TileGen: Tileable, Controllable Material Generation and Capture

Xilong Zhou
Texas A&M University
College Station, USA

Miloš Hašan
Adobe Research
San Jose, USA

Valentin Deschaintre
Adobe Research
London, UK

Paul Guerrero
Adobe Research
London, UK

Kalyan Sunkavalli
Adobe Research
San Jose, USA

Nima Khademi Kalantari
Texas A&M University
College Station, USA

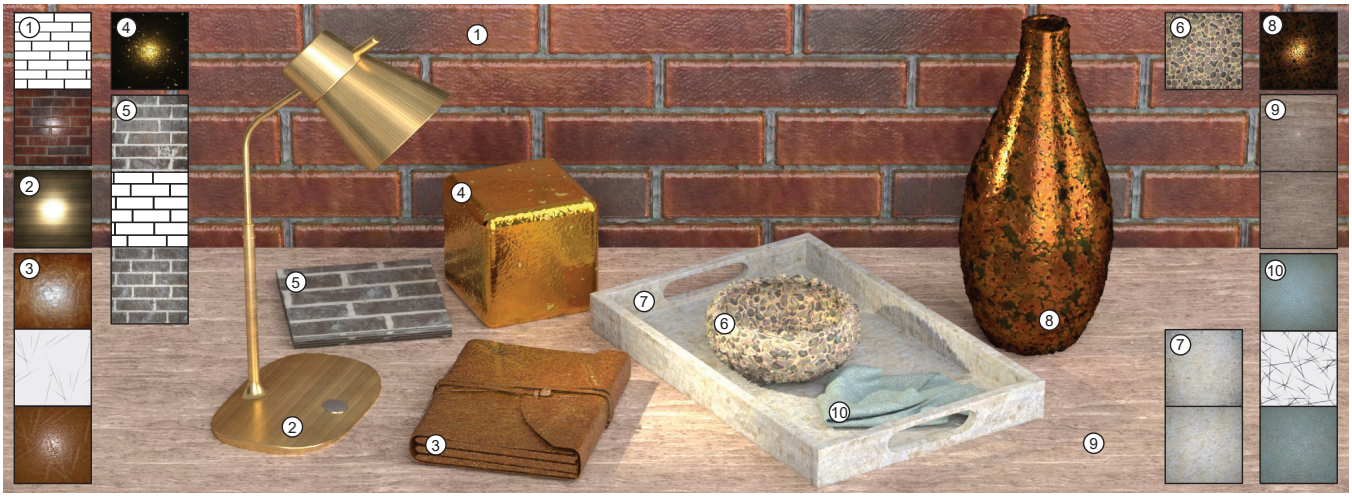

Figure 1: We present a category-specific generative model for spatially-varying materials, whose results are seamlessly tileable and optionally conditioned on patterns controlling the material structure. Our model can be used to generate new materials, or inverted to find materials matching target photographs by optimization. Here we show examples of tile, leather, stone and metal classes, either directly generated (1, 2, 4, 6, 8) or reconstructed from photographs (3, 5, 7, 9, 10) and, in some cases, conditioned on an input structure pattern (1, 3, 5, 10). The insets show, where applicable, the target photograph, condition pattern and a rendering of the synthesized material. The generated maps include height fields that are displacement-mapped in the rendered scene.

ABSTRACT

Recent methods (e.g. MaterialGAN) have used unconditional GANs to generate per-pixel material maps, or as a prior to reconstruct materials from input photographs. These models can generate varied random material appearance, but do not have any mechanism to constrain the generated material to a specific category or to control the coarse structure of the generated material, such as the exact brick layout on a brick wall. Furthermore, materials reconstructed from a single input photo commonly have artifacts and are generally not tileable, which limits their use in practical content creation pipelines. We propose TileGen, a generative model for SVBRDFs that is specific to a material category, always tileable, and optionally *conditional* on a provided input structure pattern. TileGen is a variant of StyleGAN whose architecture is modified to always produce tileable (periodic) material maps. In addition to the standard "style" latent code, TileGen can optionally take a condition image, giving a user direct control over the dominant spatial (and optionally color) features of the material. For example, in brick materials, the user can specify a brick layout and the brick color, or in leather materials, the locations of wrinkles and folds. Our inverse rendering approach can find a material perceptually matching a single target photograph by optimization. This reconstruction can also be conditional on a user-provided pattern. The resulting materials are tileable, can be larger than the target image, and are editable by varying the condition.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SA '22 Conference Papers, December 6–9, 2022, Daegu, Republic of Korea
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9470-3/22/12.
https://doi.org/10.1145/3550469.3555403

CCS CONCEPTS

• **Computing methodologies** → **Rendering**.

KEYWORDS

SVBRDF, generative adversarial model

ACM Reference Format:
Xilong Zhou, Miloš Hašan, Valentin Deschaintre, Paul Guerrero, Kalyan Sunkavalli, and Nima Khademi Kalantari. 2022. TileGen: Tileable, Controllable Material Generation and Capture. In *SIGGRAPH Asia 2022 Conference Papers (SA '22 Conference Papers), December 6–9, 2022, Daegu, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3550469.3555403

1 INTRODUCTION

High-quality materials are critical to realism in computer graphics applications. While reflectance models and rendering algorithms have reached excellent fidelity over the last decade, authoring photorealistic materials is still a time-consuming process that requires the construction and editing of complex procedural node graphs, and/or skilled manipulation of textures captured from photographs.

Given the rapid progress in generative adversarial networks (GANs) [Karras et al. 2021, 2018, 2019], such models are a natural approach for material generation. As an example, MaterialGAN [Guo et al. 2020b] is a recently proposed generative SVBRDF model trained on a large example dataset of synthetic material maps. MaterialGAN learns a latent space that can be sampled to produce high-quality materials, or leveraged as an optimization prior for SVBRDF capture from photographs. However, MaterialGAN is not tileable and does not have any mechanism for controlling the generated materials. If a user is interested in changing the layout of a synthesized brick texture, there is no simple way to do so other than by repeatedly generating a random material until the desired layout is found by chance. Additionally, the results may have artifacts such as specular highlights leaking into the estimated maps.

