Reflectance Modeling by Neural Texture Synthesis

Miika Aittala
Aalto University

Timo Aila
NVIDIA

Jaakko Lehtinen
Aalto University, NVIDIA

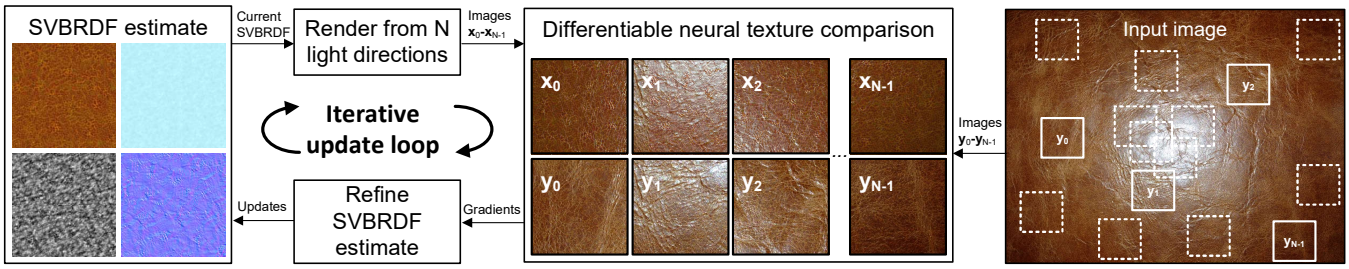


Figure 1: *Our algorithm synthesizes a spatially varying BRDF that closely matches the input flash image when rendered from different lighting directions. The optimization process iteratively updates the current estimate based on neural network-based texture statistics comparisons that are able to ignore the precise pixel arrangement inside image tiles. This snapshot is from an early stage of the optimization.*

Abstract

We extend parametric texture synthesis to capture rich, spatially varying parametric reflectance models from a single image. Our input is a single head-lit flash image of a mostly flat, mostly stationary (textured) surface, and the output is a tile of SVBRDF parameters that reproduce the appearance of the material. No user intervention is required. Our key insight is to make use of a recent, powerful texture descriptor based on deep convolutional neural network statistics for "softly" comparing the model prediction and the examplars without requiring an explicit point-to-point correspondence between them. This is in contrast to traditional reflectance capture that requires pointwise constraints between inputs and outputs under varying viewing and lighting conditions. Seen through this lens, our method is an indirect algorithm for fitting photorealistic SVBRDFs. The problem is severely ill-posed and non-convex. To guide the optimizer towards desirable solutions, we introduce a soft Fourier-domain prior for encouraging spatial stationarity of the reflectance parameters and their correlations, and a complementary preconditioning technique that enables efficient exploration of such solutions by L-BFGS, a standard non-linear numerical optimizer.

Keywords: material appearance, appearance capture, SVBRDF, convolutional neural networks, texture synthesis

Concepts: •**Computing methodologies** → **Reflectance modeling; Texturing;** *Neural networks;*

1 Introduction

Surface reflectance is important for visual realism. In this work, we attempt to capture and reproduce surface appearance by synthesizing a spatially-varying parametric reflectance model based on a single image of a flat, textured surface.

Traditionally, methods for capturing realistic spatially-varying surface reflectance, e.g., [Debevec et al. 2000; Lensch et al. 2003], fit a model to measurements taken under varying viewing and lighting conditions. They either make use of heuristics or rely on non-linear numerical optimization and an explicit, predictive mathematical model of the surface and the capture process to find parameters that yield the best reproduction. Building the model can be challenging, particularly if one wishes to gain efficiency by sharing measurements between similar surface points. To date, single-image capture of materials with rich spatial variation has remained elusive.

On the other hand, texture synthesis algorithms, e.g., [Heeger and Bergen 1995; Efros and Leung 1999], view surface texture as a stochastic process, and aim to produce new realizations that resemble an input exemplar by either copying pixels (non-parametric methods) or matching image statistics (parametric techniques). While these methods can also trivially be used to produce new realizations of existing SVBRDFs, one still needs to somehow author or capture an exemplar that can be given as input to the synthesis.

We present an algorithm that shares properties of both texture synthesis and reflectance capture. Like reflectance capture, our goal is to produce explicit, spatially-varying parametric BRDFs (SVBRDFs) that, when rendered, yield images that match the appearance of the input exemplar. Yet, analogous to texture synthesis, we produce stochastic realizations that do not directly correspond to any existing region in the input image. At heart, our algorithm employs the same core idea as parametric texture synthesis: an optimizer drives the SVBRDF solution towards values that produce renderings whose appearance matches the inputs in a statistical, soft sense. Instead of producing several novel realizations of texture, however, our goal is to summarize the texture seen in the large input image into a small, relightable SVBRDF tile, making use of the fact the input image contains regions with different illumination, cf. Figure 1. To the best of our knowledge, this general approach is novel.

Note that as non-parametric texture synthesis methods are based on copying pixel values from the exemplar onto the solution, they do not help achieve our goal: the exemplar is an RGB photo and the desired solution is a multi-channel SVBRDF map. In contrast, parametric texture synthesis methods work by iteratively modifying the solution until its chosen statistics match those of the exemplar. This suggests inserting a rendering operation — a differentiable function from SVBRDFs to RGB images — into the objective function to drive the search. This is precisely the approach we take. In particular, we build on the recent parametric texture synthesis method of Gatys et al.

[2015b], who introduce a texture descriptor built on the statistics extracted from the activations of a deep convolutional neural network (CNN) originally trained for image classification, and present state-of-the-art results that closely rival those of non-parametric methods.

Concretely, given a single head-lit flash image of a flat, textured (stationary) surface, our method recovers a small tile of SVBRDF parameters that reproduces the appearance of any region of the input when rendered under its illumination. We do this by solving

$$\underset{\mathbf{u}}{\operatorname{argmin}} \sum_k \|T_G(R(\mathbf{u};\mathbf{l}_k)) - T_G(\mathbf{y}_k)\|, \tag{1}$$

where $\mathbf{u}$ are the unknown SVBRDF parameters (a single tile of $n \times n$ pixels, with $n \sim 256$), we loop over different exemplars $\mathbf{y}_k$ with different lighting conditions $\mathbf{l}_k$, R is the rendering operator that relights the SVBRDF in lighting condition $\mathbf{l}_k$, and T_G computes the CNN-based texture descriptor for its input image.

This is a hard optimization problem. Due to their non-convexity, both texture statistics matching and SVBRDF fitting are challenging, even in isolation. Additionally, the fitting is ill-posed in the sense that the same observed pixel values can be explained by many different SVBRDFs. Combining these tasks is likely to exacerbate these issues, and indeed, our optimizer may, in this basic formulation, fall into poor local minima. To encourage balanced convergence towards high-quality solutions and mitigate issues related to ill-posedness, we describe a soft Fourier-domain prior to encourage the individual and joint local statistics of the reflectance parameters to remain constant over the solution, guiding the solver to explain similar observed behavior using similar combinations of the parameters regardless of spatial location. To further aid efficient numerical exploration of such solutions, we introduce a preconditioning scheme that makes spatially stationary descent directions easier to discover.

As we match the rendered output with the input photograph only in a statistical sense, our objective function is much more ambiguous and thus harder to optimize than those of traditional reflectance capture methods. Nevertheless, we find that many of the results capture the overall feel and the fine structure of the inputs well.

In summary, our contributions are

- The observation that soft image comparison based on CNN texture statistics can be robust enough to drive a hard reflectance fit problem with hundreds of thousands of free variables into a meaningful solution.
- A method for fitting SVBRDF model parameters to a single image of a textured head-lit surface without requiring explicit point correspondences between image patches.
- A soft Fourier-domain prior and a jointly designed preconditioner for encouraging stationarity of model parameters.

2 Related Work

2.1 Texture Synthesis

Starting from a random initial guess, the parametric approach to texture synthesis uses local optimization to match the generated image $\mathbf{x}$ and a texture exemplar $\mathbf{y}$ by minimizing their difference in terms of a *texture statistics descriptor* T:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \|T(\mathbf{x}) - T(\mathbf{y})\| \tag{2}$$

Heeger and Bergen [1995] match histograms of filter responses in a steerable pyramid, which works well for stochastic textures but fails with more structured inputs. While their original implementation sequentially enforces a series of statistics, and thus does not properly optimize Equation 2, well-defined stochastic optimization with sliced Wasserstein distance is possible [Rabin et al. 2011]. Portilla and Simoncelli [2000] consider also autocorrelation and cross-correlation statistics. Their results remained state-of-the-art in parametric texture synthesis until the recent CNN-based work by Gatys et al. (cf. Section 2.2 below). However, their method is also limited to sequentially enforcing different statistics, and does not optimize a well-defined distance metric. Random phase noise has long been popular in parametric synthesis (e.g. [Galerne et al. 2010]), but is limited to stochastic (non-structured) textures.

Visually, the most successful texture synthesis methods to date are of the non-parametric kind that copy pixels or blocks of pixels from the exemplar [Efros and Leung 1999; Efros and Freeman 2001; Hertzmann et al. 2001]. However, it is unclear how they could be used to synthesize an entire material representation (albedos, surface normals, specularity, etc) that looks like an example texture patch after relighting, as such information is not directly available in the exemplar and thus cannot be copied. While a non-parametric textural similarity metric can be defined and optimized [Kwatra et al. 2005; Kaspar et al. 2015], the discrete and combinatorial nature of this metric would lead to an unwieldy optimization task when combined with reflectance fitting.

Many texture synthesis methods can be straightforwardly applied to produce more SVBRDF texture from a small readily solved exemplar — for example, Aittala et al. [2015] suggest using Image Quilting [Efros and Freeman 2001] for this purpose. Tong et al. [2002] synthesize Bidirectional Texture Functions on 3D models, given an exemplar. We address the orthogonal problem of solving for the SVBRDF in the first place. Wei et al. [2008] propose a related unconventional use of texture synthesis. Their method reverses the usual procedure by producing a small texture exemplar that summarizes the content of a large (nonstationary) input image. Analogously, we produce a small "SVBRDF summary" of an input flash image. However, there appears to be no obvious way to adapt their algorithm to our goals.

2.2 CNN-Based Texture Statistics Descriptor

Convolutional neural networks (CNN) have recently been used in a wide range of tasks, including image classification [Szegedy et al. 2015] and object localization [Huang et al. 2015], with consistently good results. It is particularly relevant to our approach that the features learned while training for one task can be remarkably useful when used for a completely different task [Razavian et al. 2014].

