Volume xx (200y), Number z, pp. 1–13

A Semi-procedural Convolutional Material Prior

Xilong Zhou[1,2], Miloš Hašan[2], Valentin Deschaintre[2], Paul Guerrero[2], Kalyan Sunkavalli[2] and Nima Khademi Kalantari[1]

[1] Texas A&M University
[2] Adobe Research

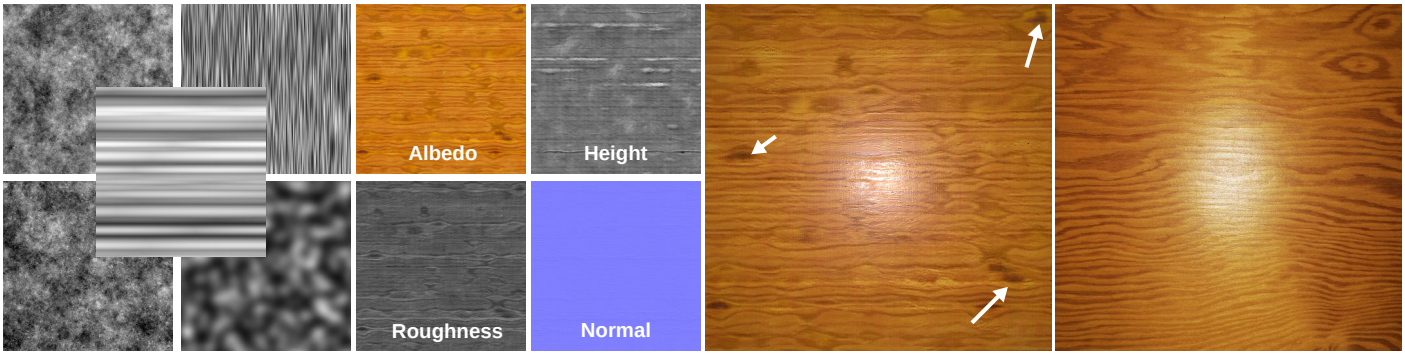

Figure 1: *We introduce a lightweight material capture method based on a* semi-procedural material prior. *Constructing our prior only requires choosing the initial patterns (left), which is a significantly easier task than designing a full procedural node graph for a material, as required by existing work [SLH*20]. These patterns are processed by a convolutional architecture inspired by procedural node graphs. We can optimize these weights to match target photographs of material samples (e.g., taken by a cell phone with flash). Note the closely matching appearance achieved for this wood example: our result even shows knots in the wood pattern, despite the initial grayscale maps having no knot-like features. Our method produces high-quality tileable results, and does not require graph structures, expert artists, nor training on large datasets of example materials.*

Abstract

Lightweight material capture methods require a material prior, defining the subspace of plausible textures within the large space of unconstrained texel grids. Previous work has either used deep neural networks (trained on large synthetic material datasets) or procedural node graphs (constructed by expert artists) as such priors. In this paper, we propose a semi-procedural differentiable material prior that represents materials as a set of (typically procedural) grayscale noises and patterns that are processed by a sequence of lightweight learnable convolutional filter operations. We demonstrate that the restricted structure of this architecture acts as an inductive bias on the space of material appearances, allowing us to optimize the weights of the convolutions per-material, with no need for pretraining on a large dataset. Combined with a differentiable rendering step and a perceptual loss, we enable single-image tileable material capture comparable with state of the art. Our approach does not target the pixel-perfect recovery of the material, but rather uses noises and patterns as input to match the target appearance. To achieve this, it does not require complex procedural graphs, and has a much lower complexity, computational cost and storage cost. We also enable control over the results, through changing the provided patterns and using guide maps to push the material properties towards a user-driven objective.

CCS Concepts

• ***Computing methodologies*** → ***Reflectance modeling;*** *Texturing;*

1. Introduction

Reconstructing spatially-varying surface reflectance from a single image is an under-constrained problem that requires appropriate

materials priors to be solved properly. Recent work on material capture has demonstrated the use of deep neural networks to learn such priors by either directly training CNNs to regress material properties from images [DAD*18, LSC18, ZK21] or by learning generative material models that can be used in an inverse rendering optimization scheme, such as in the MaterialGAN approach [GSH*20]. While these priors are quite useful, the reconstructed materials are often limited in terms of their quality, resolution, and editability, and they require training over large datasets of synthetic materials (since real material datasets are not easily available).

In contrast to such *image-based* material priors, recent work has demonstrated the use of *procedural* priors for materials [SLH*20, HDR19]. Procedural material graphs are composed of a set of generator nodes that produce initial grayscale noises and patterns, and a set of filter nodes that process the noise/patterns to produce material parameter textures. These graphs produces realistic results and allow for interactive editing and control over the generated material textures. Recent work has proposed methods to estimate procedural graph parameters from photos either by direct neural network-based parameter prediction [HDR19] or by mapping procedural graphs to differentiable programs that can be optimized to match the input photos [SLH*20]. These methods produce results that are high-quality, resolution independent and editable (by editing the graph parameters). However, the requirement of complex node graphs (which need to be manually designed by expert artists using professional software) as priors to guide the optimization is limiting.

We instead design a high-quality material prior which can be used to optimize for high-quality, tileable, resolution-independent and editable materials *without* the need for node graphs and/or large material datasets.

We note a common workflow in procedural material modeling: initial noises and patterns are processed by image filtering operations to synthesize the final texture maps. This inspires our proposed approach: we take a set of initial noises (unstructured grayscale maps) and patterns (structured grayscale maps), and process them using a sequence of filters with learnable coefficients to produce the output material textures. Our model is similar to convolutional neural networks (CNNs) operating on a fixed resolution and with specific restrictions on the convolution shapes. Unlike previous learning-based methods that train these networks on large datasets [GSH*20], and similar to procedural material estimation methods that do test-time optimization of procedural graph parameters [SLH*20], we optimize the filter/network parameters per material. We call this approach "semi-procedural" since we start from (typically procedural) initial noises and patterns, but do not require the full topology of a procedural node graph.

We observe that many nodes used in typical procedural graphs (low and high pass filters, color operations, etc.) can be exactly or approximately expressed as one of two convolution operations common in CNNs: first, a 1×1 convolution, which simply recombines input into output channels using per-pixel linear combinations (plus bias), and second, depth-wise (per-channel) $k \times k$ convolution, which applies a spatial filter to each input channel with no cross-channel communication (we use $k = 5$). These special convolutions lead to comparable inverse rendering results as general convolutional layers but with fewer optimizable weights, reducing storage of the model and providing additional regularization.

We find that the inductive bias of a network composed of these special types of layers, together with a set of initial noises, is a good material prior that does not require training on a large dataset, nor finding a suitable node graph for each material. The reason is that the operations are largely local and their expressive power is chosen to approximately capture the power of typical image processing operations in procedural node graphs. Thus, the weights of our architecture can be overfitted to specific samples; the prior is defined by the choice of initial noises/patterns and the inductive bias of the architecture, not by the weights.

Our design is related to previous work on deep image priors, which has shown that the structure of CNNs alone is a viable image prior [UVL18]. However, such a prior would be too weak for our application, as it is not specific to material textures in any way. In our inverse rendering framework, this results in basic stationary textures with no features larger than a few pixels, and is not capable of producing more global features common in tiles, wood, and many other examples; our initial patterns are critical in achieving such features. Our inductive prior limits the possible operations to a few limited convolution layers applied at fixed resolution starting from a chosen set of patterns/noises. This is an intentionally restricted architecture, which may initially seem like a limitation; however, this reduced generative power leads to a narrower (stronger) prior on plausible material appearance, in contrast to the broader MaterialGAN and even broader deep image prior.

Our semi-procedural prior reduces the complex task of constructing a full procedural node graph to a much easier task of choosing a few initial noises and patterns (for stochastic materials, the noises can even be chosen at random). Our approach retains several of the advantages of procedural approaches, such as the effectiveness of inverse optimization and tileability. We also maintain a high degree of editablity with the possibility to vary the input patterns. On the other hand, our method is specific to the inverse setting (fitting target photographs) and it is not designed to be a forward generative model for materials.

Our proposed material prior is combined with a differentiable rendering layer and a perceptual loss to build an end-to-end inverse material rendering pipeline. Similar to many SVBRDFs estimation methods [DAD*18, LSC18, DAD*19, GLD*19, GSH*20, GLT*21, ZK21], our target image is a photo of a planar surface lit by a collocated flash; however, this configuration is orthogonal to our method, and other lighting and viewing conditions can be used, simply by modifying the rendering layer.