In this work, we propose TileGen, a category-specific generative model for tileable SVBRDFs. Unlike previous methods, TileGen (i) always produces *tileable output*, (ii) enables more *direct control* over the generated materials by optionally conditioning on input structure patterns that specify where the dominant features of the material should be located, and (iii) allows plausible reconstruction of structured materials from a *single photograph*, due to the additional regularization provided by the input condition.

TileGen is a variant of StyleGAN2 [Karras et al. 2019] whose architecture is modified to always output tileable material maps. It optionally uses features extracted from a conditional input pattern in addition to the standard "style" latent. Fixing the conditional pattern while varying the style code allows generating materials with the same dominant features, but different styles. We demonstrate that our conditional method can control the structure of complex materials like tiles, or the locations of wrinkles in leather.

Based on TileGen, we propose an inverse rendering approach to find a material matching the style of a *single* target photograph taken with flash. For conditional materials, our method enables the user to provide an approximate pattern corresponding to the target photo; the pattern need not be precisely aligned with the target image. Unlike previous per-pixel approaches, the resulting materials are tileable, can be larger than the target image, and are controllable by varying the condition pattern. These additional benefits from TileGen could enable artists to easily generate and fit a wide variety of material variations to reference photographs and will become integral to practical material authoring workflows.

2 RELATED WORK

Material acquisition. Recovering material properties from images is a long-standing challenge in graphics [Dong 2019; Guarnera et al. 2016]. Some approaches propose to use specialized hardware to capture material properties [Dong et al. 2010; Kang et al. 2018] but the practice of these methods are limited by the bulky and expensive setup. Deep learning demonstrated remarkable progress in the quality of SVBRDF estimates from a single image (usually captured under flash illumination using phone) [Deschaintre et al. 2018; Li et al. 2017, 2018]. These approaches produce smooth, plausible maps, but still suffer from artifacts caused by over-exposure burn-in, despite recent architecture modifications [Guo et al. 2021] or GAN-based, mixed data augmentation [Zhou and Kalantari 2021]. Additionally, a single input is often too under-constrained, leading to inaccurate specular material properties. Recently, Henzler et al. [2021] proposed to leverage a stationarity prior to capture SVBRDFs from a single flash photograph, reducing burn-in artifacts but limiting the class of supported materials. To improve quality, few-images methods were proposed through direct inference [Deschaintre et al. 2019; Ye et al. 2021] or deep optimization [Gao et al. 2019; Guo et al. 2020b]. Specifically, MaterialGAN [Guo et al. 2020b] proposes to train an unconditional generative model for materials, allowing to optimize in the generator latent space and input noise to match the appearance of a target given a few target pictures. However, this model entangles spatial layout and style, is not tileable, and gives no explicit control over the generated materials.

As opposed to most of these approaches, we do not target exact per-pixel capture, but rather a result with matching perceptual style, guided by the photo. Our approach enables guided recovery of tileable and meaningfully controllable materials from a single picture and an optional structural pattern, through category-specific conditional GANs.

Several works explicitly explored tileability for texture synthesis [Bergmann et al. 2017] or completing textures into tiles through feature repetition and tile search [Rodriguez-Pardo and Garces 2022b]. Our approach is to instead enforce tileability through our network architecture; we do not attempt to complete non-tileable textures into tileable ones. Instead, we define a space of plausible tileable materials, which we can randomly sample, or project target photographs into it.

Guided material generation and acquisition. To provide control in the material acquisition and generation process, recent work proposed to leverage different guides. Using one [Deschaintre et al. 2020] or multiple images [Rodriguez-Pardo and Garces 2022a] alongside a small material exemplar, previous work proposes to transfer properties to large scale photograph of similar materials. Guehl et al. [2020] define a procedural generator for realistic textural structure masks, which they use to propagate existing material properties. Leveraging this procedural structure generator, Hu et al. [2022] recently proposed to generate an entire procedural material graph using a segmentation mask and material as input. As opposed to our approach, these methods rely on pre-existing material inputs; they do not target material generation or capture.

Procedural material representations in the form of node graphs [Adobe 2021], represent an artist-defined set of material appearances. Hu et al. [2019] and Shi et al. [2020] propose to optimize the

parameters of existing material graphs to match the appearance of a target photo. Our method is related to procedural materials in that it establishes a function from an input pattern (a generator in the case of procedural materials) to an SVBRDF. However, it is more general than these approaches, as it trains generators for broad material classes, where training data can come from any source. TileGen could be considered a class-level, neural procedural representation, capable of generating a large variety of samples, with conditions on the local structures and style.

Generative models. Our method is based on the StyleGAN [Karras et al. 2018, 2019] architecture which has demonstrated compelling image synthesis results on well-defined domains such as faces, landscapes, or cars. While recent approaches [Anokhin et al. 2020] propose independent per-pixel generation through a unique latent vector and positional encodings, most works leverage spatial convolutions. Isola et al. [2017] proposed an image-to-image translation model that conditions generation on an input image. More recently, different methods such as SPADE [Park et al. 2019] and CollageGAN [Li et al. 2021] proposed to condition GAN generation with semantic masks. We build on this work, conditioning the generation of materials with structure through a binary or color mask. Recently, Karras et al. [2021] highlighted the importance of careful signal processing and translation invariance in GANs to prevent the network from building priors based on absolute position in the image. We prevent dependence on absolute positions by our tileable architecture modifications and by random translations of our dataset and generated materials.

3 TILEABLE AND CONDITIONAL GAN

In this section, we describe the architecture of TileGen, as well as details about training techniques and training data.

3.1 Tileable architecture

TileGen is based on StyleGAN2 [Karras et al. 2019]. Different from StyleGAN2, our architecture is designed to preserve tilability, a property that is important for practically usable textures. This is achieved by replacing all convolution and transposed convolution operations in the model by wrap-around versions, agnostic to their location in the image or to the boundary location (please refer to supplementary material for more details). Therefore, these operations always produce smoothly tileable results if the input condition is tileable, which is always true in our results.

We observe that in our modified architecture, intermediate feature maps at all levels remain periodic (shown in Fig. 2). Furthermore, we found that this architecture can even be used to produce tileable materials from a non-tileable dataset (though the models shown in the paper use tileable datasets).

