Eurographics Symposium on Rendering (DL-only Track) (2020)
C. Dachsbacher and M. Pharr (Editors)

Joint SVBRDF Recovery and Synthesis From a Single Image using an Unsupervised Generative Adversarial Network

Yezi Zhao[1], Beibei Wang[†2], Yanning Xu[‡1], Zheng Zeng[1], Lu Wang[1], Nicolas Holzschuch[3]

[1]School of Software, Shandong University
[2]School of Computer Science and Engineering, Nanjing University of Science and Technology
[3]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK

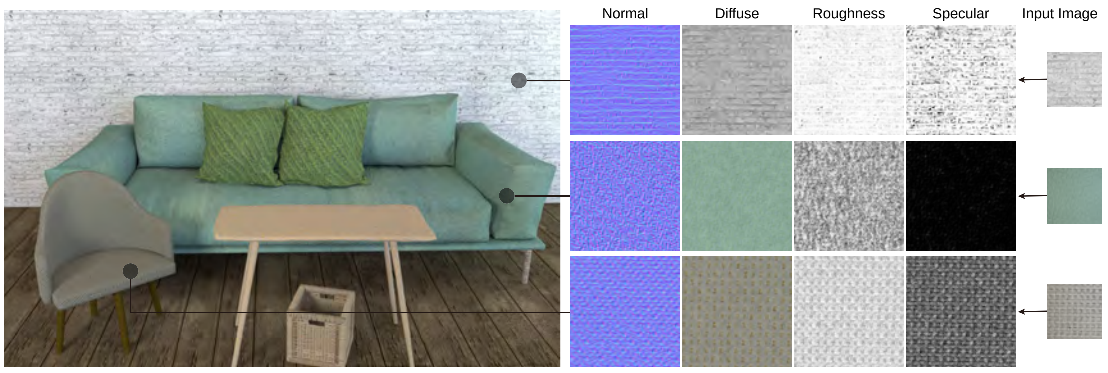

Figure 1: *A 3D scene rendered using our recovered and synthesised SVBRDF maps.*

Abstract

We want to recreate spatially-varying bi-directional reflectance distribution functions (SVBRDFs) from a single image. Producing these SVBRDFs from single images will allow designers to incorporate many new materials in their virtual scenes, increasing their realism. A single image contains incomplete information about the SVBRDF, making reconstruction difficult. Existing algorithms can produce high-quality SVBRDFs with single or few input photographs using supervised deep learning. The learning step relies on a huge dataset with both input photographs and the ground truth SVBRDF maps. This is a weakness as ground truth maps are not easy to acquire. For practical use, it is also important to produce large SVBRDF maps. Existing algorithms rely on a separate texture synthesis step to generate these large maps, which leads to the loss of consistency between generated SVBRDF maps. In this paper, we address both issues simultaneously. We present an unsupervised generative adversarial neural network that addresses both SVBRDF capture from a single image and synthesis at the same time. From a low-resolution input image, we generate a large resolution SVBRDF, much larger than the input images. We train a generative adversarial network (GAN) to get SVBRDF maps, which have both a large spatial extent and detailed texels. We employ a two-stream generator that divides the training of maps into two groups (normal and roughness as one, diffuse and specular as the other) to better optimize those four maps. In the end, our method is able to generate high-quality large scale SVBRDF maps from a single input photograph with repetitive structures and provides higher quality rendering results with more details compared to the previous works. Each input for our method requires individual training, which costs about 3 hours.

CCS Concepts
• ***Computing methodologies*** → ***Texturing;***

1. Introduction

To produce realistic images, we need to model realistic surface appearance, including reflectance, texture and surface variations. A

† Joint first author
‡ Corresponding author

possible method is to have artists and designers use specialized tools to create high-resolution texture and normal maps. Another method is to acquire the material appearance directly. Both of these two methods are time-consuming. A separate issue with textures is that we often need to apply the texture over a large area, larger than the surface used for acquisition.

Recent works have shown that it is possible to recover reflectance functions using only a few photographs of a surface [LDPT17, DAD*18, LSC18, GLD*19]. These lightweight appearance capture approaches recover Spatially-Varying BRDF maps, with diffuse and specular albedo, normal map and roughness parameters from a photograph of a real-world material sample. The capture algorithm uses convolutional neural networks (CNNs), trained with sufficient image data under the guidance of target SVRDF maps.

These learning methods are *supervised*, and require a large training dataset. Most existing works rely on synthetic data generated by a renderer. Li et al. [LDPT17] proposed a self-augmentation method to address the lack of sufficiently large datasets. Deschaintre et al. [DAD*18] provide a large dataset with various kinds of materials. But these synthetic images are still different from real-world images, and these approaches tend to fail when faced with real-world images. Their networks have a low ability to generalize to new inputs.

The generated maps for the SVBRDF are limited to the size of the input images. Even recent work [GLD*19] could produce high-resolution maps relies on a high-resolution input map. Other works have to use texture synthesis after the generation step to produce higher resolution maps. Texture synthesis is a difficult problem in itself. Texture features can be distorted by the synthesis process. Also, the consistency between four SVBRDF maps is difficult to maintain after applying texture synthesis to each separately.

In this paper, we address the same problem as prior works: to generate a large SVBRDF map from a single head-lit flash image of a flat textured surface. We tackle the two problems of SVBRDF reconstruction from the image and synthesis simultaneously: We designed an unsupervised generative adversarial neural network that generates high-resolution maps directly from a single image.

Our key insight is to employ a GAN to produce re-rendered images similar to the input images. In adversarial training, a random vector is usually given to generate "fake data". In contrast, we produce random vectors through an untrained encoder. We visualize feature maps output by the encoder and find that the textures are preserved well. Starting from random vectors with enough texture information helps the generator to converge better.

We also design a two-stream generator to divide the texture maps (normal, roughness, diffuse and specular albedo) into two groups and separate their training apart. This generator consists of one encoder and two decoders. One stream produces normal and roughness, while the other produces diffuse and specular. This two-stream generator is used during the two training stages. We found that this two-stream generator can prevent the network from overfitting the diffuse map and not optimizing the other three. A single generator tends to interpret the entire image as a diffuse map.

Our method generates large scale SVBRDF maps from a single input photograph with high quality and provides higher quality rendering results with more details compared to the previous works.

In summary, our contributions are:

- a novel GAN architecture for unsupervised high-resolution SVBRDF maps recovery and synthesis from a single image.
- a two-stream generator to separate the training of diffuse & specular maps and normal & roughness maps, which can optimize these four maps to generate more accurate rendering results.

The rest of the paper is arranged as follows. In Sec. 2, we review related works. Then we present our method and implementation in Sec. 3. After that, we show and discuss our results in Sec. 4. Finally, in Sec. 5, we summarize our method and propose future insights.

2. Related Work

2.1. Lightweight Reflectance Capture

Lightweight appearance capture refers to methods that recover reflectance parameters and shape or normals from one or a few photographs, as opposed to capturing the entire SVBRDF using an extensive acquisition system with controlled illumination and camera position. In this paper, we focus solely on lightweight appearance capture. Gao et al. [GLD*19] and Guarnera et al. [GGG*16] have presented surveys on lightweight appearance capture.

Appearance modeling based on multiple images Several works capture the SVBRDF using multiple images as input and rely on fitting or optimization to recover the information, with some assumptions, e.g. that the illumination is known or that there is sparsity in some domain. Chandraker and Manmohan [Cha14] utilize motion cues to recover simultaneously jointly recover the shape and BRDF of objects from images under known directional illumination. Hui et al. [HS15] recover SVBRDFs and shape from multiple images under known illuminations. Riviere et al. [RPG16] record a video of a spatially-varying sample using a mobile phone and lit by a flashlight, and use handcrafted heuristics to identify specular and diffuse reflections. Hui et al. [HSL*17] also record a video using a cellphone camera and flash. They use a dictionary-based reflectance prior to derive a robust technique for per-pixel normal and BRDF estimation assuming sparsity. Dong et al. [DCP*14] and Xia et al. [XDPT16] recover the reflectance and shapes from a video of rotating object under unknown natural illumination, assuming the sparsity of the strong edges in the incident lighting.

Appearance modeling based on a single or few images Another group of work only uses one or few images as input, assuming spatial sparseness of materials or stationary. Aittala et al. [AWL*15] used two photographs (one with flash and one without) to recover the reflectance, assuming the maps are stationary. Aittala et al. [AAL16] improved the approach, using one image as input and leveraging the deep convolutional neural network for fitting. Xu et al. [XNY*16] used two images from a near-field perspective camera, and assume spatial relation for reflectance recovery.