CNNs consist of a sequence of layers that apply a set of convolution kernels to input from the preceding layer, followed by a non-linear mapping [LeCun et al. 1998]

$$\mathbf{a}_i^l = \sigma(\sum_j \mathbf{a}_j^{l-1} * \mathbf{k}_{ij}^l). \tag{3}$$

Here $*$ denotes convolution, $\mathbf{a}_i^l$ is the scalar 2D image of the ith activation on the lth layer, $\mathbf{k}_{ij}^l$ is the jth 2D convolution kernel associated with output activation i at level l, and σ is typically a rectified linear unit, i.e. $max(0,\cdot)$. The process is bootstrapped by considering the input RGB image as a layer with three activations.

Relevant to our goal, Gatys et al. [2015b] use the pre-trained VGG-19 [Simonyan and Zisserman 2014], a deep CNN that was trained for image classification, and intriguingly, show that ensembles of Gram matrices that describe statistics of network activations on the convolutional layers form an excellent descriptor of texture, and even low-level artistic style [Gatys et al. 2015a]. (Fully connected

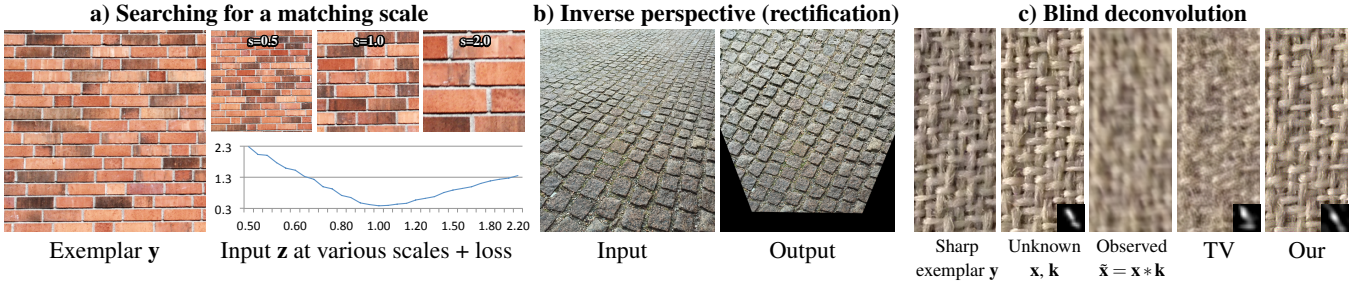


Figure 2: *Toy examples. Using the texture descriptor T_G for a) determining the matching scale between textures, b) rectifying a perspective image, and c) using a second image of the same material as a prior in blind deconvolution instead of a total variation (TV) smoothness term.*

layers are omitted.) For the lth convolutional layer, the entries of the Gram matrix $\mathbf{G}^l$ are given by

$$G^l_{ij} = \text{mean}\{\mathbf{a}^l_i \odot \mathbf{a}^l_j\}, \tag{4}$$

where $\odot$ denotes entrywise product. The descriptor $T_G(\mathbf{x})$ is obtained by vectorizing the Gram matrices and concatenating them together across different layers with per-level weights w^l_G applied:

$$T_G(\mathbf{x}) = \left[w^1_G \,\text{vec}(\mathbf{G}^1),\ w^2_G \,\text{vec}(\mathbf{G}^2),\ \ldots\ w^{\#\text{layers}}_G \,\text{vec}(\mathbf{G}^{\#\text{layers}})\right] \tag{5}$$

Here vec$(\cdot)$ takes in a matrix and reorders its elements into a row vector. (The precise order is irrelevant as long as it is constant.) Comparing two descriptors $T_G(\mathbf{x})$ and $T_G(\mathbf{y})$ happens by simply evaluating a vector norm $\|T_G(\mathbf{x}) - T_G(\mathbf{y})\|$. We use the L_1 norm instead of L_2. This has a small positive effect on our results.

To synthesize a texture using the descriptor, the algorithm starts from random noise. The current image $\mathbf{x}$ is iteratively fed to the CNN, the descriptor $T_G(\mathbf{x})$ evaluated, its difference to the exemplar's descriptor $T_G(\mathbf{y})$ computed, and finally the image is updated so that the difference is reduced. Since T_G is (piecewise) differentiable, the gradient $\partial\|T_G(\mathbf{x}) - T_G(\mathbf{y})\|/\partial\mathbf{x}$ can be computed efficiently using backpropagation and used for numerical optimization. Gatys et al. demonstrate state-of-the-art results in parametric texture synthesis. We base our synthesis algorithm on theirs.

2.3 Reflectance Capture

Weyrich et al. [2009] provide a comprehensive overview of reflectance capture from radiometric measurements. Even assuming pure surface reflection (no subsurface scattering), a dense sampling over the entire 6D space of surface position, lighting direction, and view direction is cumbersome. Hence, various assumptions about the underlying reflectance are relied on to reduce data volume, complexity of the capture process, and computational requirements. For instance, several techniques exploit homogeneity or "texturedness" where many surface points share a common appearance [Lensch et al. 2003; Zickler et al. 2006; Wang et al. 2008; Dong et al. 2010; Aittala et al. 2015]. A major challenge in these methods is to explicitly identify which points share apperance. Our key contribution is a method that similarly relies on repetition to share information between surface points, but does not require explicit correspondences.

Reflectance can be extracted from a single image aided by strong assumptions — for instance diffuse only, e.g., [Barron and Malik 2015] or spatial homogeneity and known geometry [Boivin and Gagalowicz 2001]. While we also make use of a strong assumption (texturedness), we aim to support more general reflectance with per-pixel variations in glossiness, albedos, and normals. Another class of methods relies on user input for interactive modeling, e.g., [Clark 2010; Dong et al. 2011]. We aim to perform a full model fit without user intervention.

Tools from texture synthesis have only rarely been applied in reflectance capture. Wang et al. [2011] capture the surface normal statistics from perfectly specular surfaces from a single image under step illumination. The range of spatial structures and materials that lead to faithful synthesis results is narrow compared to our full SVBRDF models. Ngan and Durand [2005] model Bidirectional Texture Functions of stochastic materials by the steerable pyramid statistics computed from images of samples taken under different viewing and lighting conditions. They relight by forcing the measured statistics onto a "master" image using the Heeger-Bergen algorithm. While their method is purely data-driven and does not result in an explicit SVBRDF map, it is an important inspiration to us, as they make use of image statistics matching for reflectance modeling. Aittala et al. [2015] also employ Heeger-Bergen as part of their textured SVBRDF capture pipeline to massage intermediate fitting results to look more like the input images. Unlike us, they do not make use of statistics during the model fit, but instead rely on explicit point correspondences estimated from a guide image. Texture synthesis has also been applied as a post-process for transferring captured facial reflectance [Weyrich et al. 2006] and enhancing its microgeometry [Graham et al. 2013].

3 Model Fitting by Neural Texture Statistics

Traditional parametric texture synthesis (Equation 2) directly alters the colors of pixels in the result $\mathbf{x}$ so that its texture descriptor $T(\mathbf{x})$ would match that of the exemplar $T(\mathbf{y})$. As we desire to reproduce appearance by an analytic reflectance model, our unknowns are per-pixel reflectance parameters that cannot be directly optimized in the same framework: comparison between reflectance parameters and exemplar images is meaningless.

To enable optimizing for model parameters instead, we introduce a mapping $f(\mathbf{p})$ that, given parameters $\mathbf{p}$, produces an image $\mathbf{x}$, i.e., $\mathbf{x} = f(\mathbf{p})$. The question now becomes how to choose the model parameters $\mathbf{p}$ such that the descriptors match between $f(\mathbf{p})$ and $\mathbf{y}$:

$$\underset{\mathbf{p}}{\text{argmin}}\ \|T(f(\mathbf{p})) - T(\mathbf{y})\|. \tag{6}$$

This forms the core of our approach.

Before describing our main application, we begin with a few toy examples of model fitting using CNN-based texture statistics. We strongly emphasize that these examples do not constitute attempts to advance the state of the art in any of the associated problems.

Toy Example 1: Find texture scale (1D) Our simplest example is a 1D optimization for estimating scale (zoom level) of a given texture, given a reference realization. More precisely, we are given

an exemplar $\mathbf{y}$ of a texture, and another realization $\mathbf{z}$ of the same texture imaged at a different (unknown) scale. The problem is to find the scale s where the zoomed input matches the exemplar: $\operatorname{argmin}_s \|T_G(\text{scale}(\mathbf{z},s)) - T_G(\mathbf{y})\|$.

Figure 2a plots the descriptor difference over scale s for two different crops of the same texture, leading to the correct minimum around 1.0. (It is typical that the loss function does not reach zero.) We observe similar behavior over a range of different input textures, and conclude that the descriptor T_G robustly identifies scale for highly similar textures. Comparing visibly different exemplars, e.g., brick wall of different composition or color, does not yield robust results.

Toy Example 2: Texture rectification (3D) A slightly more complex toy example is the automatic rectification (inverse warping) of a texture under perspective, a simple form of a shape-from-texture problem [Super and Bovik 1995]. Under the correct inverse warp, the local statistics of a homogeneous texture should be similar in all suitably large neighborhoods. Using a three-parameter model for perspective distortion (f for field of view, θ_x for pitch, θ_y for yaw) of a plane through the origin, and assuming the input image $\mathbf{x}$ is generated by the mapping $\text{warp}(f,\theta_x,\theta_y;\mathbf{x})$, we solve for

$$\operatorname*{argmin}_{f,\theta_x,\theta_y} \sum_{i=1}^{N}\sum_{j=1}^{N} \|T_G(E(\text{warp}^{-1}(f,\theta_x,\theta_y;\mathbf{x}),i)) - T_G(E(\text{warp}^{-1}(f,\theta_x,\theta_y;\mathbf{x}),j))\| \quad (7)$$

where $\text{warp}^{-1}(f,\theta_x,\theta_y;\mathbf{x})$ performs the inverse warp of the input image $\mathbf{x}$, and $E(\cdot,i)$ extracts the ith subregion from the image. Figure 2b shows a result using a real (non-synthetic) input image using $N=4$ regions. We find that texture similarity across regions robustly identifies a perspective transform that results in visually homogeneous inverses. Note that unlike most shape-from-texture algorithms, we do not attempt to reason about the effect of perspective projection on the statistics and rely on brute force optimization instead.