We demonstrate that our inverse material estimation pipeline is able to produce tileable, high-resolution and high-quality results (see Fig. 1) comparable to the state-of-the-art approach [SLH*20], despite not requiring pre-existing node graphs nor training on large material datasets. Moreover, we provide ways to edit the results. We summarize our paper and its contributions as follows:

- We propose a semi-procedural material prior that maps procedural input patterns/noises to material maps using layers of learnable 1×1 and depth-wise convolutions.
- Using this prior, we demonstrate inverse material estimation re-

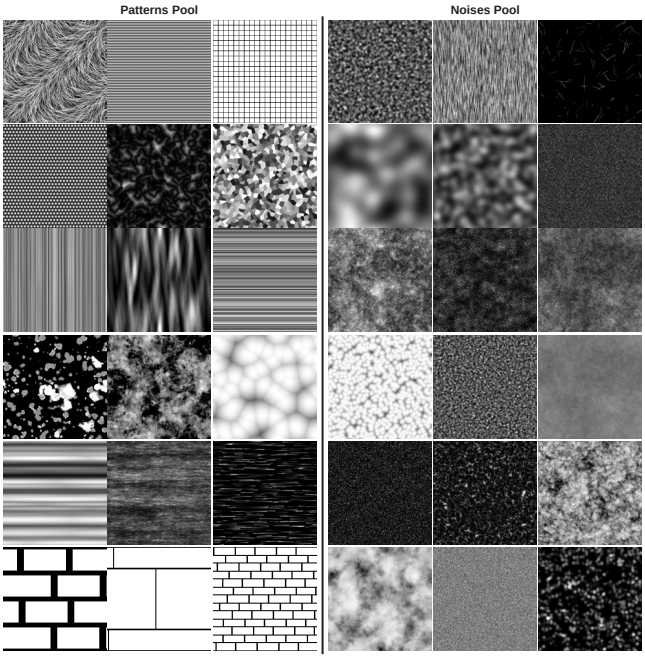

Figure 2: *Examples of initial patterns (left) and noises (right). Patterns are structured grayscale maps that are chosen by the user based on the material category and content; noises are general grayscale spatial variations that help adding detail to the material, and can often be chosen at random.*

sults comparable or better than state-of-the-art approaches with significantly less data requirements, code complexity, computational cost, and memory requirements (under 1.5K parameters, or under 6 kilobytes).
- We demonstrate additional editing capabilities by changing initial maps or providing guide maps.

2. Related Work

2.1. Material Representation

Spatially varying materials are traditionally represented as a set of material parameter maps, where each pixel represents parameters for analytical shading models such as GGX or Phong [WMLT07, Pho75]. This representation is practical because it is compact, but lacks editability as parameters are stored independently per pixel. There is no simple way to create such material parameter maps from scratch. To tackle this challenge, the graphics industry turned to procedural representations of materials, enabling editable material content creation. Different procedural models exist in the industry [Ado22a] and research literature [GHYZ20, HHD*22]; however, the creation of realistic editable procedural materials remains a challenge and relies typically on node graphs with complex topology. Recently, neural material representations were proposed [KMX*21, RJGW19, RGJW20, SRRW21], allowing to encode the shading model itself in neural networks and query a reflectance given a light and view direction; this is orthogonal to our goal of representing and capturing the spatial variation of material parameters. In this work we propose a material representation which relies on the recombining initial noises and patterns (picked from a pool or procedurally generated) and a limited set of neural layers and operations, building a prior on the space of plausible materials obtainable from the initial noises and patterns.

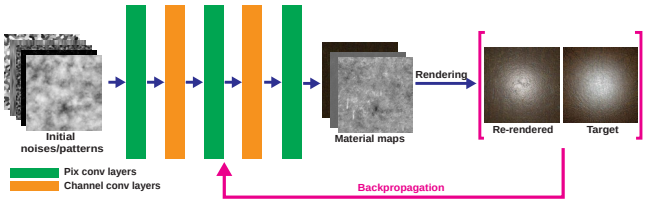

Figure 3: *Our architecture. Initial noise and pattern maps are processed alternatingly by* PIXCONV *and* CHANNELCONV *layers, resulting in material parameter maps (albedo, height, roughness) that are turned by a differentiable rendering operator into a material image lit by flash illumination. This rendered image is optimized to match a target image using a perceptual loss. Backpropagation updates the parameters of the* PIXCONV *and* CHANNELCONV *layers, minimizing the difference between the rendered image and the target.*

2.2. Material Acquisition

Material acquisition has been an active research area for decades [GGG*16], and recent work has been focusing on lightweight acquisition, typically using deep learning for reconstruction. Different approaches were proposed to capture material from a single [DAD*18, GLT*21, LSC18, ZK21, MRR*22, VPS21], multiple [DAD*19, GLD*19, GSH*20, HHG*22] photographs or a video [YDPG21]. These methods focus on the recovery of parameters maps per pixel and do not allow for much post-acquisition control. With our method we propose to train a specialized material prior, powered by a carefully chosen set of operations, allowing to better constrain single image reconstruction and avoid typical artifacts such as burned-in specular highlights. Recently Henzler et al. [HDMR21] proposed to use a U-net like architecture to reshape noise into material parameters given a single flash picture. In contrast to our work, their neural architecture is heavier and their approach is limited to stationary materials as they rely on noise and purely statistical metrics, losing any non-local spatial arrangement of the original image, as they do not use input patterns.

2.3. Material control

Naive control over analytical SVBRDF maps requires an artist to edit the parameters of each maps individually, using image editing tools. To facilitate this process different methods were proposed to decompose SVBRDF in different components, easier to individually edit [LBAD*06, LL11]. More recently, learning-based methods were proposed to transfer material properties [DDB20, RPG21, FR22] or interpolate and resample them [HDMR21]. Our method allows to edit the input noises and patterns, as well as provide guide maps to constrain the optimization result, providing control over the generated material structure (see Fig. 10).

2.4. Procedural material modeling

As mentioned, procedural materials representations maintain controllability for users; however, their creation requires significant ex-

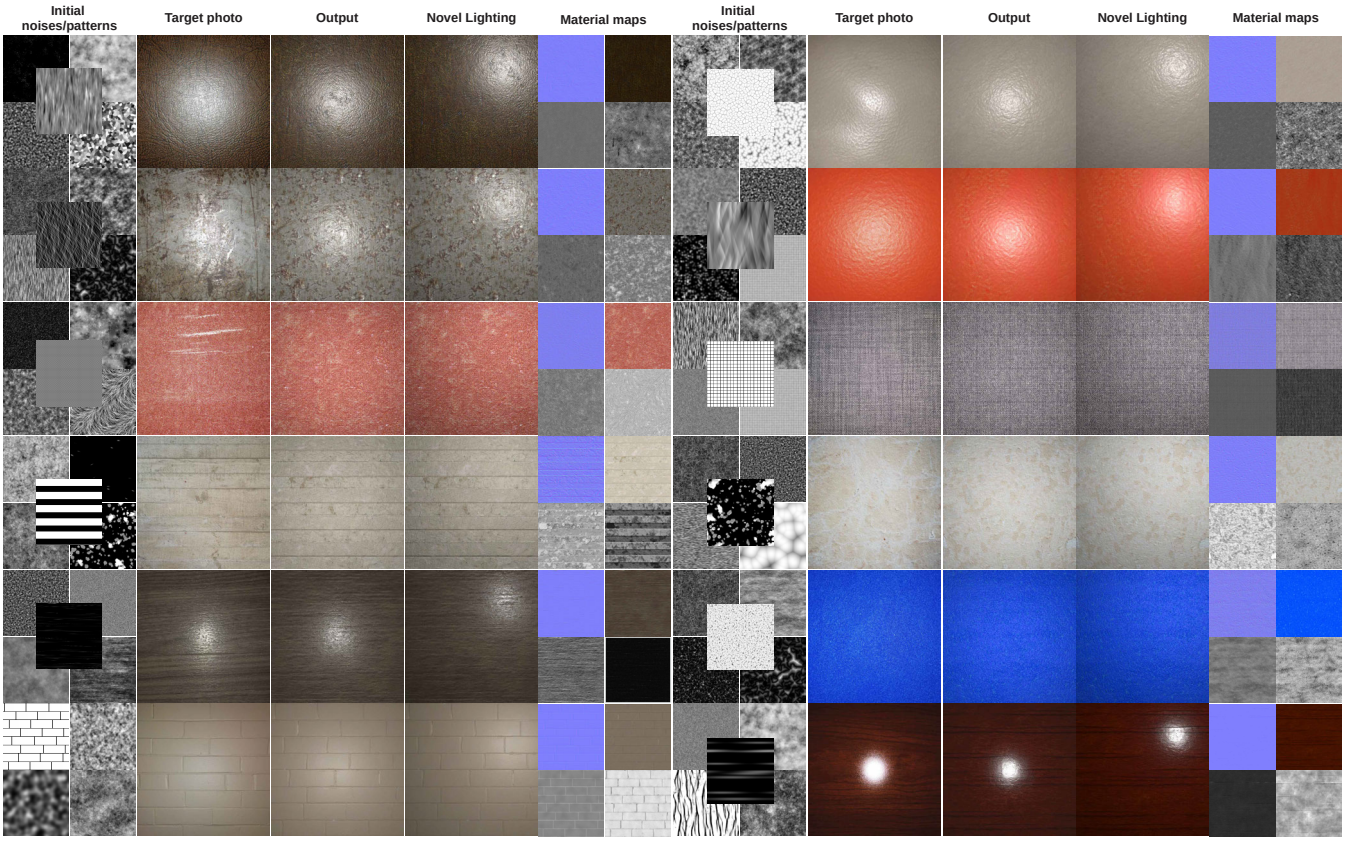

Figure 4: *Single-image material capture (inverse rendering) examples. For each triple, we demonstrate the input noises/patterns, our corresponding optimized result and relighting results given the target photo. Note the close match in appearance achieved by our method, even given input patterns that are very simple and do not explicitly contain some of the appearance features found in the resulting materials.*