3.2 Conditional architecture

CollageGAN [Li et al. 2021] is a recent conditional GAN approach for generating realistic images given segmentation maps. Though we do not currently collage multiple GAN models, we use a conditional architecture (shown in Fig. 2) inspired by CollageGAN.

The main idea of the CollageGAN architecture is that the conditional pattern $\boldsymbol{p}$ is encoded into features $\boldsymbol{\phi}$, which feed into the initial layer of the architecture at size 32×32, replacing the learned

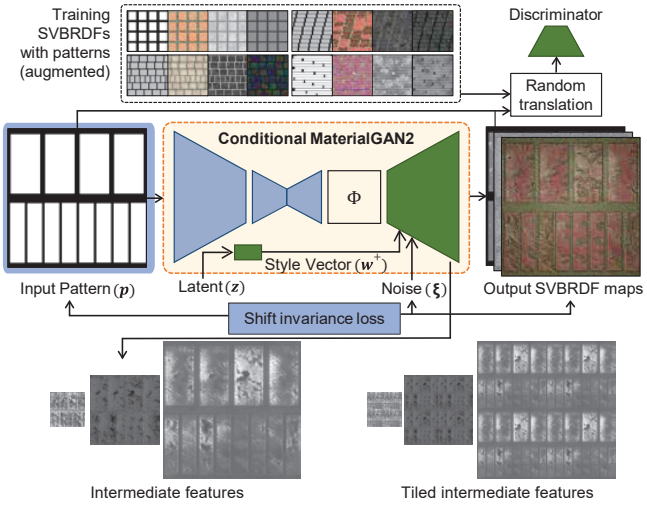

Figure 2: The conditional version of TileGen is trained on a dataset of SVBRDF parameter maps with corresponding condition patterns. Our conditional generator has a CollageGAN-like encoder that maps the input pattern p into features Φ at the start of the StyleGAN2-based decoder (green); the decoder also receives the latent code z (via a mapping network) and noise. The encoder and decoder have been modified to only use tileable operations. The resulting SVBRDF maps, together with the condition, are randomly translated and fed to a StyleGAN2 discriminator. *Differences between conditional model and unconditional model are shown in light blue. In unconditional model, we do not have input patterns and encoder-decoder. See Sec. 3 for more details.*

initial constant of StyleGAN2. This approach is also used in our architecture; however, a major difference from CollageGAN is that our latent vector $\boldsymbol{z}$ is generated at random from the normal distribution, instead of depending on $\boldsymbol{p}$. We therefore do not apply KL-divergence loss to regularize the latent vector either. This provides a certain level of disentanglement between the pattern $\boldsymbol{p}$ and the "style" of the material, given by the latent vector $\boldsymbol{z}$. This property is important for our inverse rendering, and does not exist in CollageGAN, where the result depends on the condition $\boldsymbol{p}$.

Our trained generator can be written as a function $\mathcal{G}$ that generates the material parameter maps (in our case, 5 channels for non-metallic materials: diffuse albedo $\boldsymbol{a}$, height $\boldsymbol{h}$, roughness $\boldsymbol{r}$), given the latent code $\boldsymbol{z}$ and pattern $\boldsymbol{p}$. For our metal material class, the generator also outputs one extra channel: the metallic amount $\mathbf{m}$. It could be extended to output different number of channels driving different shading models.

3.3 Synthetic dataset design

We prepare class-specific datasets based on Substance Source; each Substance graph created by artists is tagged as the corresponding material class (like stone, leather, etc.). Using these tags, we create datasets for four categories (leather, tile, stone, and metal). One reason that we choose class-specific training is that it enables us to define conditional patterns per class. We trained four TileGen models using prepared four synthetic datasets: unconditional stone and metal models, and conditional tile/brick and leather models.

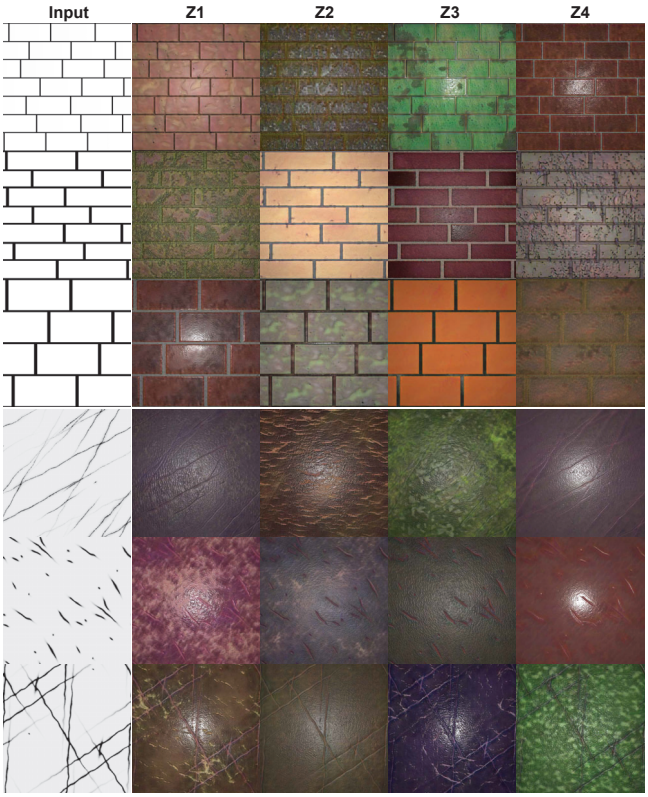

Figure 3: Randomly sampled conditional materials from the tile and leather classes. We feed the conditional pattern on the left to TileGen, along with four different random latent vectors $z_1, \ldots, z_4$, to produce corresponding material instances. The results have varied appearance with a constant layout condition. The texture maps are shown in supplementary materials.

For a given class, we collected a number of Adobe Substance node graphs [Adobe 2021]. We sampled their parameters in reasonable ranges, and saved the output material maps: diffuse albedo $\boldsymbol{a}$, height (displacement) $\boldsymbol{h}$, roughness $\boldsymbol{r}$, and metallicity $\mathbf{m}$ for metals (though other material categories such as fabrics or fur could output other maps). We further augment the training examples, by applying augmentation to the height and roughness channels. We apply small color augmentation to remain within realistic color tones for a given material class.

