loss for a single ray is given by:

$$L_{\text{cycle}} = \sum_i w_i \|F_{\text{uv}}^{-1}(F_{\text{uv}}(\mathbf{x}_i)) - \mathbf{x}_i\|_2^2. \tag{12}$$

Mask loss. We also additionally provide a loss to supervise a foreground-background mask. Basically, the transmittance (Eqn. 2) of the the last shading point T_N on a pixel ray indicates if the pixel is part of the background. We use the ground truth mask M_{gt} per pixel to supervise this by

$$L_{\text{mask}} = \|M_{\text{gt}} - (1 - T_N)\|_2^2. \tag{13}$$

We found this mask loss is necessary when viewpoints do not cover the object entirely. In such cases, the network can use the volume density to darken (when the background is black) renderings and fake some shading effects that should be in the texture. When the view coverage is dense enough around an object, this mask loss is often optional.

Full loss. Our full loss function L during training is:

$$L = L_{\text{render}} + a_1 L_{\text{cycle}} + a_2 L_{\text{mask}}. \tag{14}$$

We use $a_1 = 1$ for all our scenes in our experiments. We use $a_2 = 1$ for most scenes, except for those that already have good view coverage, where we remove the mask loss by setting $a_2 = 0$.

4. Implementation Details

4.1. Network details

All four sub-networks, F_σ, F_{tex}, F_{uv}, and F_{uv}^{-1}, are designed as MLP networks. We use unit vectors to represent viewing direction $\mathbf{d}$ and UV coordinate $\mathbf{u}$ (for spherical UV). As proposed by NeRF, we use positional encoding to infer high-frequency geometry and appearance details. In particular, we apply positional encoding for our geometry network F_σ and texture network F_{tex} on all their input components including $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{d}$. On the other hand, since the texture mapping is expected to be smooth and uniform, we do not apply positional encoding on the two mapping networks. Please refer to the supplemental materials for the detailed architecture of our networks.

4.2. Training details

Before training, we normalize the scene space to the unit box. When generating rays, we sample shading points on each pixel ray inside the box. For all our experiments, we use stratified sampling (uniform sampling with local jittering) to sample 256 point on each ray for ray marching. For each iteration, we randomly select a batch size of 600 to 800 pixels (depending on GPU memory usage) from an input image; we take 2/3 pixels from the foreground and 1/3 pixels from the background.

Our inverse mapping network F_{uv}^{-1} maps the 2D UV space to a 3D surface, which is functionally similar to AtlasNet [14] and can be trained as such, if geometry is available. We thus initialize F_{uv}^{-1} with a point cloud from COLMAP [39] using a Chamfer loss. However, since the MVS point cloud is often very noisy, this Chamfer loss is only used during this initialization phase. We find this initialization facilitates training, though our network still works without it for most cases (see Fig. 3). Usually, this AtlasNet-style initialization is very sensitive to the MVS reconstruction noise and leads to a highly non-uniform mapping surface. However, we find that our final inverse mapping network can

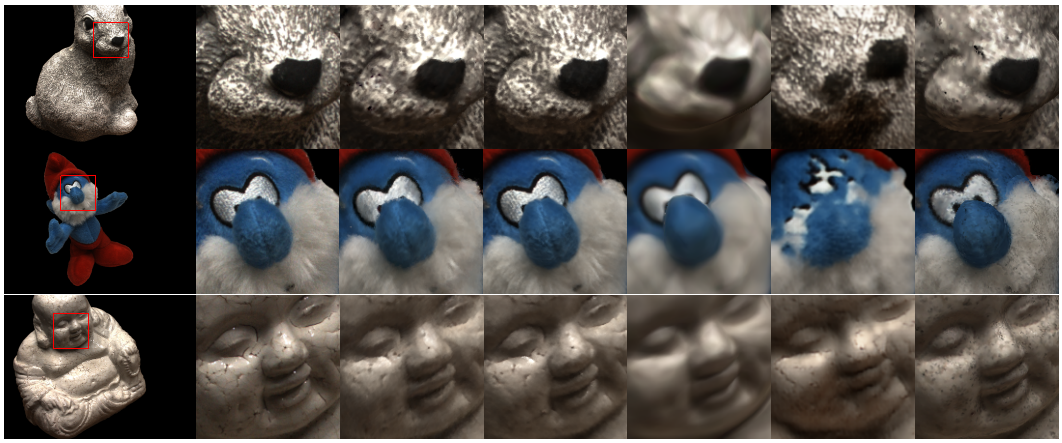


Figure 4. Comparisons on DTU scenes. Note how visually close our method is to the state-of-the-art, while enabling editing.