pertise. To simplify their use, Hu et al. [HDR19] proposed to train a neural network to map photos of material samples to parameter vectors of a procedural node graph. Later, Guo et al. [GHYZ20], Shi et al. [SLH*20] and Hu et al. [HGH*22] used differentiable procedural material definitions (in the form of hand-written programs or node graphs, respectively) to optimize parameters to match a user-provided photograph. To alleviate the need for a pre-existing procedural material database, Hu et al. [HHD*22] proposed a semi-automatic pipeline which, given a set of material maps and their segmentation, generates a simple material graph reproducing its appearance. However, it requires already having a material (and its segmentation) to proceduralize, and is not a material capture method per se. As opposed to previous work, we do not require a full procedural node graph; we instead use a compact convolutional neural architecture which starts with a few chosen noises and patterns, and combines and processes them to reproduce a target photograph.

3. Method

Our goal is to design a differentiable material prior defining a subspace of plausible material textures. Specifically, this prior takes the form of a function M_θ that maps a set of initial patterns and noises into material texture maps. The function is defined by a neural architecture parameterized by weights θ, fitted per material. Thus, M_θ functions as a deep texture prior, additionally limited to textures obtainable from the initial patterns/noises using a small number of restricted convolutions. Because M_θ is differentiable with respect to its weights θ, we can use a differentiable renderer and back-propagation to optimize the weights to match a given target photo of a real-world material.

In practice, our pipeline takes a series of noises $\boldsymbol{n}$ and patterns $\boldsymbol{p}$ as input and outputs material parameter maps (for example, but not limited to, the albedo map $\boldsymbol{a}$, height map $\boldsymbol{h}$ and roughness map $\boldsymbol{r}$. While we could also directly infer normals, inferring height instead of normal leads to more plausible maps and optionally enables displacement [SLH*20, HHD*22, HDMR21]):

$$(\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{r}) = M_\theta(\boldsymbol{n}, \boldsymbol{p}). \tag{1}$$

All initial noises and patterns are grayscale images of equal size (typically but not necessarily square); see Fig. 2. The output maps are of the same size, and are typically grayscale or RGB (albedo).

To design the architecture for M_θ, we first observe the node types used in a typical procedural node graph, such as procedural graphs built using Adobe Substance Designer [Ado22a]. Generator nodes produce initial grayscale maps that are similar to our noises and patterns. These are combined and adjusted by filter

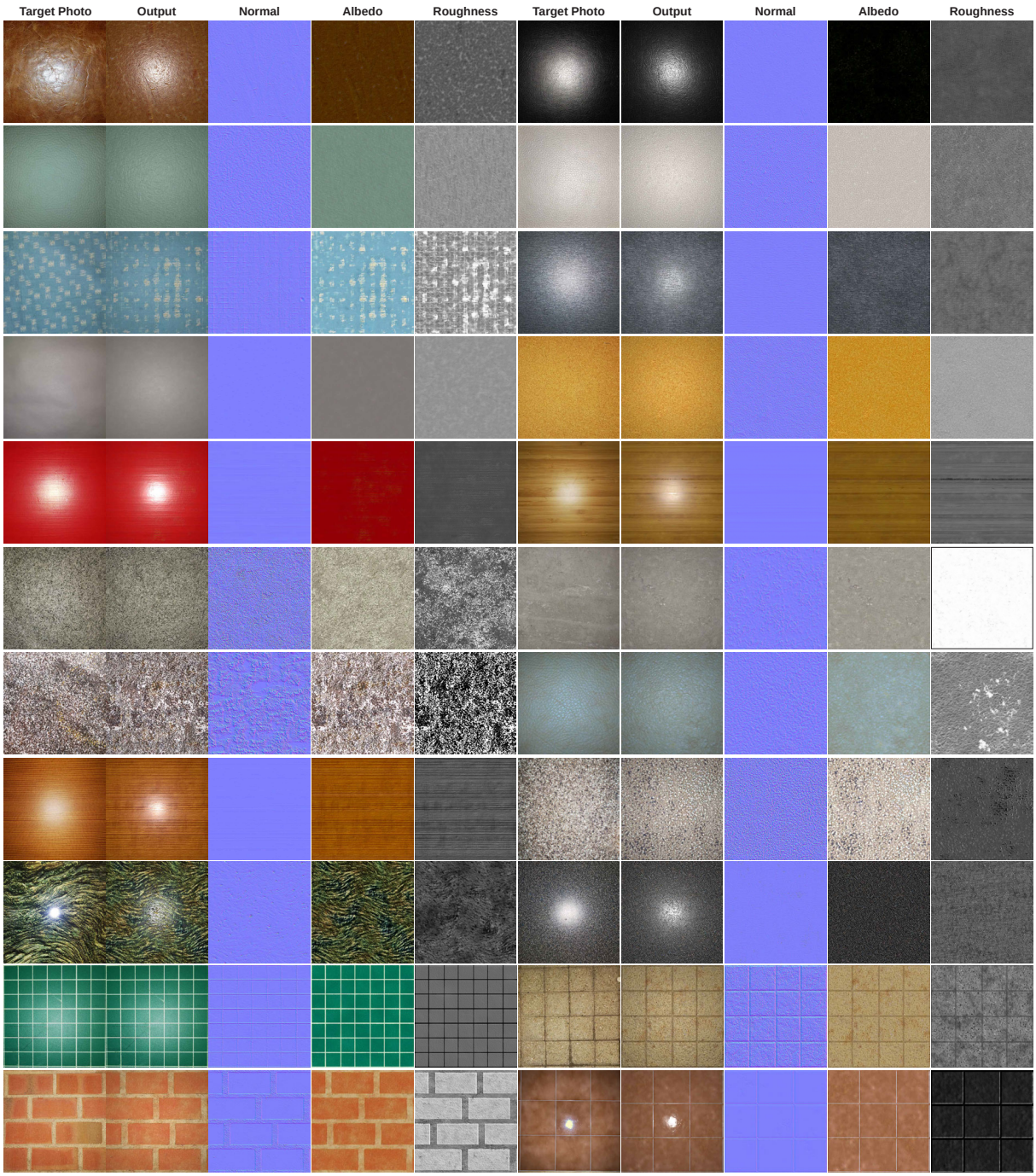

Figure 5: *More material capture examples, including both stochastic and structured materials, also showing the resulting material maps. Note that the maps are artifact-free, tileable and directly usable in 3D content creation pipelines. The corresponding initial noises/patterns are in the supplementary materials.*

nodes, which implement basic image processing operations: blending, blur, sharpen, channel shuffle, per-pixel curves, directional blur, spatial offsets, rotations or warps, etc. Typically, all intermediate and final maps are of the same resolution. The common image processing operations can be classified into three types.

- Per-pixel linear or non-linear operations on multiple channels.
- Spatial convolutions within each channel.
- Spatial transformations, such as translation, scaling, and rotation, and warping.

Inspired by these operation types, our material prior is designed as

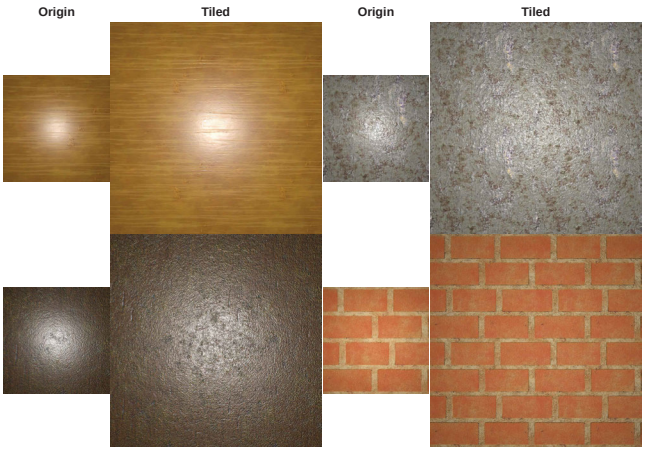

Figure 6: *Demonstration of tileability. We tile our resulting material maps 2x2 to show our results are seamlessly tileable.*

a special kind of convolutional neural network that uses two types of intermediate layers:

- PIXCONV: 1×1 convolution layers, which compute per-pixel linear combinations of channels plus bias, and
- CHANNELCONV: $k \times k$ per-channel convolution layers, with no channel cross-talk.