Learning-based appearance modeling Deep learning has been very useful for SVBRDF recovering. Li et al. [LXR*18] estimates shape and part of the BRDF using only a single flash image as input. They recover normal and diffuse albedo, as well as roughness, but ignore the specular albedo. Boss et al. [BJK*20] use two images captured by a cellphone with flash both on and off to estimate the shape and SVBRDF. Li et al. [LDPT17] adopt an encoder-decoder architecture to estimate diffuse reflectance and normal maps. They also introduce self-augmentation to expand a small synthetic training set. Li et al. [LSC18] train a CNN to regress an SVBRDF and surface normals from images. They design an in-network rendering layer to model appearance and a material classifier to provide additional supervision during training. They further refine the results from the network using a dense CRF module. Deschaintre et al. [DAD*18] enrich the encoder-decoder architecture with a secondary network that extracts global features at each stage of the network. They also introduce an in-network renderer which computes derivatives automatically during backpropagation to further enhance the estimated reflectance parameters. We adopt this strategy in our work as well. Deschaintre et al. [DAD*19] proposed a learning based SVBRDF estimation system that supports arbitrary number of input. They first use a network (similar to [DAD*18]) to extract latent features from each individual input image. Then the features are assembled and sent to another few convolutional layers to generate final SVBRDFs. Gao et al. [GLD*19] can estimate the SVBRDFs of a planar exemplar from an arbitrary number of input photographs. Their method needs an initial value of SVBRDF maps, and they encode the initial value and directly optimize the parameters in latent space. With more input images, the SVBRDF maps get more precise.

All supervised learning approaches are limited because the resolution of their output is the resolution of their input. On top of that, some methods have to take low resolution input images [LDPT17, LSC18, DAD*18]. Even recent methods [GLD*19] that can deal with large resolution images have this limitation. In comparison, our method is unsupervised. The only thing we need is a photo of the desired surface appearance. During training, we send small tiles cropped from the photo into the network, and this makes the training fast.

2.2. Texture synthesis

Several surveys [WLKT09, AYD*18, RDDM18] cover a wide spectrum of example-based texture synthesis methods and provide a comprehensive overview. Texture synthesis methods can be categorized into three different kinds. The first kind is called *by expansion*, including the classic image quilting methods [EL99, WL00, EF01, LH05] and also modern solutions using Generative Adversarial Networks (GANs) [JBV16, ZZB*18]. Zhou et al. [ZZB*18] use GAN for non-stationary texture synthesis. Their network is trained to double the spatial extent of texture blocks extracted from a specific texture exemplar, and then used to expand the size of the entire exemplar. Our method is inspired by this work, but we improve their network and training procedure to better fit our problem. Hu et al. [HDR19] introduced a framework for inverse procedural texture modeling: trained an unsupervised clustering model to select a most appropriate procedural model and then used a CNN pool to map images to parameters. It solved a different problem, compared to our method.

The second kind is *tiling methods*, such as Wang tiles [Wan61, CSHD03]. These methods first create small tiles from the input texture. These tiles are designed to allow seamless stitching to others, and are thus used as building blocks to generate larger textures. However, Wang tiles do not work well on structured or anisotropic textures. More recent work used a Generative Adversarial Networks [FAW19] to synthesize very large terrains from a rough guidance map. However, it requires thousands of tiles in the training step, which is not suitable for our problem.

The third kind is *blending methods*, or by-example noise methods. They assume that any point on the resulting texture is blended from several patches from the input texture (example). Different blending methods of the example patches are possible, from simple linear blending (prone to "ghosting" artifacts), to more advanced variance preserving [YNBH11] and histogram preserving [HN18] methods. Gatys et al. [GEB15] propose the first deep learning method for example-based texture synthesis. They use the pre-trained image classification network VGG-19 to extract features at different layers and use Gram matrices to compare differences in a statistical way. However, the blending methods do not work well on structured images, and relied on a high resolution target image.

2.3. Generative adversarial network

GANs [GPAM*14] are powerful at generating data from a given prior distribution. They consist of two networks, a generator G and a discriminator D. The generator tries to create fake but plausible images, while the discriminator tries to distinguish fake images (produced by the generator) from real images. To train the networks, the loss function is formulated as:

$$\min_G \max_D \mathbb{E}[\log D(\mathbf{x})] + \mathbb{E}[\log(1 - D(G(\mathbf{z})))] \qquad (1)$$

where $\mathbf{z}$ denotes a noise vector, $\mathbf{x}$ denotes the real images. During training, the discriminator is jointly updated with the generator.

Recently this adversarial learning technique has been employed in various image manipulation tasks. To our knowledge, we are the first to bring it to the appearance modeling task.

Usually, a generator generates images from random noise (often called latent vector/code) from Gaussian distribution. Starting from a prior distribution far away from target distribution makes the training of GANs challenging. Recently image generating works [IZZE17, BCW*17] use Variational Autoencoders (VAE) to get latent vectors at a good starting point. VAEs encode the latent variables and learn the prior distribution of the data that is to be generated. The Encoder is able to learn the distribution of the data and latent vectors can be sampled from this distribution rather than generating random noise. We also use an encoder to produce latent vectors, but the encoder is untrained.

3. Our Method

Given a single entire head-lit flash image of a flat, textured surface taken by phone, we aim to recover and also synthesize the SVBRDF maps with higher resolution. We proposed a novel unsupervised generative adversarial network for these joint tasks (Section 3.1), and introduced a combined loss function (Section 3.2).

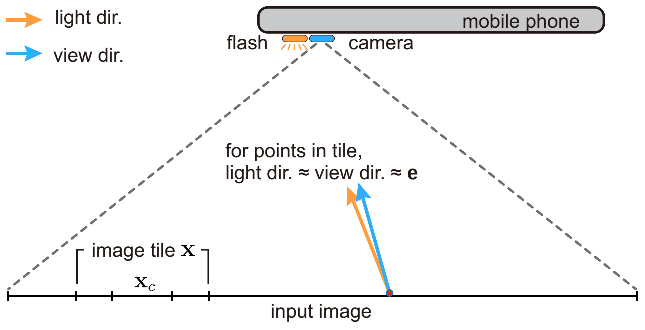

Figure 2: *Imaging setup.*

Preliminaries. We follow the assumption and the setup proposed by [AAL16] to get an image. As shown in Figure 2, the light source is very close to camera, so at each individual pixel, the lighting and viewing directions can be considered identical. Given an input image with size $m \times m$, our network generates SVBRDF maps encoded in a multi-channel image $\mathbf{u}$ with size $2m \times 2m$, consisting of the diffuse albedo $\rho_d \in \mathbb{R}^3$, specular albedo $\rho_s \in \mathbb{R}$, roughness $\alpha \in \mathbb{R}$, and surface normal $\mathbf{n} \in \mathbb{R}^3$, where $\mathbf{u} = [\rho_d, \rho_s, \alpha, \mathbf{n}]$. We use Cook-Torrance reflectance model [CT82] for rendering. The rendered result in a given lighting condition is denoted as $\mathbf{y} = R(\mathbf{u}; \mathbf{e}, \mathbf{l})$, where $\mathbf{e}$ is the half-vector, $\mathbf{l}$ is the irradiance value, and they are estimated similarly to Aittala et al. [AAL16].

3.1. Network architecture

Our GAN consists of a *two-stream generator* and a *patch discriminator* (see Figure 3). From the input image, we first randomly choose a candidate tile $\mathbf{x}$ (with the size of $2n \times 2n$), crop a tile $\mathbf{x}_c$ (with the size of $n \times n$) from the center of $\mathbf{x}$ and feed it into the generator. The size of the input image ($m \times m$) should be at least 4 times of the tile size ($n \times n$). In the generator, we generate the maps of the tile with two groups separately: normal and roughness, diffuse and specular, called two-stream generator. Then we use the predicted SVBRDF maps to render an image and distinguish the rendered image and the candidate tile $\mathbf{x}$ from the input image in the discriminator.

Generator. Our generator consists of one untrained encoder En and two decoders $De_{n,\alpha}$, De_{ρ_d,ρ_s}, where $De_{n,\alpha}$ produces normal and roughness and De_{ρ_d,ρ_s} produces diffuse and specular.

The encoder (see Figure 4(a)) takes in a $n \times n$ image and extracts high dimensional features. The encoder has 4 convolutional layers, with each followed by an instance normalization and leaky-ReLu activation. The extracted features are taken by the decoders $De_{n,\alpha}$ and De_{ρ_d,ρ_s} to produce n, α, ρ_d, ρ_s:

$$G_{n,\alpha} = De_{n,\alpha}(En(x_c)) = [n, \alpha] \tag{2}$$

$$G_{\rho_d,\rho_s} = De_{\rho_d,\rho_s}(En(x_c)) = [\rho_d, \rho_s] \tag{3}$$

Figure 4(c) shows the decoder architecture. Both of the two decoders have five transposed convolutional layers. The first four layers are followed by instance normalization and leaky-ReLU activation function. The two decoders differ in the output channel of the last transposed convolutional layer. $De_{n,\alpha}$ outputs a two-channel image of SVBRDF parameters: one channel for height map, and one channel for roughness. De_{ρ_d,ρ_s} outputs three channels for RGB diffuse albedo, and one channel for specular albedo. We use the height map rather than the normal map directly, for the network stability reason.

Discriminator. The generated SVBRDF maps from the decoder are rendered into a re-rendering image. Both this rendered image and the candidate tile with size $2n \times 2n$ from the input image are fed to the discriminator (denoted as Dis) to determine the correctness of the generated SVBRDF maps. More specially, we employ the patch discriminator architecture proposed by [IZZE17] (see Figure 4(b)). It consists of five convolutional layers. The first 4 convolutional layers are followed by instance normalization and leaky-ReLu activation. After the fifth convolutional layer, the high dimensional features are projected into a scalar using 1×1 convolution and sigmoid activate function.