Method	PSNR	SSIM
SRN [41]	26.05	0.837
DeepVoxels [40]	20.85	0.702
Colmap [39]	24.63	0.865
NeRF[28]	**30.73**	**0.938**
Ours	**28.23**	**0.894**

Table 1. Average PSNR/SSIM for novel view synthesis on 4 held-out views on 5 DTU scenes. See supplementary for full table.

output a much smoother surface as shown in Fig. 7, after jointly training with our rendering and cycle losses.

Specifically, we initially train our method using a Chamfer loss together with a rendering loss for 50,000 iterations. Then, we remove the Chamfer loss and train with our full loss (Eqn. 14) until convergence, after around 500,000 iterations. Finally, we fine-tune our texture network F_{tex} until convergence, freezing the other networks (F_σ, F_{uv} and F_{uv}^{-1}), which is useful to get better texture details. The whole process takes 2-3 days on a single RTX 2080Ti GPU.

5. Results

We now show experimental results of our method and comparisons against previous methods on real scenes.

5.1. Configuration

We demonstrate our method on real scenes from different sources, including five scenes from the DTU dataset [1] (Fig. 1, 4, 5), two scenes from Neural Reflectance Fields [4] obtained from the authors (Fig. 6), and three scenes captured by ourselves (Fig. 5). Each DTU scene contains either 49 or 64 input images from multiple viewpoints. Each scene from [4] contains about 300 images. Our own scenes each contain about 100 images. For our own data, we capture the images using a held-hanld cellphone and use the structure from motion implementation in COLMAP [38] for camera calibration. For other scenes, we directly use the provided camera calibration in the dataset. Since our method focuses on the capture and surface discovery of objects, we require the input images to have a clean, easily segmentable background. We use U2Net [36] to automatically compute masks for our own scenes. For the DTU scenes, we use the background masks provided by [51]. The images from [4] are captured under a single flash light, which already have very dark background; thus we do not apply additional masks for these images.

5.2. View synthesis results on DTU scenes

We now evaluate and compare our view synthesis results on five DTU scenes. In particular, we compare with NeRF [28], two previous neural rendering methods, SRN [41] and DeepVoxels [40], and one classical mesh reconstruction method COLMAP [39]. We use the released code from their authors to generate the results for all the comparison methods. For COLMAP, we skip the structure from motion, since we already have the provided camera calibration from the dataset. We hold-out 4 random views as testing views from the original 49 or 64 input views and run all methods on the remaining images for reconstruction.

We show qualitative visual comparisons on zoomed-in crops of testing images of two DTU scenes in Fig. 4 (the other scenes are shown in supplementary materials), and quantitative comparison results of the averaged PSNRs and SSIMs on the testing images across five scenes in Tab. 1. Our method achieves high-quality view synthesis results as reflected by our rendered images being close to the ground truth and also our high PSNRs and SSIMs. Note that NeuTex enables automatic texture mapping that none of the other comparison methods can do. Even a traditional mesh-based method like COLMAP [39] needs additional techniques or tools to unwrap its surface for texture mapping, whereas our method unwraps the surface into a tex-

