Deferred Neural Rendering: Image Synthesis using Neural Textures

JUSTUS THIES, Technical University of Munich
MICHAEL ZOLLHÖFER, Stanford University
MATTHIAS NIESSNER, Technical University of Munich

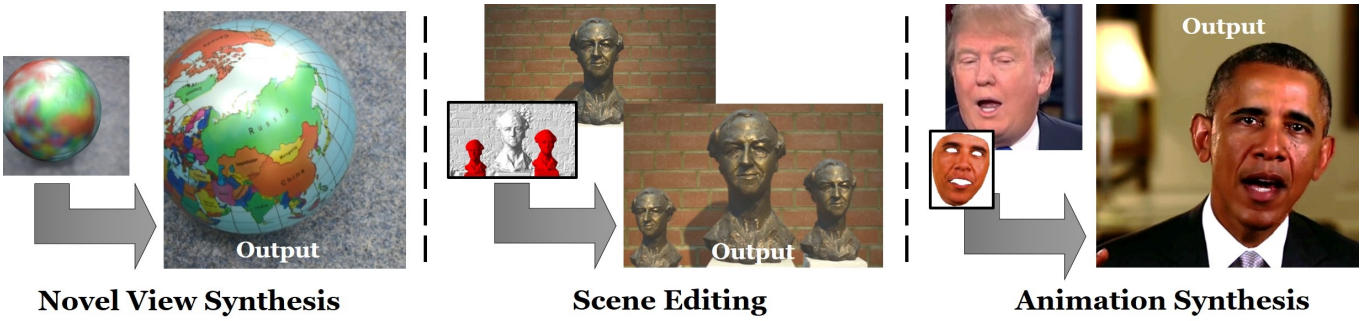

Fig. 1. We present an image synthesis approach that learns object-specific neural textures which can be interpreted by a neural renderer. Our approach can be trained end-to-end with real data, allowing us to re-synthesize novel views of static objects, edit scenes, as well as re-render dynamic animated surfaces [0].

The modern computer graphics pipeline can synthesize images at remarkable visual quality; however, it requires well-defined, high-quality 3D content as input. In this work, we explore the use of imperfect 3D content, for instance, obtained from photo-metric reconstructions with noisy and incomplete surface geometry, while still aiming to produce photo-realistic (re-)renderings. To address this challenging problem, we introduce *Deferred Neural Rendering*, a new paradigm for image synthesis that combines the traditional graphics pipeline with learnable components. Specifically, we propose *Neural Textures*, which are learned feature maps that are trained as part of the scene capture process. Similar to traditional textures, neural textures are stored as maps on top of 3D mesh proxies; however, the high-dimensional feature maps contain significantly more information, which can be interpreted by our new deferred neural rendering pipeline. Both neural textures and deferred neural renderer are trained end-to-end, enabling us to synthesize photo-realistic images even when the original 3D content was imperfect. In contrast to traditional, black-box 2D generative neural networks, our 3D representation gives us explicit control over the generated output, and allows for a wide range of application domains. For instance, we can synthesize temporally-consistent video re-renderings of recorded 3D scenes as our representation is inherently embedded in 3D space. This way, neural textures can be utilized to coherently re-render or manipulate existing video content in both static and dynamic environments at real-time rates. We show the effectiveness of our approach in several experiments on novel view synthesis, scene editing, and facial reenactment, and compare to state-of-the-art approaches that leverage the standard graphics pipeline as well as conventional generative neural networks.

Authors' addresses: Justus Thies, Technical University of Munich, justus.thies@tum.de; Michael Zollhöfer, Stanford University, zollhoefer@cs.stanford.edu; Matthias Nießner, Technical University of Munich, niessner@tum.de.

0730-0301/2019/7-ART66 $15.00
https://doi.org/10.1145/3306346.3323035

CCS Concepts: • **Computing methodologies** → **Computer vision**; **Computer graphics**.

Additional Key Words and Phrases: neural rendering, neural texture, novel view synthesis, facial reenactment

ACM Reference Format:
Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred Neural Rendering: Image Synthesis using Neural Textures. *ACM Trans. Graph.* 38, 4, Article 66 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3323035

1 INTRODUCTION

The current computer graphics pipeline has evolved during the last decades, and is now able to achieve remarkable rendering results. From a fixed function pipeline around the rasterization unit, the graphics pipeline has turned into a programmable rendering pipeline. Based on this pipeline the Siggraph community has established rendering techniques that now achieve nearly photo-realistic imagery. While the visuals are stunning, a major drawback of these classical approaches is the need of well-defined input data, including a precise definition of the surface geometry, the underlying material properties, and the scene illumination. In movie or video game productions, this underlying 3D content is manually-created by skilled artists in countless working hours. An alternative is to obtain 3D content from real-world scenes by using 3D reconstruction techniques. However, given the inherent limitations of state-of-the-art 3D reconstruction approaches, such as noisy, oversmoothed geometry or occlusions, the obtained 3D content is imperfect. From this captured content, it is nearly impossible to re-synthesize photo-realistic images with the existing computer graphics pipeline and rendering techniques.

In this work, we assume that captured 3D content will always suffer from reconstruction artifacts in one way or another. Rather than aiming to fix the artifacts in the 3D content, we propose to

[0] Third Party Material: Trump, MSNBC (https://youtu.be/Tsh_V3U7EfU) [Fair Use] Obama, The Obama White House (https://youtu.be/d-VaUaTF3_k) [Public Domain]

change the paradigm of the rendering pipeline to cope with these imperfections. To this end, we introduce *Deferred Neural Rendering* which makes a step towards a learnable rendering pipeline, combining learned *Neural Textures* with the traditional graphics pipeline. Neural textures, similar to classical textures, are stored in 2D maps on top of a 3D mesh, and may be transformed along with the underlying 3D geometry. However, the core idea behind neural textures is that they are composed of a set of optimal feature maps, rather than simple RGB values, that are learned during the scene capture process. The rich signal stored in these high-dimensional neural textures encodes a high-level description of the surface appearance, and can be interpreted by our new deferred neural renderer. Both the neural textures and the deferred neural renderer are trained in an end-to-end fashion, which enables us to achieve photo-realistic (re-)rendering results, even from imperfect geometry.

This learnable rendering pipeline enables a wide range of practical new application scenarios. Our main motivation is to produce photo-realistic images from imperfect 3D reconstructions, for instance, obtained from a video sequence using multi-view stereo where the geometry typically is noisy, oversmoothed from strong regularization, or has holes. Here, we show that we can synthesize novel view points of a scene while the geometry acts as a 3D proxy that forces generated images to be temporally-coherent with respect to the new camera view points. However, the ability to explicitly control the 3D geometry not only allows synthesizing static view points, but also editing the underlying 3D content. For instance, we can duplicate scene objects, along with their neural textures, and then coherently manipulate the re-renderings of the captured 3D environment. We show several examples of such editing operations, e.g., copy or remove, in static 3D environments, which would have not been feasible with existing capture or editing techniques. In addition, we demonstrate that we can edit dynamic scenes in a similar fashion. Specifically, we show examples on human faces, where we first reconstruct the 3D face as well as the neural textures, and then modify the 3D facial animation and re-render a photo-realistic output. This way, our unified neural rendering approach can easily achieve results that outperform the quality of existing facial re-enactment pipelines [Thies et al. 2016].

In comparison to existing black-box, generative neural networks that are defined by a series of 2D convolutions, our approach is inherently embedded in 3D space. As such, we implicitly obtain generated video output that is temporally coherent due to the underlying 3D embedding. In addition, we have active control in manipulations, rather than making modifications to a semantically-uncorrelated random vector defining the latent space, as in 2D GAN approaches [Goodfellow et al. 2014].

In summary, we combine the benefits of traditional graphics-based image synthesis with learnable components from the machine learning community. This results in a novel paradigm of a learnable computer graphics pipeline with the following contributions:

- *Neural Rendering* for photo-realistic image synthesis based on imperfect commodity 3D reconstructions at real-time rates,
- *Neural Textures* for novel view synthesis in static scenes and for editing dynamic objects,
- which is achieved by an end-to-end learned novel deferred neural rendering pipeline that combines insights from traditional graphics with learnable components.

2 RELATED WORK

Deferred Neural Rendering presents a new paradigm of image synthesis with learned neural textures and renderer. Such learned computer graphics components can be useful for a variety of problems in computer graphics and computer vision. In this work, we focus on novel view synthesis and synthesis of novel scene edits and animations, such as the animation of reconstructed human faces.