All convolutional layers preserve the input size, so the entire architecture operates at a fixed resolution. All CHANNELCONV convolutions use circular padding, so that tileable (periodic) inputs always lead to tileable outputs. We have experimented with using spatial transformation layers as well, but found that the above two convolution types are already sufficient for a large number of materials.

Our architecture consists of three PIXCONV layers interleaved with two CHANNELCONV layers (Fig. 3). The LeakyReLU function is used in between all layers. The last layer uses a sigmoid function to map the output values into a desired range. In our results, we use 16 channels for all internal layers. The starting channel count is given by the number of desired initial noises and patterns, and the final channel count is 5 when using albedo, roughness and height as the output maps. For CHANNELCONV layers, we use a kernel size of 5×5.

These design decisions enable us to reproduce most of the image processing power of typical procedural node graphs, but using a compact differentiable neural architecture with a very small number of parameters and with lower risk of overfitting to a target image than with more powerful networks.

3.1. Matching target images

Given a target image I^* (not necessarily tileable), our goal is to produce high-quality tileable SVBRDF maps, which can be re-rendered into an image I with a similar style as I^*. This requires combining our prior with a differentiable rendering layer and a perceptual loss to capture the style of the target image. Therefore, we use a differentiable direct illumination renderer to render an image under flash lighting. More specifically, we convert the height map into a normal map using central finite differences (where height scaling is itself an optimizable parameter), use the albedo map as a Lambertian term, and compute the specular component using the roughness map and a standard microfacet BRDF with the GGX normal distribution [WMLT07]. We assume a dielectric Schlick Fresnel term with a typical index of refraction 1.5; other Fresnel terms or specular/metallic maps could be added trivially. The rendered image I is computed as:

$$I = Ren(\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{r}) = Ren(M_\theta(\boldsymbol{p}, \boldsymbol{n})), \tag{2}$$

where *Ren* is the differentiable rendering operation, following the BRDF model described earlier. To capture the style of target image we use the style loss based on Gram matrices of VGG features as proposed by Gatys et al. [GEB, GEB16] instead of a per-pixel loss; this loss has been used for procedural material capture before [GHYZ20, SLH*20]. Additionally, we use an L_1 loss between downsampled images to match color and highlight shape better. So our full optimization loss is defined as:

$$\mathcal{L}_{\text{full}} = \lambda_1 \|T_G(I) - T_G(I^*)\|_1 + \lambda_2 \|I_d - I_d^*\|_1. \tag{3}$$

We minimize $\mathcal{L}_{\text{full}}$ over θ by gradient descent optimization. (In practice, we also optimize height scaling and light intensity as additional scalar parameters.) Here T_G represents concatenated Gram matrices of five deep feature layers (after each pooling operation) of the pretrained VGG19 network [SZ15]. I_d and I_d^* represent downsampled $d \times d$ resolution images, where we set d to 16 or 32; adding the donwsampled L1 difference to the loss helps matching low-frequency features like color and highlight shape, while avoiding matching high-frequency features like the exact locations of scratches or bumps. The weights λ_1 and λ_2 are set to 1.0 and 0.1, respectively.

3.2. Patterns and noises

As discussed, we classify the initial grayscale maps into patterns $\boldsymbol{p}$ and noises $\boldsymbol{n}$. Our noise pool contains around 200 noises that are widely used in Substance Source [Ado22b] graphs across material classes. The pattern pool contains more structured maps that are widely used in specific material classes. Here we consider six material classes: wood, leather, stone, metal, tiles and fabric. We build this pattern pool using 88 Substance graphs from the above material classes. In our results, we typically use 3 initial noises and 2 initial patterns for most classes; for tiles, we use 3 initial noises and 1 initial tile pattern. The input noises are typically selected randomly from our noise pool, while the patterns are chosen by the user. Note that all the noises and patterns are tileable, and that our architecture is designed to preserve this tileability in all generated material maps. Moreover, all noises and patterns can be constructed at arbitrary resolutions and can be described by a small number of parameters and random seeds. We will release our pattern and noise pool upon publication.

3.3. Implementation details and performance

We implement our approach in PyTorch, including the rendering operator. To implement the CHANNELCONV operation, we use the group convolution operation in PyTorch, setting the number of groups to the number of channels (16).

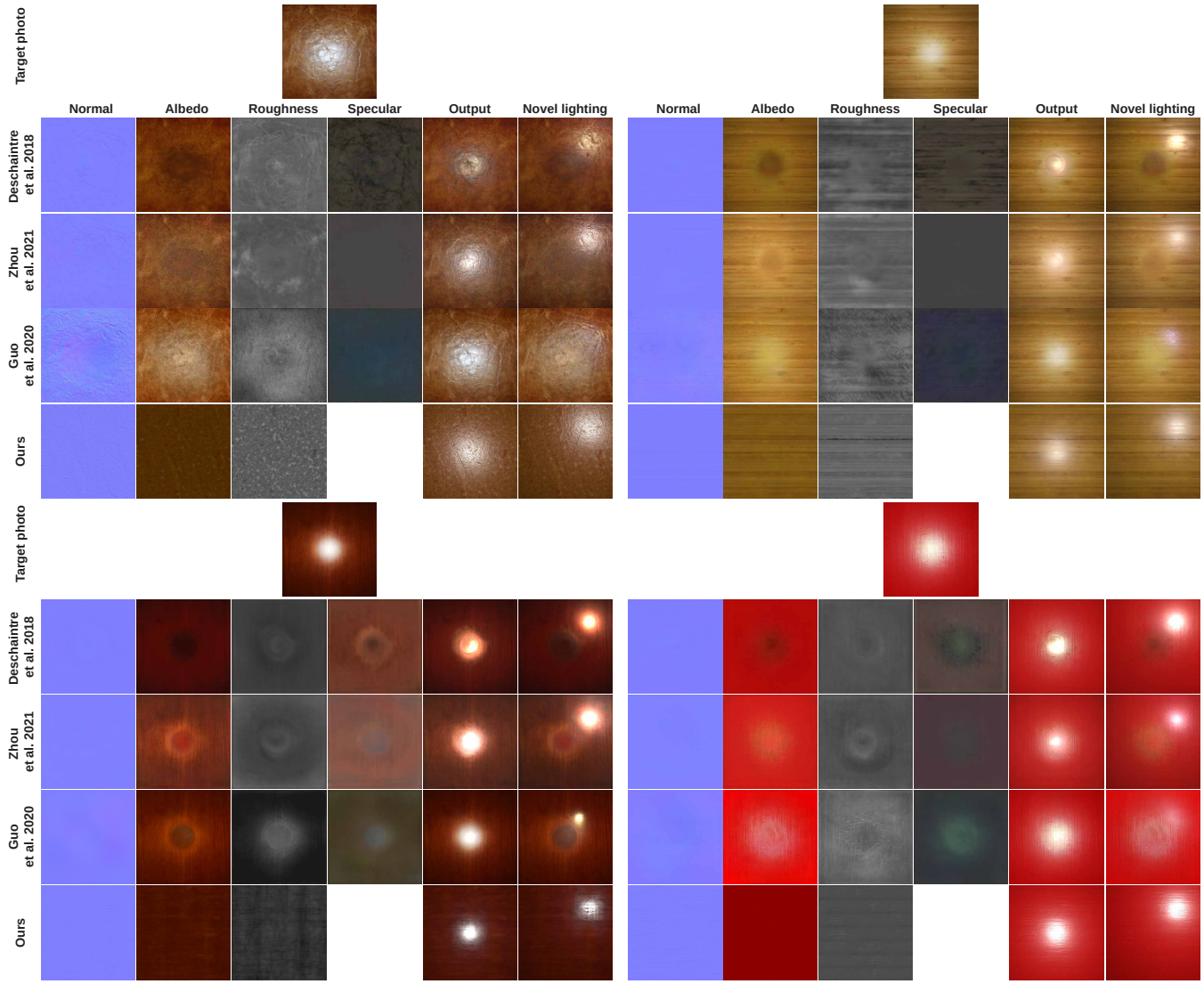

Figure 7: *Comparison with other per-pixel single image SVBRDF estimation approaches. We compare our approach with [DAD*18, ZK21, GSH*20]. Other single image SVBRDF estimation approaches have strong highlight burn-in onto material maps, but the estimated material maps of our approach are consistent and do not show artifacts. Note that the albedo brightness can be inconsistent between methods because albedo and light intensity have an inherent ambiguity. This issue could be easily resolved by calibration with a target of known reflectance, and is orthogonal to all methods tested.*

We use the Xavier approach [GB10] to initialize our network and do not require any form of pre-training. For optimization, we use the Adam optimizer [KB14] with an initial learning rate of 0.005. The internal hyper-parameters of Adam are kept at their defaults in PyTorch. For fine-tuning network during editing, we keep the same optimizer settings. We typically perform 2000 iterations for optimization and 50 iterations for editing. Optimization takes around 2-3 minutes and editing takes less than 5 seconds on single Nvidia 2080 Ti GPU. Our method requires 2GB of GPU memory and takes around 2 minutes for 2k optimization steps; using Shi et al. [SLH*20]'s method, the memory and runtime depend on the complexity of the input graphs. Our observation shows that on average Shi et al.'s method takes 4∼6GB and 10∼20 minutes for 1k optimization steps. Both were tested on a single RTX2080 Ti.

