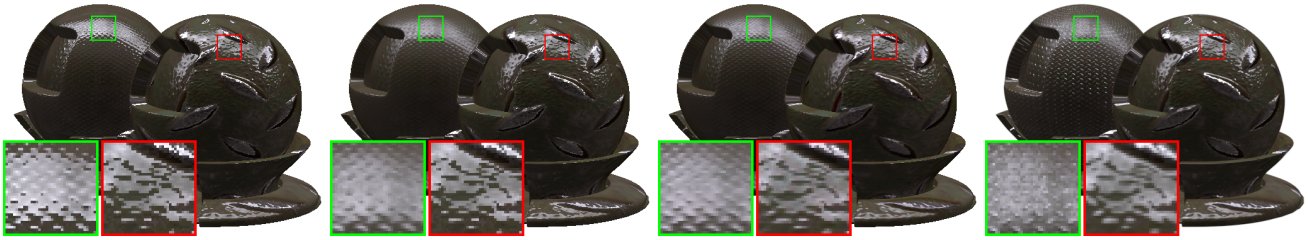
Eurographics Symposium on Rendering (2023)
T. Ritschel and A. Weidlich (Editors)

Data-driven Pixel Filter Aware MIP Maps for SVBRDFs

Pauli Kemppinen[1], Miika Aittala[2] and Jaakko Lehtinen[1,2]

[1]Aalto University, Finland
[2]Nvidia Helsinki, Finland



(a) Naive linear **(b)** MIPNet **(c)** Ours **(d)** Reference

Figure 1: *Our goal is to find a level of detail representation for surface appearance that produces high quality output without supersampling. From left to right: Linear downsampling, MIPNet [GFL*22], our method, ground truth reference. Linear, MIPNet and ours are shaded once per pixel. Note the smooth highlights in the closer object: in the reference image, these are caused by the high quality pixel filter. We optimize our SVBRDF maps to match this appearance when shaded only once per pixel, obtaining a good match to the ground truth at all scales.*

Abstract

We propose a data-driven approach for generating MIP map pyramids from SVBRDF parameter maps. We learn a latent material representation where linear image downsampling corresponds to linear prefiltering of surface reflectance. In contrast to prior work, we explicitly model the effect of the antialiasing pixel filter also at the finest resolution. This yields high-quality results even in images that are shaded only once per pixel with no further processing. The SVBRDF maps produced by our method can be used as drop-in replacements within existing rendering systems, and the data-driven nature of our framework makes it possible to change the shading model with little effort. As a proof of concept, we also demonstrate using a shared latent representation for two different shading models, allowing for automatic conversion.

1. Introduction

Rendering is the process of turning a virtual scene description into a digital image. This image is a regularly sampled digital signal, so its frequency content must be limited prior to sampling according to the Nyquist-Shannon sampling theorem [Sha49]. In principle, we can think of this process as first producing an infinitely sharp image and then appropriately low pass filtering it before sampling. In rendering, this low pass is typically realized as a convolution with a pixel filter, such as a Gaussian or a function from the Mitchell-Netravali family [MN88]. Since the infinitely sharp input image cannot be obtained or stored, the convolution is typically approximated by supersampling or Monte Carlo integration.

Level of detail (LoD) techniques for texture maps are designed to help the pixel filter approximation or replace it altogether. The idea is to transform the pixel filter into the texture domain and apply the filter there aided by a precomputed structure. This structure is typically a MIP map pyramid that stores several prefiltered and downsampled versions of the texture. In real-time rendering it is typical to sample the texture only once per output pixel and use some other approach to filter the effects of visibility.

In their basic form, texture LoD techniques assume that the final shading result depends linearly on the contents of the texture map so that shading commutes with pixel filtering. This holds for simple reflectance models, such as the spatially-varying albedo of a diffuse surface, but breaks down when the texture is used to drive a more complex surface appearance model such as a spatially varying BRDF (SVBRDF). As parametric BRDF models are typically not linear with respect to their parameters, rendering does not commute with pixel filtering: shading with a full resolution SVBRDF and low pass filtering the image is not the same as low pass filter-

DOI: 10.2312/sr.20231124

ing the SVBRDF parameter textures and shading the result at a low sampling rate. However, the shading result *is*, under distant illumination, linear w.r.t. the spatial average of the pointwise directional values of the BRDF (the so-called "apparent BRDF"). This motivates a long line of prior work for representing prefiltered apparent BRDFs with varying simplifying assumptions [BM93; Tok05; OB10]. A concurrent work [GFL*22] gives a data driven alternative. We will elaborate on this below.