2.1 Novel-view Synthesis from RGB-D Scans

A traditional building block of novel-view synthesis approaches is to first obtain a digital representation of a real world scene. In particular in the context of 3D reconstruction with commodity RGB-D sensors, researchers have made significant progress, enabling robust tracking [Choi et al. 2015; Dai et al. 2017; Izadi et al. 2011; Newcombe et al. 2011; Whelan et al. 2016] and large-scale 3D scene capture [Chen et al. 2013; Nießner et al. 2013; Zeng et al. 2013]. Images from novel view points can be synthesized by rendering these reconstructions. Given the inherent limitations in current state-of-the-art 3D reconstruction approaches, the obtained 3D content is imperfect, for instance, reconstructed geometry is noisy and/or oversmoothed, or has holes due to occlusion; this makes it nearly impossible to re-synthesize photo-realistic images. A large body of work is also focused on the digitization of surface appearance; here, vertex colors can be estimated from the observations based on weighted averaging, but tracking drift and insufficient geometric resolution leads to blur. Textures are an alternative that tackles the problem of missing spatial resolution; here, the geometry and color resolutions are decoupled, however, one must find a consistent uv-mapping. One approach to compensate for camera drift and slightly wrong geometry is based on finding non-rigid warps [Huang et al. 2017; Zhou and Koltun 2014]. A major benefit of this line of work is that the reconstructed 3D content can be visualized by standard rendering techniques, and the approach directly generalizes to 4D capture as long as non-rigid surfaces can be reliably tracked [Dou et al. 2016; Innmann et al. 2016; Newcombe et al. 2015]. However, at the same time, imperfections in the reconstructions directly translate to visual artifacts in the re-rendering, which currently is the major hurdle towards making content creation from real-world scenes accessible.

2.2 Image-based Rendering

An alternative direction is to only utilize a very coarse geometry proxy and fill in the missing content based on high-resolution 2D textures [Huang et al. 2017]. Image-based rendering (IBR) pushes this to the limit, where the 3D geometry proxy is only used to select suitable views for cross-projection and view-dependent blending [Buehler et al. 2001; Carranza et al. 2003; Hedman et al. 2016; Heigl et al. 1999; Zheng et al. 2009]. The advantage is that the visual quality of the re-rendered images does not exhibit the common artifacts caused by a low-resolution geometric representation. However, many IBR approaches suffer from ghosting artifacts and problems at

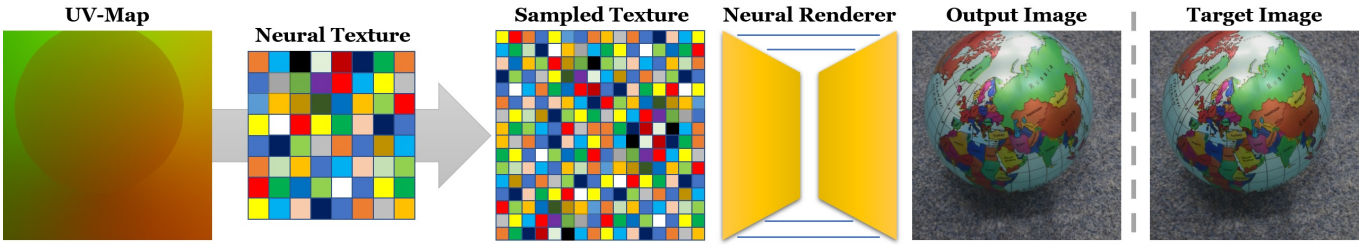

Fig. 2. Overview of our neural rendering pipeline: Given an object with a valid uv-map parameterization and an associated *Neural Texture* map as input, the standard graphics pipeline is used to render a view-dependent screen-space feature map. The screen space feature map is then converted to photo-realistic imagery based on a *Deferred Neural Renderer*. Our approach is trained end-to-end to find the best renderer and texture map for a given task.

the occlusion boundaries. These artifacts can be reduced using optical flow alignment [Casas et al. 2015; Du et al. 2018; Eisemann et al. 2008] or by different view-specific proxies [Chaurasia et al. 2013]. An alternative is to directly model uncertainty [Penner and Zhang 2017]. Our approach also changes the standard rendering pipeline to address the shortcomings of imperfect 3D surface geometry for rendering photo-realistic imagery. However, in contrast to these approaches, *Deferred Neural Rendering* is more flexible, since we learn so-called *neural textures* that efficiently encode the appearance of an object in a normalized texture space.

2.3 Light-field Rendering

There exists a wide range of light-field rendering approaches [Gortler et al. 1996; Levoy and Hanrahan 1996; Magnor and Girod 1999]. In particular, our approach is closely related to the work on surface light fields. Surface light fields store the direction-dependent radiance of every point on the surface, can be used for novel view synthesis, and are able to handle scenes with complex non-Lambertian surface properties. Similar to our approach, a proxy geometry is required. Lumitexels/Lumispheres are used to store the direction-dependent radiance samples at each surface point. To store and query these lumitexels several different methods have been proposed [Chen et al. 2002; Miandji et al. 2013; Wood et al. 2000]. Our neural textures can be seen as a learned analog to these lumitexels, but instead of hand-crafted features, we employ end-to-end learning to find optimal features that can be interpreted by a neural network such that the original images are best reproduced. Recently, Chen et al. presented Deep Surface Light Fields [Chen et al. 2018], which reduces the required number of surface light field samples required for view interpolation using a neural network. The used encoder-decoder network structure that estimates per-vertex colors is specially designed for the surface light field use case and learns to fill missing sample data across vertices. Instead of predicting per vertex colors, we propose an end-to-end trained feature representation of scene appearance based on neural textures and a deferred neural renderer. The deferred neural renderer is a convolutional network that takes the neighborhood of the rasterized surface points into account and, thus, is able to correct for reconstruction errors of the underlying geometry. In contrast, the Deep Surface Light Fields approach needs high quality reconstructions, since a 'slight misalignment can lead to strong artifacts such as ghosting and blurring' [Chen et al. 2018]. In our results section, we demonstrate how our approach can handle a decreasing level of geometry quality and show a comparison to a per-pixel fully connected network.

2.4 Image Synthesis using Neural Networks

With the recent success of deep learning, neural networks can now also be utilized to synthesize artificial 2D imagery. In particular, generative adversarial networks (GANs) [Goodfellow et al. 2014] and auto-regressive networks [Oord et al. 2016] achieve very impressive results in synthesizing individual images. Pure synthesis approaches can be extended to a conditional setting, which is normally tackled with generator networks that follow a classical encoder-decoder architecture [Hinton and Salakhutdinov 2006; Kingma and Welling 2013]. Conditional synthesis can be used to bridge the gap between two different domains, i.e., renderings of incomplete computer vision reconstructions and photo-realistic imagery. Nowadays, conditional GANs (cGANs) are the de facto standard for conditional image synthesis [Isola et al. 2017; Mirza and Osindero 2014; Radford et al. 2016]. Recent approaches use generator networks based on a U-Net [Ronneberger et al. 2015] architecture, a convolutional encoder-decoder architecture with skip connections. In both settings, high-resolution [Karras et al. 2018; Wang et al. 2018b] synthesis has been demonstrated. While such generative approaches have shown impressive results for the synthesis of single, isolated images, synthesizing 3D and temporally-consistent imagery is an open problem. One step in this direction is the vid2vid [Wang et al. 2018a] approach that employs a recurrent network for short-term temporal coherence. Unfortunately, the produced results are not 3D consistent and lack photo-realism as well as long-term coherence. The lack of 3D consistency is an artifact of trying to learn complex 3D relationships with a purely 2D image-to-image translation pipeline, i.e., in view-dependent screen space. Rather than relying on a black-box neural network with a series of 2D convolutions, we propose *neural textures*, which combine the knowledge of 3D transformations and perspective effects from the computer graphics pipeline with learnable rendering. To this end, we efficiently encode the appearance of an object in normalized texture space of a 3D model, learned in an end-to-end fashion, and are hence able to combine the benefits from both graphics and machine learning.