Toy Example 3: Texture prior for blind deconvolution Given a single blurred image $\tilde{\mathbf{x}}$, blind deconvolution methods aim to simultaneously recover a sharp image $\mathbf{x}$ and the blur kernel $\mathbf{k}$ most likely to have produced the observed image through convolution: $\tilde{\mathbf{x}} = \mathbf{x} * \mathbf{k}$ (the latter two are unknown). As the problem is severely ill-posed, priors such as sparse derivatives or total variation are used for picking out the solution that most resembles a natural image.

In our final, most complex example, we assume that in addition to a blurred image of a textured surface, we assume a *different* sharp exemplar $\mathbf{y}$ of the same texture is given as input to the algorithm. We proceed to simultaneously minimize

$$\operatorname*{argmin}_{\mathbf{x},\mathbf{k}} \|\mathbf{x}*\mathbf{k} - \tilde{\mathbf{x}}\| + \lambda \|T_G(\mathbf{x}) - T_G(\mathbf{y})\| \quad (8)$$

where the first term encourages the convolved result to match the input, the second term encourages the result to be similar to the sharp input exemplar $\mathbf{y}$ in terms of the texture descriptor, and $\lambda = 0.01$ is a mixing weight. Figure 2c shows an example. Note how the true solution $\mathbf{x}$ and exemplar $\mathbf{y}$ are quite different pixel to pixel; yet the texture prior is powerful enough to drive this optimization problem with $\sim$ 200K variables to a much more reasonable solution than the standard total variation (TV) prior. We performed the optimization using L-BFGS. Again, we stress that the comparison to TV is not fair as we use more information, and fully acknowledge that this method is not applicable to general images.

In summary, we have observed that texture descriptor T_G can be used for guiding nontrivial optimization problems. We will now tackle the full SVBRDF recovery problem.

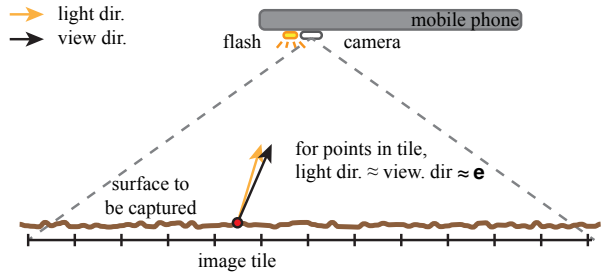


Figure 3: *Imaging setup. Image courtesy of Aittala et al. [2015]*

4 Method

Given a head-on flash image of a mostly flat textured surface, our aim is to find an SVBRDF that, when rendered, produces similar photorealistic texture as that found in the input. We approach this by representing the unknown material as a *single* tile of SVBRDF parameters, and comparing its renderings to several *different* patches of the input image in terms of how well they match in terms of a texture descriptor. Note that the images cannot be directly compared pointwise using e.g. squared pixel differences — as would be done in standard reflectance capture — because the textural features are arranged differently in each input tile.

Input data The input image is a fronto-parallel image of a mostly flat surface under headlight illumination (i.e., the light source is very close to camera). The setup, shown in Figure 3, is similar to that used by Aittala et al. [2015]. The shooting geometry is invariant to distance, up to an unknown and irrelevant global scale. Due to headlight illumination, the lighting and viewing directions are equal at each individual pixel (= a backscatter measurement), but vary between pixels. At each pixel, the lighting conditions are determined by the half-vector $\mathbf{e} \in \mathbb{R}^3$ and an irradiance value $\mathbf{I} \in \mathbb{R}^3$. They are simple to estimate based on the position of the highlight center in the image (computed as the centroid of the positions of the 5% brightest pixels), the known field-of-view of the camera, and known angular distribution of flash emission (cf. Appendix).

Output data The unknown SVBRDF is a multi-channel image tile $\mathbf{u}$ of size $n \times n$, with n around one tenth of the input image width. At each pixel j, the vector of unknowns $\mathbf{u}^j$ consists of the diffuse albedo $\rho_d \in \mathbb{R}^3$, specular albedo $\rho_s \in \mathbb{R}^3$, glossiness $\alpha \in \mathbb{R}$, and surface normal $\mathbf{n} \in \mathbb{R}^3$. We denote the rendered result in the given lighting conditions by $R(\mathbf{u};\mathbf{e},\mathbf{I})$. To render, we use the Blinn reflectance model:

$$R(\mathbf{u};\mathbf{e},\mathbf{I}) = \mathbf{I}\max(0,\mathbf{e}\cdot\mathbf{n})\,(\rho_d + \rho_s \max(0,\mathbf{e}\cdot\mathbf{n})^{\alpha}). \quad (9)$$

4.1 The Loss Function

Concretely, we seek an SVBRDF $\mathbf{u}$ of size $n \times n$ that solves

$$\operatorname*{argmin}_{\mathbf{u}} \sum_k \| T_G(R(\mathbf{u};\mathbf{e}_k,\mathbf{I}_k)) - T_G(\mathbf{y}_k)\|. \quad (10)$$

Here $\mathbf{e}_k$ and $\mathbf{I}_k$ are the half-vectors and irradiances at each pixel in the kth tile. To get a sampling of differently lit pieces of the target material, we choose a set of 15 tiles $\mathbf{y}_k$ of the same size $n \times n$ from the input image $\mathbf{y}$ around the highlight center (cf. Figure 1).

Problem properties As pointed out in Section 1, both the reflectance fitting and texture statistics matching are hard, non-convex problems. Thus it can be expected that the combined problem is

Non-stationary texture | Stationary texture

Figure 4: *These two textures are equally close to the exemplar in terms of the descriptor T_G: the CNN statistics are not strong enough to enforce stationarity of synthesis results.*

at least as hard. A particularly interesting issue that shows up immediately is stationarity, by which we mean spatial homogeneity, "looking the same" across the image (we will be more precise later). In our experience, given a stationary initial guess, e.g. white noise, the method of Gatys et al. [2015b] produces a stationary result. However, there is nothing in their formulation that explicitly enforces this. Figure 4a demonstrates that non-stationarity is not captured by the position-oblivious Gram statistics (Equation 4): while the left image has large-scale contrast variation, its descriptor distance to the exemplar is the same as that of the right, stationary image.

Given the above, it is to be expected that any non-stationarity introduced in the course of optimization will remain as it is not penalized in any way. This is indeed what we observe if we attempt to directly minimize Equation 10: even through the initial guesses are stationary, non-stationarity is immediately introduced to the SVBRDF parameter maps, and hence the renderings, by the badly conditioned reflectance fit that tends to greedily favor different explanations for similar observed reflectance in different parts of the tile.

We address this by introducing a prior that encourages the SVBRDF parameters to be individually stationary (have the same local mean and same local variance everywhere), and jointly stationary (have the same local correlation everywhere). This prior is quite loose: it does not encourage particular values for particular variables or penalize non-smoothness. It merely says "above some scale, look the same everywhere; and when you conspire with another variable, do it in the same way everywhere". Details are given in Section 4.3. Figure 5 show an example of the tempering effect of the priors.

In addition to the stationarity prior, we find that a frequency-domain preconditioner that weights spatial frequencies by weights derived from the input image tiles $\mathbf{y}_k$ helps the optimizer find good descent directions that leads to reasonably uniform convergence across the image. In practice, this means we internally represent the SVBRDF components as the conjugate symmetric parts of their Fourier transforms. Full details are given in Section 4.4, along with other choices of parameterization. Figure 5 demonstrates its effect on convergence.

Full loss function All together, our entire problem is

$$\operatorname*{argmin}_{\tilde{\mathbf{u}}} \sum_k \| T_G(R(P(\tilde{\mathbf{u}}); \mathbf{e}_k, \mathbf{I}_k)) - T_G(\mathbf{y}_k) \| + \lambda Q(P(\tilde{\mathbf{u}})), \quad (11)$$

where $P(\tilde{\mathbf{u}})$ is the preconditioner that evaluates the primal-domain SVBRDF parameters $\mathbf{u}$ (and other terms required by the prior) given the raw optimization variables $\tilde{\mathbf{u}}$, and $Q(\cdot)$ is the prior (Section 4.3). The weight λ is set to the number of tiles multiplied by 0.001.

Finally, to enable gradient-aware local optimization, we need the derivative of the loss function w.r.t. the internal parameters. This is easily obtained from the chain rule:

$$\frac{\partial \text{loss}}{\partial \tilde{\mathbf{u}}} = \left[\sum_k \frac{\partial \| T_G(\mathbf{x}) - T_G(\mathbf{y}) \|}{\partial \mathbf{x}} \overset{=R(\mathbf{u})}{\frac{\partial \mathbf{x}}{\partial \mathbf{u}}} \overset{=P(\tilde{\mathbf{u}})}{\frac{\partial \mathbf{u}}{\partial \tilde{\mathbf{u}}}} \right] + \lambda \frac{\partial Q}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \tilde{\mathbf{u}}}. \quad (12)$$

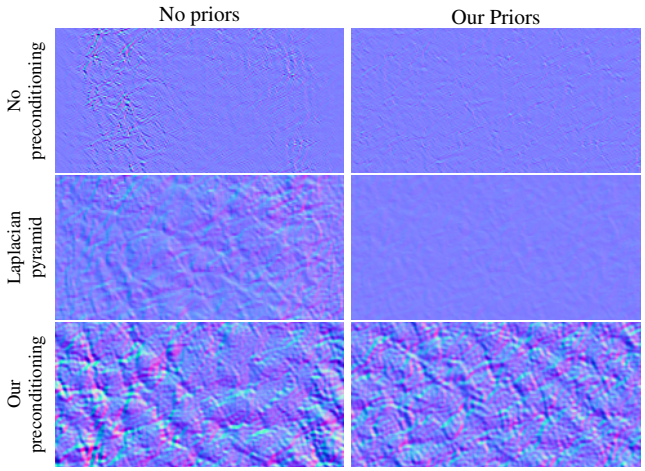


Figure 5: *The effect of priors and preconditioning, visualized by the effect they have on the solution normal map of the* leather_amber *dataset after 100 L-BFGS iterations.* ***The columns:*** *Solutions computed without and with the stationarity priors.* ***The rows:*** *Solutions computed using no preconditioning, a Laplacian pyramid preconditioner, and our preconditioner. Priors improve stationarity for all preconditioners. Without preconditioning, the solution has not progressed very far from the initial guess. Laplacian pyramid preconditioning helps somewhat, and our preconditioning gives a much more detailed and believable result. This is not surprising, as our priors and preconditioner are designed jointly. On the last row, note how the prior helps remove the non-stationarity from the otherwise plausible solution.*

Gatys et al. [2015b] describe how to compute the gradient $\partial \| T_G(\mathbf{x}) - T_G(\mathbf{y}) \| / \partial \mathbf{x}$ using the standard backpropagation machinery from convolutional neural networks. We build on the same machinery to evaluate our additional terms: the derivative of the rendered result w.r.t. the SVBRDF parameters $\partial R / \partial \mathbf{u}$, the prior $\partial Q / \partial \mathbf{u}$, and the preconditioner $\partial P / \partial \tilde{\mathbf{u}}$. Note that explicit full Jacobians are never required by L-BFGS, and would be infeasible to compute; we only ever need the gradient of the scalar loss function. Additional details on the optimization are given in Appendix A.