Discussions:

- We used an untrained encoder in our network, as starting from random vectors with enough texture information helps the generator to converge better. Figure 5 visualizes several layers outputted by the encoder, and shows the textures are preserved well.
- We used a two-stream generator: one stream for normal and roughness and the other for diffuse and specular in our decoder. The key idea behind is to force the network to optimize maps with balance. In comparison, other methods [LDPT17, LSC18, DAD*18] used one stream for all the maps and trended to overestimate the diffuse map. Figure 15 compares our two-stream generator against one steam (encoder-decoder) architecture generator.
- We also considered using one decoder for each map or even one encoder-decoder network for each map, but it increases both the training time and redundant variables. Since the encoder extracts features from low-dimensional inputs, the SVBRDF maps are highly correlated, so that it's not necessary to use four encoders. As for the decoders, we observe that two decoders are already satisfied and also save training time.

3.2. Loss function

We proposed to guess the diffuse map and use it as the ground truth for the diffuse map, called *guessed diffuse map*, as our method is unsupervised and we do not have the ground truth maps. We choose to guess the diffuse map ground truth, as it's the most obvious one among the four maps from an input image. Firstly, we compute the normalized input image $\mathbf{X}^*$, as in [AAL16]:

$$\mathbf{X}^* = \left(\mathbf{X} - \mathbf{X}^{\text{mean}}\right) / \left(\mathbf{X}^{\sigma} + 10^{-4}\right) \tag{4}$$

$$\mathbf{X}^{\text{mean}} = \text{blur}(\mathbf{X}) \tag{5}$$

$$\mathbf{X}^{\sigma} = \sqrt{\text{blur}\left((\mathbf{X} - \mathbf{X}^{\text{mean}})^2\right)} \tag{6}$$

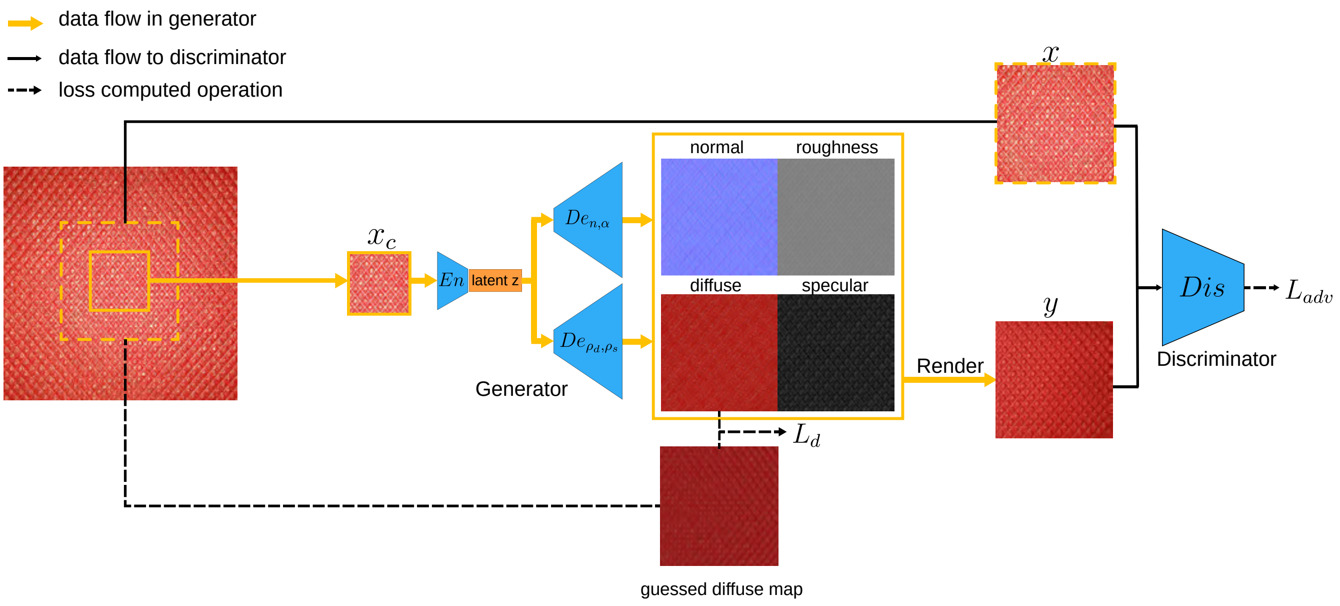

Figure 3: *Framework of our GAN architecture. When training, we randomly choose a candidate tile* **x**, *crop a tile* $\mathbf{x}_c$ *from the center of* **x** *and feed it into the generator. The generator generates the maps of the tile with two groups separately: normal and roughness, diffuse and specular. The discriminator distinguishes between the rendered image and the candidate tile. A guessed diffuse map (computed from* **x***) is used as guidance for the predicted diffuse map. The network is trained using a joint loss function of adversarial loss* L_{adv} *and L1 loss* L_d*.*

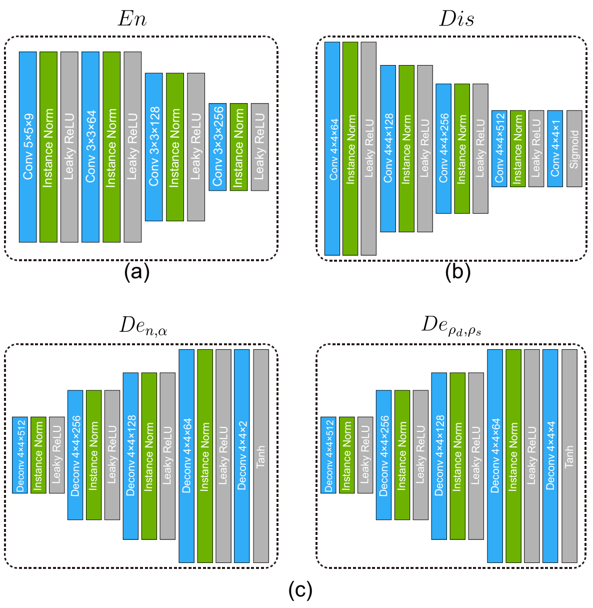

Figure 4: *Network architecture. (a) Encoder architecture. (b) Discriminator architecture. (c) Decoder architecture. Blue blocks stand for convolutional layers, green blocks stand for instance normalization, gray blocks stand for activation functions. Kernel size and number of feature channels are marked on convolutional layers.*

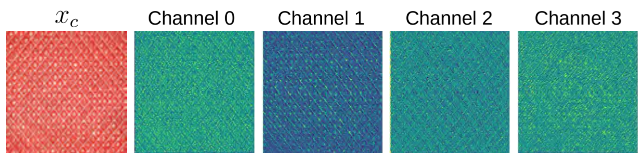

Figure 5: *Visualization of the first 4 layers latent vector from encoder. Even with untrained parameters, the encoder still can preserve texture information.*

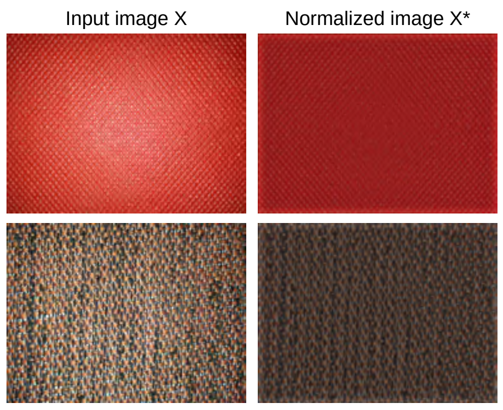

Figure 6: *Examples of normalized images. Normalization reduces the significant lighting variation.*

We subtract the $\mathbf{X}^{\text{mean}}$ (computed as the strongly Gaussian blurring) from $\mathbf{X}$ and point-wise divide by the standard deviation $\mathbf{X}^{\sigma}$. Figure 6 shows examples of normalized images.

Then we use $\mathbf{X}^*$ as the *guessed diffuse map* for diffuse maps:

$$\tilde{\rho_d} = \mathbf{X}^* \tag{7}$$

Based on the guessed diffuse maps, we proposed a joint loss function $\mathcal{L}_{final}$: L1 loss $\mathcal{L}_d(G)$ between generated diffuse map and the guessed diffuse map, and a weighted adversarial loss $\mathcal{L}_{GAN}(G,D)$:

$$\mathcal{L}_{final} = \lambda \mathcal{L}_{GAN}(G,D) + \mathcal{L}_d(G), \tag{8}$$

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}[\log Dis(\mathbf{x})] + \mathbb{E}[\log(1 - Dis(\mathbf{y})], \tag{9}$$

$$\mathcal{L}_d(G) = \mathbb{E}\left[||\tilde{\rho_\mathbf{d}} - \rho_\mathbf{d}||_1\right]. \tag{10}$$

Discussions: We use L1 distance rather than L2, because L1 loss produces less blurring results. We also try L1 loss between a re-rendering image $\mathbf{y}$ and an image tile $\mathbf{x}$:

$$\mathcal{L}_{render}(G) = \mathbb{E}\left[||\mathbf{x} - \mathbf{y}||_1\right]. \tag{11}$$

But the results are not satisfying. When there are highlights on the input image, the diffuse map is highly affected and overestimated, resulting in incorrect rendering under novel view (see Figure 16).