2.5 View Synthesis using Neural Networks

Neural networks can also be directly applied to the task of view synthesis. Novel views can be generated based on a large corpus of posed training images [Flynn et al. 2016], synthesized by learned

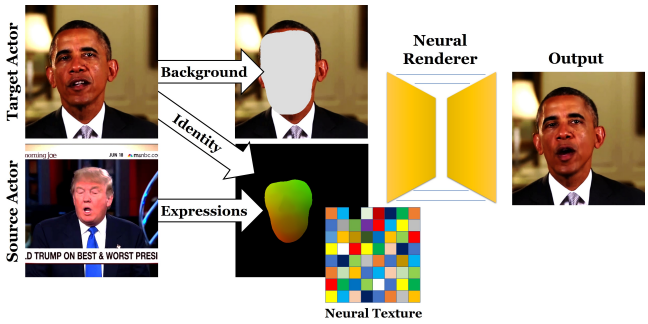

Fig. 3. Overview of our reenactment synthesis pipeline. Using expression transfer, we generate an altered uv map of the target actor matching the expression of the source actor. This uv map is used to sample from the neural texture of the target actor. In addition we provide a background image to the neural renderer, to output the final reenactment result [1].

warping [Park et al. 2017; Zhou et al. 2016], or a layered scene representation [Tulsiani et al. 2018; Zhou et al. 2018]. Disentangled representations with respect to rotation or other semantic parameters can be learned using deep convolutional inverse graphics networks [Kulkarni et al. 2015]. Images can be synthesized based on low-dimensional feature vectors [Eslami et al. 2018; Yan et al. 2016] that obey geometric constraints [Cohen and Welling 2014; Worrall et al. 2017]. For example, latent variables based on Lie groups can be used to better deal with object rotation [Falorsi et al. 2018]. Recent learning-based IBR-approaches learn the blend function using a deep network [Hedman et al. 2018], employ separate networks for color and disparity estimation [Kalantari et al. 2016], explicitly learn view-dependent lighting and shading effects [Thies et al. 2018], or integrate features into a persistent Cartesian 3D grid [Sitzmann et al. 2019]. While these approaches enforce geometric constraints, they yet have to demonstrate photo-realistic image-synthesis and do not directly generalize to dynamic scenes. Our approach combines 3D knowledge in the form of neural textures with learnable rendering and gives us explicit control over the generated output allowing for a wide range of applications.

3 OVERVIEW

The motivation for our work is to enable photo-realistic image synthesis based on imperfect commodity 3D reconstructions. At the core of our approach are our *Neural Textures* that are learned jointly with a *Deferred Neural Renderer*. Neural Textures are a new graphics primitive that can have arbitrary dimension and store a high-dimensional learned feature vector per texel. Using the standard graphics pipeline, neural textures are sampled, resulting in a feature map in the target image space (see Fig. 2). Based on a trained *Deferred Neural Renderer*, the sampled image space feature map is then interpreted. The renderer outputs the final image that photo-realistically re-synthesizes the original object.

[1] Third Party Material: Trump, MSNBC (https://youtu.be/Tsh_V3U7EfU) [Fair Use] Obama, The Obama White House (https://youtu.be/d-VaUaTF3_k) [Public Domain]

Neural Textures are the basis for a wide variety of applications ranging from novel-view synthesis to video editing. Here, we concentrate on the use cases that are most relevant to computer graphics: 1) *Neural Textures* can be used to texture a given mesh and, thus, can be easily integrated into the standard graphics pipeline. In particular, for 3D scanned objects (e.g., via KinectFusion [Izadi et al. 2011; Newcombe et al. 2011] or multi-view stereo reconstructions [Schönberger and Frahm 2016]), where additional ground truth color images are available, they enable learning photo-realistic synthesis of imagery from novel view points (see Fig. 3). 2) *Neural Textures* can also be used to edit dynamic scenes in a similar fashion. Specifically, we show reenactment examples for human faces (e.g., Face2Face [Thies et al. 2016]), where we first reconstruct the 3D face as well as a neural texture, and then modify and realistically re-render the 3D facial animation.

In the following, we detail the stages of our deferred neural rendering pipeline, see Sec. 4. Next, we show different use cases, including comparisons to approaches based on standard computer graphics algorithms, see Sec. 5. Finally, we want to inspire the reader to further investigate the benefits of neural textures, see Sec. 6.

4 DEFERRED NEURAL RENDERING

Deferred Neural Rendering combines principles from the traditional graphics pipeline with learnable components. The classical rendering pipeline consists of several stages that can potentially be made learnable. In this work, we are focusing on *Neural Textures* and *Deferred Neural Rendering*, see Fig. 2, which enable us to realize a variety of applications. Lets consider the task of realistically re-rendering an object based on a noisy commodity 3D reconstruction. Given a 3D reconstructed object with a valid uv-texture parameterization, an associated *Neural Texture* map, and a target view as input, the standard graphics pipeline is used to render a view-dependent screen-space feature map. This feature map is then converted to photo-realistic imagery based on a *Deferred Neural Renderer*. Our approach can be trained end-to-end to find the best renderer $\mathcal{R}$ and texture map $\mathbf{T}$ based on a training corpus of N posed images $\{\mathbf{I}_k, \mathbf{p}_k\}_{k=1}^{N}$. Here, $\mathbf{I}_k$ is the k-th image of the training corpus and $\mathbf{p}_k$ are the corresponding camera parameters (intrinsic and extrinsic). We phrase finding the best neural texture $\mathbf{T}^*$ and the best deferred neural renderer $\mathcal{R}^*$ for a specific task as a joint optimization problem over the complete training corpus:

$$\mathbf{T}^*, \mathcal{R}^* = \underset{\mathbf{T},\mathcal{R}}{\operatorname{argmin}} \sum_{k=1}^{N} \mathcal{L}(\mathbf{I}_k, \mathbf{p}_k | \mathbf{T}, \mathcal{R}) \ . \tag{1}$$

Here, $\mathcal{L}$ is a suitable training loss, i.e., a photometric re-rendering loss. In the following, we describe all components in more detail.

4.1 Neural Textures

Texture maps are one of the key building blocks of modern computer graphics. Typically, they contain appearance information, such as the albedo of an object, but they can also store custom attributes, such as high-frequency geometric detail in the form of normal or displacement maps. These textures can be thought of as low-dimensional hand-crafted feature maps that are later on interpreted by programmed shader programs. With this analogy in

mind, *Neural Textures* are an extension of traditional texture maps; instead of storing low-dimensional hand-crafted features, they store learned high-dimensional feature maps capable of storing significantly more information and can be interpreted by our new deferred neural rendering pipeline. Instead of rebuilding the appearance of a specific object using hand-crafted features, we learn them based on a ground truth training corpus of images. Both the neural textures and the deferred neural renderer are trained end-to-end, enabling photo-realistic image synthesis even if the original 3D content was imperfect. In our experiments, we use 16 feature channels. It is possible to employ an intermediate re-rendering loss to enforce that the first 3 feature channels represent the average color of the object, i.e., mimic a classical color texture map.

4.2 Neural Texture Hierarchies

Choosing the right texture resolution for a given scene and viewpoint is a challenging problem. In particular, for complex scenes with high depth complexity there might not exist a single optimal choice. Given a fixed texture resolution, if parts of the scene are far away from the viewer, the texture will be under-sampled, i.e., texture minification. In contrast, if parts of the scene are close to the virtual camera, the texture is over-sampled, i.e., texture magnification. Both minification and magnification might appear at the same time for different parts of the scene. In classical computer graphics, Mipmaps are used to tackle this challenge. Inspired by classical Mipmaps, we propose to employ *Neural Texture Hierarchies* with K levels. We access the texture hierarchy by sampling values from all K levels using normalized texture coordinates and bi-linear sampling. The final color estimate is then obtained by adding all per-level sampling results (Laplacian Pyramid). During training, our goal is to learn the best *Neural Texture Hierarchy* that stores low frequency information on the coarse levels, while high frequency detail is represented on the finer levels. We enforce this split during training based on a soft-constraint, i.e., we apply no regularization to the coarse levels and an increasing amount of ℓ_2 regularization to the features channels of the finer levels. *Neural Texture Hierarchies* enable us to obtain higher quality results than with *Neural Textures* alone; for an evaluation, see Fig. 8. The improved quality is due to the fact that if only one high resolution neural texture would be used, which solves the minification problem, we run into overfitting problems during training due to texture magnification. Overfitting the training data is problematic, since it leads to sampling issues during test time where unoptimized pixel values might be sampled. Even though we do not explicitly specify a Mipmap interpolation scheme, we want to highlight that the network as well as the texture is trained end-to-end; thus, the 2D renderer has to learn a proper mapping of the Laplacian Pyramid along the entire training-set, including the compensation of aliasing effects in image space.