Figure 6 visualizes the entire computation graph used for evaluating the objective, the prior, and the preconditioner. The reader may want to refer to it during the next sections, when we describe the individual components.

4.2 Texture Descriptor Computation

We first pre-process both the input photo and the corresponding rendered tiles into a format suitable for the CNN. The VGG network was trained with natural images in the range $[0, 255]$, with mean subtracted. Hence, we expect the descriptor to work best with similarly formatted inputs. Furthermore, transforming each tile to a similar range of pixel values results in more balanced convergence, as each tile will produce objective values in a similar range regardless of its intensity and contrast.

With this in mind, we first clamp the rendered values to 1 to emulate saturation of the camera pixels. We then normalize the images according to the local statistics of the input images before evaluating the descriptor T_G. See Figure 7 for an illustration. We first estimate the local mean and standard deviation of the input image by computing $\mathbf{y}^{\text{mean}} = \text{blur}(\mathbf{y})$ and $\mathbf{y}^{\sigma} = \sqrt{\text{blur}((\mathbf{y} - \mathbf{y}^{\text{mean}})^2)}$. To normalize an image $\mathbf{x}$ of the kth tile, be it either a rendering or the original $\mathbf{y}_k$, we first subtract the corresponding tile of the strongly blurred

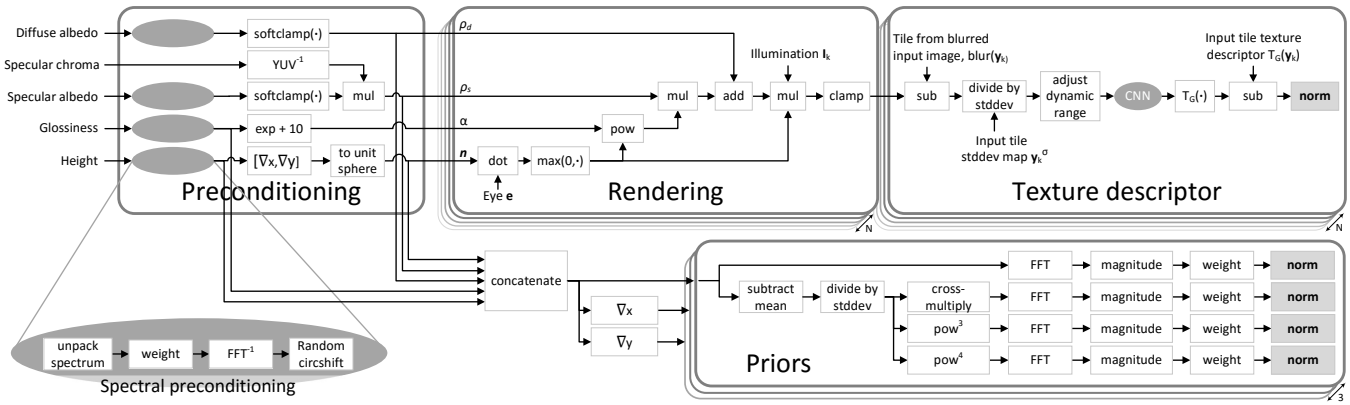
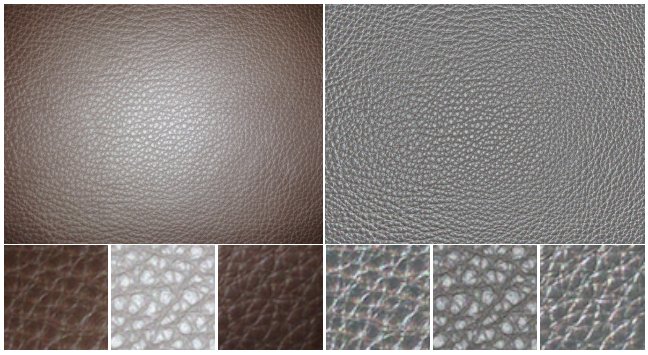


Figure 6: *The detailed computation graph used for evaluating the preconditioner, priors, the loss function, as well as its gradient. The computation is shown as a directed acyclic graph, where each node performs a single operation for which an explicit gradient computation is simple. The forward evaluation follows the arrows, whereas the matching gradient backpropagation happens in the opposite direction along the same arrows. Rendering and Texture statistics are executed for each N light directions, Preconditioning only once, and Priors three times. The total loss is obtained by summing the bolded "norm" boxes from all instances of texture statistics and priors.* $\text{softclamp}(x) = \frac{1}{2}(x + (x^2 + \varepsilon)^{0.5})$.



Input image $\mathbf{y}$ After normalization $(\mathbf{y} - \mathbf{y}^{\text{mean}})/\mathbf{y}^{\sigma}$

Figure 7: *The input image has significant lighting variation, which is greatly reduced by local mean and contrast normalization (Section 4.2). The bottom row shows closeups from top-left corner, center, and bottom-right corner. After normalization, the regions look almost the same except for the lighting direction.*

input image and pointwise divide by the local standard deviation: $(\mathbf{x} - \mathbf{y}_k^{\text{mean}})/(\mathbf{y}_k^{\sigma} + 10^{-4})$. Finally, we multiply all RGB values by 255 to match VGG's expected dynamic range, and further by 0.2 to match the typical range in photos it was trained with.

Note that this transformation does not bias the expected solution because it is always performed using the statistics of the input tile. Hence, the solution is still required to reproduce the colors, intensity levels, and contrast of the original untransformed input. The pre-processing helps the optimizer and the similarity metric by improving the conditioning of the problem.

After pre-processing, the image is fed to the neural network for statistics computation. We use the first 17 convolutional layers of a pre-trained VGG-19 network [Simonyan and Zisserman 2014], with average pooling instead of max pooling [Gatys et al. 2015b]. The fully connected layers are omitted. The layer weights w_G^l (Equation 5) are set to the number of output features of the respective layer l, which gives more weight to the higher layers (e.g. $w_G^1 = 64$, $w_G^3 = 128$, and w_G^{17}=512). This weighting scheme was observed to slightly but consistently improve the results over uniform weights.

4.3 Stationarity Priors

To prevent the situation where different parts of the SVBDRF end up with radically different solutions, we explicitly require the reflectance parameters to be loosely stationary throughout the tile. By stationarity, we mean spatial homogeneity of local statistics: measured over any sufficiently large region, the mean, (co-)variance, skewness and kurtosis of the parameters should remain constant. Our prior explicitly enforces this. Figure 8 illustrates the effects of constant vs. varying statistics over a texture. Figure 8a,e,g show stationary textures whose local mean and variance are constant (skewness and kurtosis not shown). Compare this to Figure 8b,c where respectively local mean and contrast vary slowly.

Our prior does not constrain the local statistics of the variables, e.g. by encouraging or discouraging the variables to be correlated. It merely states that if certain statistical features are present in a given region, they should be present in others as well. This may be contrasted with classical smoothness priors, which require similarity between neighboring points, but are blind to the global picture.

4.3.1 Mean Value Prior

Let us first consider the mean. The prior should discourage behavior where parts of the image are significantly brighter than others, i.e., the local average changes slowly over the image. Texture A_1 in Figure 8b shows an example.

Convolving the image with a low-pass filter measures local average. If the result is a constant, any local average variations are smaller than the size of the kernel. Conceptually, our prior evaluates this convolution and penalizes any variation left. In practice, the prior is efficiently implemented in frequency domain. As the low-pass convolution is evaluated by weighting down all but low frequencies in the Discrete Fourier Transform (DFT) of the parameter maps, it is clear that a varying local average is directly visible in those low (non-DC) frequencies. For example, the bottom row of Figure 8b shows energy inside the red circle. Concretely, we add to the objective function

$$P_1(\mathbf{u}) = w_1 \|\mathbf{w}_f \odot \mathcal{F}\{\mathbf{u}\}\|^2 \tag{13}$$

where $w_1 = 3 * 10^{-2}$ is the overall weight of the prior, $\mathcal{F}$ computes the Fast Fourier Transform (FFT), $\mathbf{w}_f$ is a frequency-wise weighting and $\odot$ denotes pointwise product. See Figure 6 for the DAG that

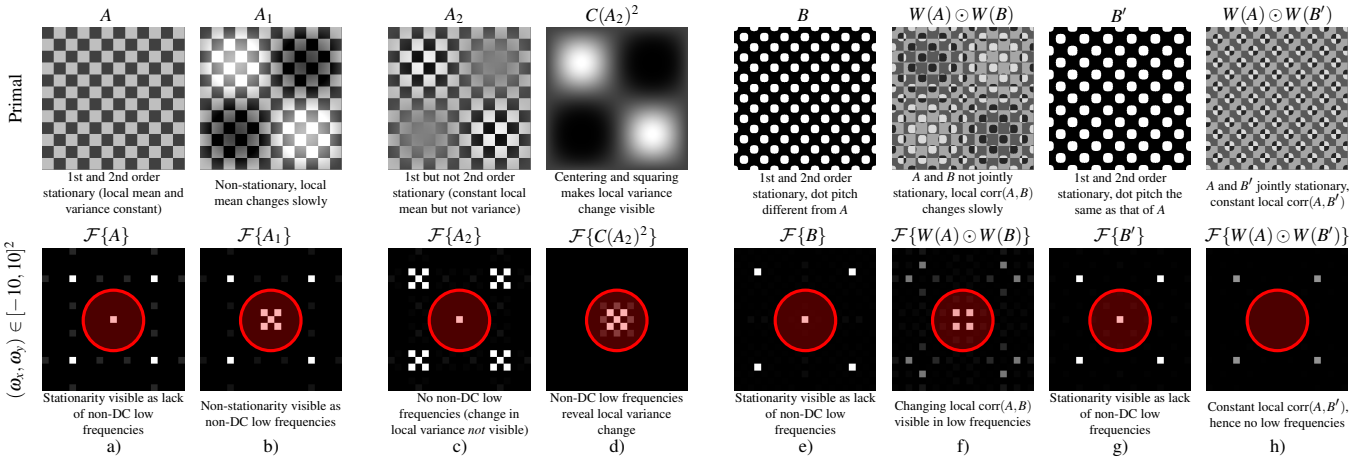


Figure 8: *Illustration of constant vs. varying local statistics (cf. Sections 4.3.1 and 4.3.2)*

implements this (and later) priors. For $\mathbf{w}_f$ we use the magnitude spectrum of a normalized Gaussian with standard deviation $1/6$th of the tile size, with the DC component zeroed. Ideally the size of this filter would be proportional to the size of the textural features in the input. However, we found this fixed value to work reasonably well.