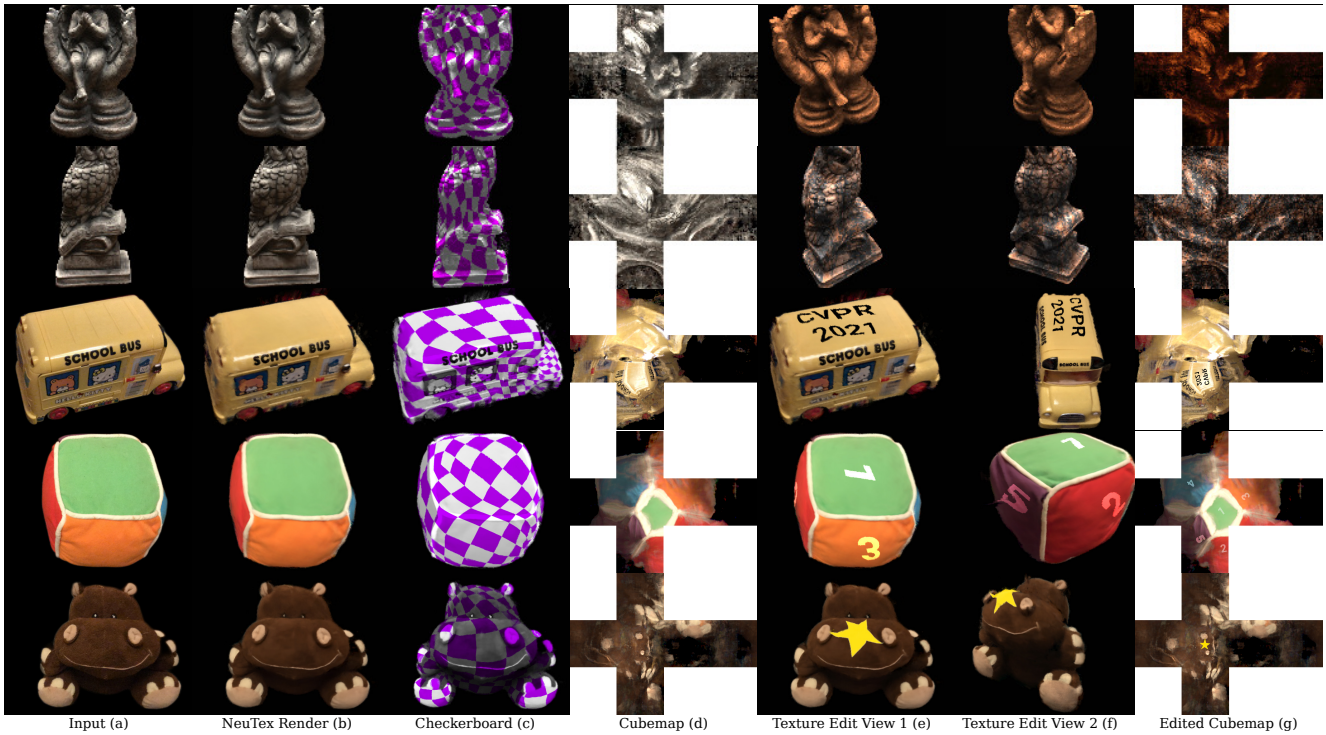


Figure 5. Texture editing on DTU (rows 1-2) and our (rows 3-5) scenes. Since our neural texture (F_{tex}) depends on a view direction, we show the cubemap texture (d) with pixelwise maximum values across views. Each texture is edited by multiplying a new specified texture; the resulting texture is shown in (g). The images rendered with the edited textures are shown from two different views in (e) and (f).

ture while doing reconstruction in a unsupervised way. To achieve this challenging task, NeuTex is designed in a more constrained way than NeRF. As a result, our rendering quality is quantitatively slightly worse than NeRF. Nonetheless, as shown in Fig. 4, our rendered results are realistic, reproduce many high-frequency details and qualitatively look very close to NeRF's results.

In fact, our results are significantly better than all other comparison methods, including both mesh-based reconstruction [39] and previous neural rendering methods [40, 41] in both qualitative and quantitative comparisons. In particular, COLMAP [38] can reconstruct reasonable shapes, but it cannot recover accurate texture details and intrinsically lacks view-dependent appearance effects (due to the Lambertian materials assumption). DeepVoxels [40] leverages a non-physically-based module for geometry and occlusion inference. While this works well on scenes that have hundreds of input images, it does not work well on DTU scenes that have only about 40 to 60 images, leading to incorrect shapes and serious artifacts in their results. SRN [41], on the other hand, can reproduce reasonable shape in the rendering; however it cannot generate high-frequency appearance details like our method. Our approach is based on physically-based volume rendering, which models scene geometry and appearance accurately, leading to photo-realistic rendering results. More importantly, our approach achieves texture mapping and enables surface appearance editing in a 2D texture space that it automatically discovered, which cannot be done by NeRF nor any other previous neural rendering approaches.

5.3. Texture mapping and appearance editing

We now demonstrate our unique results on texture mapping and texture-space appearance editing that previous neural rendering approaches cannot achieve. Figure 5 shows such results on diverse real objects of DTU scenes and our own captured scenes. Our method can synthesize realistic view synthesis results (Fig. 5.b) that are very close to the ground truth. In addition, our method successfully unwraps the object surface into a reasonable texture (Fig. 5.d); the discovered texture space meaningfully expresses the 2D surface and distributes uniformly, as illustrated by the checkerboard rendering shown in Fig. 5.c.