4.3 Differentiable Sampling of Neural Textures

One key property of traditional texture maps is that they can be sampled at arbitrary floating point image locations. To this end, the returned color value is computed based on an interpolation scheme. Our *Neural Textures*, similar to the standard graphics pipeline, support bi-linear interpolation for sampling the stored high-dimensional feature maps in a differentiable manner. This enables end-to-end training of our *Neural Textures* together with the *Deferred Neural Renderer*. With the support for bi-linear sampling, we can use the standard graphics pipeline to rasterize the coarse proxy geometry and sample the neural texture. This results in a view-dependent screen space feature map. Note, during training we emulate the graphics pipeline, such that the forward and backward pass will exactly match the operations at test time.

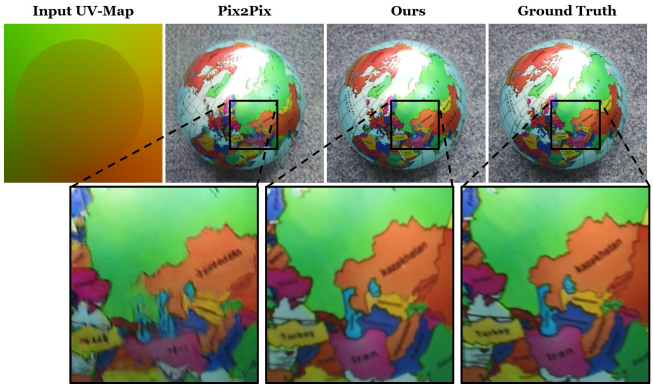

Fig. 4. Comparison to the image-to-image translation approach Pix2Pix [Isola et al. 2016]. As can be seen, the novel views synthesized by our approach are higher quality, e.g., less blurry. Our results are close to the ground truth.

4.4 Deferred Neural Renderer

The task of the *Deferred Neural Renderer* is to form a photo-realistic image given a screen space feature map; this can be thought of as an analogy to classical deferred rendering. We obtain the screen space feature map by rendering the coarse geometric object proxy, which is textured with the neural texture, from the ground truth viewpoint using the traditional rasterization pipeline. Before training commences, we precompute the required texel-to-pixel mapping for each of the training images (referred to as uv map in this paper). At test time, the rasterizer of the graphics pipeline can be employed. Based on this precomputed mapping, differentiable sampling can be used to obtain the screen space feature map via a lookup. Our *Deferred Neural Renderer* is based on recent advances in learning image-to-image mappings based on convolutional encoder-decoder network with skip-connection, similar to U-Net [Ronneberger et al. 2015] (see Appendix A). Our network can have additional inputs such as a view-direction. For view-dependent effects, we explicitly input the view-direction using the first 3 bands of spherical harmonics resulting in 9 feature maps. Prior to the actual network, we multiply the sampled features with the evaluated spherical harmonic basis functions (feature channels $4-13$). This allows us to rotate features with respect to the view-direction. In Fig. 13, we show the advantage of using such a spherical harmonics layer. We train our neural texture representation end-to-end with the rendering network.

4.5 Training

We train our neural rendering approach end-to-end using stochastic gradient descent. Since training images are static, we are able to precompute texture look-up maps which we will refer to as *uv-maps* in the following. We build training pairs consisting of an *uv*-map and the corresponding ground truth color image.

In all experiments, we employ an ℓ_1 photometric reproduction loss with respect to the ground truth imagery. The ℓ_1 loss is defined on a random crops (position and scale of the crops varies) of the full frame image. Random cropping and scaling acts as a data augmentation strategy and leads to significantly better generalization to new views at test time. We employ the Adam optimizer [Kingma and Ba 2014] built into PyTorch [Paszke et al. 2017] for training. The *Deferred Neural Renderer* and the *Neural Textures* are jointly trained using a learning rate of 0.001 and default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \cdot e^{-8}$ for Adam. We run approximately $50k$ steps of stochastic gradient descent.

4.6 Training Data

Our rendering pipeline is trained based on a video sequence that is also used to reconstruct the object. The synthetic sequences consist of 1000 random training views and a smooth trajectory of 1000 different test views on the hemisphere. The synthetic objects are provided by Artec3D[2]. The average angular difference between the novel view direction and their nearest neighbor in the training set is 2.13° (with a minimum of 0.04° and a maximum of 6.60°). In our ablation study w.r.t. the number of training images (see Fig. 9), the angular difference increases with a decreasing number of images: 500 images results in an average of 3.30° (0.11° min / 9.76° max), 250 images in 4.4° (0.11°/12.67°) and 125 images in 6.27° (0.23°/18.79°). For real sequences, the training corpus size varies between 800 and 1700 frames depending on the sequence and the target application with similar angular differences from the test set to the training set (e.g., Sequence 10 with 800 training images and a mean angular difference of 0.92° (0.04°/3.48°), and Sequence 6 containing 1700 training images with a mean angular difference of 0.85° (0.02°/2.3°)). Note that the angular differences in the viewing directions do not consider the positional changes in the cameras used for novel view synthesis.

For facial reenactment, a variety of facial expressions in the training video is necessary, otherwise it looses expressiveness. In our experiments, we used 650 training images for Macron, 2400 for Obama, and 2400 for Sequence 17.

5 RESULTS

The experiments on 2D neural textures are based on RGB input data. We first reconstruct the geometry of the target object and estimate a texture parametrization. The training data is created by re-rendering the *uv*-maps of the mesh that correspond to the observed images. Using this training data, we optimize for the neural texture of the object, allowing us to re-render the object under novel views (see Sec. 5.1) or animate the object (see Sec. 5.2). We used neural textures with a resolution of 512×512 with 4 hierarchy level, containing 16 features per texel and a U-Net with 5 layers as a neural renderer.

[2]https://www.artec3d.com/3d-models

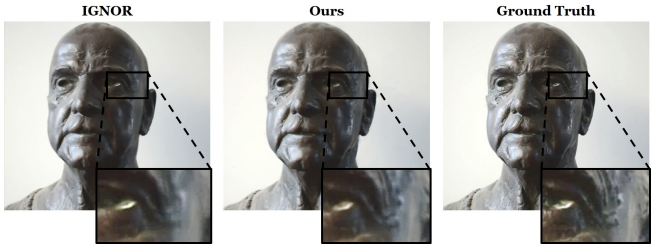

Fig. 5. Comparison to the novel view point synthesis approach IGNOR [Thies et al. 2018]. Our approach better reproduces high frequency specular highlights.

The quality of our image synthesis approach can best be judged from the results shown in the supplemental video.

5.1 Novel View Point Synthesis

For the novel view point synthesis of static objects, we require a 3D reconstruction and camera poses. The training video sequence has been captured at a resolution of 1920×1080 pixels at 30 Hz using a DSLR camera. We obtain a coarse geometric proxy and the camera parameters (intrinsic and extrinsic) using the COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016] structure-from-motion approach. To handle video sequences, which results in several hundred input images, we only use a subset of frames (every $25th$ frame) for the dense reconstruction. The other frames are registered to this reconstruction. The *uv*-parameterization is computed based on the Microsoft *uv*-atlas generator [3].

The learnable rendering pipeline allows us to re-render objects in a photo-realistic fashion. We only consider one object during training time, which allows us to optimize for the object-specific texture and appearance. Object-specific learning for novel view synthesis is known from recent publications [Sitzmann et al. 2019; Thies et al. 2018]. Fig. 4 shows a novel view synthesis generated by our approach in comparison to an image-to-image translation network (Pix2Pix [Isola et al. 2016]) that directly predicts the output image based on the input *uv*-map. Our approach outperforms Pix2Pix in terms of image quality with much sharper results. Our results are also more temporally coherent (c.f. supplemental video). In Fig. 5 we compare our approach to the image-guided neural object rendering approach (IGNOR) of Thies et al. [2018]. In contrast to this image-guided rendering approach, our technique demonstrates a temporally more stable re-rendering of objects. Our approach also does not need to store multiple of the original images for re-rendering, instead we store a single neural texture that enables us to synthesize the object under new views.