3.4. Editing

Note that our goal is to propose a compact alternative to complex procedural node graphs. Our proposed architecture achieves high quality tileable SVBRDFs comparable to procedural graphs; however procedurals also enable the user to perform flexible and interactive editing. For example, in Substance graphs, artists can control the appearance of the generated material maps either by changing the input generators or tweaking the parameters of different graph nodes. We propose similar editing operations for our framework:

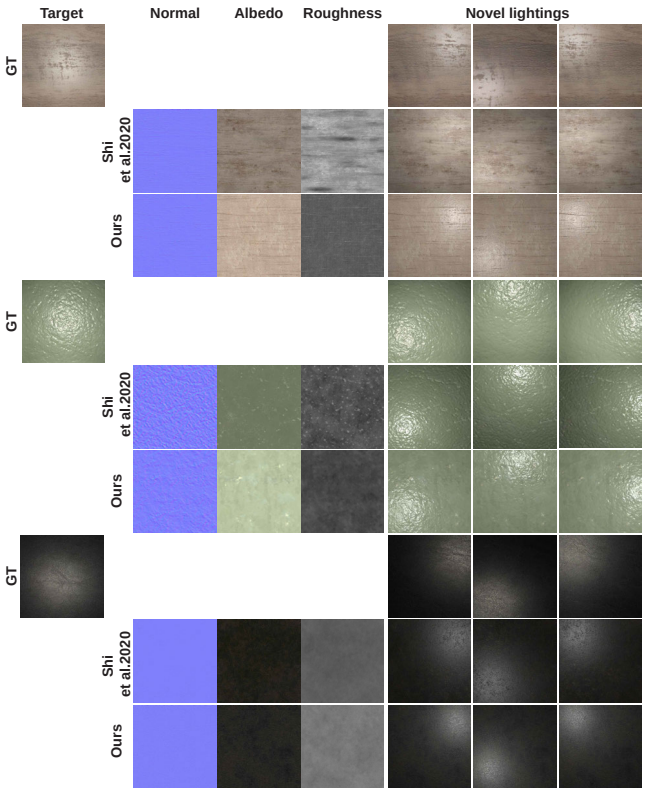

Figure 8: *Comparison with MATch [SLH*20] on real images with ground truth pictures with a different lighting. We find that both methods produce high-quality tileable material maps without over-exposure artifacts. However, MATch requires a complete procedural node graph which can represent the material of interest, while our approach requires only a few noises/patterns.*

Table 1: *Numerical Gram Matrix difference comparison with per-pixel single image SVBRDF estimation approaches [DAD*18, ZK21, GSH*20] on 53 synthetic images and 51 captured real images (Real1) with ground truth; as well as comparison with MATch [SLH*20] on 11 captured real images with ground truth, which we know can be reproduced by procedural materials (Real2). On synthetic data the error is computed over comparable material maps and on real data over rendered images. Our approach reaches state of the art quality on synthetic data, without being trained on it and exceeds it on real data while not requiring a predesigned procedural material graph. (The bold values represents the lowest values under each category)*

	Synthetic		Real1	Real2
	albedo	rendered	rendered	rendered
Des18	0.0133	0.0084	0.0108	-
Zhou21	**0.0129**	**0.0056**	0.0089	-
Guo20	0.0145	0.0078	0.0092	-
MATch	-	-	-	0.0037
Ours	0.0137	0.0059	**0.0055**	**0.0036**

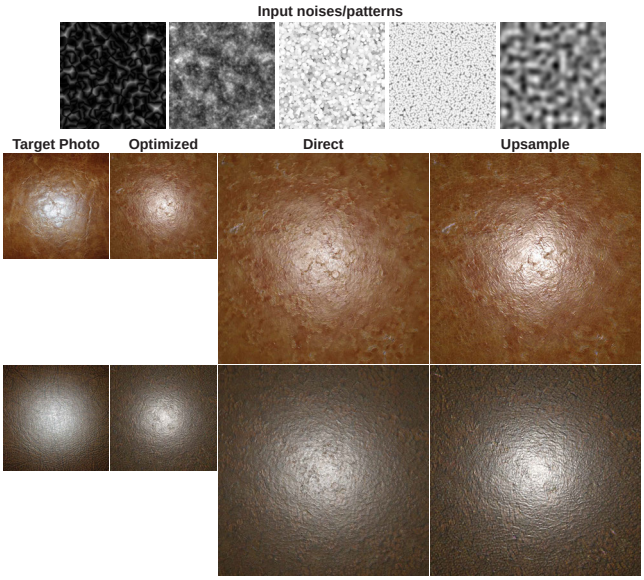

Figure 9: *Demonstration of higher resolution. Optimize: optimized* 512×512 *results given specific target image. Direct: directly applying the optimized network to high resolution* 1024×1024 *initial maps. Upsample: we upsample the convolutional kernels of the optimized network and use these with* 1024×1024 *initial maps.*

we would like to control the estimated material maps by changing the initial maps or performing fine-tuning on the optimized network given a guide map provided by the user.

Changing input noises. In procedural modeling, one common way to control generated material maps is to control the initial pattern generators. This approach maps directly into our pipeline. The intuition is that after our convolutional architecture is optimized to represent a target image, a user can simply edit the input pattern/noises (by slightly adjusting them or completely changing them), resulting in another plausible material.

However, our observation shows that only changing the initial maps, with no change to the weights θ, can cause some artifacts. Therefore, we propose to perform a small number of fine-tuning steps on the optimized weights under the new noises and patterns. Our experiments show that this fine-tune process usually takes less than 50 gradient descent steps (compared to about 2000 when starting from scratch), which enables users to edit the initial maps quickly and conveniently.

Guide map. Apart from changing initial patterns and noises, in the classic procedural graph pipeline, users can play with the parameters of graph nodes to adjust the different image processing operations, which further control the resulting material maps. In our pipeline, no precisely analogous operation exists, but we propose an operation with a similar editing power. Specifically, a user can provide a guide map (for example, an albedo map or roughness map), and these guide maps can be used as a constraint to guide optimization to a different desired result. For example, given that users provide a guided albedo map $\boldsymbol{a}_g$, the loss used to fine-tune the system can be written as:

$$\mathcal{L}_{\text{edit}} = \mathcal{L}_{\text{full}} + \lambda_e \| T_G(\boldsymbol{a}) - T_G(\boldsymbol{a}_g) \|_1, \quad (4)$$

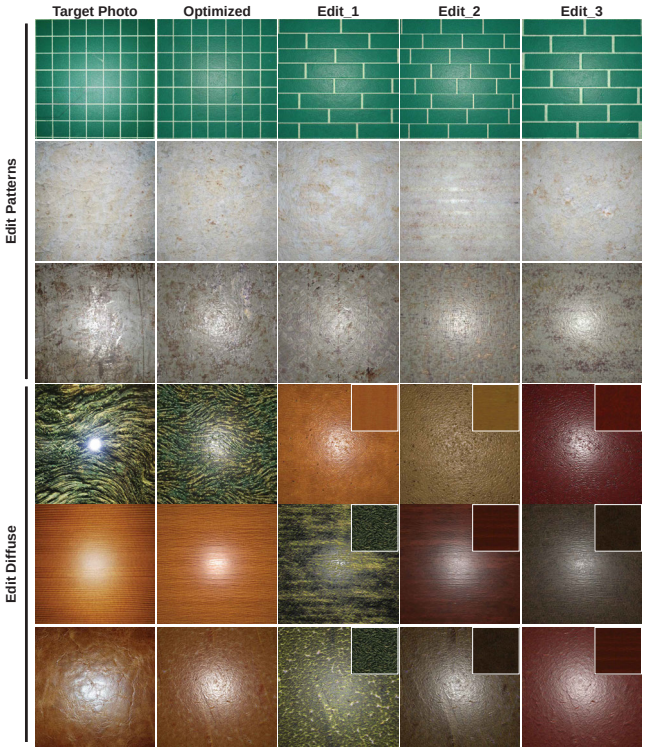

Figure 10: *Material Editing. We show two types of edits. In rows 1-3, we show editing the initial patterns and noises. In rows 4-6, we show editing results by providing different guide albedo maps. All examples are generated by re-optimizing network for 50 steps. The corresponding edited patterns are demonstrated in the supplementary material.*

where λ_e represents the magnitude of the editing term. We use $\lambda_e = 100$ to ensure the guide is closely followed. Similarly, we can design losses for other material maps as well. Our experiments show that this fine-tuning takes a small number of steps and lets a user apply a wide range of edits to the resulting material maps.