Texture editing. Our high-quality texture mapping enables flexible appearance editing applications as shown in Fig. 5.e-g. In these examples, we show that we can use a specified full texture map to modulate the original texture, which entirely changes the object appearance. For example, the object in the 1st row is interestingly changed from a stone-like object to a wooden one. We also demonstrate that we can locally modify the texture to add certain patterns on

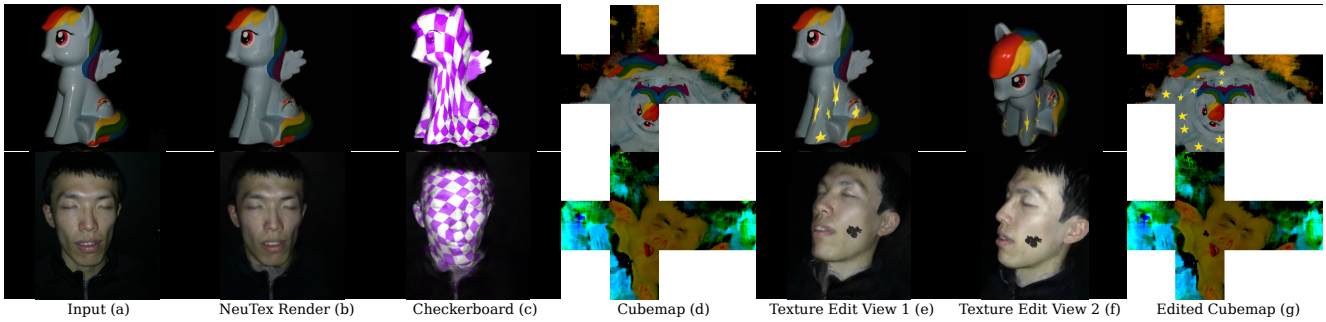


Figure 6. NeuTex in reflectance fields setting. We edit captured diffuse albedo (d) as shown in (g) to produce results shown in (e,f).

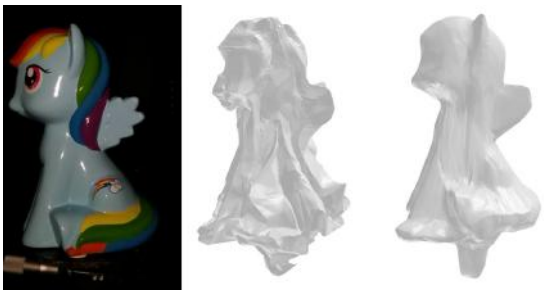


Figure 7. Our parametric surface (F_{uv}^{-1}) is strongly affected by noise in the COLMAP point cloud when trained with a Chamfer loss (center). Finetuning with our cycle and rendering losses without point cloud supervision (right) gives a smoother surface.

the object surface, such as the CVPR logo, the numbers, and the star in the last three rows. Note that, all our appearance editing is directly done in the texture space, which changes the essential surface appearance and naturally appears consistent across multiple view points, as shown in the rendered images from two different views in Fig. 5.e and f. Our NeuTex successfully disentangles the geometry and appearance of real objects and model the surface appearance in a meaningful texture space that explains the surface.

Inverse mapping. We further demonstrate the role of our inverse mapping network F_{uv}^{-1} in discovering this reasonable texture space. When we remove F_{uv}^{-1} from our network and train the system with only the rendering loss, the result always leads to a degenerate texture mapping, where large regions of 3D points are mapped to the same UV coordinates, as illustrated by the checkerboard-texture rendering shown in Fig. 3. In contrast, our full network with the cycle loss generally discovers a uniform space. As described in Sec. 4.2, we initialize our inverse network using a point cloud with a Chamfer loss; this is done mainly to help the network converges quickly to a reasonable stage. In Fig. 3, we also show that our inverse network can still function well even without the point cloud initialization, using only the cycle loss. Note that, our initial point clouds come from an MVS reconstruction [39], which contains a lot of noise, leading to a noisy surface out of F_{uv}^{-1} as shown in Fig. 7. To prevent a degradation in the surface texture quality, we remove the supervision on this initial point cloud after initialization. Instead, our cycle loss can continue improving the noisy initialization and let the inverse mapping network F_{uv}^{-1} discover a smooth surface as shown in Fig. 7.

5.4. Extension to reflectance fields

NeuTex can be incorporated into different volume rendering pipelines. We now discuss combining it with the recent Neural Reflectance Fields [5, 4] that reconstructs BRDFs in volume rendering from flash images.

Instead of directly outputting radiance $\mathbf{c}$, [4] regresses normal $\mathbf{n}$ and reflectance parameters $\mathbf{r}$ at each shading point, and introduces a reflectance-aware volume rendering that computes radiance from these shading properties under given viewing and lighting condition. We correspondingly modify our geometry network F_σ to jointly regress volume density and normal, and change the texture regression network F_{tex} to regress the reflectance parameters in the texture space. Our central texture mapping and inverse mapping networks remain the same for this case. The modified network naturally provides the required volume properties in the reflectance-aware volume rendering process.