4.3.2 Local Contrast, Correlation, and Higher Orders

The mean value prior eliminates the most obvious non-stationarities, but leaves room for significant higher order variation: in particular, while it forbids regions to have different means, it still allows them to have arbitrarily different variances and covariances. We apply similar reasoning as above to discourage such differences.

As variance is the second central moment, we begin by subtracting the mean of each parameter map from itself. The local variance is then measured by raising the centered maps to second power, and locally averaging by convolution with a low-pass filter. Again, any remaining spatial variation is visible as energy in the non-DC low-frequencies of the Fourier transform, which is easily penalized by a weighted norm on the spectrum like above:

$$P_2(\mathbf{u}) = w_2 \| \mathbf{w}_f \odot \mathcal{F}\left[C(\mathbf{u})^2 \right] \|^2 \tag{14}$$

where $w_2 = 3 * 10^{-5}$ and $C(\mathbf{u}) = \mathbf{u} - \text{mean}(\mathbf{u})$ centers each of the maps separately.

We further discourage nonstationarity by penalizing variations in covariances between variables. To make the prior robust to the scale of each individual variable (which also varies significantly across materials), we further whiten each centered map by division with its standard deviation; hence, we are in fact penalizing variations in the normalized correlation. We then take the pointwise product of each pair of whitened images and penalize their variations, leading to

$$P_{\text{corr}}(\mathbf{u}) = w_{\text{corr}} \sum_j \sum_{j'>j} \| \mathbf{w}_f \odot \mathcal{F}\left\{ W(\mathbf{u}_j) \odot W(\mathbf{u}_{j'}) \right\} \|^2 \tag{15}$$

where $w_{\text{corr}} = 3 * 10^{-5}$, j and j' loop over all variables, $\mathbf{u}_j$ is the 2D image of the jth variable, and $W(\mathbf{x}) = (\mathbf{x} - \text{mean}(\mathbf{x}))/(\sqrt{\text{var}(\mathbf{x})} + 10^{-4})$ is the whitening transform. Since normal vectors are derived from height, we set a zero weight for their correlation to avoid numerical issues.

Illustrative examples Let us illustrate how the above captures second-order variations. For instance, an image with constant local average may have low contrast in some regions, and high contrast in others. Cf. texture A_2 in Figure 8c; note that there are no low frequencies apart from the DC in the magnitude spectrum $\mathcal{F}\{A_2\}$, i.e., the changing local variance (contrast) is *not* visible there. It does, however, show up as the low-frequency energy in the spectrum $\mathcal{F}\{C(A_2)^2\}$ of the centered and squared signal (Figure 8d).

Furthermore, even if two variables are both first and second order stationary, their local correlation may vary across the image. For example, textures B and B' in Figure 8e,g both have unchanging local mean and variance (contrast), but the local correlation between A and B varies as the dot pitch in B is different from the checker pitch in A. This manifests itself as low-frequency energy in the spectrum of the pointwise product $W(A) \odot W(B)$ (Figure 8f). Contrast this to Figure 8g: the equal dot and checker pitches in A and B' mean in the local correlation is unchanging, and hence the spectrum contains no low frequencies (Figure 8h).

Higher-order moments By the same reasoning, we also penalize global variations in third and fourth moments of the whitened parameters (i.e. skewness and kurtosis), by formulas

$$P_3(\mathbf{u}) = w_3 \| \mathbf{w}_f \odot \mathcal{F}\left[W(\mathbf{u})^3 \right] \|^2, \tag{16}$$

$$P_4(\mathbf{u}) = w_4 \| \mathbf{w}_f \odot \mathcal{F}\left[W(\mathbf{u})^4 \right] \|^2 \tag{17}$$

where $w_3 = 3 * 10^{-6}$ and $w_4 = 3 * 10^{-7}$. We find that these terms generally help, although are not as important as the first and second order priors.

Gradient statistics In addition to applying the priors to SVBRDF parameter maps, we apply the same priors to their finite difference gradients. This imposes additional stationarity constraints related to the spatial size of features: while the scaling of an image does not change the magnitude of pixel values, it does change the magnitude of gradients. We exclude the height from the gradient prior because its gradient (normals) has already been taken into account. Like skewness and kurtosis, we find that applying the prior also to gradients yields a small but consistent improvement.

4.4 Internal Parameterization (Preconditioning)

This subsection describes our internal reflectance parameterization designed make the loss function friendly to numerical optimization. This preconditioning changes the loss function landscape, but not

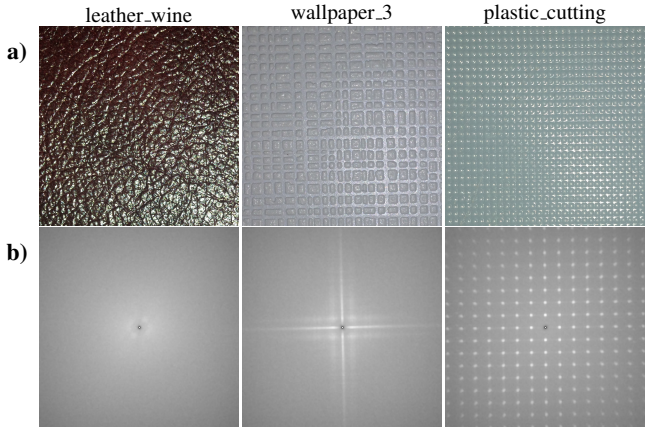


Figure 9: *Frequency weights for preconditioning.* ***a)*** *Representative tiles from input images.* ***b)*** *Average magnitude spectra* $\mathbf{w}_g$ *taken over 300 tiles from input images. Together with the prior weights, these form the per-frequency weights* $\mathbf{w}_s$ *used in preconditioning.*

the optimal solutions. Our preconditioner consists of two parts. First, we internally store all SVBRDF components as conjugate symmetric Fourier spectra; second, after transformation into the spatial domain, additional non-linearities are applied to yield final glossiness values, surface normals and specular albedos.

4.4.1 SVBRDF Components

The glossiness parameter α appears as an exponent in Equation 9, and its perceptual effect is highly non-linear. We optimize it in the better-suited logarithmic space, and exponentiate before feeding into the renderer. Furthermore, we limit the glossiness value to ≥ 10 from below in order to prevent it from taking such low values that it might be confused with the diffuse component.

We parameterize the surface normal field $\mathbf{n}$ as a per-pixel height map $\mathbf{h}$. We convert the height map to a unit normal vector field by first convolving with finite difference kernels to derive $\tilde{\mathbf{n}} = [\nabla\mathbf{x}\ \nabla\mathbf{y}\ 1]$ and then normalizing it to the unit sphere $\mathbf{n} = \tilde{\mathbf{n}}/|\tilde{\mathbf{n}}|$. We use the 2×2 kernel $\frac{1}{2}[-1\ 1\ ;\ -1\ 1]$ (and its transpose for y-direction) to reduce numerical artifacts related to one-sided derivatives.

We also constrain the specular chromaticity to be constant across the solution. This is a reasonable assumption for most materials, and significantly reduces the dimensionality and ambiguity of the problem. We reduce the specular albedo ρ_s to a scalar field, and introduce two global scalar variables, $\rho_{s,U}$ and $\rho_{s,V}$ that represent the two YUV chromaticity components. At each pixel, the full RGB specular albedo is computed as $\rho_s^i \mathbf{Y}^{-1}[1\ \rho_{s,U}\ \rho_{s,V}]^{\mathrm{T}}$, where $\mathbf{Y}$ is the RGB-to-YUV conversion matrix.

4.4.2 Spectral Preconditioning

Because our stationarity prior induces strong dependencies between variables, we find that optimizing the reflectance variables (with the above remappings applied) does not yield reasonable local minima. Values for an individual pixel cannot move much by themselves, because they would soon incur a significant non-stationarity penalty. Indeed, any given update should either be made to all image regions simultaneously, if beneficial, or not at all. In other words, the updates themselves should be stationary.

Transforming the optimization variables into the Fourier domain helps significantly. In this space, each variable controls the magnitude of a single plane wave in a given parameter image. Above some frequency, the plane waves are all stationary by our definition, and thus give rise to descent directions that are in tune with the prior. To enforce real-valued results, we use only the conjugate symmetric part of the Fourier transform as the raw parameterization. To strongly discourage non-stationary steps, we weigh down the lowest frequencies as described below.

To further encourage the optimizer to primarily consider steps that would produce similar frequency content as present in the input, we further scale the frequencies with weights derived from the input photograph. Before starting the optimization, we pick a few hundred random tiles from the input, whiten them, compute their average windowed FFT magnitude spectrum, and use it as a per-frequency weight $\mathbf{w}_g$.