Pairing each synthetic material with its conditional pattern is dependent on the specific material category. For the tile/brick class, we identified a node in the corresponding graph whose output approximately matches our desired pattern, having a high value denoting a tile and a low value denoting the gap between tiles. For the leather class, we similarly identified a node whose output closely matches the final wrinkle pattern. We either threshold these outputs to make them binary for tile or keep these outputs as grayscale for leather, yielding the pattern $\boldsymbol{p}$, which we save as an additional texture output. We kept the stone and metal categories unconditional, as there is no obvious structure shared by them (though specific subsets of stones or metals could have structure and we could make specific models for them). Even though we focus on four material categories in the paper, our approach is flexible and it is possible to prepare any dataset, e.g. mixing materials which share similar types of conditions. Please refer to supplementary materials for more implementation details and dataset examples.

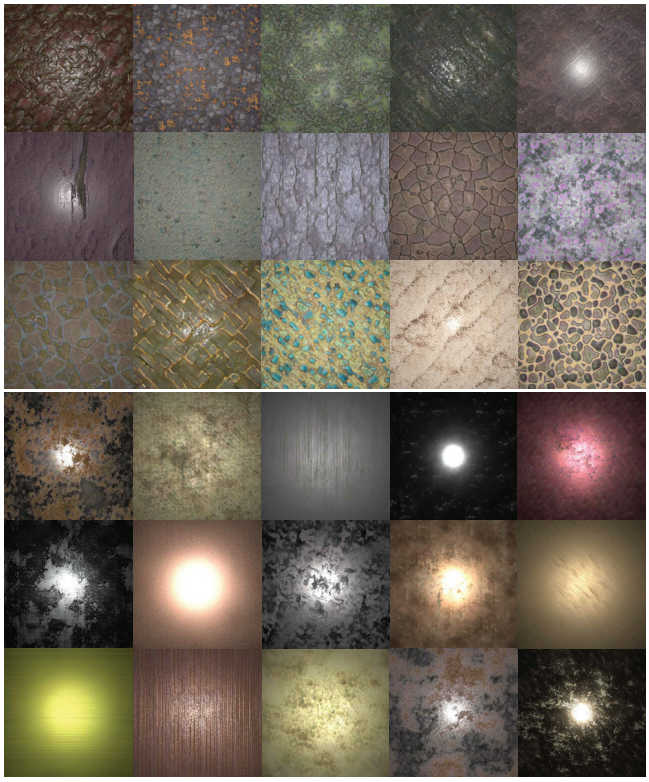

Figure 4: Randomly sampled unconditional materials from the stone and metal classes, showing the diversity of the results within each class. The corresponding texture maps are shown in supplementary materials.

3.4 Losses

We train our networks with an adversarial loss L_{adv}. We follow the StyleGAN2 [2019] discriminator architecture and regularization, except with more channels provided to the discriminator, representing our parameter texture maps.

For the conditional models, we also feed the condition pattern to the discriminator and randomly translate the pattern and maps (both ground-truth and generated) before feeding them. When training conditional models with only the adversarial loss above, we find that the model bakes in some material structure variations into the latent code $\boldsymbol{z}$ (i.e., varying $\boldsymbol{z}$ also changes the dominant structures in the generated material). However, we want the input pattern to drive the material structure; we introduce a second loss L_{shift} ensuring that shifting the input pattern also shifts the generated material maps by the same amount. We run the pattern $\boldsymbol{p}$ through the generator $\mathcal{G}$ twice: the second time with a random translation (shift) T, maintaining the same latent vector $\boldsymbol{z}$. A further detail is that we also shift the random noise (used in StyleGAN2 as additional inputs to the generator) by the same amount. Finally, the shift loss is defined as: $L_{\text{shift}} = \|T(\mathcal{G}(\boldsymbol{p}, \boldsymbol{z}, \boldsymbol{\xi})) - \mathcal{G}(T(\boldsymbol{p}), \boldsymbol{z}, T(\boldsymbol{\xi}))\|_1$.

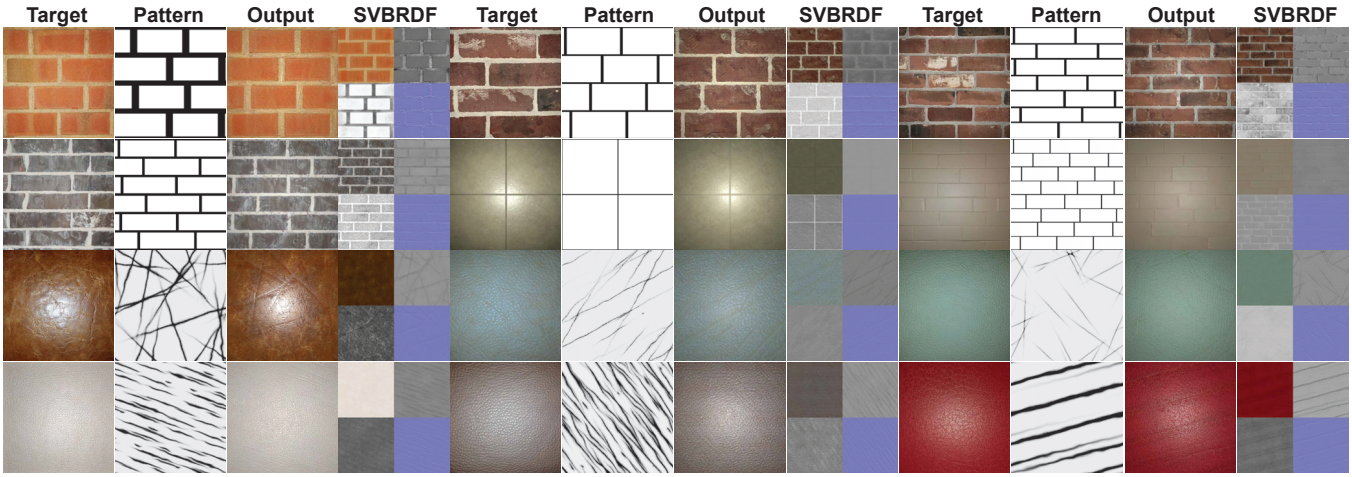

Figure 5: Examples of reconstructing conditional materials from a *single* target photograph with flash illumination for tiles (two top rows) and leather (two bottom rows). For each materials class, the target image is on the left, followed by the input pattern created by the user. For tiles we assume the user pattern has approximately matching feature sizes to the photo, though the feature layout can be arbitrary as we do not seek per-pixel correspondence. For leathers we assume the patterns represents wrinkles, though these wrinkles may not exist in the target images. The third column shows our resulting rendered material, followed by the predicted parameter maps (top-left: diffuse map, top-right:height map, bottom-left: roughness map, bottom-right:normal map).