In Fig. 6, we also show comparisons to classical IBR methods. As baseline we implemented the IBR method of Debevec et al. [Debevec et al. 1998]. This approach directly reprojects the appearance of the captured views into the target view. The original technique uses a per-triangle nearest neighbor view selection to reproject the appearance into the target view, instead, we do this view selection on a per-pixel basis, resulting in higher quality. The nearest neighbors are selected among the entire training set (i.e., 1686 views). This per-pixel selection is far from real-time, but ensures optimal

[3]https://github.com/Microsoft/UVAtlas

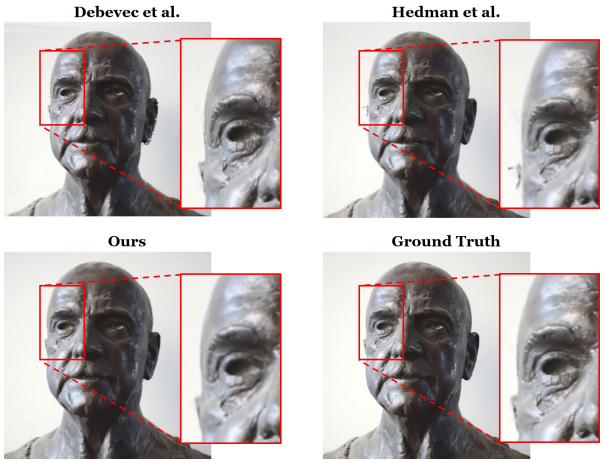

Fig. 6. Comparison to Debevec et al. [Debevec et al. 1998] (based on 1686 views) and Hedman et al. [Hedman et al. 2016] (based on the 99 views used for 3D reconstruction, since this approach requires the stereo reconstructed depth of the frames).

per-pixel results. Note, that this IBR method heavily relies on the reconstructed geometry. The MSE is 37.58 compared to 20.65 for our method. Especially, on occlusion boundaries the classical IBR method shows artifacts (see supplemental video). The authors of Hedman et al. [Hedman et al. 2016] ran a comparison on the same sequence. Besides the reconstructed model, they also use the reconstructed per frame depth. Thus, their approach is limited to the frames used for the multi-view stereo reconstruction (in this sequence every $25th$ frame, resulting in a set of 99 frames). Their approach is able to generate high quality results with an MSE of 28.05; improving the handling of occlusion boundaries in comparison to Debevec et al.. In comparison to both image-based rendering approaches, our technique does not require access to the training data during test time. Only the texture ($512 \times 512 \times 16$) and the rendering network (16 million parameters) has to be provided which is magnitudes lower than storing hundreds of high resolution images. In case of InsideOut [Hedman et al. 2016], also the depth of each frame has to be stored. Note that our approach needs to be trained on a specific sequence which is a drawback in comparison to the IBR methods. Training takes a similar amount of time as the stereo reconstruction and ensures high-quality synthesis of novel views.

Fig. 7 and Fig. 8 show a study on the influence of the resolution of the employed neural texture. As can be seen, the best results for a single texture without hierarchy are obtained for a resolution of 256×256 achieving an MSE of 0.418. This sweet spot is due to tradeoffs in texture minification and magnification. The hierarchical neural texture is able to further improve quality while increasing texture resolution, i.e., an MSE of 0.38 at a resolution of 2048×2048. Our approach is able to synthesize novel views based on a relatively small training set, but the quality of view-dependent effects, such as specular highlights, is slowly degrading with a decreasing number of training images (see Fig. 9).

In Fig.10 we show an ablation study on the influence of the proxy geometry. We gradually reduce the geometry resolution using

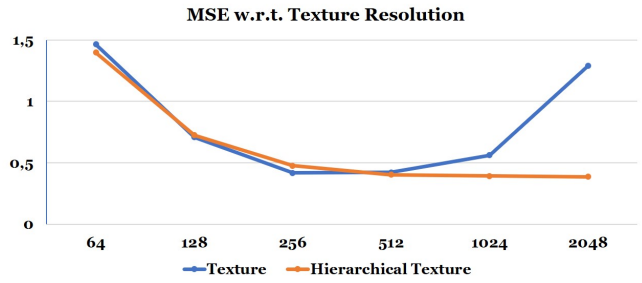

Fig. 7. Influence of the neural texture resolution on the re-rendering quality. The graph is showing the MSE of the image synthesis vs. the ground truth test data with respect to the neural texture resolution. In this experiment, a single texture without hierarchy has its sweet spot at a resolution of 256×256 achieving an MSE of 0.418 with an increasing error for higher resolutions (at a resolution of 4096×4096 it reaches an MSE of 8.46). In contrast, the hierarchical texture performs better on higher resolutions, i.e., an MSE of 0.38 at a resolution of 2048×2048. The MSE is computed on color values in the range of [0, 255] using a test sequence of 1000 frames, based on a synthetic rendering of a vase with Phong shading (see Fig. 8).

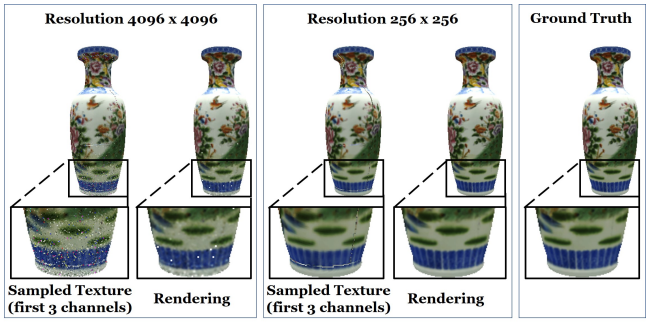

Fig. 8. Influence of the neural texture resolution on re-rendering. Sample images of re-renderings using single neural textures. In addition to the re-renderings, we also show the first three channels of the sampled neural texture which are regularized to be the mean color.

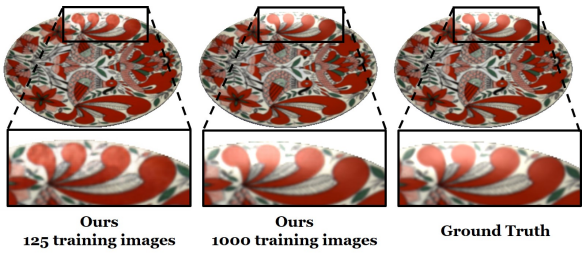

Fig. 9. The number of images used for training influences the reconstruction ability. In particular, view-dependent effects such as specular highlights slowly degrade. Given the object above we achieve an MSE of 2.2 for 1000, 2.2 for 500, 6.5 for 250 and an MSE of 16.9 for only 125 images in the training set (assuming color values in the range of [0, 255]).

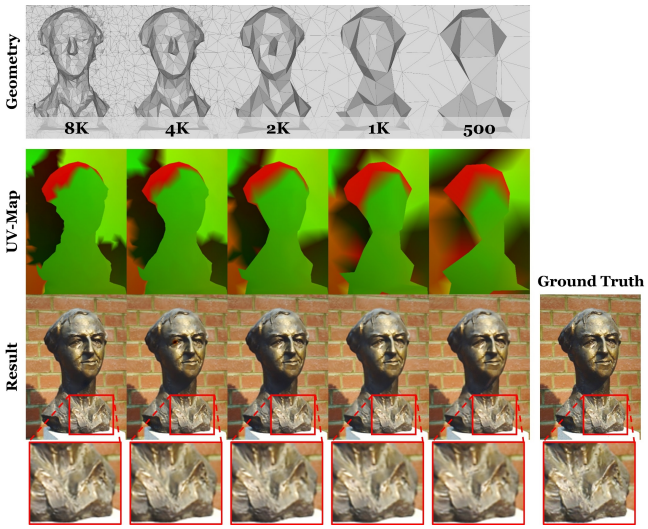

Fig. 10. Ablation study w.r.t. the resolution of the underlying geometry proxy. Using quadric edge collapse we gradually reduce the number of triangles of the geometry from 8000 to 500. Even with the lowest resolution, a photo-realistic image can be generated. The MSE measured on the test sequence increases from 11.068 (8K), 11.742 (4K), 12.515 (2K), 18.297 (1K) to 18.395 for a proxy mesh with 500 triangles (MSE w.r.t. color channels in [0, 255]).

quadric edge collapse. As can be seen, our approach is also able to reproduce a reasonable output image given a very coarse mesh.