3.3. Training and Implementation

During training, in each iteration, we randomly crop a 256×256 candidate tile and its corresponding values as the ground truth. The center of the candidate tile 256×256 with size 128×128 is cropped and fed to the generator. The generated maps are rendered to a re-rendered image. Then re-rendered image and the candidate tile are fed into the discriminator. Among one iteration, we take 5 gradient descent steps on the generator $G_{n,\alpha}$, one step on the generator G_{ρ_d,ρ_s} and one step on the discriminator. We train the network for 20,000 iterations, with different randomly cropped tiles in each iteration.

We use the Adam optimization algorithm [KB14] with a fixed learning rate of 2e-5. Weights of the untrained encoder layers are initialized from a Gaussian distribution with mean 0 and standard deviation 0.02. The hyper-parameter λ is set to 0.1. It takes approximately 3 hours to train our network for one image using a TitanX GPU.

During predicting, our network is able to generate SVBRDF maps with $2\times$ resolution of the input image. To produce higher resolution SVBRDF maps, the neural network should be run iteratively. For instance, given a 512×512 input image, it requires to run our network three times to get a $4K \times 4K$ (See Figure 8). One advantage of our network is the trained model can be used for input image with an arbitrary size, so the second and following expansions of the image can use the trained model directly.

4. Results and discussion

4.1. Results

Recovered and synthesized SVBRDFs. Figures 7 shows a selection of representatives of input photographs, and our generated SVBRDF maps. The input images are from the 72-material iPhone 5 flash-no-flash dataset [AWL*15]. We discard the outer 37% of the 3264×2448 images, move the brightest part to image center and then resize the remaining part to 1632×1224 resolution. The size of the generated SVBRDF maps are 3264×2448. The results demonstrate that our method reproduces a rich set of reflectance effects for leathers, plastics, fabrics successfully. For both images with regular structures (fabric and plastic) and irregular structures (leather), our method can synthesize higher resolution SVBRDF maps without any artifacts. Comparing the zoom-in of the rendered images and the input image, the original patterns are well preserved. As expected, the SVBRDFs maintain consistent structure across the entire image, and rich in details.

Iterative expansion. Our model can expand an image to twice of its input size. By repeating the expansion, we can synthesize higher resolutions, until reaching the limitation of the GPU memory. This is useful when the input image has low resolution. We crop a 512×512 tile from the *fabric_yellow* image from the dataset [AWL*15] as the input image, and use it to train a model. After training, we feed in the image tile repeatedly and get 4096×4096 SVBRDF maps. Figure 8 shows the results of SVBRDF maps and renderings. The details and structures are preserved well in the iteratively synthesized SVBRDFs.

3D scenes rendering. To further validate our method, we used our synthesized maps to render a 3D scene (Figure 1). In this scene, maps for sofa, chair, and bucket are recovered from captured images. The white brick wall and the wood floor are recovered from rendered images. The scene is rendered with Arnold renderer[1]. Our generated high-resolution maps produce a seamless appearance.

4.2. Comparison with previous works

We compare our outputs with existing methods [DAD*18, GLD*19] using both rendered images and real world captured photos. The reference maps are from some free texture and material websites[2,3]. We notice that GANs aim at generating images look like the real images, but are not pixel-to-pixel corresponding. For better pixel-to-pixel comparison with reference maps in Figure 9, we replace the last transpose convolutional layer with a convolutional layer, to output SVBRDF maps with the same size as the input image. In Figure 10 and Figure 11 the reference maps of real world photos are not available, so we still use our original transposed convolutional layer.

Low-resolution SVBRDFs. Exiting works, e.g. [DAD*18], can only generate low-resolution SVBRDFs. To compare with them, we cropped a small image (256×256) and predict it's SVBRDF maps. We show results for both rendered

[1]https://www.arnoldrenderer.com/arnold/arnold-for-maya
[2]https://texturehaven.com
[3]https://freepbr.com

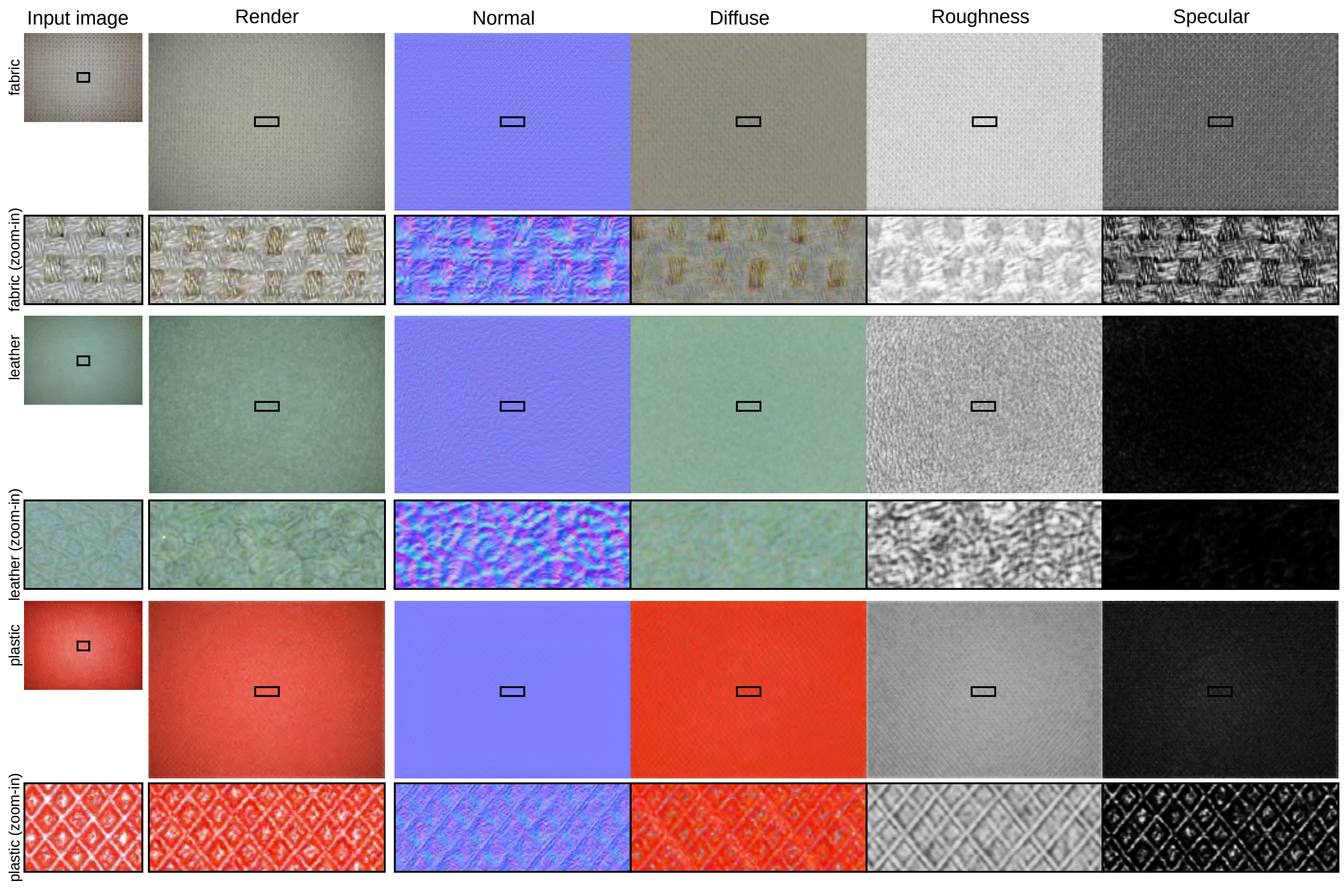

Figure 7: *Generated diffuse, specular, roughness and normal maps and rendering results with our method from several input images.*

images (Figure 9) and captured images (Figure 10). In Figure 9, our network is trained using a rendered 1024 × 1024 image with known view and light direction. For Gao et al. [GLD*19], we use the reference maps to render 20 images as inputs (N=20), and [DAD*18]'s outputs as initialization. The errors (mean square error, MSE) between the maps / rendered images and the references are shown under the images. For the re-rendered results, our method produces the best rendered result, while Gao et al. [GLD*19] is slightly different from the input image, and Deschaintre et al. [DAD*18] produces the least accurate result. For the recovered SVBRDFs, our method generates better diffuse and specular map, while the roughness map is less accurate. Comparing the novel view rendered results, although Gao et al. [GLD*19] has the lowest error, our result is more similar to the reference visually, while Gao et al. [GLD*19] produces blurry highlights, due to the inaccuracy of the specular map.

In Figure 10, we also compare SVBRDF maps using captured photos (from Aittala et al. [AWL*15]). We trained our model using a 1632 × 1224 input image, and cropped a 128 × 128 image for SVBRDF maps prediction. For Gao et al. [GLD*19], we use a single image as input (N=1), as images with more views and light directions are not available. Both Gao et al. [GLD*19] and our method generate similar results to the reference, while Deschaintre et al. [DAD*18] generates less accurate rendered result. Comparing our method and [GLD*19], our method generates more accurate specular. Although, in theory, from one input image, it's impossible to decide which map contributes more, it's more convincing that a leather should have some specular from real life observations. Thus, our SVBRDF maps are more plausible and result in a better novel view rendered image.