4 MATCHING TARGET IMAGES

In this section, we use TileGen to solve the inverse problem of finding an SVBRDF that, when rendered, matches the appearance of a target photograph of a physical material sample.

We assume the input image is a single photograph of a planar sample taken with a cell phone camera with flash. We use a differentiable rendering operator $\mathcal{R}$ that takes as input the parameter maps, and synthesizes corresponding images of the material lit by the flash illumination. We turn the height map into a normal map using central finite differences, and shade the diffuse component using a Lambertian term, and the specular component using a standard microfacet BRDF with the GGX normal distribution [Walter et al. 2007]. This could be easily extended to other lighting/shading setups or to multiple inputs, as these are independent of TileGen and are just modifications to the rendering operator.

Given a target image $\boldsymbol{I}$, we would like to find material maps that render to an image similar to $\boldsymbol{I}$, under a suitable loss $\mathcal{L}$. More specifically, we define the vector $\boldsymbol{u}$ to be the pair of style vector $\boldsymbol{w}^+$ and noise vector $\boldsymbol{\xi}$ in the $\mathcal{W}^+\mathcal{N}$ space (illustrated in Fig. 2), instead of the original random code $\boldsymbol{z}$, as detailed by Guo et al. [2020b]. The optimization problem then becomes:

$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \mathcal{L}(\mathcal{R}(\mathcal{G}(\boldsymbol{u}, \boldsymbol{p})), \boldsymbol{I}), \tag{1}$$

where $\mathcal{G}$ is the learned conditional TileGen generator. (For unconditional generators, we drop the $\boldsymbol{p}$ dependence.) Given that both $\mathcal{G}$ and $\mathcal{R}$ are differentiable operations, Eq. (1) can be optimized via gradient-based methods to estimate $\boldsymbol{u}^*$ and the corresponding SVBRDF maps $\mathcal{G}(\boldsymbol{u}^*, \boldsymbol{p})$.

A key question is which loss $\mathcal{L}$ to use. Per-pixel losses (used by MaterialGAN) cause overfitting, especially when run on single images. This leads to unrealistic maps with undesirable artifacts, such as flash leaking into the albedo, that happen to lead to lower loss than any realistic maps would [Deschaintre et al. 2018; Gao et al. 2019]. Our main loss term is a style loss based on the Gram matrix [Gatys et al. 2015] of VGG features [Simonyan and Zisserman 2015], which approximates the difference between distributions of neural feature activations, and is insensitive to pixel alignment. This loss has been successfully used for material capture without pixel correspondence for procedural materials [Guo et al. 2020a; Shi et al. 2020]. The style loss drives the overall appearance, while condition and prior learned by our generator ensures the local coherence and tileability of the results, and that the optimized appearance remains in the subspace of realistic materials for a targeted class.

We combine the Gram matrix loss with an L_1 loss computed between downsampled images of resolution 16×16. This improves global matching of overall color and roughness. Furthermore, at every other iteration, we apply random translations to the generated material maps before passing them to the rendering operator: essentially, we are looking for a material that renders to a close perceptual match to the target image across all of its translations. This further makes any remaining overfitting negligible.

In contrast, the original MaterialGAN cannot be based on a global loss, because its material prior is not strong enough by itself. Using only a style loss and the low-resolution L_1 loss with our models works because they are class-specific and (in the case of highly structured materials like tiles) conditional on structure patterns. Tileability of the resulting materials is also only possible when using a style loss; a per-pixel match to the original (non-tileable) target image would of course not be able to produce tileable results.

5 RESULTS

We trained several versions of TileGen: conditional models for brick/tile and leather materials, and unconditional models for stone and metal materials. We additionally trained a color-conditioned