We show that our neural texture mapping can enable high-quality BRDF texture extraction in this setting in Fig. 6 on the two scenes from [4]. Our approach achieves realistic rendering (Fig. 6.b) that reproduces the original appearance, discovers a reasonably uniform texture space (Fig. 6.c), successfully unwraps the surface BRDFs into this space (as shown by the albedo maps in Fig. 6.d), and enables realistic rendering with BRDF editing in the texture space (Fig. 6.e-g). These results demonstrate the generality of our neural texture mapping framework and inspire potential future applications of our technique on other neural rendering tasks.

6. Conclusion

We have presented a novel approach that enables texture mapping in neural volumetric rendering. We introduce a novel disentangled neural scene representation that models geometry as a 3D volume and models appearance as a 2D

texture in a automatically discovered texture space. We propose to jointly train a 3D-to-2D texture mapping network and a 2D-to-3D inverse mapping network to achieve surface reasoning and texture space discovery, using a surface-aware cycle consistency loss. As demonstrated, our approach can discover a reasonable texture space that meaningfully explains the object surface. Our method enables flexible surface appearance editing applications for neural volumetric rendering.

Acknowledgement This research was supported by gifts from Adobe, Kwai, and Qualcomm.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, 120(2):153–168, 2016. 6

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *ICML*, pages 40–49, 2018. 2

[3] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Transaction on Graphics*, 36(4):106–1, 2017. 2

[4] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020. 2, 3, 6, 8

[5] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. *arXiv preprint arXiv:2007.09892*, 2020. 1, 2, 8

[6] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, pages 425–432. ACM, 2001. 2

[7] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1538–1547, 2019. 2

[8] Zhang Chen, Anpei Chen, Guli Zhang, Chengyuan Wang, Yu Ji, Kiriakos N. Kutulakos, and Jingyi Yu. A neural rendering framework for free-viewpoint relighting. In *CVPR*, June 2020. 2

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *arXiv preprint arXiv:1812.02822*, 2018. 2

[10] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2524–2534, 2020. 2

[11] Paul Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques' 98*, pages 105–116. Springer, 1998. 2

[12] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009. 2

[13] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. Shape and viewpoint without keypoints. *arXiv preprint arXiv:2007.10982*, 2020. 3

[14] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 3, 5

[15] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018. 2

[16] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8356–8364, 2020. 3

[17] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *CVPR*, pages 2821–2830, 2018. 2

[18] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: An end-to-end 3D neural network for multiview stereopsis. In *ICCV*, pages 2307–2315, 2017. 2

[19] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018. 2, 3

[20] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 1, 2

[21] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *ICCV*, 38(3):199–218, 2000. 2

[22] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2015. 2

[23] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[24] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 2

[25] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 2

[26] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 2, 4

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *arXiv preprint arXiv:1812.03828*, 2018. 2

[28] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020. 1, 2, 3, 4, 5, 6

[29] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, pages 3504–3515, 2020. 2

[30] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4531–4540, 2019. 3

[31] Michael Oechsle, Michael Niemeyer, Lars Mescheder, Thilo Strauss, and Andreas Geiger. Learning implicit surface light fields. *arXiv preprint arXiv:2003.12406*, 2020. 3

[32] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. *ACM Transactions on Graphics*, 38(4):1–14, 2019. 2

[33] Omid Poursaeed, Matthew Fisher, Noam Aigerman, and Vladimir G Kim. Coupling explicit and implicit surface representations for generative 3d modeling. *arXiv preprint arXiv:2007.10294*, 2, 2020. 3

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2

[35] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 2

[36] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020. 6

[37] Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano, and Hao Li. Photorealistic facial texture inference using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5144–5153, 2017. 3

[38] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 6, 7

[39] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 3, 5, 6, 7, 8

[40] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3D feature embeddings. In *CVPR*, pages 2437–2446, 2019. 2, 6, 7

[41] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1119–1130, 2019. 2, 6, 7

[42] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018. 2

[43] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 3

[44] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017. 2

[45] Jinglu Wang, Bo Sun, and Yan Lu. Mvpnet: Multi-view point regression networks for 3D object reconstruction from a single image. *arXiv preprint arXiv:1811.09410*, 2018. 2