The full per-frequency weighting $\mathbf{w}_s$ is obtained as $(1-\mathbf{w}_f)\odot\mathbf{w}_g$, which is normalized to sum to 1, with DC then set to 0.001. Here, $\mathbf{w}_f$ is the spectral weighting from the priors. Some examples are shown in Figure 9. The entire spectral preconditioning transformation is shown in Figure 6 and given by the formula

$$\mathbf{u} = w_p \mathcal{F}^{-1}\left[\mathbf{w}_s \odot U(\tilde{\mathbf{u}})\right], \tag{18}$$

where $\tilde{\mathbf{u}}$ are the raw optimization variables, and $\mathbf{u}$ are the transformed primal variables fed into the renderer (gloss, specular albedo, and surface normal additionally undergo the transformations described in 4.4.1). U is the linear transform that converts the conjugate-symmetric part $\tilde{\mathbf{u}}$ into an entire complex-valued spectrum so that the inverse FFT produces a real-valued image. The leading multiplier $w_p = 1000n^2$ balances the weight of the per-point variables against the global specular chroma variables. n is the tile pixel dimension.

Because the height map does not directly cause frequency content in the rendering — the normals do — we additionally multiply its weighting $\mathbf{w}_s$ by $150(|\omega|+3)^{-1}$ (biased inverse magnitude of the frequency). The idea is that the height-to-normals differentiation (essentially multiplication by ω_x and ω_y in Fourier domain) approximately cancels this extra weighting, so that the intended effect of the preconditioning carries over to the normals. We further multiply the weight of the glossiness variables by 4, as otherwise they tend to change slowly.

We note a similarity to Barron and Malik [2015] who optimize weighted Laplacian pyramid coefficients as opposed to direct pixel values, with dramatically improved results. Refer to Figure 5 for a comparison of different preconditioners.

4.4.3 Periodicity

We aim for a seamlessly tileable result, both because it is obviously useful in practice, but also because it prevents artifacts from forming at the edges. We remove any special significance from the edges by using periodic boundary conditions in the height-to-normals convolution and the gradient stationarity priors. The various Fourier transforms already use periodic boundaries intrinsically.

To eliminate boundary effects from the statistics comparisons, we remove the zero-padding from the convolutions in VGG evaluation, instead letting it shrink the activation maps on each convolution. This leads to spatially uneven results, as the statistics of the shrunken activation maps contribute little to the outer edges of the tile. We eliminate this effect by randomly circularly shifting the parameter maps before feeding them to the renderer, by a per-tile amount that stays constant throughout the optimization. The effect is that the contributions from the statistics will evenly land to multiple locations within the periodic tile, instead of the center only.

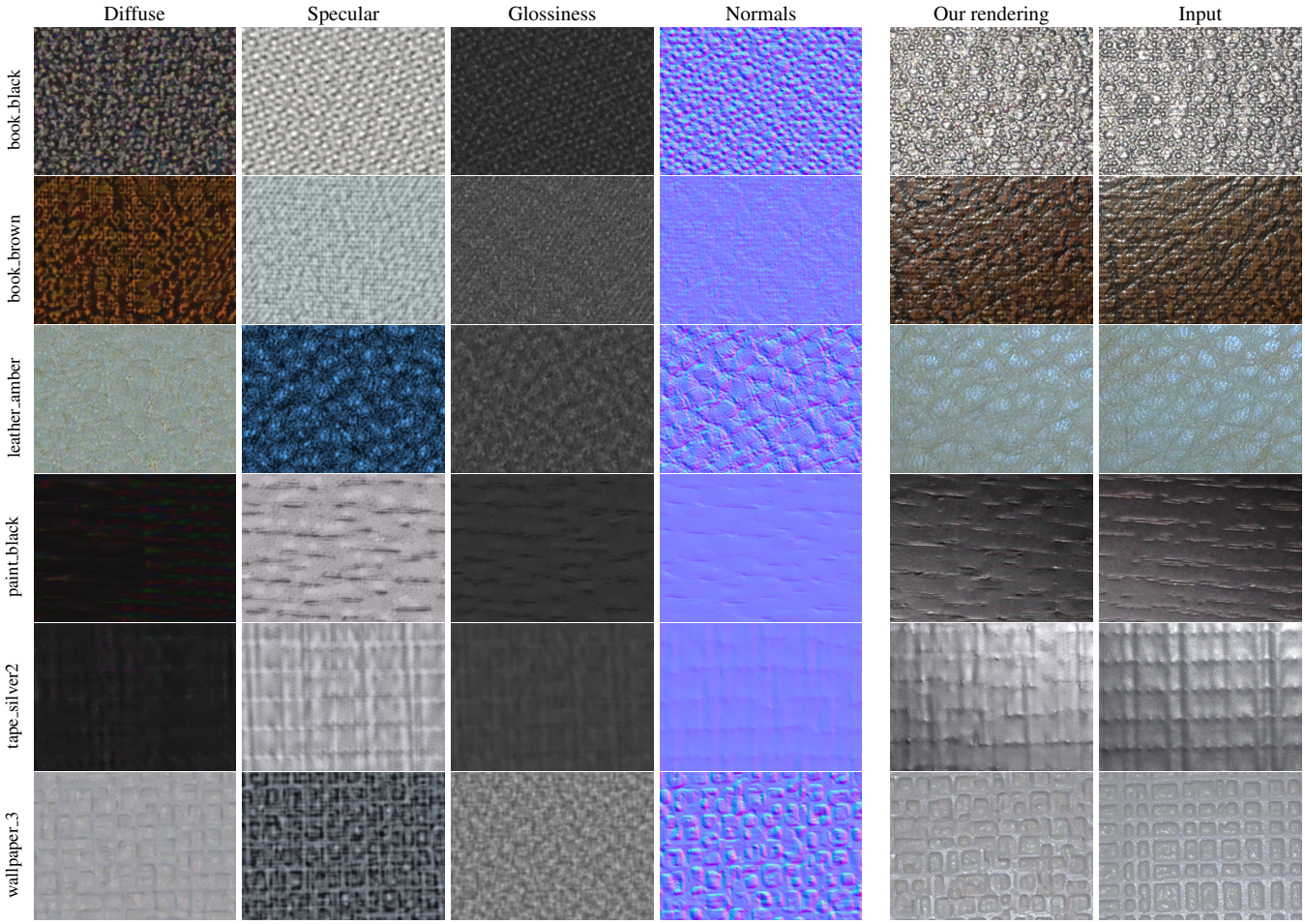


Figure 10: *Diffuse, specular, glossiness and normal maps, and comparison of re-renderings and input for select datasets (cropped vertically).*

5 Results and Discussion

5.1 Implementation and Input Data

We have implemented our algorithm in Matlab using the MatConvNet package [Vedaldi and Lenc 2014]. Our solver takes approx. 2 hours per dataset on a single NVIDIA Quadro K6000 GPU using 1200 L-BFGS iterations with 15 tiles, 256x256 pixels each.

We evaluate our method on the freely-available 72-material iPhone 5 flash-no-flash dataset[1] from Aittala et al. [2015]. We scale the input flash images to 1632x1224, and ignore the no-flash guide images. The inputs are JPEG files that have undergone an unknown processing pipeline that includes e.g. color processing, gamma, and noise reduction. We do not attempt a radiometric calibration except for approximate inverse gamma. The set includes categories such as metal, wood, plastic, papers, fabrics, and leathers. Some of the inputs feature defects such as the edge of the material sample. We preprocess these by manually introducing a crop rectangle. All solution data, along with the input images, is presented for inspection in the supplemental material.

5.2 Synthesis Results

As our objective function only attempts to match the input in a statistical sense, it is much more ambiguous than the explicit constraints used in most reflectance capture techniques. This ambiguity leads one to presume the synthesis results may not reach the same quality. Bearing this in mind, we nevertheless find that the results generally convey the feel of the materials well both in terms of overall appearance and individual structured features. Figure 10 shows a subset of the synthesized SVBRDF maps, along with side-by-side comparisons to the input. To directly visualize how the main objective of the optimization is fulfilled, Figure 11 shows several input photographs, each with 15 rendered SVBRDF tiles overlaid, that feature solutions of varying degrees of success. Most often, the rendered tiles blend in with the background quite well. As expected, the stationarity assumption means large-scale features in the input photographs are missing from the results. However, the solver appears mostly robust to input non-stationarity. The supplemental material contains videos featuring novel-view relightings of all inputs, demonstrating generalization.

Interestingly, we note that also many materials that strongly violate our model assumptions (non-planarity, anisotropy, etc., e.g. *fabric_orange, fabric_zigzag, seed*) generally blend in with the input photographs. However, these results often fail to faithfully generalize to novel views, as can be seen in the accompanying relighting videos. Some results feature strong intensity differences towards the edges. We attribute this to our imprecise model of the flash falloff.

5.3 Discussion

Effect of neural statistics To study the importance of the neural texture descriptor on the results, we implemented a version of our al-