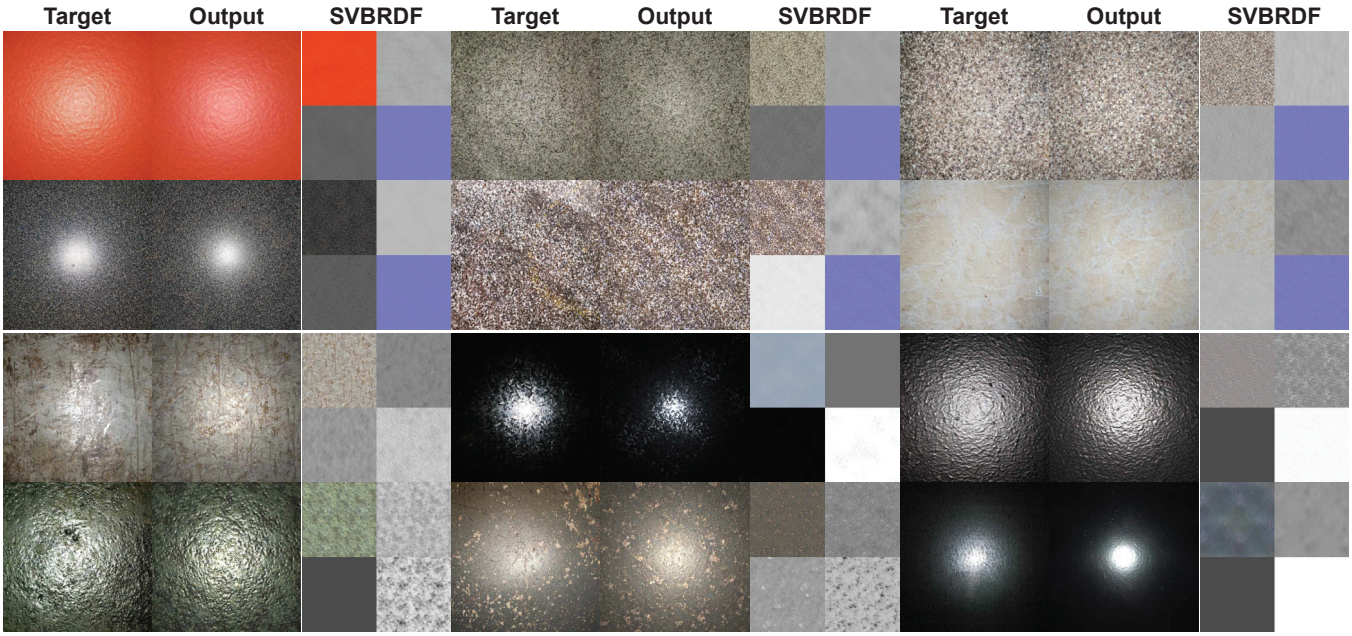

Figure 6: Examples of reconstructing unconditional materials from a *single* target photograph with flash illumination for stone (two top rows) and metal (two bottom rows). For each materials class, the target image is on the left, followed by our resulting rendered material and the predicted parameter maps (for stone, the order of feature maps is same as Figure 5; for metal, top-left: base color, top-right: height map, bottom-left: roughness, bottom-right: metallic map).

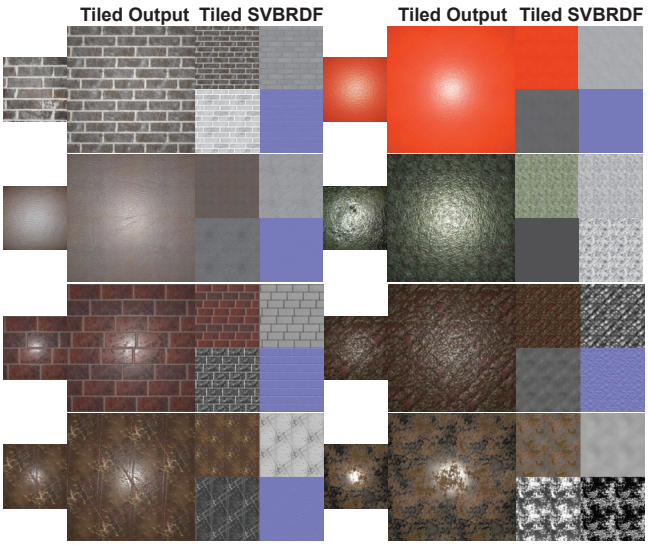

Figure 7: Demonstration of tileability for both inverse rendering (two top rows) and randomly sampled results (two bottom rows). The leftmost image is either the target for inverse rendering or the original generated material for random sampling, followed by a rerendered image using tiled texture maps and the corresponding tiled texture maps. The results show seamless tileability (periodicity) of our resulting textures, even though the target image is not tileable in the inverse rendering examples.

version shown in supplemental materials. Here, we demonstrate the results of applying these models to both forward generation and inverse optimization tasks, with and without conditions. We show more results in supplemental materials.

Forward generation. We demonstrate the results of using our models in a forward manner for material generation. In Figure 3, we show the results of several input patterns provided to our conditional generator, with four randomly sampled styles (latent codes) for each pattern, and showcase the quality and variety of the generated results. This demonstrates that the layout of the results indeed follows the input condition, while the style of the result is separated and depends on the latent code. For conditional models, all input conditions are tileable. Such patterns are easy to create using existing tools such as Substance Designer's generator nodes. In Figure 4, we show the results of unconditional generators, showing semantically meaningful variety as we sample new materials.

Material capture. Next, we show examples of using our models as priors for optimization-based material capture from real photographs. In Figure 5, we show conditional material capture from target images. We only assume that the input patterns match the approximate feature size and layout in the target image, with no pixel correspondence required. Note how our reconstructions capture the appearance of the material samples while remaining faithful to the input condition, even though it is not exactly aligned with the target image. We also show that we can change the conditional patterns, allowing to control the result details while preserving the target appearance. Next, in Figure 6 we show examples of materials recovered using our unconditional models: stone (2 top rows) and metal (2 bottom rows). By using a class-specific generative model, we can relax the optimization loss from per-pixel to global, preventing the flash artifacts typical in previous single image acquisition methods, and obtaining tileable results as an additional benefit.

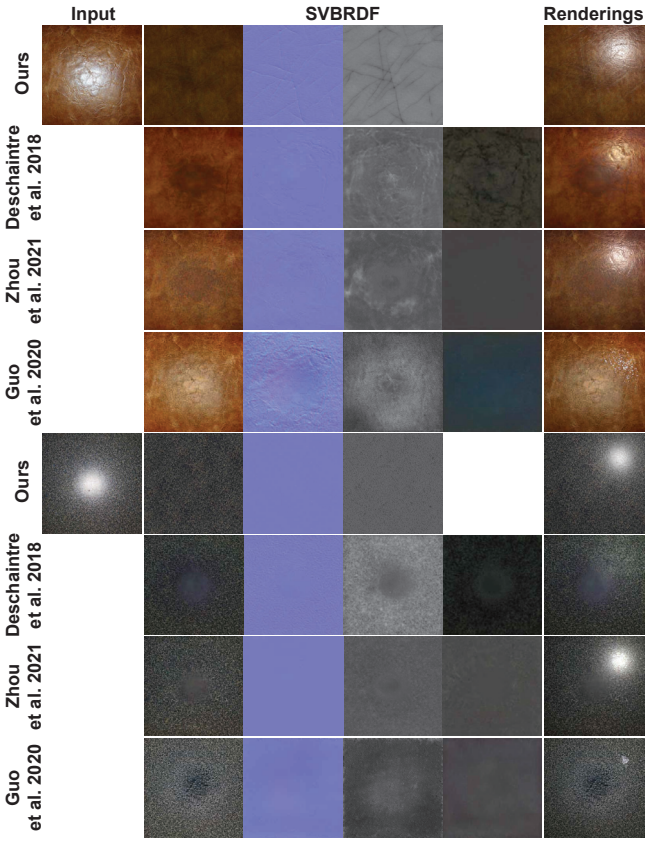

Figure 8: Comparison with three SVBRDF estimation approaches [Deschaintre et al. 2018; Guo et al. 2020b; Zhou and Kalantari 2021] on SVBRDF capture from a single target image (left). All of these approaches generate unclean feature maps, baking in the flash highlight. In contrast, our material maps and re-renderings are clean and plausible.