4. Results

In this section, we first show a number of results of applying our semi-procedural prior to inverse rendering (single-image material capture). Next, we demonstrate editing capabilities, comparisons with other material capture approaches, and demonstrate resolution invariance. For all stochastic material results, we randomly select the input noises. For more structured materials, we roughly pick a matching pattern, but show in Fig. 12 that our approach can still perform well with loosely corresponding patterns.

4.1. Inverse rendering

In Fig. 4, we show material capture results for a number of materials across multiple categories, both stochastic and structured (provided a similar structure is provided in the input patterns): leather, wood, brick, fabric, wall paint, concrete. In each example, we show the initial noise/pattern images (left), the resulting re-rendered material (middle) and the target photo (right). In Fig. 5, we show

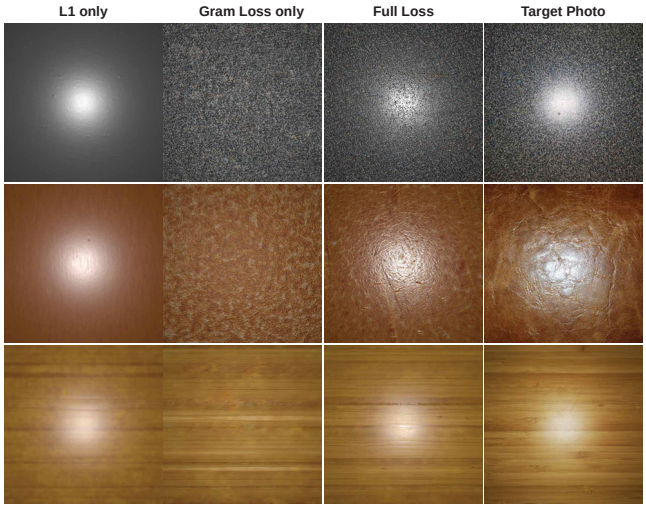

Figure 11: *Effect of different losses. In this figure, we compare our full loss with L1 loss and Gram loss alone. As shown, the L1 loss alone misses high frequency details, while the Gram loss alone fails to capture the specular behaviour. By combining both losses, we capture both specular behaviour and texture details.*

further examples, including the resulting material maps (albedo, roughness, and normal; we convert height to normal for visualization, to facilitate comparison to most previous works which generate normal maps. These results show that our approach is capable of producing high-quality, visually plausible results covering many different material classes. Note that the resulting material maps are free of artifacts, and never burn in specular highlights or light falloff. Our method always produces tileable results given tileable initial patterns and noises, as is shown in Fig. 6.

4.2. Comparison with other approaches

We compare our approach with per-pixel single-image estimation approaches [DAD*18, ZK21, GSH*20] as well as a previous procedural material capture approach [SLH*20]. We use the code provided by authors to generate the results of [DAD*18, ZK21]. For MaterialGAN [GSH*20], as discussed in the paper, we first optimize the latent and noise space to embed the results of [DAD*19] for initialization, and then optimize further to match the input image directly. As is shown in Fig. 7, our approach does not bake highlights into material maps, which is a common problem with other single-image SVBRDF approaches. Furthermore, our maps are tileable and ready for use in a 3D content creation system.

To compare our results with MATch [SLH*20], we directly use the output material maps from their paper. Both our approach and MATch produce high-quality results (Figure 8). However, our method is more compact and computationally efficient than MATch. We simply require the user to choose initial patterns and noises from a pool, which is substantially easier than constructing a complete procedural node graph. Additionally, node graphs may not be available for all types of target photos, while our pool of patterns and noises, together with our prior, can cover a potentially

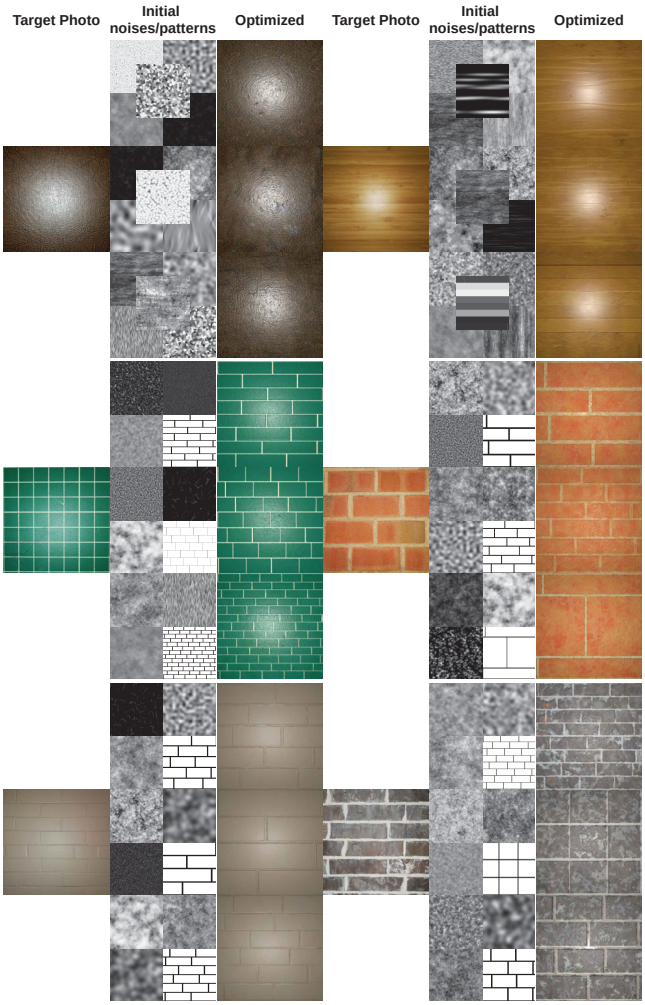

Figure 12: *Effect of input noises and patterns. We optimize our network for a given target image with different input noises and patterns. We see that even with different noises/patterns, our approach is able to generate outputs that correctly match the target style. In the highly structured tiles examples, we show input patterns which significantly differ from the one in the target, yet our method still matches the desired appearance.*

much larger range of materials, and is easier to extend by simply adding new patterns and noises.

We further analyze our results quantitatively in Tab. 1 and show that despite not being trained on synthetic data to recover the material per pixel, our results quality on synthetic data is on par with previous work trained on synthetic data, and does better on real data. We also quantitatively compare to MATch [SLH*20], confirming that we achieve similar quality, without requiring pre-designed graphs. For real data we use 51 real photographs to compare with most methods (Real1) and a subset of 11 that have matching procedural graphs, to compare to MATch (Real2).

4.3. Higher resolutions

We demonstrate that our results can be easily extended to higher resolutions than the target resolution used for optimization. As shown in Figure 9, the model optimized at a 512×512 resolution can be directly used with high resolution (1024×1024) initial maps; however, the output lacks high frequency details (see third column). This is because our 5×5 CHANNELCONV operations are adapted to the original resolution. To adapt the model correctly to higher resolutions, we upsample the convolutional kernels from 5×5 to 9×9 (or equivalently, double their radius from to 2 to 4) and then normalize the sum of the original kernel weights to preserve energy. By using this technique, we are able to generate high-resolution results from low-resolution targets without losing high-frequency details.

4.4. Editing results

As discussed in Sec. 3, we perform editing by fine-tuning the network given different input noises or guide maps. Our experiments show that for most examples this re-optimization will converge within 50 iterations (less that 5 seconds). Edits are illustrated in Fig. 10. Given different input noises, we can quickly obtain the different material maps with the same appearance and different initial patterns; given an albedo guide map, we can also change the appearance of material maps to match the guide map. Our observation shows that this indirect editing strategy works for most scenes. Limitations of this strategy are discussed in Sec. 5.2

5. Discussion

In this section, we add ablation studies, and discuss limitations and future work of our approach. In the ablation studies, we analyze the effects of different losses, network architectures and input noises.

5.1. Ablation Study

Effect of different losses. We first analyze the effect of different losses on our approach. More specifically, we compare our full loss results with L1 loss only and Gram matrix loss only, as shown in Fig 11. With only L1 loss, results are overall blurry and lack fine details; with only Gram loss, results are sharp but fail to reproduce the specular reflection. With our full loss, combining the downsampled L1 loss and Gram loss, our results capture the style correctly as well as reproduce the specular reflection.