To analyze the effect of the U-Net-like renderer, we also trained a per-pixel fully connected network that outputs the final image (see Fig.11). Thus, this network does not leverage neighborhood information and is solely based on the per-pixel sampled texture values. We use the same structure as our rendering network and replace all convolutions by 1×1 convolutions with stride 1. The idea of taking a per-pixel network is similar to the approach of Chen et al. [Chen et al. 2018] which estimates view-dependent per-vertex colors using a fully-connected network. Since we are using a hand-held video sequence to reconstruct the object, the geometry contains errors. As shown in our experiment, the neighborhood information is crucial to correct for these geometry errors. In contrast, a per-pixel network leads to more blurry outputs (see supplemental video). This observation is also mentioned in the paper of Chen et al. [Chen et al. 2018] and, thus, they have to take special care to reconstruct high quality geometry as input.

Similar to other learning-based approaches, view extrapolation results in artifacts (see Fig.12). As can be seen, our approach is still able to synthesize reasonable views of the object in the areas that where captured in the training video.

Scene Editing. Using our rendering technique based on neural textures, we are able to apply edits to a scene (see Fig. 14). Given a reconstruction of the scene, we can move objects around, remove, or duplicate them (see Fig. 16). In contrast to a standard image-to-image translation network, our method better generalizes to scene edits and generates temporally stable, high quality results (see Fig. 15).

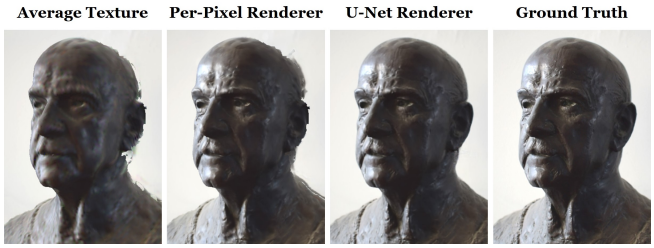

Fig. 11. Here, we show a comparison of a U-Net renderer vs. a per-pixel fully connected render network. Since the U-Net renderer uses neighborhood information, it is able to correct the reconstruction errors of the underlying geometry. We also show the rendering based on a classical average texture.

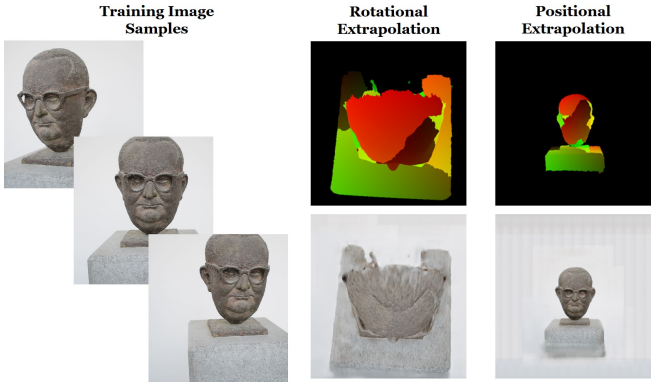

Fig. 12. Here, we show an extrapolation example, where we use a video of a bust captured from the front. Both rotational and positional extrapolation leads to reasonable results.

5.2 Animation Synthesis

In this section, we show the capability to re-render dynamic objects, which allows us to implement applications such as facial reenactment (see Fig. 3). To this end, we compare against the facial reenactment pipeline Face2Face [Thies et al. 2016]. We leverage the 3D face reconstruction of Face2Face and the parametrization of the template mesh to generate the training data. As a result of the training procedure, we optimize for a person-specific neural texture and a person-specific neural renderer. We use the deformation transfer technique of Face2Face to re-render novel uv-maps of the target person corresponding to the expressions of a source actor which is the input to our technique. Fig. 17 shows the advantage of the trained neural texture. It better captures the idiosyncrasies of the target actor, thus, enabling us to improve over state-of-the-art reenactment approaches that are either computer graphics based Thies et al. [2016] (Face2Face) or learned Kim et al. [2018] (DeepVideoPortraits).

Note that we only have to train a single neural texture and renderer once for a target actor. We do not have to store multiple textures for different expressions like in paGAN [Nagano et al. 2018]. Reenactment can be done with any new source actor. Even real-time reenactment is possible since the rendering only takes $\approx$ 4ms (mean evaluation time for the test sequence in Fig. 17 on a Nvidia 1080Ti) in comparison to 10ms for the rendering process of Face2Face.

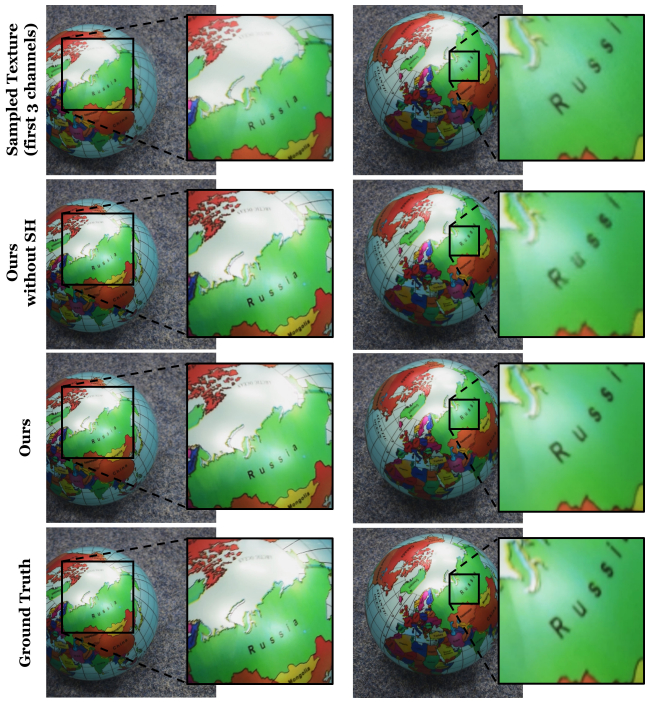

Fig. 13. Novel View Synthesis: in contrast to a static mean texture, specular highlights are consistently reproduced with our approach. We also show the result without the spherical harmonics layer. As can be seen, the output is still photo-realistic, but it has a high MSE of 58.1 compared to 48.2 with our full pipeline (measured on a test sequence with 250 images). The right column shows the differences in visual quality. With the spherical harmonics layer, the re-rendering is sharper and has a clearer specular highlight.

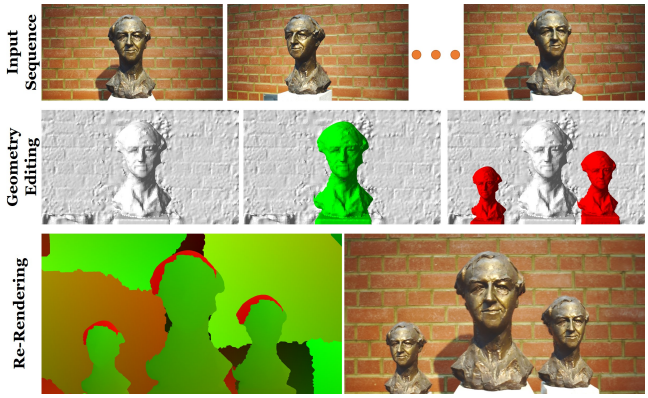

Fig. 14. Overview of our scene editing pipeline. Given a video sequence, we estimate the geometry of the scene. We are then able to modify the geometry, e.g., by move and copy edits. During the edits, we keep track of the uv-parametrization of the vertices, which allows us to render a uv-map that is used as input to our neural renderer. The renderer and its corresponding texture is optimized using the original input images and the original geometry. Using the modified uv-map as input, we can apply the renderer to produce a photo-realistic output of the edited scene.

While our approach is able to generate photo-realistic reenactment results, it raises ethical concerns. Our digital society is strongly

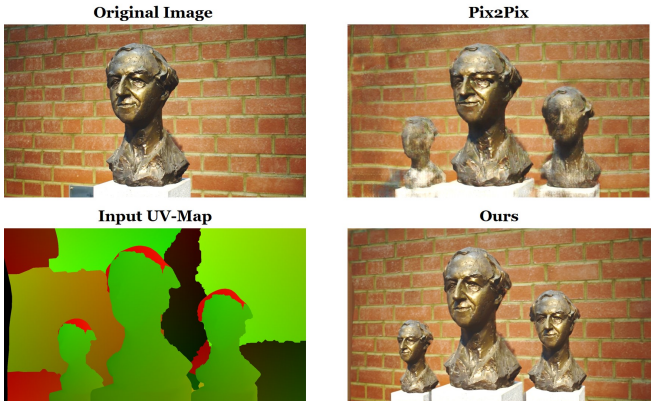

Fig. 15. Editing comparison to Pix2Pix [Isola et al. 2016]. As can be seen, our approach better generalizes to scene edits.