We evaluate further properties of our method: the tileability of our results (Figure 7). The supplementary materials also demonstrate that the effect of input patterns on results, the invariance to translation of the input pattern, the use of a colored condition, using conditions of higher resolutions.

Comparison. We first compare our approach to MaterialGAN [Guo et al. 2020b], Zhou et al. [2021] and Deschaintre et al. [2018] in Figure 8. Please refer to supplementary materials for more comparisons with MaterialGAN. We see that the MaterialGAN optimization approach, when used with a single input photograph, does not sufficiently constrain the space of realistic materials and generates unrealistic material parameters. The forward methods by Zhou et al. and Deschaintre et al. also suffer from flash artifacts and do not recover tileable materials. Our method does not suffer from these problems even when matching a single target image, thanks to its architecture, the pattern conditioning, the use of a style loss rather than a per-pixel comparison, and the random translations during optimization. While our results are not exactly aligned with the target photograph, they capture the overall appearance more accurately and are significantly more practical in a content creation

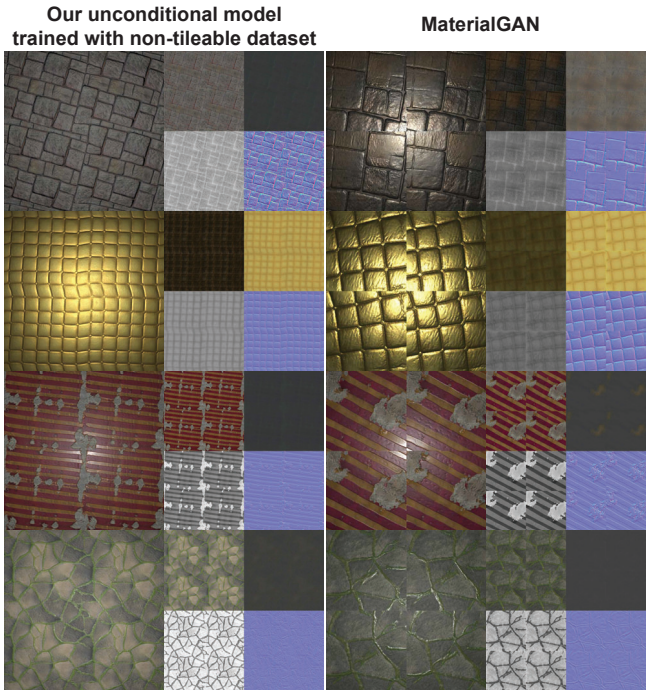

Figure 9: Comparison of randomly sampled results of original MaterialGAN and our unconditional model trained on the same non-tileable dataset MaterialGAN was trained on, showing 2x2 tiled results and the corresponding tiled material maps. Even when trained with a non-tileable dataset, our unconditional model can produce seamless texture maps.

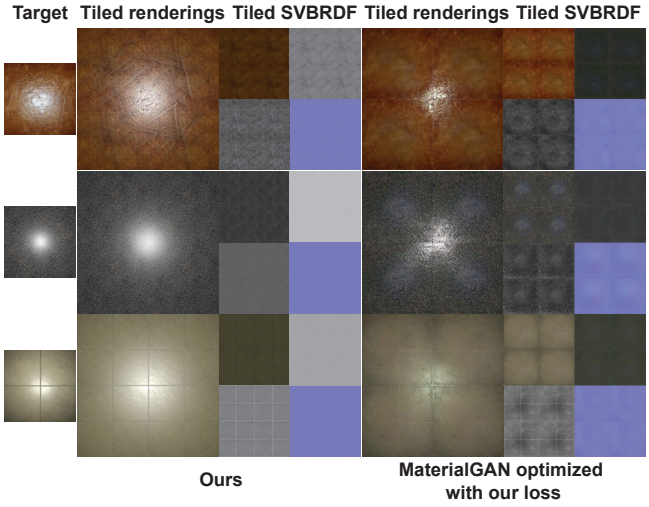

Figure 10: Comparison to the original MaterialGAN optimized with our global loss. Even when using a non pixel-wise loss, the results of MaterialGAN present highlight baking artifacts and are not tileable, unlike our results.

workflow. We further compare our tileable architecture to MaterialGAN by training our unconditional model with the original dataset from Deschaintre et al. that MaterialGAN was trained on (Figure 9), showing that our results are tileable even if the dataset is not. We also compare our inverse rendering results with the original

MaterialGAN but using our loss function in Figure 10, showing that just changing the loss is not sufficient to improve the results.

5.1 Discussion and Limitations

Our approach is trained per material semantic class, which enables meaningful style navigation and conditioning, but requires training one network per class and to have sufficient amount of dataset for each class. Our current generators have a resolution of 512×512 and have been trained for five days on 4 NVidia V100 GPUs. Higher-resolution generative models with faster training time remain a desirable future improvement. Our optimization takes around two minutes per target image, which is usable in practice, but a faster solver (perhaps based on a neural network predicting larger steps than basic gradient descent methods) would be a valuable direction.

6 CONCLUSION AND FUTURE WORK

We propose a new material authoring approach conditioned on class-specific networks, and show that we can condition them on easily authored inputs such as structure for tiles or cracks for leather. We train a semantically meaningful material generator, separating style from structure in the available control. We demonstrate material generation, acquisition matching the appearance of a single photograph, as well as additional effects such as interpolation, on multiple material classes. We believe that our method simplifies material design, making it accessible to novice user with extended control, without relying on complex material graphs.

We hope that future work will increase the resolution and training performance of the conditional generators. Meanwhile, it would be interesting to study the influence of material categorization on TileGen. Training TileGen on actual material sample photos from specific categories is also an exciting future direction, creating highly valuable material generators based on real data.