Effect of different initial noises and patterns. We also analyze the effect of different input noises and patterns. Given the same target image, our neural network is optimized using different input combinations as well as different number of inputs. In Fig. 12, we show all five input noises and their corresponding optimized results. As shown, our approach is able to produce high quality results that matches the style of target images even though different input noises are taken as input. In Fig. 13, we show the effect of progressively adding inputs. In the first wood example (left-most in the figure), when only a structured pattern is fed as input, the optimized results only follows this structure, without any fine details, which appear as we add more inputs noises. Similarly, if only input noises are fed as input, the results only simulate the style

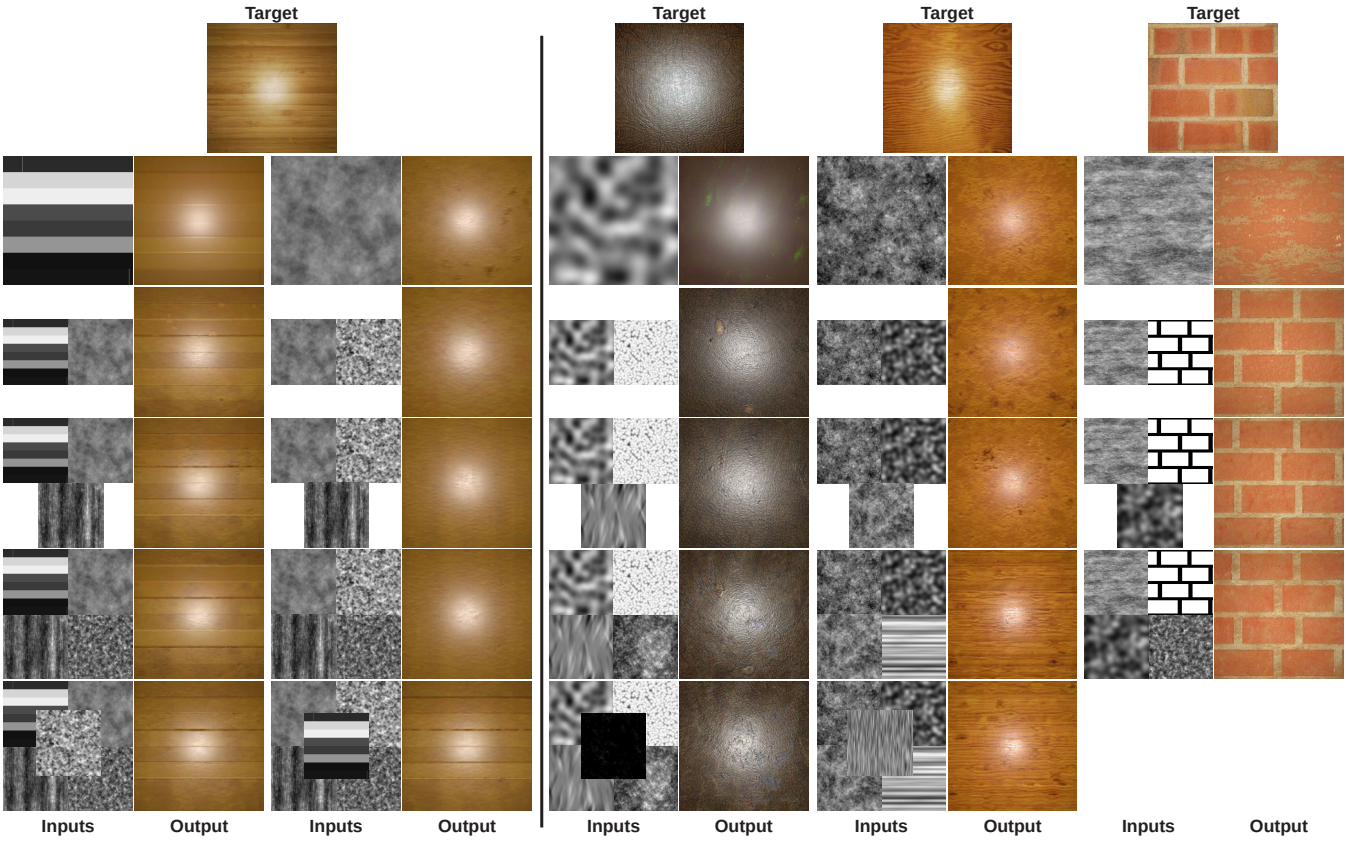

Figure 13: *Effect of number of input on the optimized results. Given target photos, we increase the number of input noises and patterns from 1 to 5 (4 for the tile example) and show the corresponding optimized results. As shown in the figure, as we add inputs signals, results show finer details and become more visually pleasing. For a target with strong structured patterns (left-most example), we show the effect of different input signals ordering. With only the structure input (top-left, first column), the optimized results are well-structured but lack realistic fine details, and without only noise (top-left, third column), our approach focuses on the overall appearance, but cannot reproduce structured patterns until a suitable signal is provided as input (5th row, 4th column).*

of target, losing structured patterns, which is easily solved by providing a pattern (shown in the 3rd and 4th column). We empirically set the number of input as five to keep enough fine details using small network. In practice, thanks to the flexibility of our approach, users can select patterns from our provided pattern pool or create any preferred patterns as input, using any pattern generator [GAD*20, Ado22a, HHD*22].

Network architecture. We compare our proposed architecture with PIXCONV layers only, as well as standard CNN convolution layers (Fig. 14). With PIXCONV layer only, the output images are the results of per-pixel operations only, without considering any spatial coherence, leading to unrealistic re-rendered images following the initial maps too closely and lacking details. Compared with a standard CNN with full convolutions, our proposed approach produces comparable results with significantly fewer training parameters. With our current architecture, regular convolutions would have 10,437 training parameters, while our proposed model contains only 1,258 training parameters. Combined with the fact that the initial input patterns and noises are typically themselves procedural (and thus described by at most tens of parameters), our method also provides an extremely storage-efficient material representation.

5.2. Limitations and future work

Even though our approach is lightweight and can produce high-quality materials, some limitations remain. As shown in Fig. 15, given strongly structured target photos, initial patterns that are not chosen well for the target appearance will fail to capture its style. Therefore, for these strongly structured target photos, users need to manually provide initial patterns that are sufficiently close to the target structure (even though pixel correspondence is not required). Similarly, if multiple structured patterns are provided to represent the same signal, the optimisation risks getting into a local minimum that combines both patterns. Automatic initial pattern selection is an interesting future direction.

Even though our method allows for several editing operations, it still does not provide as direct control over some material properties as a native procedural model. Optimization with guide maps may in some cases take longer to fine-tune or cause material map entanglement issues as shown in Fig 16. An interesting future solu-

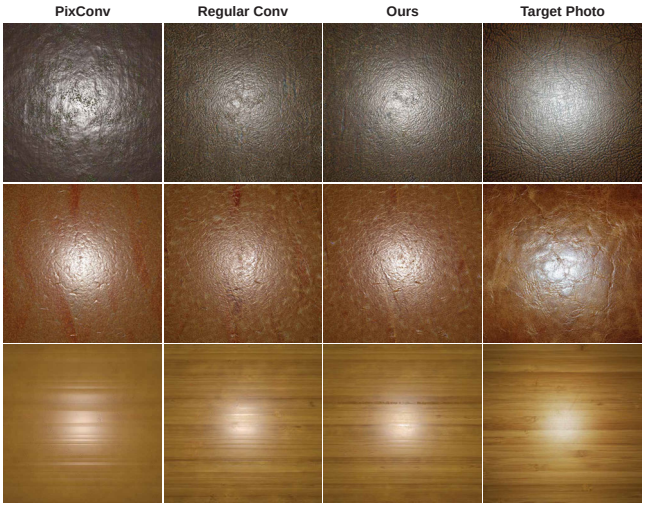

Figure 14: *Effect of different network architectures. We compare our architecture with using only* PIXCONV *layers and with standard CNN operations. Our results are much closer to target than* PIXCONV *and comparable with standard convolution, while using significantly fewer parameters.*

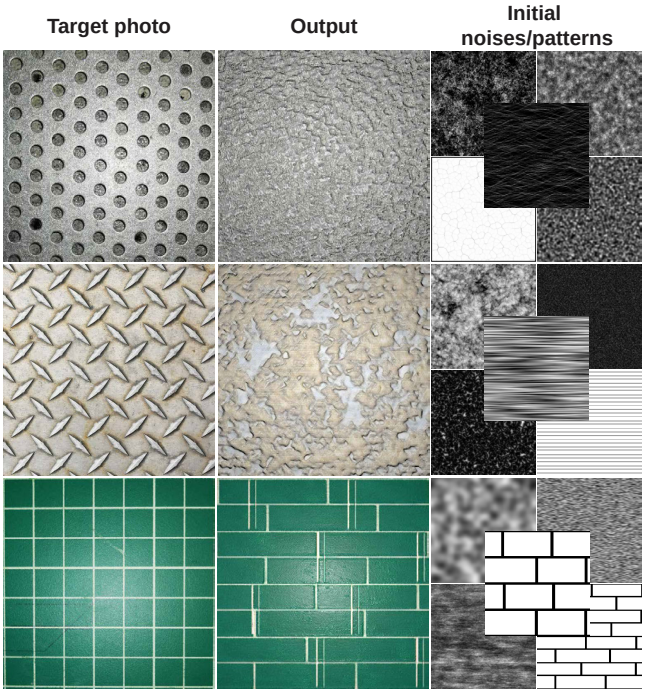

Figure 15: *In strongly structured target images, using initial patterns that are too distant from the target structure results in outputs that do not match the desired appearance (first two lines). Further, if multiple patterns matching a signal are provided as input, our optimisation may fail (third line). A better choice of initial patterns is required to resolve this.*

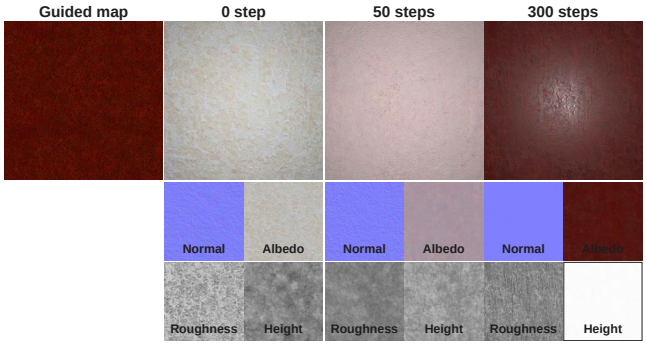

Figure 16: *Limitation: Given this guide albedo map, our method takes more than 50 steps to converge and shows some entanglement with the roughness being modified too.*

tion could rely on a hypernetwork, capable of directly editing our network's weights to achieve direct and more accurate editing.