High-resolution SVBRDFs. Gao et al. [GLD*19] can recover high-resolution SVBRDFs from high-resolution input images. Figure 11 compares SVBRDF maps recovered from 1024 × 1024 captured images. We train our model using 1632 × 1224 input images, and crop a 512 × 512 image to generate 1024 × 1024 SVBRDF maps. For Gao et al. [GLD*19], we also use a single image as input (N=1), due to the availability of images with more views and light directions. We do not provide errors in this comparison, as there are no reference images, except the input image. Both methods generate a very close rendered image to the input image. Regarding the SVBRDFs, Gao et al. [GLD*19] has slightly artifacts at the center of the diffuse maps. This artifact is also visible in the novel view rendering result. Our method generates more plausible maps and novel view renderings.

Brute force SVBRDFs synthesis. Directly applying texture synthesis methods to low-resolution SVBRDF maps can also resulting

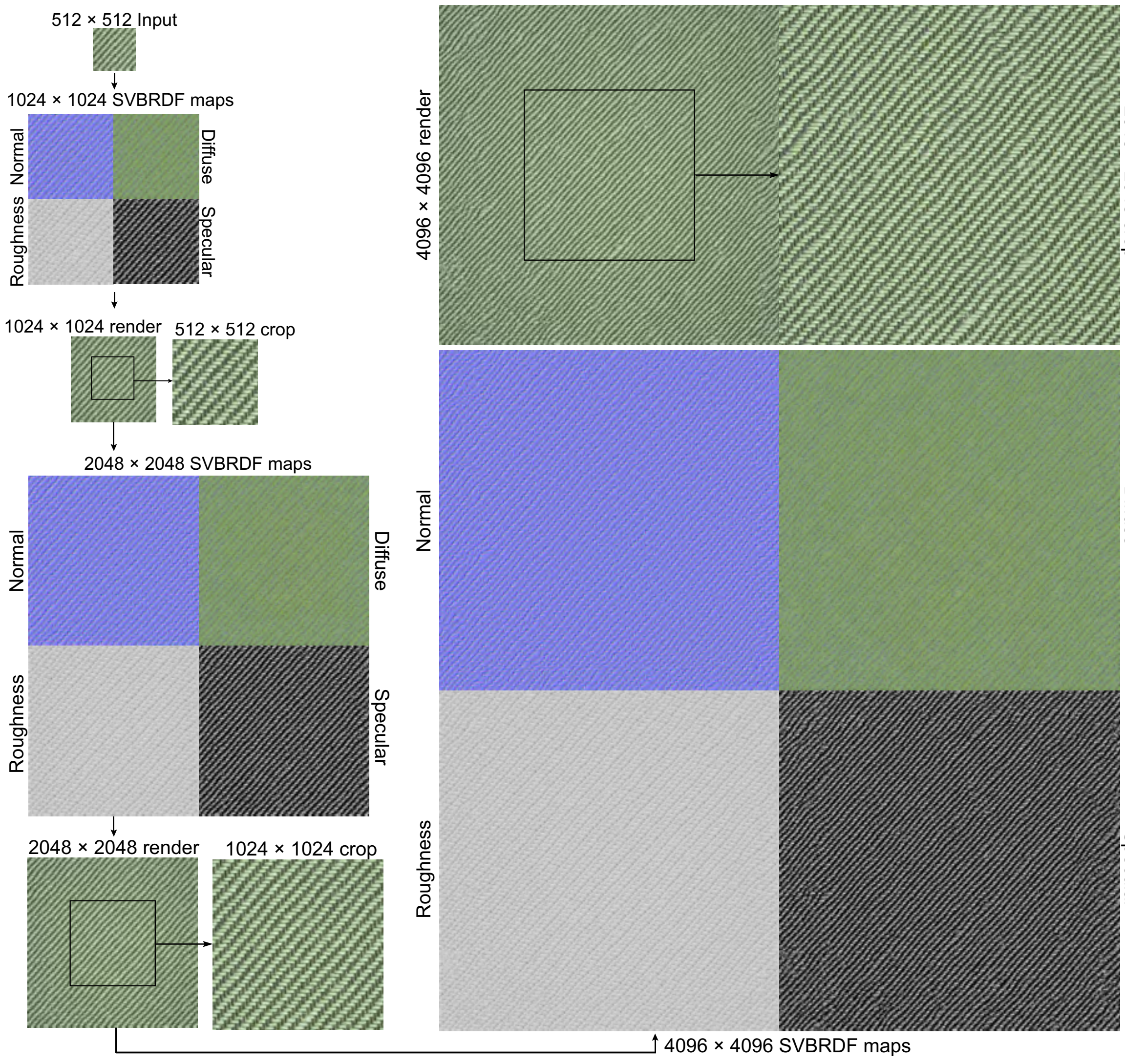

Figure 8: *Using our model three times to generate SVBRDFs with size* $4K \times 4K$ *from a* 512×512 *input image.*

in higher-resolution SVBRDFs, referred as Brute force SVBRDFs synthesis. Since the maps are synthesized respectively, there is no guarantee that the maps can preserve the same pattern as the input image. Figure 12 shows a comparison between our method and the brute force method. The errors are not shown, as the generated maps or rendered results used texture synthesis, so they are not able to match the references exactly. As expected, the synthesized diffuse map and normal map from the brute force method have misaligned patterns and strong inconsistency (see bottom row close-up view). In contrast, our method can preserve the consistency among the maps very well. Besides the quality, our method only needs to be trained once on the input image, and then it is ready to generate higher-resolution SVBRDFs. As for the brute force method, after recovering low-resolution SVBRDFs from the input image, it then requires training on all four maps and synthesizing higher-resolution SVBRDFs respectively.

We then compare our method with Aittala et al. [AAL16], which also targets at inputs with repetitive structures. Aittala et al. [AAL16] can only recover SVBRDFs and relies on other methods for synthesis. We used histogram preserved blending [HN18] for synthesis. There are several other texture synthesis methods, such as Gatys et al. [GEB15], and TileGAN [FAW19]. But they require a high-resolution target image or a lot of input images, so they are not suitable for this problem. In Figure 13, we compare our method with Aittala et al. [AAL16] on a low-resolution output (left image), and compare with Aittala et al. [AAL16] combined with Heitz and Neyret [HN18] on a high-resolution output. We used the same parameters (glossiness rather than roughness) and the same range for the parameters as Aittala et al. [AAL16]. Regarding the reflectance model, we used the Cook-Torrance reflectance model and they used the Blinn reflectance model. By comparison, for SVBRDF recovery only (left image), the diffuse map of [AAL16]

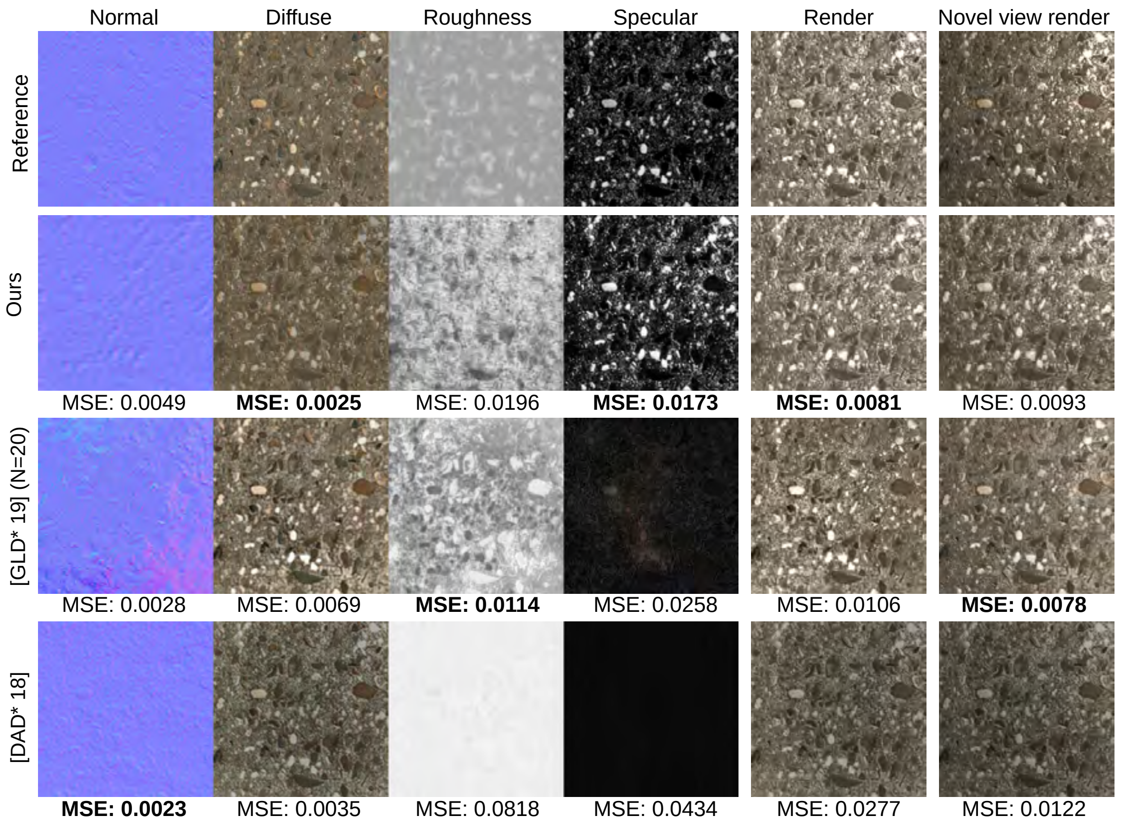

Figure 9: *SVBRDF maps of* 256×256 *from rendered images, compared with Deschaintre et al. [DAD* 18] and Gao et al. [GLD* 19]. The errors (MSE) under the images show the difference between the image and the reference. The lowest error is marked in blod.*

has obvious discontinuous artifacts, which are amplified in the synthesized diffuse map and rendered image (right image), while our method is able to preserve the structural feature.