ACKNOWLEDGEMENTS

We thank Krishna Kumar Singh, Yijun Li and Jingwan Lu for help with CollageGAN set up and training details.

REFERENCES

Adobe. 2021. Substance. https://substance3d.adobe.com/assets.

Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. 2020. Image Generators with Conditionally-Independent Pixel Synthesis. *arXiv preprint arXiv:2011.13775* (2020).

Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. 2017. Learning Texture Manifolds with the Periodic Spatial GAN. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 469–477. https://proceedings.mlr.press/v70/bergmann17a.html

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-image SVBRDF Capture with a Rendering-aware Deep Network. *ACM Trans. Graph.* 37, 4 (2018), 128:1–128:15.

Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. 2019. Flexible SVBRDF Capture with a Multi-Image Deep Network. *Computer Graphics Forum* 38, 4 (2019).

Valentin Deschaintre, George Drettakis, and Adrien Bousseau. 2020. Guided Fine-Tuning for Large-Scale Material Transfer. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 39, 4 (2020). http://www-sop.inria.fr/reves/Basilic/2020/DDB20

Yue Dong. 2019. Deep appearance modeling: A survey. *Visual Informatics* 3, 2 (2019), 59–68.

Yue Dong, Jiaping Wang, Xin Tong, John Snyder, Yanxiang Lan, Moshe Ben-Ezra, and Baining Guo. 2010. Manifold Bootstrapping for SVBRDF Capture. *ACM Trans. Graph.* 29, 4 (2010), 98:1–98:10.

Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. 2019. Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. *ACM Trans. Graph.* 38, 4 (2019).

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. A Neural Algorithm of Artistic Style. arXiv:1508.06576

Dar'ya Guarnera, Giuseppe Claudio Guarnera, Abhijeet Ghosh, Cornelia Denk, and Mashhuda Glencross. 2016. BRDF Representation and Acquisition. *Computer Graphics Forum* (2016).

Pascal Guehl, Remi Allègre, Jean-Michel Dischler, Bedrich Benes, and Eric Galin. 2020. Semi-Procedural Textures Using Point Process Texture Basis Functions. *Computer Graphics Forum* (2020). https://doi.org/10.1111/cgf.14061

Jie Guo, Shuichang Lai, Chengzhi Tao, Yuelong Cai, Lei Wang, Yanwen Guo, and Ling-Qi Yan. 2021. Highlight-Aware Two-Stream Network for Single-Image SVBRDF Acquisition. *ACM Trans. Graph.* 40, 4, Article 123 (jul 2021), 14 pages. https://doi.org/10.1145/3450626.3459854

Yu Guo, Milos Hasan, Lingqi Yan, and Shuang Zhao. 2020a. A Bayesian Inference Framework for Procedural Material Parameter Estimation. arXiv:1912.01067

Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. 2020b. MaterialGAN: Reflectance Capture using a Generative SVBRDF Model. *ACM Trans. Graph.* 39, 6 (2020), 254:1–254:13.

Philipp Henzler, Valentin Deschaintre, Niloy J. Mitra, and Tobias Ritschel. 2021. Generative Modelling of BRDF Textures from Flash Images. *ACM Trans. Graph.* 40, 6, Article 284 (dec 2021), 13 pages.

Yiwei Hu, Julie Dorsey, and Holly Rushmeier. 2019. A Novel Framework for Inverse Procedural Texture Modeling. *ACM Trans. Graph.* 38, 6 (2019), 186:1–186:14.

Yiwei Hu, Chengan He, Valentin Deschaintre, Julie Dorsey, and Holly Rushmeier. 2022. An Inverse Procedural Modeling Pipeline for SVBRDF Maps. *ACM Trans. Graph.* 41, 2, Article 18 (jan 2022), 17 pages. https://doi.org/10.1145/3502431

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

Kaizhang Kang, Zimin Chen, Jiaping Wang, Kun Zhou, and Hongzhi Wu. 2018. Efficient reflectance capture using an autoencoder. *ACM Trans. Graph.* 37, 4 (2018), 127–1.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2021. Alias-Free Generative Adversarial Networks. In *Proc. NeurIPS*.

Tero Karras, Samuli Laine, and Timo Aila. 2018. A Style-Based Generator Architecture for Generative Adversarial Networks. arXiv:1812.04948

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2019. Analyzing and Improving the Image Quality of StyleGAN. arXiv:1912.04958

Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. 2017. Modeling Surface Appearance from a Single Photograph Using Self-Augmented Convolutional Neural Networks. *ACM Trans. Graph.* 36, 4 (2017), 45:1–45:11.

Yuheng Li, Yijun Li, Jingwan Lu, Eli Shechtman, Yong Jae Lee, and Krishna Kumar Singh. 2021. Collaging Class-specific GANs for Semantic Image Synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14418–14427.

Zhengqin Li, Kalyan Sunkavalli, and Manmohan Chandraker. 2018. Materials for Masses: SVBRDF Acquisition with a Single Mobile Phone Image. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 11207)*. 74–90.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2337–2346.

Carlos Rodriguez-Pardo and Elena Garces. 2022a. Neural Photometry-guided Visual Attribute Transfer. *IEEE Transactions on Visualization and Computer Graphics* (2022).

Carlos Rodriguez-Pardo and Elena Garces. 2022b. SeamlessGAN: Self-Supervised Synthesis of Tileable Texture Maps. *IEEE Transactions on Visualization and Computer Graphics* (2022).

Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. 2020. MATch: Differentiable Material Graphs for Procedural Material Capture. *ACM Trans. Graph.* 39, 6 (Dec. 2020), 1–15.

Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet Models for Refraction Through Rough Surfaces. *EGSR 2007* (2007), 195–206.

Wenjie Ye, Yue Dong, Pieter Peers, and Baining Guo. 2021. Deep Reflectance Scanning: Recovering Spatially-varying Material Appearance from a Flash-lit Video Sequence. *Computer Graphics Forum* 40, 6 (2021), 409–427. https://doi.org/10.1111/cgf.14387 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14387

Xilong Zhou and Nima Khademi Kalantari. 2021. Adversarial Single-Image SVBRDF Estimation with Hybrid Training. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.142635