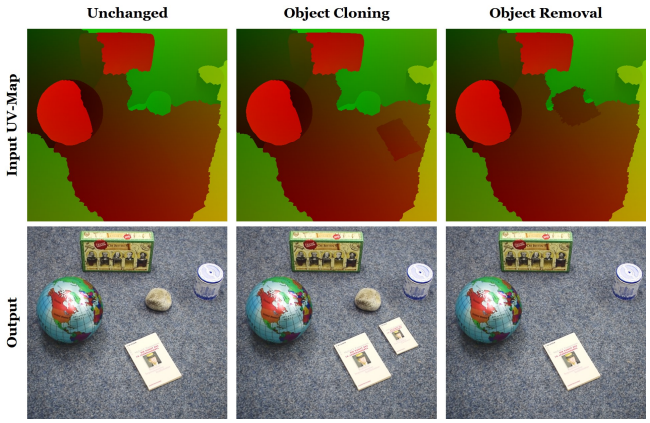

Fig. 16. Editing a scene with multiple objects, including removal and cloning. Our approach obtains photo-realistic results.

relying on the authenticity of images and video footage. On one hand researchers try to close the gap between computer generated images and real images to create fantastic looking movies, games and virtual appearances. On the other hand, these techniques can be used to manipulate or to create fake images for evil purposes. Image synthesis approaches, especially the ones related to humans, are therefore of special interest to the digital media forensics community. Recently, techniques have been proposed that employ deep neural networks to detect such generated images [Rössler et al. 2018, 2019]. The drawback of these methods is that they rely on a huge training set. Transferability to other synthesis approaches is very limited, but is getting into the focus of researchers [Cozzolino et al. 2018].

6 CONCLUSION

In this work, we have presented neural textures, and demonstrated their applicability in a large variety of applications: novel view synthesis, scene editing, and facial animation editing. Interestingly, our rendering is also significantly faster than traditional reenactment, requiring only a few milliseconds for high-resolution output. However, we believe this is only a stepping stone to much wider

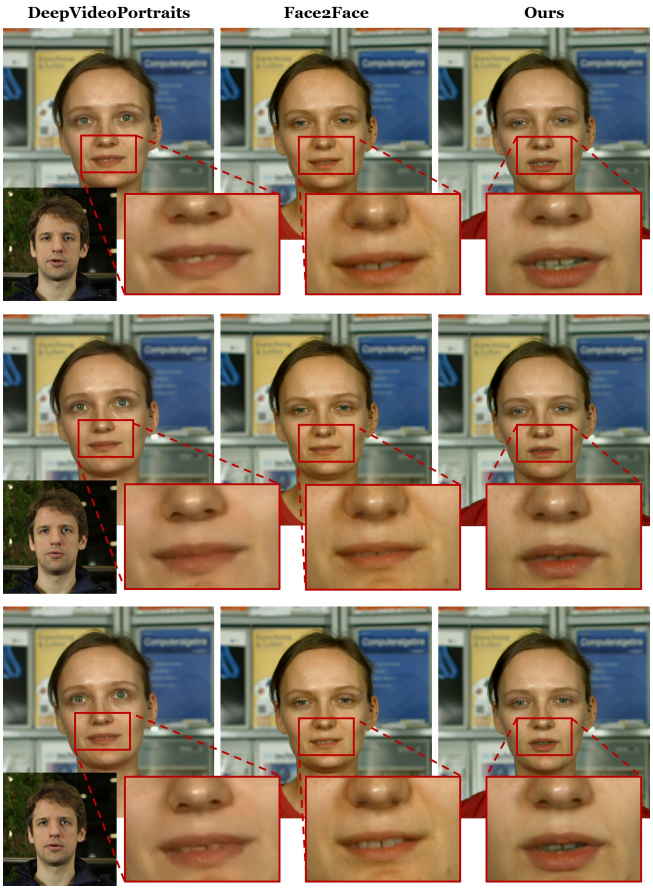

Fig. 17. Comparison to the state-of-the-art facial reenactment approaches of Kim et al. [2018] (DeepVideoPortraits) and Thies et al. [2016] (Face2Face). As can be seen, our learning-based approach better captures the person-specific idiosyncrasies of the target actor. The mouth interior has more variety, is much sharper, and has less stretching artifacts.

range of applications where we use imperfect 3D content in order to generate photo-realistic output. Our approach relies on a geometry proxy that has to be reconstructed in a preprocessing step. If the geometry is too coarse, the result quality gracefully degrades, i.e., the re-rendered images get more blurry. As the appearance of each object is unique, we must train the neural texture for every new object. However, we believe that neural texture estimation based on a few images in conjunction with a generalized neural renderer is a very promising direction for future work. Preliminary results on synthetic data suggest that our neural renderer can be generalized; currently, on real data, the size of our dataset (<10 objects) is limiting, but we believe that this generalization has potential for many new applications; e.g., transfer learning, where a renderer can be exchanged by another renderer that, for example, renders segmentations or the like.

The idea of neural textures is also not only bound to the two-dimensional setting that is discussed in our work. It would be straightforward to extend the idea to higher dimensions or other data structures (e.g., a volumetric grid). Beyond finding the right scene representation, there are also many other problems that can be tackled, such as disentangled control over the scene illumination and surface reflectance. In our work, we assume static illumination and, thus, we are not able to relight the scene. In addition, we believe there is a unique opportunity to revisit other components and algorithms of the traditional rendering pipeline, such as novel shading and lighting methods, or even use a full differentiable path tracing framework [Li et al. 2018].

In conclusion, we see a whole new field of novel computer graphic pipeline elements that can be learned in order to handle imperfect data captured from the real world. With neural textures, we demonstrated a first step towards this avenue, showing applications for photo-realistic novel view point synthesis, scene editing, as well as animation synthesis. We hope to inspire follow-up work in this direction.

ACKNOWLEDGMENTS

We thank Angela Dai for the video voice over and Peter Hedman for the comparison. We gratefully acknowledge the support by the AI Foundation, Google, a TUM-IAS Rudolf Mößbauer Fellowship, the ERC Starting Grant *Scan2CAD* (804724), and a Google Faculty Award. This work was supported by the Max Planck Center for Visual Computing and Communications (MPC-VCC).

REFERENCES

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured Lumigraph Rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 425–432. DOI: http://dx.doi.org/10.1145/383259.383309

Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. 2003. Free-viewpoint Video of Human Actors. *ACM Trans. Graph. (Proc. SIGGRAPH)* 22, 3 (July 2003), 569–577.

Dan Casas, Christian Richardt, John P. Collomosse, Christian Theobalt, and Adrian Hilton. 2015. 4D Model Flow: Precomputed Appearance Alignment for Real-time 4D Video Interpolation. *Comput. Graph. Forum* 34, 7 (2015), 173–182.

Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth Synthesis and Local Warps for Plausible Image-based Navigation. *ACM Trans. Graph.* 32, 3, Article 30 (July 2013), 12 pages.

Anpei Chen, Minye Wu, Yingliang Zhang, Nianyi Li, Jie Lu, Shenghua Gao, and Jingyi Yu. 2018. Deep Surface Light Fields. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 1, Article 14 (July 2018), 17 pages. DOI: http://dx.doi.org/10.1145/3203192

Jiawen Chen, Dennis Bautembach, and Shahram Izadi. 2013. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 4 (2013), 113.

Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk. 2002. Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*. ACM, New York, NY, USA, 447–456. DOI: http://dx.doi.org/10.1145/566570.566601

Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. 2015. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5556–5565.

Taco S Cohen and Max Welling. 2014. Transformation properties of learned visual representations. *arXiv preprint arXiv:1412.7659* (2014).

Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. 2018. ForensicTransfer: Weakly-supervised Domain Adaptation for Forgery Detection. *arXiv* (2018).

Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. 2017. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 76a.

Paul Debevec, Yizhou Yu, and George Boshokov. 1998. Efficient View-Dependent IBR with Projective Texture-Mapping. EG Rendering Workshop.

Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, and others. 2016. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 114.

Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. 2018. Montage4D: Interactive Seamless Fusion of Multiview Video Textures. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '18)*. ACM, New York, NY, USA, Article 5, 11 pages. DOI: http://dx.doi.org/10.1145/3190834.3190843

M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. 2008. Floating Textures. *Computer Graphics Forum (Proc. EUROGRAHICS* (2008). DOI: http://dx.doi.org/10.1111/j.1467-8659.2008.01138.x

SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, and others. 2018. Neural scene representation and rendering. *Science* 360, 6394 (2018), 1204–1210.

Luca Falorsi, Pim de Haan, Tim R Davidson, Nicola De Cao, Maurice Weiler, Patrick Forré, and Taco S Cohen. 2018. Explorations in Homeomorphic Variational Auto-Encoding. *ICML Workshop on Theoretical Foundations and Applications of Generative Models* (2018).

John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. Deepstereo: Learning to predict new views from the world's imagery. In *Proc. CVPR*. 5515–5524.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proc. NIPS*. 2672–2680.

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The Lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. ACM, New York, NY, USA, 43–54. DOI: http://dx.doi.org/10.1145/237170.237200

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 257.

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 231.

Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc J. Van Gool. 1999. Plenoptic Modeling and Rendering from Image Sequences Taken by Hand-Held Camera. In *Proc. DAGM*. 94–101.

Geoffrey E. Hinton and Ruslan Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 (July 2006), 504–507. DOI: http://dx.doi.org/10.1126/science.1127647

Jingwei Huang, Angela Dai, Leonidas Guibas, and Matthias Nießner. 2017. 3DLite: Towards Commodity 3D Scanning for Content Creation. *ACM Transactions on Graphics 2017 (TOG)* (2017).

Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. 2016. VolumeDeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*. Springer, 362–379.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *arxiv* (2016).

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *Proc. CVPR*. 5967–5976. DOI: http://dx.doi.org/10.1109/CVPR.2017.632

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and others. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 559–568.

Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-Based View Synthesis for Light Field Cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)* 35, 6 (2016).

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *Proc. ICLR*.

H. Kim, P. Garrido, A. Tewari, W. Xu, J. Thies, N. Nießner, P. Pérez, C. Richardt, M. Zollhöfer, and C. Theobalt. 2018. Deep Video Portraits. *ACM Transactions on Graphics 2018 (TOG)* (2018).

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980

Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *CoRR* abs/1312.6114 (2013).

Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. 2015. Deep Convolutional Inverse Graphics Network. In *Proc. NIPS*. 2539–2547.

Marc Levoy and Pat Hanrahan. 1996. Light Field Rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. ACM, New York, NY, USA, 31–42. DOI: http://dx.doi.org/10.1145/237170.237199

Tzu-Mao Li, Michaël Gharbi, Andrew Adams, Frédo Durand, and Jonathan Ragan-Kelley. 2018. Differentiable programming for image processing and deep learning in halide. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 139.

Marcus Magnor and Bernd Girod. 1999. Adaptive Block-based Light Field Coding. In *Proc. 3rd International Workshop on Synthetic and Natural Hybrid Coding and Three-Dimensional Imaging IWSNHC3DI'99, Santorini, Greece*. 140–143.

Ehsan Miandji, Joel Kronander, and Jonas Unger. 2013. Learning Based Compression for Real-time Rendering of Surface Light Fields. In *ACM SIGGRAPH 2013 Posters (SIGGRAPH '13)*. ACM, New York, NY, USA, Article 44, 1 pages. DOI: http://dx.doi.org/10.1145/2503385.2503434

Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. (2014). https://arxiv.org/abs/1411.1784 arXiv:1411.1784.

Koki Nagano, Jaewoo Seo, Jun Xing, Lingyu Wei, Zimo Li, Shunsuke Saito, Aviral Agarwal, Jens Fursund, and Hao Li. 2018. paGAN: real-time avatars using dynamic textures. 1–12. DOI: http://dx.doi.org/10.1145/3272127.3275075

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 343–352.

Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 127–136.

Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. 2013. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)* 32, 6 (2013), 169.

Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. 2016. Conditional Image Generation with PixelCNN Decoders. In *Proc. NIPS*. 4797–4805.

Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C. Berg. 2017. Transformation-Grounded Image Generation Network for Novel 3D View Synthesis. *CoRR* abs/1703.02921 (2017).

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

Eric Penner and Li Zhang. 2017. Soft 3D Reconstruction for View Synthesis. *ACM Trans. Graph.* 36, 6, Article 235 (Nov. 2017), 11 pages.

Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *Proc. ICLR*.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proc. MICCAI*. 234–241. DOI: http://dx.doi.org/10.1007/978-3-319-24574-4_28

Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. 2018. FaceForensics: A Large-scale Video Dataset for Forgery Detection in Human Faces. *arXiv* (2018).

Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. 2019. FaceForensics++: Learning to Detect Manipulated Facial Images. *arXiv* (2019).

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*.

V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer. 2019. DeepVoxels: Learning Persistent 3D Feature Embeddings. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Justus Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. 2016. Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In *Proc. CVPR*.

J. Thies, M. Zollhöfer, C. Theobalt, M. Stamminger, and M. Nießner. 2018. IGNOR: Image-guided Neural Object Rendering. *arXiv 2018* (2018).

Shubham Tulsiani, Richard Tucker, and Noah Snavely. 2018. Layer-structured 3D Scene Inference via View Synthesis. In *Proc. ECCV*.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018a. Video-to-Video Synthesis. In *Proc. NeurIPS*.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018b. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Proc. CVPR*.

Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. 2016. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research* 35, 14 (2016), 1697–1716.

Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. 2000. Surface Light Fields for 3D Photography. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 287–296. DOI: http://dx.doi.org/10.1145/344779.344925

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. 2017. Interpretable transformations with encoder-decoder networks. In *Proc. ICCV*, Vol. 4.

Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. 2016. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Proc. NIPS*. 1696–1704.

Ming Zeng, Fukai Zhao, Jiaxiang Zheng, and Xinguo Liu. 2013. Octree-based fusion for realtime 3D reconstruction. *Graphical Models* 75, 3 (2013), 126–136.
Ke Colin Zheng, Alex Colburn, Aseem Agarwala, Maneesh Agrawala, David Salesin, Brian Curless, and Michael F. Cohen. 2009. Parallax photography: creating 3D cinematic effects from stills. In *Proc. Graphics Interface*. ACM Press, 111–118.
Qian-Yi Zhou and Vladlen Koltun. 2014. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 155.
Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo Magnification: Learning View Synthesis Using Multiplane Images. *ACM Trans. Graph.* 37, 4, Article 65 (July 2018), 12 pages.
Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. 2016. View synthesis by appearance flow. In *Proc. ECCV*. Springer, 286–301.

A NETWORK ARCHITECTURE

Our rendering network is based on a U-Net [Isola et al. 2017] with 5-layers, i.e., an encoder-decoder network with skip connections. For our experiments, we are using the following architecture (see Fig. 18). Based on an image containing 16 features per pixel (i.e., the rendered neural texture), we apply an encoder that is based on 5 convolutional layers each with instance normalization and a leaky ReLU activation (negative slope of 0.2). The kernel size of each convolutional layer is 4 (stride of 2) with output features 64 for the first and 128, 256, 512, 512 respectively for the other layers. The decoder mirrors the encoder, i.e., the feature channels are the same as in the respective encoder layer, kernel size is 4 and stride is 2. For the final output layer, we are using a $TanH$ activation as in Pix2Pix [Isola et al. 2017].

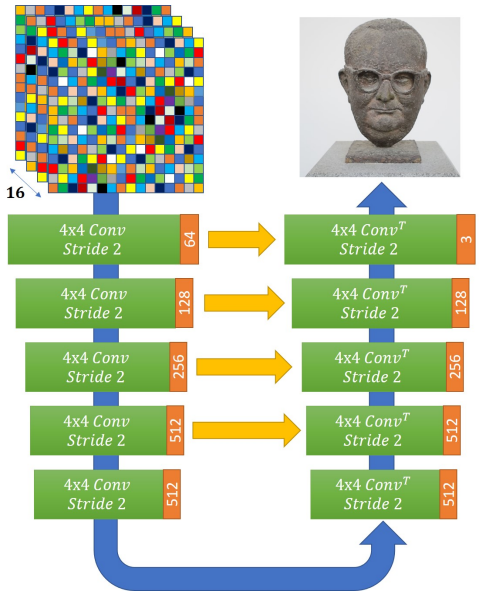

Fig. 18. U-Net architecture of our rendering network. Given the rendered neural textures, we run an encoder-decoder network with skip connections (in yellow), to generate the final output image. Number of output features of the layers are noted in orange.