4.3. Network Analysis

Using an untrained encoder. In Figure 14, we compare the generated SVBRDFs and re-rendered results of our encoder (untrained encoder) and the trained encoder. Both the maps and the rendered result are closer to the references. The network with a trained encoder has more parameters, thus it has stronger fitting capabilities. But in visual and quantitative comparisons, the network with an untrained encoder can produce results that are not much different from the previous approach, and the MSE of the diffuse map and the roughness map are even lower. Besides, our untrained encoder saves training time.

Effects of the two-stream generator. Figure 15 shows the generated SVBRDFs and rendered results of both our method (two-stream generator) and one stream generator. Our method generates much closer maps and the rendered results to the reference, while the one-stream generator suffers from artifacts in both the normal map and the rendered result.

Comparison of different loss functions. Figure 16 compares the outputs with different loss functions: our joint loss function ($L_{adv} + L_d$), a joint loss function with rendering loss and adversarial loss ($L_{adv} + L_{render}$) and adversarial loss only. We observe that the uneven light affects the maps when using $L_{adv} + L_{render}$ or just using L_{adv} only. The network trained using our designed loss function produces more correct maps (second row). The maps are not affected by the lighting in the input image, thus they produce better rendered result under novel light directions.

4.4. Limitation

Our method is able to synthesize SVBRDFs for input image with regular patterns, like the fabric (Figure 7), and also for input images with irregular patterns, like leather (Figure 7). But our method can not handle input images with global structures. In Figure 17, we generate the SVBRDFs and rendered results from an image without repetitive features. Our method fails to synthesize the images, and only increases the resolution. Our method relies on cropping a lot of small tiles from the input image to train the model. However, the tiles from images without repetitive features are not similar, and can not lead to successful training.

Another limitation of our method is that the synthesized output size is limited by the graphics card memory. Furthermore, the size of the input image should be at least 4 times the size of the tiles to make sure there are enough tiles used for training. This is to make

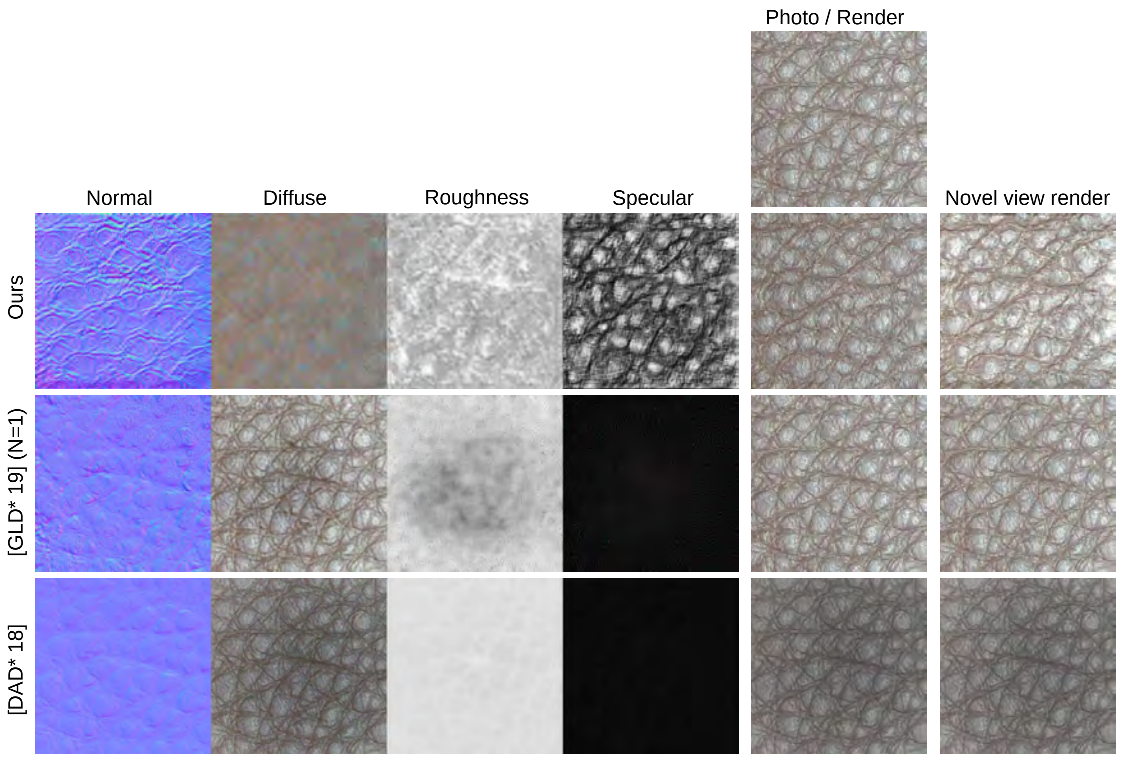

Figure 10: *SVBRDF maps of* 256×256 *from captured images, compared with Deschaintre et al. [DAD* 18] and Gao et al. [GLD* 19].*

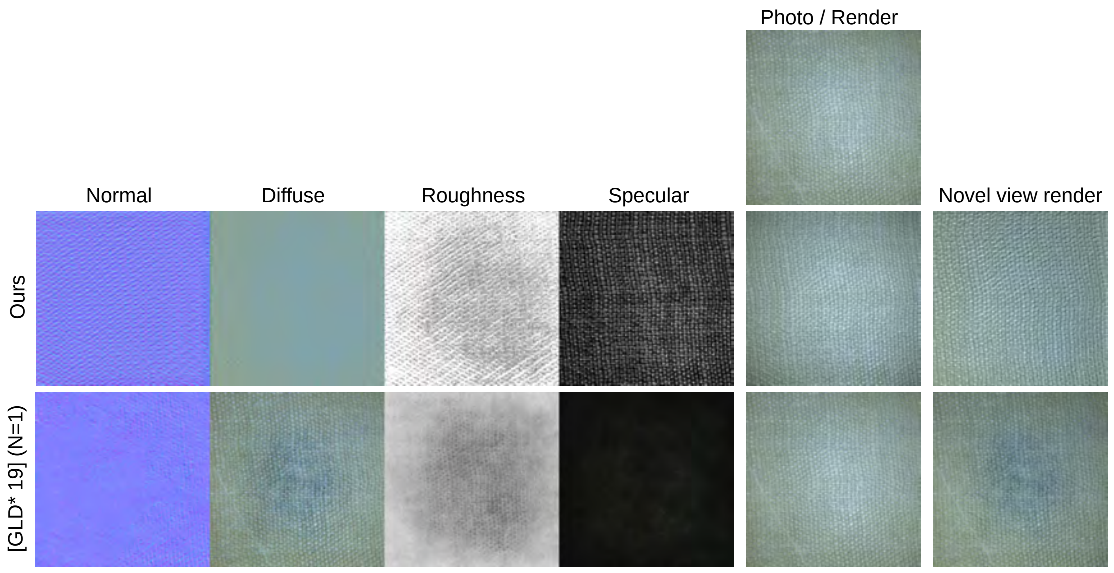

Figure 11: *SVBRDF maps of* 1024×1024 *from captured images, compared with Gao et al. [GLD* 19].*