[1] http://tinyurl.com/TwoShotSVBRDF/inputs_and_solutions

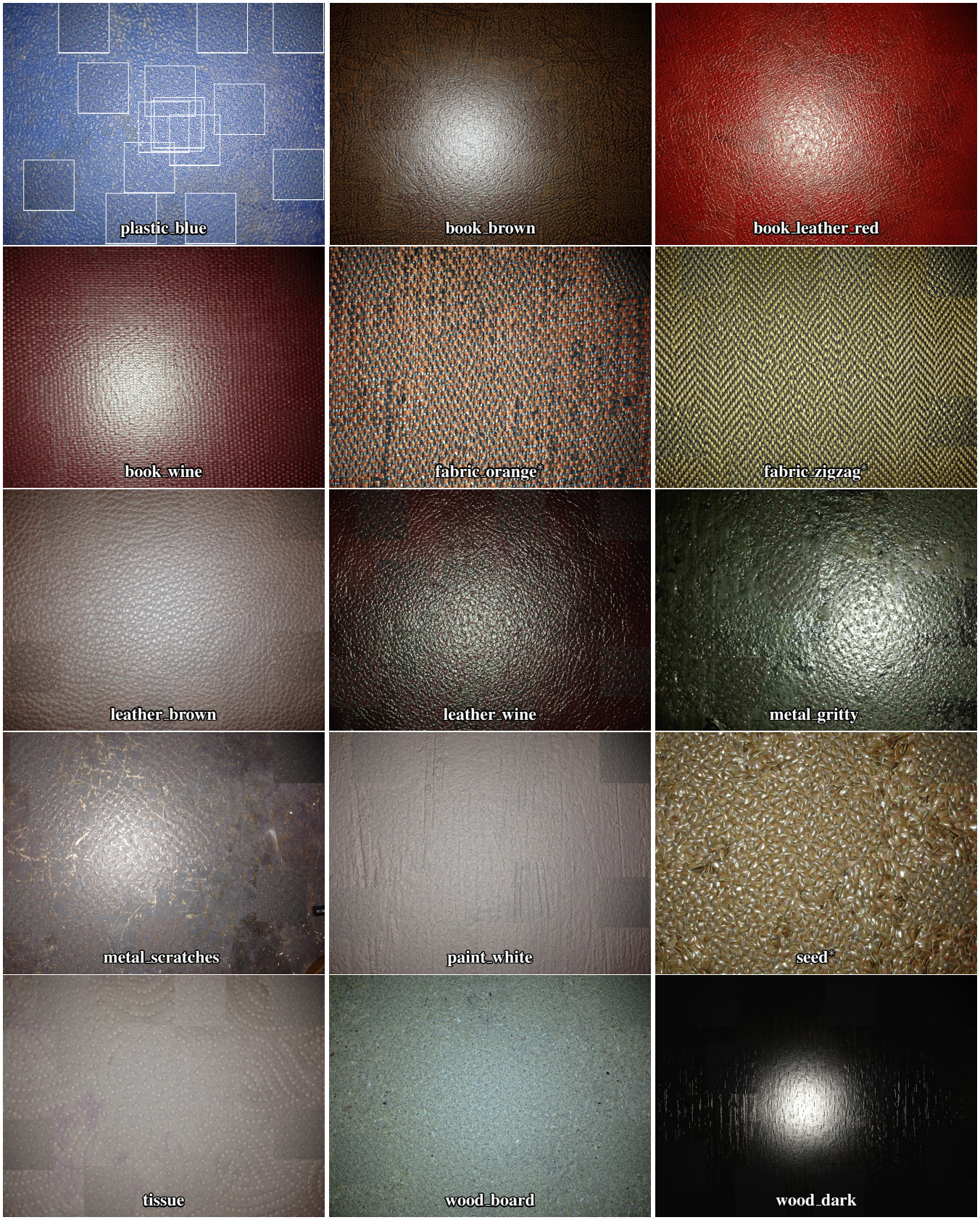


Figure 11: *A selection of results. Each of the input images have 15 relit synthesized tiles embedded in them. The tiles are marked in the top-left image, and are roughly in the same places in the other images. Inputs marked with * strongly violate our input assumptions by significant anisotropy or non-planarity. All remaining results are included in the supplemental material.*

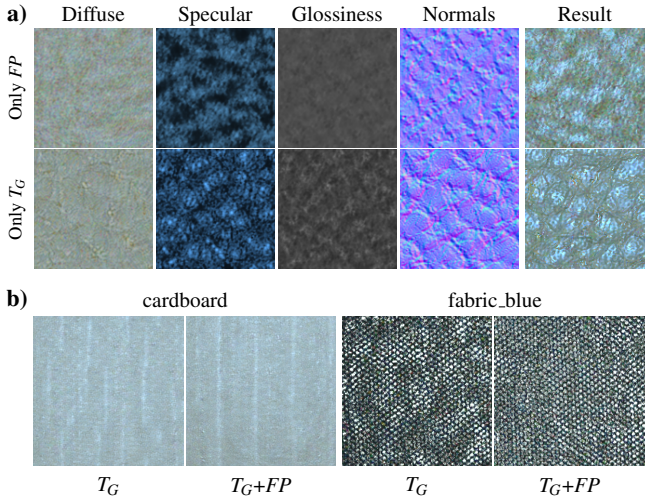


Figure 12: ***a)*** *Comparison of T_G and the Fourier power spectrum texture descriptor (FP). FP works reasonably well for stochastic textures, but falls apart with more structured inputs, such as* leather_amber. ***b)*** *A result computed using both texture descriptors simultaneously. Interestingly, in some cases FP helps to recover larger structures, such as the vertical streaks in* cardboard*. In* fabric_blue*, the combined descriptor helps the optimizer overcome its confusion resulting from the significant anisotropy. The result is still overall unfaithful as this is not part of our reflectance model.*

gorithm that uses the absolute difference in Fourier power spectrum $\text{abs}(|\mathcal{F}\{W \odot R(\mathbf{u}; \mathbf{e}_k, \mathbf{l}_k)\}|^2 - |\mathcal{F}\{W \odot \mathbf{y}_k\}|^2)$ as the texture descriptor instead of $T_G(\cdot)$. W is the Hann windowing function. This is essentially a random phase noise descriptor. Figure 12a compares results to ours in a representative set. Interestingly, the result manages to capture overall shininess and color, and looks plausible viewed from far away. However, it breaks down into Perlin noise -like patterns on closer inspection, as the descriptor fails to capture the local shapes of the textural features. This is to be expected, given the rather narrow range of textures represented well by Fourier modulus.

Failure cases Figure 13 describes representative failures due to model assumption violations and bad conditioning. Please refer to the caption for analysis. A particularly interesting issue is strong repetitive structure, such as a brick wall or the wallpaper in Figure 13a). This continues to be challenging for parametric texture synthesis, including the CNN-based approach we build on. This carries over to our results. By seeding synthesis with initial guesses that feature slight regular structure on top of white noise, we have noticed that the loss function *is* able to distinguish between a regular and irregular solution, but with the purely stochastic initial guesses the L-BFGS optimizer gets stuck in a local minimum, unable to move to the basin of convergence of the regular structure.

Interestingly, we observe that mixing the absolute Fourier power spectrum difference descriptor in with T_G in the loss function may help our solver reach more regular solutions (Figure 12b). The effect is not general, but the result indicates that further research into suitable statistics is warranted.

We also tested the method on a dataset that clearly violates our stationarity assumption. Figure 13e) shows a material with distinct regions of metal, wood, and shiny paint. As expected, the method cannot find a solution tile that reconstructs each of these materials simultaneously, and instead produces a noisy solution that roughly reproduces the average colors and the apparent highlight size.

Normal maps comparison Figure 14 compares some of our normal maps to the results of Aittala et al. [2015] and their submicron-accurate GelSight ground truth scans. As is to be expected, our results are not as accurate, but often convey the structure reasonably well. In hard cases, the ambiguity posed by lack of point correspondences may lead to a clearly inferior result (*plastic_cutting*).

Synthetic data comparison We tested our method on a set of rendered synthetic input images with known ground truth. To obtain plausible SVBRDF data, we adapted the solved maps from Aittala et al. [2015] to our rendering model. Figure 15 shows the synthetic flash photo, the corresponding ground truth, and our solution for three materials of varying color, bumpiness, and specularity. Visual and statistical comparison of the maps demonstrates that our method generally recovers a good approximation of the ground truth. The most notable deviation can be seen in the albedo channels of *leather_brown* dataset, where the specular lobe is wider than the camera FOV, and hence not uniquely separable from the diffuse component under our assumptions. To evaluate the visual significance of the deviations, we also render our solutions from a novel view and light angle, and compare the result to a corresponding ground truth rendering. While somewhat more irregular, our solutions reproduce the appearance well.

6 Conclusion

We have described a method for capturing a rich appearance model of a textured surface based on a single input image. Inspired by parametric texture synthesis, we introduced a novel way to drive the model fit by statistical image comparison without explicit matches between model predictions and the input. Plausible results are obtained for a range of different materials.

Acknowledgements

We thank Samuli Laine, Tero Karras, and Frédo Durand for fruitful discussions. This work was supported by the Academy of Finland (grant 277833). We acknowledge the computational resources provided by the Aalto Science-IT project.

A Implementation

We implement our algorithm in Matlab using the MatConvNet package [Vedaldi and Lenc 2014], which evaluates directed acyclic graphs (DAGs) of layers and computes the gradients using the chain rule with backpropagation. It automatically maps onto a GPU. The texture descriptors T_G can readily be computed and differentiated using this machinery [Gatys et al. 2015b], and we formulate the rest of our approach, i.e., preconditioning, rendering, and priors as a sequence of layers in the same DAG. The full design is shown in Figure 6. This approach requires us to implement a differentiation operation for the new layer types, such as FFT, but it also allows us to differentiate very complicated compositions of functions.

The objective function is evaluated by feeding the optimization variables and per-tile constants into the network entry nodes, running the network, and summing the outputs of the texture statistics comparison layers and the priors. Conversely, the derivatives are evaluated by tagging these output nodes with a derivative value of 1, and backpropagating the derivatives through the network in a standard manner. This is repeated for each tile k, each time substituting in the lighting conditions I_k and target texture descriptors $T_G(\mathbf{y}_k)$, and accumulating the results. The priors are processed separately in a similar manner. The target texture descriptors are computed

a) b) c) d) e)

Figure 13: *Failure cases.* ***a)*** *Materials with highly regular structure may fail to recover the precise arrangement, and the solution is left with several incompatible smaller regions that fade into each other. Individual features may still be captured well in parts of the resuls.* ***b)*** *Anisotropy and gross violations of planarity may lead to results blending in with the input rather well, but the synthesized surface shape and reflectance are not faithful. Here, the anisotropic bumps are explained by curved smooth ridges that have little to do with the real shape.* ***c)*** *Here the solver attempts to explain an almost pure albedo variation with normal changes. In this case, manually setting a penalty for normals fixes the issue.* ***d)*** *Strongly anisotropic materials, such as the cloth, do not produce faithful synthesis results as they fall outside the scope of our model.* ***e)*** *Mixtures of materials – in this case metal, wood, and shiny paint – violate our stationarity assumption and lead to dubious results.*

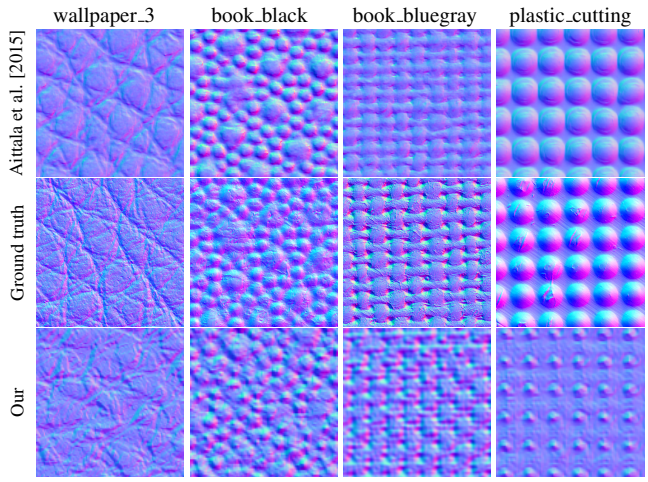


Figure 14: *Comparison between ground truth GelSight scans, the results of Aittala et al. [2015], and our result. In the first three sets we observe a generally good resemblance, but in* plastic_cutting *the estimation has failed and most of the structure is missing.*

and stored in a pre-pass by feeding the tiles into the same DAG (bypassing the rendering step), and reading back their statistics.