Our prior is not a generative model and is designed to be used through inverse rendering. It would be interesting to extend our approach to a generative model in the future.

6. Conclusion

We present a differentiable semi-procedural material prior, defining a set of plausible materials constructible from a set of initial patterns and noises. We rely on a set of restricted convolution kernels to constrain the generative output space of our prior. We demonstrate the use of this prior for single-image material capture, achieving state-of-the-art results previously achievable only when using pre-existing procedural graphs as priors. The benefits of our approach include tileability, resolution independence and editability. Our prior can handle non-stationary materials with larger features, does not suffer from flash artifacts typical in per-pixel methods, and does not require pre-existing complex procedural materials, artistic [KB14] expertise, nor training on [RPG21] large datasets. [GSH*20]

Acknowledgements

This project was in part funded by a generous gift from Adobe.

References

[Ado22a] ADOBE: Substance designer, 2022. 3, 4, 11

[Ado22b] ADOBE: Substance source, 2022. 6

[DAD*18] DESCHAINTRE V., AITTALA M., DURAND F., DRETTAKIS G., BOUSSEAU A.: Single-image svbrdf capture with a rendering-aware deep network. *ACM Trans. Graph. 37*, 4 (Aug 2018). 2, 3, 7, 8, 9

[DAD*19] DESCHAINTRE V., AITTALA M., DURAND F., DRETTAKIS G., BOUSSEAU A.: Flexible svbrdf capture with a multi-image deep network. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 38*, 4 (July 2019), 1–13. 2, 3, 9

[DDB20] DESCHAINTRE V., DRETTAKIS G., BOUSSEAU A.: Guided fine-tuning for large-scale material transfer. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 39*, 4 (2020), 91–105. 3

[FR22] FISCHER M., RITSCHEL T.: Metappearance: Meta-learning for visual appearance reproduction. *ACM Trans Graph (Proc. SIGGRAPH Asia) 41*, 4 (2022). 3

[GAD*20] GUEHL P., ALLÈGRE R., DISCHLER J.-M., BENES B., GALIN E.: Semi-Procedural Textures Using Point Process Texture Basis Functions. *Computer Graphics Forum* (2020). 11

[GB10] GLOROT X., BENGIO Y.: Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)* (2010), pp. 249–256. 7

[GEB] GATYS L. A., ECKER A. S., BETHGE M.: A neural algorithm of artistic style. *Journal of Vision 16*. 6

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016). 6

[GGG*16] GUARNERA D., GUARNERA G. C., GHOSH A., DENK C., GLENCROSS M.: BRDF Representation and Acquisition. *Computer Graphics Forum 35*, 2 (2016), 625–650. 3

[GHYZ20] GUO Y., HAŠAN M., YAN L., ZHAO S.: A bayesian inference framework for procedural material parameter estimation. *Computer Graphics Forum 39*, 7 (2020), 255 – 266. 3, 4, 6

[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 Trans. Graph. 38*, 4 (July 2019). 2, 3

[GLT*21] GUO J., LAI S., TAO C., CAI Y., WANG L., GUO Y., YAN L.-Q.: Highlight-aware two-stream network for single-image svbrdf acquisition. *ACM Trans. Graph. 40*, 4 (July 2021). 2, 3

[GSH*20] GUO Y., SMITH C., HAŠAN M., SUNKAVALLI K., ZHAO S.: Materialgan: Reflectance capture using a generative svbrdf model. *ACM Trans. Graph. 39*, 6 (2020). 2, 3, 7, 8, 9, 12

[HDMR21] HENZLER P., DESCHAINTRE V., MITRA N. J., RITSCHEL T.: Generative modelling of brdf textures from flash images. *ACM Trans Graph (Proc. SIGGRAPH Asia) 40*, 6 (2021). 3, 4

[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. 2, 4

[HGH*22] HU Y., GUERRERO P., HASAN M., RUSHMEIER H., DESCHAINTRE V.: Node graph optimization using differentiable proxies. In *ACM SIGGRAPH 2022 Conference Proceedings* (New York, NY, USA, 2022), SIGGRAPH '22, Association for Computing Machinery. 4

[HHD*22] HU Y., HE C., DESCHAINTRE V., DORSEY J., RUSHMEIER H.: An inverse procedural modeling pipeline for svbrdf maps. *ACM Transactions on Graphics (TOG) 41*, 2 (2022), 1–17. 3, 4, 11

[HHG*22] HU Y., HAŠAN M., GUERRERO P., RUSHMEIER H., DESCHAINTRE V.: Controlling material appearance by examples. *Computer Graphics Forum 41*, 4 (2022), 117–128. 3

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). 7, 12

[KMX*21] KUZNETSOV A., MULLIA K., XU Z., HAŠAN M., RAMAMOORTHI R.: Neumip: Multi-resolution neural materials. *Transactions on Graphics (Proceedings of SIGGRAPH) 40*, 4 (July 2021). 3

[LBAD*06] LAWRENCE J., BEN-ARTZI A., DECORO C., MATUSIK W., PFISTER H., RAMAMOORTHI R., RUSINKIEWICZ S.: Inverse shade trees for non-parametric material representation and editing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (July 2006). 3

[LL11] LEPAGE D., LAWRENCE J.: Material matting. *ACM Trans. Graph. 30*, 6 (dec 2011), 1–10. 3

[LSC18] LI Z., SUNKAVALLI K., CHANDRAKER M.: Materials for masses: SVBRDF acquisition with a single mobile phone image. *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 72–87. 2, 3

[MRR*22] MARTIN R., ROULLIER A., ROUFFET R., KAISER A., BOUBEKEUR T.: Materia: Single image high-resolution material capture in the wild. *Computer Graphics Forum 41*, 2 (2022), 163–177. 3

[Pho75] PHONG B. T.: Illumination for computer generated pictures. *Commun. ACM 18*, 6 (jun 1975), 311–317. 3

[RGJW20] RAINER G., GHOSH A., JAKOB W., WEYRICH T.: Unified neural encoding of btfs. *Computer Graphics Forum 39*, 2 (2020), 167–178. 3

[RJGW19] RAINER G., JAKOB W., GHOSH A., WEYRICH T.: Neural btf compression and interpolation. *Computer Graphics Forum 38*, 2 (2019), 235–244. 3

[RPG21] RODRIGUEZ-PARDO C., GARCES E.: Neural photometry-guided visual attribute transfer. *IEEE Transactions on Visualization and Computer Graphics 29*, 3 (2021), 1818–1830. doi:10.1109/TVCG.2021.3133081. 3, 12

[SLH*20] SHI L., LI B., HAŠAN M., SUNKAVALLI K., BOUBEKEUR T., MECH R., MATUSIK W.: Match: Differentiable material graphs for procedural material capture. *ACM Trans. Graph. 39*, 6 (Dec. 2020), 1–15. 1, 2, 4, 6, 7, 8, 9, 10

[SRRW21] SZTRAJMAN A., RAINER G., RITSCHEL T., WEYRICH T.: Neural BRDF representation and importance sampling. *Computer Graphics Forum 40*, 6 (Sept. 2021), 332–346. 3

[SZ15] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015), Bengio Y., LeCun Y., (Eds.). 6

[UVL18] ULYANOV D., VEDALDI A., LEMPITSKY V.: Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018). 2

[VPS21] VECCHIO G., PALAZZO S., SPAMPINATO C.: Surfacenet: Adversarial svbrdf estimation from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 12840–12848. 3

[WMLT07] WALTER B., MARSCHNER S. R., LI H., TORRANCE K. E.: Microfacet models for refraction through rough surfaces. *EGSR 2007* (2007), 195–206. 3, 6

[YDPG21] YE W., DONG Y., PEERS P., GUO B.: Deep reflectance scanning: Recovering spatially-varying material appearance from a flash-lit video sequence. *Computer Graphics Forum 40*, 6 (2021), 409–427. 3

[ZK21] ZHOU X., KALANTARI N. K.: Adversarial single-image svbrdf estimation with hybrid training. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 315–325. 2, 3, 7, 8, 9