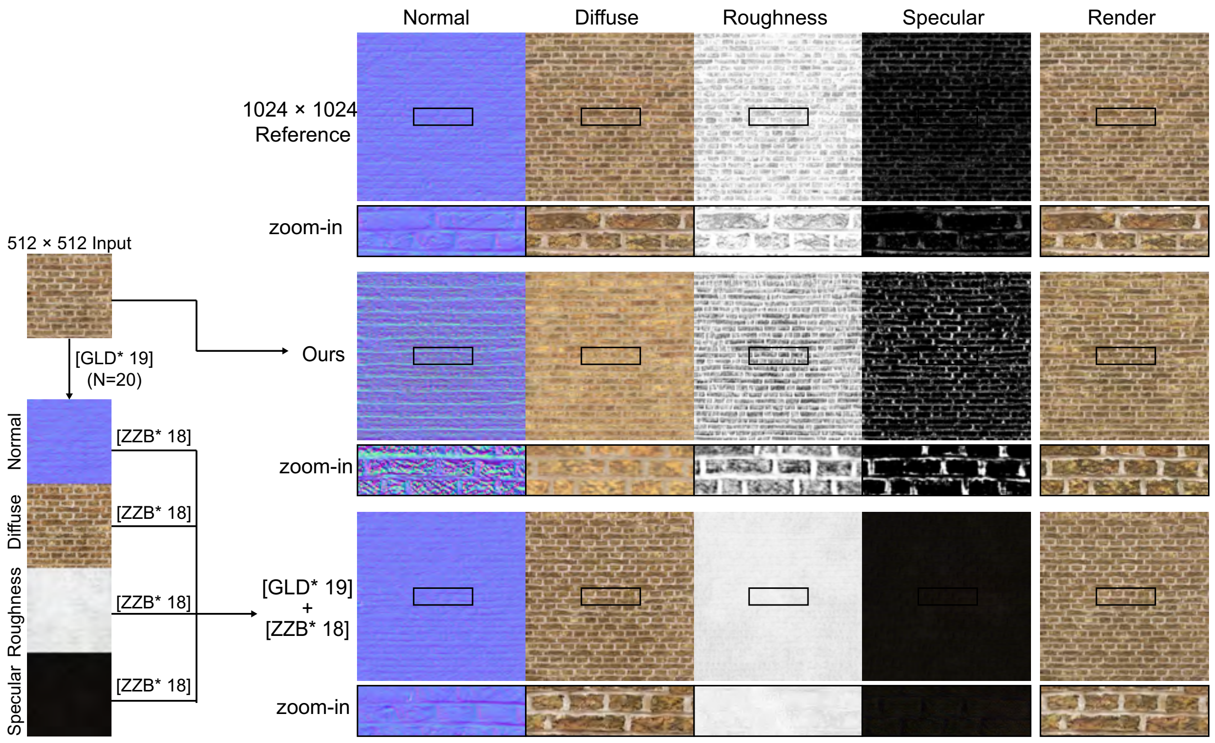

Figure 12: *Comparison between our method and Gao et al. [GLD* 19] with a texture synthesis method [ZZB* 18]. Recovering SVBRDF maps first and then synthesizing them separately leads to inconsistent high-resolution maps (bottom row). Note that the features in the normal map and the diffuse map are not aligned. Since the maps are synthesized separately, they have different brick patterns, while our result can preserve the consistency among the maps very well.*

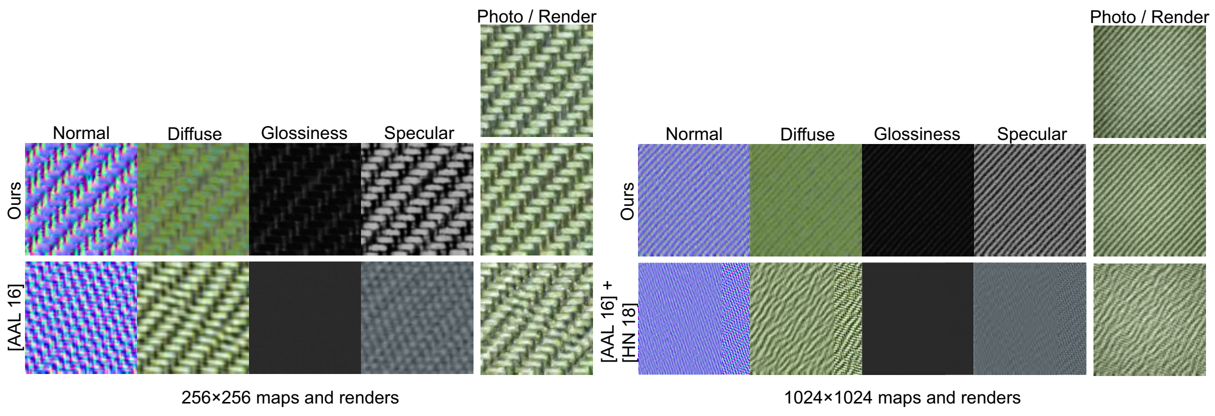

Figure 13: *Comparison between our method and Aittala et al. [AAL16] with a texture synthesis method [HN18]. Our method preserves the structural features very well, while their result has obvious artifacts.*

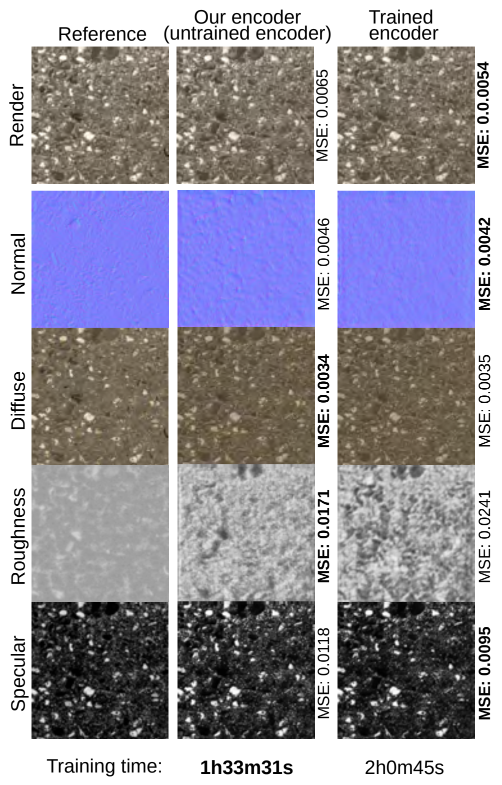

Figure 14: *Comparison of our encoder (untrained) and a trained encoder. The training time is shown under the images. The results of the two approaches are not much different from both visual and quantitative comparisons, and the untrained encoder saves training time.*

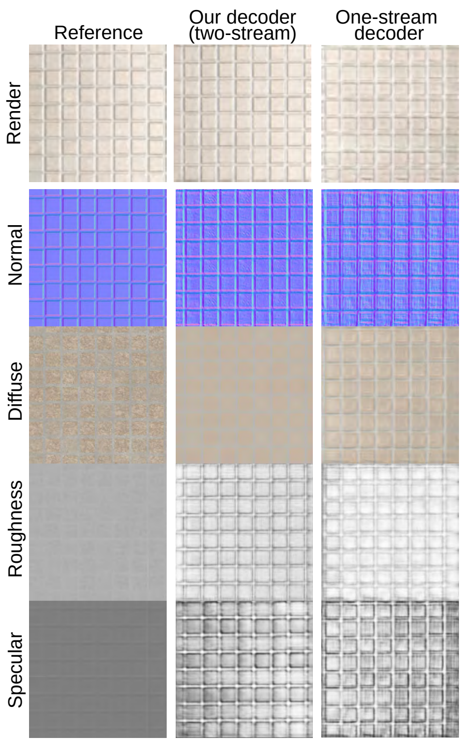

Figure 15: *Comparison of output SVBRDF maps using common encoder-decoder architecture generator (last column) and our two-stream generator (second column). Our two-stream generator removes lighting from diffuse map and generates more plausible normal map.*

sure that the randomly cropped tiles are different enough from each other, otherwise the GAN might fall into mode collapse or gradient vanishing.

Overall, our method is able to produce high quality rendered images and novel view rendered images. Regarding the SVBRDF maps, our method weakened the diffuse map and enhanced the specular map, due to the two-stream framework, compared to other methods (Deschaintre et al. [DAD*18] and Gao et al. [GLD*19]). This mechanic produces more reasonable maps for set of materials, like leathers, while overestimates the specular map for less shiny materials, like fabric. A deeper work on digging the material appearance in real life will be a possible solution for this issue.

5. Conclusion

We have presented an unsupervised generative adversarial network (GAN) for simultaneous recovery and synthesis of SVBRDF maps from a single image. We proposed a two-stream generator to enhance the specular maps and a novel joint loss function considering both adversarial loss and an L1 loss. Our model does not rely on a heavy dataset, as it is an unsupervised method. As far as we know, this is the first unsupervised GAN model for recovery and synthesis of SVBRDF maps. The model is trained using tiles cropped from the input image and then used for SVBRDF maps recovery and synthesis of the input image. The training time is acceptable. Our model can be used iteratively to synthesize extremely high-resolution images. In the end, our model is able to generate SVBRDFs and rendered images closer to the references, which are more plausible than the results from the state-of-the-art methods.

With a single input image, the problem is under-constrained and several maps could contribute to the observed features. Our separation between normal and roughness on one side and diffuse and specular albedo on the other tends to give more weight to the normal map, but also to the specular component. In future work, we want to introduce existing knowledge about the material, for example that leather is more specular, and fabric more diffuse.

Acknowledgments. We thank the reviewers for the valuable comments. This work has been partially supported by the National Key R&D Program of China under grant No. 2017YFB0203000, the National Natural Science Foundation of China under grant No. 61802187 and 61872223, the Natural Science Foundation of Jiangsu under grant No. BK20170857 and the fundamental research funds for the central universities No. 30920021133.