In optimization, we initialize the raw optimization variables $\tilde{\mathbf{u}}$ so that after preconditioning their values are $\mathbf{n} = [0\ 0\ 1]$, $\alpha = \exp(3)$, $\rho_s = [0.5\ 0.5\ 0.5]$, and ρ_d computed as the average color of the outer edges of the input image, where the effect of the specular highlight is the weakest. We also add a tiny amount of noise (standard deviation 0.01) to each variable. We approximate the flash emission by the empirical function $\exp(-0.5 * \tan(\tan(\gamma) * 1.6)^2)$, where γ is the angle from the camera central axis.

References

AITTALA, M., WEYRICH, T., AND LEHTINEN, J. 2015. Two-shot SVBRDF capture for stationary materials. *ACM Trans. Graph. 34*, 4, 110:1–110:13.

BARRON, J., AND MALIK, J. 2015. Shape, illumination, and reflectance from shading. *IEEE TPAMI (to appear)*.

BOIVIN, S., AND GAGALOWICZ, A. 2001. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In *Proc. ACM SIGGRAPH*, 107–116.

CLARK, R., 2010. Crazybump. http://www.crazybump.com, Last access: 16 Jan 2016.

DEBEVEC, P., HAWKINS, T., TCHOU, C., DUIKER, H.-P., SAROKIN, W., AND SAGAR, M. 2000. Acquiring the reflectance field of a human face. In *Proc. ACM SIGGRAPH*, 145–156.

DONG, Y., WANG, J., TONG, X., SNYDER, J., LAN, Y., BEN-EZRA, M., AND GUO, B. 2010. Manifold bootstrapping for SVBRDF capture. *ACM Trans. Graph. 29*, 4, 98:1–98:10.

DONG, Y., TONG, X., PELLACINI, F., AND GUO, B. 2011. Appgen: interactive material modeling from a single image. *ACM Trans. Graph. 30*, 6, 146:1–146:10.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proc. ACM SIGGRAPH*, 341–346.

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *Proc. International Conference on Computer Vision (ICCV '99)*, vol. 2, 1033–1038.

GALERNE, B., GOUSSEAU, Y., AND MOREL, J.-M. 2010. Random phase textures: Theory and synthesis. *IEEE Transactions in Image Processing 20*, 1, 257–267.

GATYS, L. A., ECKER, A. S., AND BETHGE, M. 2015. A neural algorithm of artistic style. *CoRR abs/1508.06576*.

GATYS, L. A., ECKER, A. S., AND BETHGE, M. 2015. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems 28*.

GRAHAM, P., TUNWATTANAPONG, B., BUSCH, J., YU, X., JONES, A., DEBEVEC, P., AND GHOSH, A. 2013. Measurement-Based Synthesis of Facial Microgeometry. *Computer Graphics Forum 32*, 2pt3, 335–344.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proc. ACM SIGGRAPH*, 229–238.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *Proc. ACM SIGGRAPH*, 327–340.

HUANG, L., YANG, Y., DENG, Y., AND YU, Y. 2015. Densebox: Unifying landmark localization with end to end object detection. *CoRR abs/1509.04874*.

KASPAR, A., NEUBERT, B., LISCHINSKI, D., PAULY, M., AND KOPF, J. 2015. Self Tuning Texture Optimization. *Computer Graphics Forum 34*, 2.

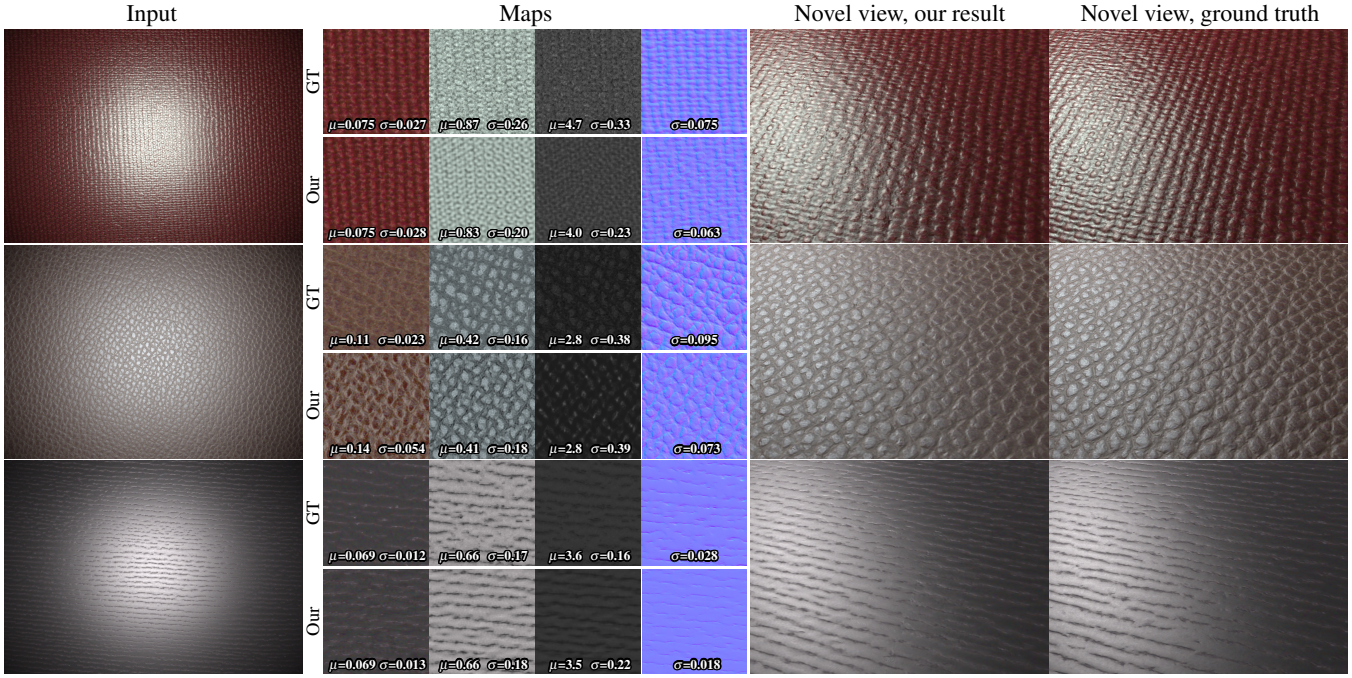


Figure 15: *Results for three synthetic datasets. The leftmost column shows the input flash image, which was rendered using the ground truth SVBRDF shown in the middle column. Our solution from this input is also shown in the center column along with mean and standard deviation of the maps. The statistics of the glossiness values are computed from logarithms of the Blinn exponents. The rightmost columns show a comparison of novel-view renderings of the ground truth and our solution.*

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Trans. Graph. 24*, 3, 795–802.

LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. 1998. Gradient-based learning applied to document recognition. In *Proc. IEEE*, 2278–2324.

LENSCH, H. P. A., KAUTZ, J., GOESELE, M., HEIDRICH, W., AND SEIDEL, H.-P. 2003. Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph. 22*, 2, 234–257.

NGAN, A., DURAND, F., AND MATUSIK, W. 2005. Experimental analysis of BRDF models. In *Proc. Eurographics Symposium on Rendering*, 117–226.

PORTILLA, J., AND SIMONCELLI, E. P. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision 40*, 1, 49–70.

RABIN, J., PEYRÉ, G., DELON, J., AND BERNOT, M. 2011. Wasserstein barycenter and its application to texture mixing. In *Proc. Scale Space and Variational Methods in Computer Vision (SSVM)*, vol. 6667, 435–446.

RAZAVIAN, A. S., AZIZPOUR, H., SULLIVAN, J., AND CARLSSON, S. 2014. CNN features off-the-shelf: An astounding baseline for recognition. In *Proc. CVPR*.

SIMONYAN, K., AND ZISSERMAN, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556*.

SUPER, B., AND BOVIK, A. 1995. Shape from texture using local spectral moments. *IEEE TPAMI 17*, 4.

SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., AND RABINOVICH, A. 2015. Going deeper with convolutions. In *Proc. CVPR*.

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. Graph. 21*, 3, 665–672.

VEDALDI, A., AND LENC, K. 2014. MatConvNet - convolutional neural networks for MATLAB. *CoRR abs/1412.4564*.

WANG, J., ZHAO, S., TONG, X., SNYDER, J., AND GUO, B. 2008. Modeling anisotropic surface reflectance with example-based microfacet synthesis. *ACM Trans. Graph. 27*, 3, 41:1–41:9.

WANG, C.-P., SNAVELY, N., AND MARSCHNER, S. 2011. Estimating dual-scale properties of glossy surfaces from step-edge lighting. *ACM Trans. Graph. 30*, 6, 172:1–172:12.

WEI, L.-Y., HAN, J., ZHOU, K., BAO, H., GUO, B., AND SHUM, H.-Y. 2008. Inverse texture synthesis. *ACM Trans. Graph. 27*, 3, 52:1–52:9.

WEYRICH, T., MATUSIK, W., PFISTER, H., BICKEL, B., DONNER, C., TU, C., MCANDLESS, J., LEE, J., NGAN, A., JENSEN, H. W., AND GROSS, M. 2006. Analysis of human faces using a measurement-based skin reflectance model. *ACM Trans. Graph. 25*, 3, 1013–1024.

WEYRICH, T., LAWRENCE, J., LENSCH, H., RUSINKIEWICZ, S., AND ZICKLER, T. 2009. Principles of appearance acquisition and representation. *Foundations and Trends in Computer Graphics and Vision 4*, 2, 75–191.

ZICKLER, T., RAMAMOORTHI, R., ENRIQUE, S., AND BELHUMEUR, P. N. 2006. Reflectance sharing: predicting appearance from a sparse set of images of a known shape. *IEEE TPAMI 28*, 8, 1287–1302.