[46] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018. 2

[47] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 2

[48] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics*, 38(4):76, 2019. 2

[49] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics*, 37(4):126, 2018. 2

[50] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018. 2

[51] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 6

[52] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37(4):1–12, 2018. 2

NeuTex: Neural Texture Mapping for Volumetric Neural Rendering Supplementary Material

Fanbo Xiang[1], Zexiang Xu[2], Miloš Hašan[2], Yannick Hold-Geoffroy[2], Kalyan Sunkavalli[2], Hao Su[1]

[1] University of California, San Diego
[2] Adobe Research

1. Cube map Clarification

We briefly clarify our texture visualization (with cubemaps) used in our main paper. As discussed in Sec. 3.3 in the paper, we use spherical UVs for our texture mapping, where $\mathbf{u}$ represents a point on the surface a unit sphere. For all the figures in the main paper, we use cubemaps [1] to visualize the spherical domain. A cubemap consists of sixes faces of a unit box, recording all the color information projected from a unit sphere (as shown in Fig 1), which is widely used in graphics for spherical mapping. An alternative standard way to visualize a spherical function is to use a equirectangle map. We show the correspondence between a cube map and a equirectangle map in Fig. 2. We use cubemaps in the paper since they involve less distortion, avoiding the distorted regions in the top and bottom of equirectangle maps.

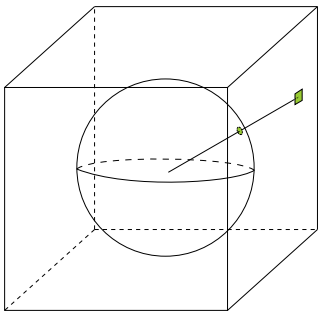


Figure 1. Cube map projection. The color at each point on the unit sphere is projected to a point on the cube centered at the origin. A cubemap is obtained by "opening up" the cube.

2. Network Implementation Details

2.1. Network structure

We show the detailed network architecture for $F_\sigma, F_{uv}, F_{uv}^{-1}$ and F_{tex} in Figure 3.

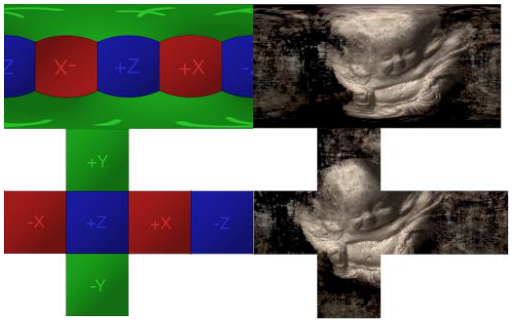


Figure 2. Cube maps on the second row corresponds to the equirectangle maps on the first row. They are different projections of the same spherical texture. A cubemap has a smaller distortion on the Y direction.

2.2. Training details in initialization

Here we describe in detail how we do the initialization stage mentioned in Sec 4.2. in the paper. Given a point cloud from Colmap, we first downsample it to one with 2,000-3,000 points. We denote this point cloud as P_{gt}. We then sample 2,500 points uniformly in the UV space (the unit sphere). We denote the set of UV coordinates as P.

Chamfer loss. The Chamfer loss is simply the Chamfer distance between $F_{uv}^{-1}(P)$ and P_{gt}, where $F_{uv}^{-1}(P)$ corresponds to the point cloud generated by inverse-mapping very UV in P to the 3D space using the network F_{uv}^{-1}.

$$L_{\text{chamfer}} = \text{Chamfer}(F_{uv}^{-1}(P), P_{gt})$$

Inverse loss. We also leverage a loss that is similar to our cycle loss to let the initialization also influence the texture mapping network F_{uv}. In particular, instead of the 3D-to-2D-to-3D cycle mapping used the cycle loss in Eqn. 12 of the paper, we leverage a 2D-to-3D-to-2D cycle mapping in the initialization, given by:

$$L_{\text{cycle2}} = ||F_{uv}(F_{uv}^{-1}(P)) - P||_2^2$$

.