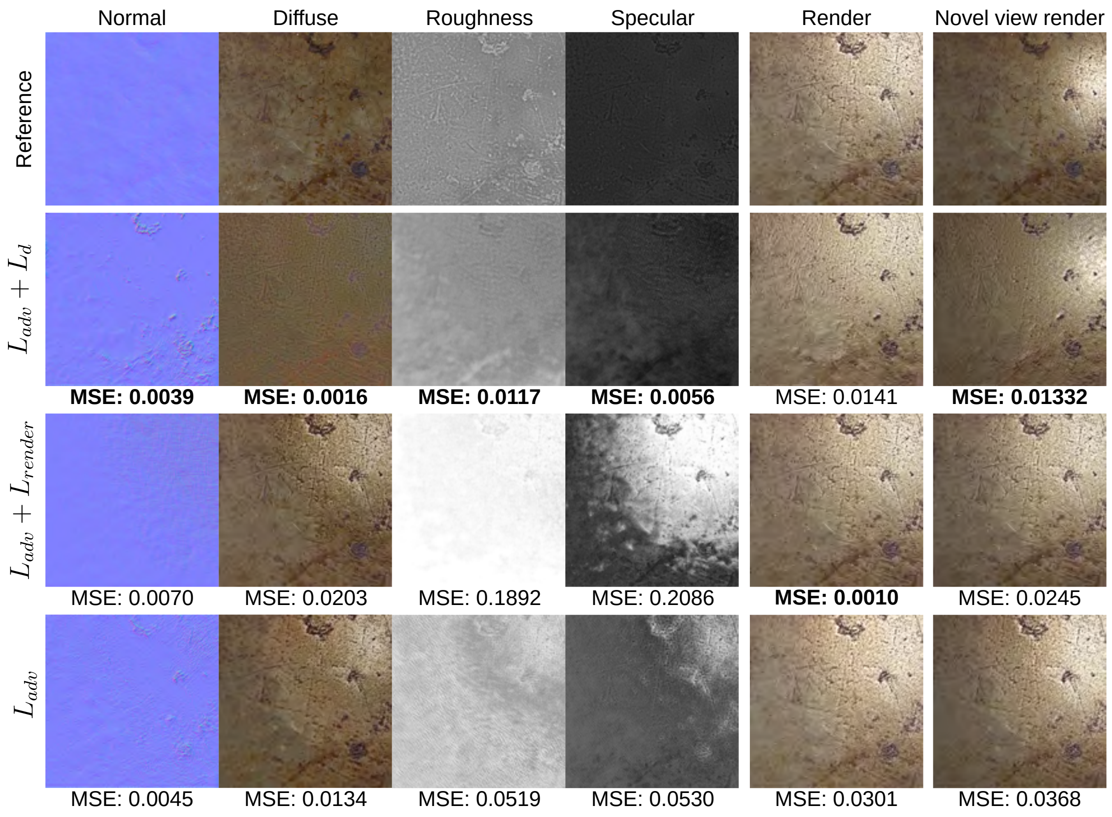

Figure 16: *Comparison of using different loss functions: our joint loss function ($L_{adv} + L_d$), a joint loss function with rendering loss and adversarial loss ($L_{adv} + L_{render}$) and adversarial loss only.*

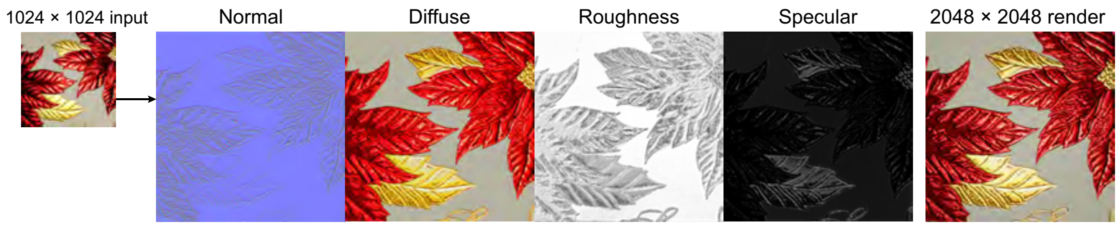

Figure 17: *Our method failed to synthesis textures when the input image has a global structure.*

References

[AAL16] Aittala M., Aila T., Lehtinen J.: Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 65. 2, 4, 8, 11

[AWL*15] Aittala M., Weyrich T., Lehtinen J., et al.: Two-shot svbrdf capture for stationary materials. *ACM Trans. Graph. 34*, 4 (2015), 110–1. 2, 6, 7

[AYD*18] Akl A., Yaacoub C., Donias M., Da Costa J.-P., Germain C.: A survey of exemplar-based texture synthesis methods. *Computer Vision & Image Understanding* (2018), S1077314218300523. 3

[BCW*17] Bao J., Chen D., Wen F., Li H., Hua G.: Cvae-gan: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2745–2754. 3

[BJK*20] Boss M., Jampani V., Kim K., Lensch H. P., Kautz J.: Two-shot spatially-varying brdf and shape estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020). 3

[Cha14] Chandraker M.: On shape and material recovery from motion. In *European Conference on Computer Vision* (2014), Springer, pp. 202–217. 2

[CSHD03] Cohen M. F., Shade J., Hiller S., Deussen O.: Wang tiles for image and texture generation. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 287–294. 3

[CT82] Cook R. L., Torrance K. E.: A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG) 1*, 1 (1982), 7–24. 4

[DAD*18] Deschaintre V., Aittala M., Durand F., Drettakis G., Bousseau A.: Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 128. 2, 3, 4, 6, 7, 9, 10, 12

[DAD*19] Deschaintre V., Aittala M., Durand F., Drettakis G., Bousseau A.: Flexible svbrdf capture with a multi-image deep network. *Computer Graphics Forum 38*, 4 (2019). 3

[DCP*14] Dong Y., Chen G., Peers P., Zhang J., Tong X.: Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Transactions on Graphics (TOG) 33*, 6 (2014), 193. 2

[EF01] Efros A. A., Freeman W. T.: Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001* (August 2001), 341–346. 3

[EL99] Efros A. A., Leung T. K.: Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision* (Corfu, Greece, September 1999), pp. 1033–1038. 3

[FAW19] Frühstück A., Alhashim I., Wonka P.: Tilegan: synthesis of large-scale non-homogeneous textures. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–11. 3, 8

[GEB15] Gatys L., Ecker A. S., Bethge M.: Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems* (2015), pp. 262–270. 3, 8

[GGG*16] Guarnera D., Guarnera G., Ghosh A., Denk C., Glencross M.: Brdf representation and acquisition. *Computer Graphics Forum 35*, 2 (2016), 625–650. 2

[GLD*19] Gao D., Li X., Dong Y., Peers P., Xu K., Tong X.: Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 134. 2, 3, 6, 7, 9, 10, 11, 12

[GPAM*14] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y.: Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680. 3

[HDR19] Hu Y., Dorsey J., Rushmeier H.: A novel framework for inverse procedural texture modeling. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–14. 3

[HN18] Heitz E., Neyret F.: High-performance by-example noise using a histogram-preserving blending operator. *Proc. ACM Comput. Graph. Interact. Tech. 1*, 2 (2018), 31:1–31:25. 3, 8, 11

[HS15] Hui Z., Sankaranarayanan A. C.: A dictionary-based approach for estimating shape and spatially-varying reflectance. In *2015 IEEE International Conference on Computational Photography (ICCP)* (2015), IEEE, pp. 1–9. 2

[HSL*17] Hui Z., Sunkavalli K., Lee J.-Y., Hadap S., Wang J., Sankaranarayanan A. C.: Reflectance capture using univariate sampling of brdfs. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 5362–5370. 2

[IZZE17] Isola P., Zhu J.-Y., Zhou T., Efros A. A.: Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1125–1134. 3, 4

[JBV16] Jetchev N., Bergmann U., Vollgraf R.: Texture synthesis with spatial generative adversarial networks. *CoRR abs/1611.08207* (2016). 3

[KB14] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 6

[LDPT17] Li X., Dong Y., Peers P., Tong X.: Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 45. 2, 3, 4

[LH05] Lefebvre S., Hoppe H.: Parallel controllable texture synthesis. In *ACM SIGGRAPH 2005 Papers*. 2005, pp. 777–786. 3

[LSC18] Li Z., Sunkavalli K., Chandraker M.: Materials for masses: Svbrdf acquisition with a single mobile phone image. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 72–87. 2, 3, 4

[LXR*18] Li Z., Xu Z., Ramamoorthi R., Sunkavalli K., Chandraker M.: Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–11. 2

[RDDM18] Raad L., Davy A., Desolneux A., Morel J.-M.: A survey of exemplar-based texture synthesis. *Annals of Mathematical Sciences and Applications 3*, 1 (2018), 89–148. 3

[RPG16] Riviere J., Peers P., Ghosh A.: Mobile surface reflectometry. *Computer Graphics Forum 35*, 1 (2016), 191–202. 2

[Wan61] Wang H.: Proving theorems by pattern recognition - ii. *The Bell System Technical Journal 40*, 1 (1961), 1–41. 3

[WL00] Wei L.-Y., Levoy M.: Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 479–488. 3

[WLKT09] Wei L.-Y., Lefebvre S., Kwatra V., Turk G.: State of the art in example-based texture synthesis. 3

[XDPT16] Xia R., Dong Y., Peers P., Tong X.: Recovering shape and spatially-varying surface reflectance under unknown illumination. *ACM Trans. Graph. 35*, 6 (Nov. 2016). 2

[XNY*16] Xu Z., Nielsen J. B., Yu J., Jensen H. W., Ramamoorthi R.: Minimal brdf sampling for two-shot near-field reflectance acquisition. *ACM Trans. Graph. 35*, 6 (Nov. 2016). 2

[YNBH11] Yu Q., Neyret F., Bruneton E., Holzschuch N.: Lagrangian texture advection: Preserving both spectrum and velocity field. *IEEE Transactions on Visualization and Computer Graphics 17*, 11 (2011), 1612–1623. 3

[ZZB*18] Zhou Y., Zhu Z., Bai X., Lischinski D., Cohen-Or D., Huang H.: Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–13. 3, 11