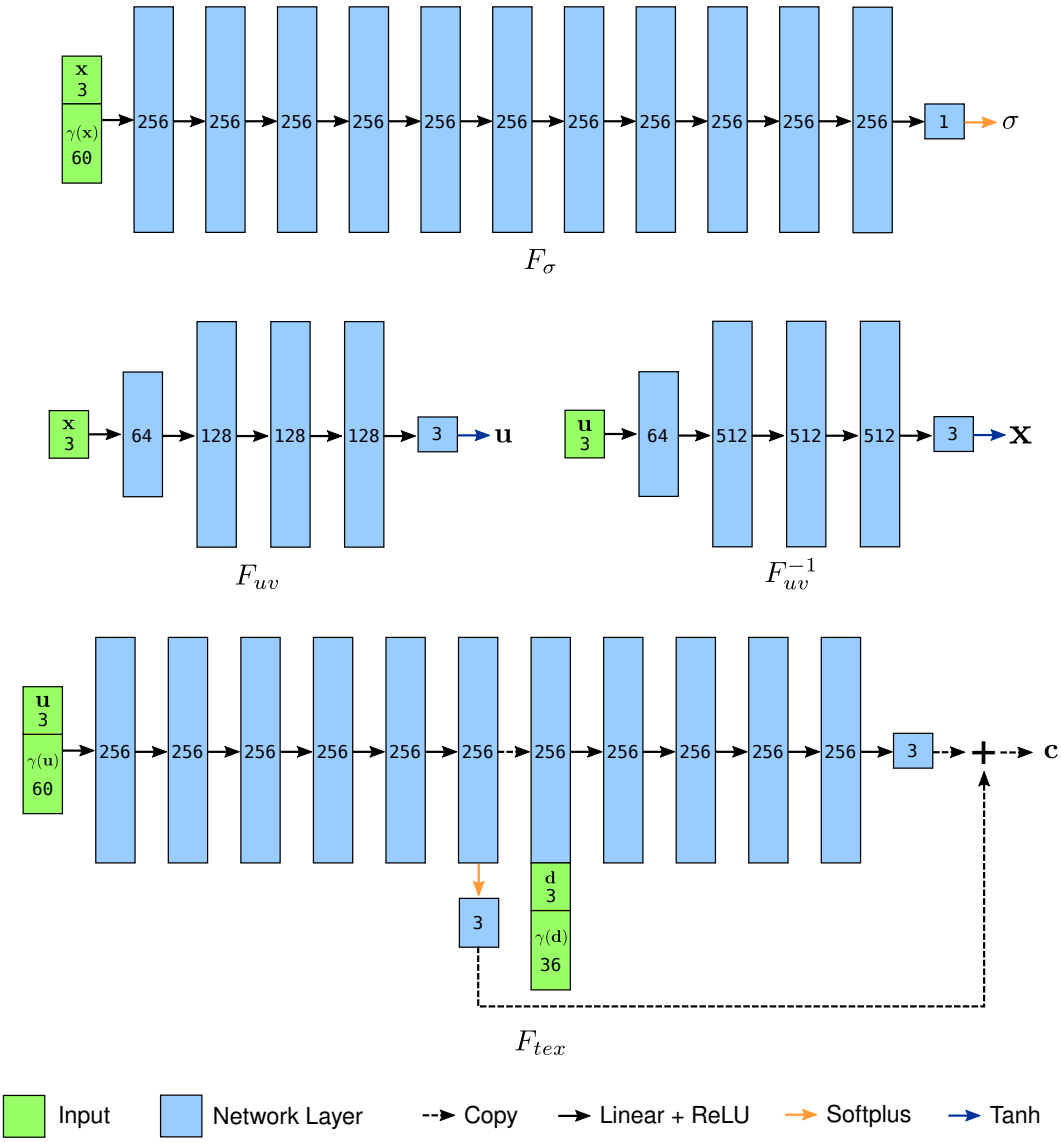


Figure 3. Network structure for the 4 networks. **x** represents 3D coordinates. γ denotes positional encoding. **u** represents texture-space points (3D points on the unit sphere). **d** represents a 3D unit vector for viewing direction. σ is predicted volumetric density. **c** is predicted radiance.

Rendering and mask loss. The same rendering and mask loss as described in section 4.1 are also applied in the initialization stage. So the loss at initialization stage is

$$L_{\text{init}} = L_{\text{chamfer}} + aL_{\text{cycle2}} + bL_{\text{render}} + cL_{\text{mask}}$$

where we set $a = 100$, $b = c = 1$.

3. Additional Results

3.1. Full quantitative comparison

We have shown the averaged quantitative results across five DTU scenes in Tab. 1 of the paper. Detailed comparisons on individual scenes are provided in Table 1. Similar to the average scores, though slightly worse than NeRF, our method significantly outperforms other traditional and neural rendering methods.

3.2. Additional visual results

Figure 4 shows the visual comparison of different methods on the remaining 2 DTU scenes. Figure 5 shows additional texture editing results. Please refer to the attached video for more results on view synthesis and editing.

References

[1] Ned Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applica-*

Method	55		83		114		118		122	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
SRN[5]	21.35	0.673	28.68	0.929	23.75	0.808	28.74	0.900	27.75	0.877
DeepVoxels[4]	17.21	0.532	23.76	0.858	17.97	0.606	23.18	0.764	22.12	0.748
Colmap[3]	21.25	0.784	27.11	0.921	20.69	0.809	27.43	0.907	26.66	0.905
NeRF[2]	**26.78**	**0.913**	**31.77**	**0.952**	**27.38**	**0.918**	**33.98**	**0.954**	**33.72**	**0.955**
Ours	**22.67**	**0.808**	**30.61**	**0.931**	**26.45**	**0.891**	**30.67**	**0.916**	**30.75**	**0.925**

Table 1. PSNR/SSIM for novel view synthesis quality on 4 held-out views on 5 DTU scenes.

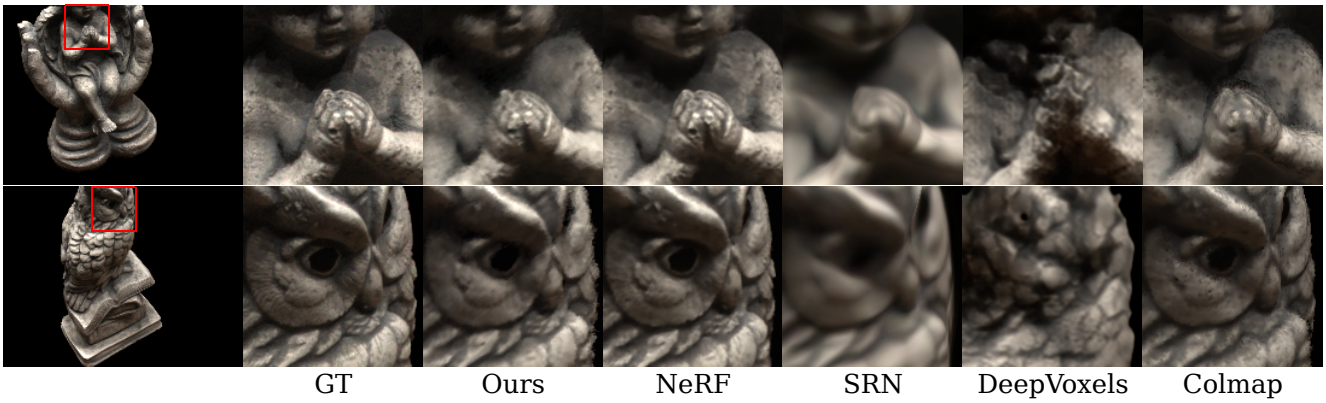


Figure 4. Comparison on the remaining DTU scenes.

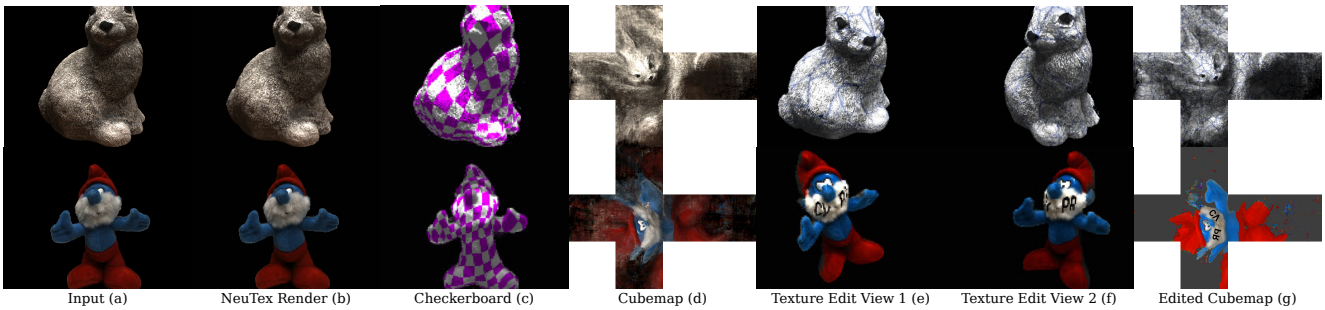


Figure 5. Additional texture editing on DTU scenes.

tions, 6(11):21–29, 1986. 1

[2] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020. 3

[3] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 3

[4] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3D feature embeddings. In *CVPR*, pages 2437–2446, 2019. 3

[5] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1119–1130, 2019. 3