We propose to learn a low pass operator that preserves the surface appearance and suitably limits the frequency content of the subsequent renderings. The operator is realized as a pair of convolutional neural networks (CNN) that encode to and decode from a latent texture representation. The latent representation is optimized to be such that the appearance-preserving low-pass operation is linear in this space. This enhances training stability compared to a simpler iterative encoder that autoregressively produces one MIP map layer at a time. The encoder and decoder are trained in the Noise2Noise fashion [LMH*18] using a rendering loss that employs the desired pixel filter. As we target real-time rendering, we choose to output standard MIP maps that drive a standard real-time SVBRDF model, so that no changes to the target rendering code are required; only the MIP map generation is affected. In line with this, we focus on the case of shading the material once per pixel. Using our method in a supersampled offline renderer is still possible by applying the proper MIP bias. Our method is fully data driven, and can in principle be used for any BRDF model or shader, so long as a differentiable implementation is available.

Interestingly, all prior work appears to treat only minification of the surface reflectance function. We contribute the observation that this picture is incomplete: even when rendering using no minification, a proper antialiasing pixel filter has a smoothing effect that we believe should be modeled by the reflectance prefiltering algorihm. For example, rendering a perfect bumpy mirror with one sample per pixel will generally clearly alias and produce the well-known "shimmering" in specular highlights. We explicitly account for pixel filtering also in processing the highest-resolution SVBRDF. This results in noticeable smoothing in sharp specular reflections and a much better match to the supersampled reference, as apparent in the foreground crop in Figure 1 and our further experiments.

We also demonstrate training a single encoder and two different decoders, each of which outputs parameters for a different SVBRDF model. This offers automatic conversion between different parametric BRDF models, with potential applications in targeting hardware platforms of different capabilities from the same source material.

2. Previous work

2.1. Prefiltering

MIP mapping [DSS78] with trilinear fetches [Wil83] or their anisotropic extension [MPFJ99] is the standard technique for prefiltering color images in real-time rendering.

For bump [Bli78] and normal [COM98][CMRS98] maps no prefiltering solution is ubiquitous. However, several methods have been proposed for specific material models with a varying set of assumptions. A simple rule of thumb for Blinn-Phong materials was given by [Tok05]. LEAN mapping [OB10] improves this by approximating Blinn-Phong materials with the Beckmann distribution, and derives a formula for the roughness assuming normal variation to be Gaussian and independent of other material parameters. Notably, LEAN mapping accounts for anisotropy that may result from normal maps even if the base SVBRDF is isotropic. LEADR mapping [DHI*13] extends this to displacement mapping. These methods do not require any changes to the rendering system itself. In contrast, Han et al. [HSRG07] use spherical harmonics to represent and filter the distribuiton of normals, and Xu et al. [XWZB17] fit a von Mises-Fisher mixture to a BRDF and evaluate it adaptively.

In concurrent work, Gauthier et al. [GFL*22] propose a data driven approach that learns to produce MIP maps with a multi-level rendering loss. Their problem setting is identical to ours. However, their method does not explicitly consider the pixel filter and any bandlimiting emerges purely from the downsampling operation. They also downsample some channels, such as the diffuse and specular albedos, using conventional methods, and the roughnesses and normals using their machine learning pipeline. This means that they assume statistical independence of the albedo channels and the specular lobes, which does not hold for all real-world materials as they (and we, see Figure 11) demonstrate.

In the most extreme cases, highly specular materials and normal maps cause extremely bright and small subpixel highlights which produce a glinty appearance. Due to the remarkably complex distribution of normals (see Figure 8), rendering such materials efficiently requires a specialized algorithm such as the hierarchical method of Yan et al. [YHJ*14]. In recent years, progress has been made to render such materials in real time [TZX*22][CLS*21], but these are still noticeably more costly than a standard single-lobe specular shader.

In addition to normal maps, the rapidly varying normals on high curvature surfaces can cause aliasing. Kaplanyan et al. introduce a way to filter this variation based on the local geometric setup [KHPL16], while Tokuyoshi et al. refine the approach for real-time shading in forward rendering [TK21]. These approaches require knowledge of the specific scene configuration, and are thus not easily combined into a prefiltering method. However, they work well as a post-processing step for an already filtered local shading model as shown by [KHPL16]. This being the case, we only consider planar local geometry and distant viewing and light directions.

Another related approach is path space regularization [KD13][WDH*21], where BRDFs are mollified (*made less sharp* in a specific sense) to enable rendering of some extremely difficult kinds of light paths, and reduce variance in configurations close to these. Here the BRDFs are modified not due to local variations, but to make the global light transport more feasible.

2.2. Machine learning

Convolutional neural networks (CNNs) [LBH15] are a standard tool for processing images and other signals in machine learning. They consist of layers of convolutions that are learned from data, and fixed nonlinearities between the layers. Stochastic gradient descent (SGD) or Adam [KB14] is typically used to optimize the con-

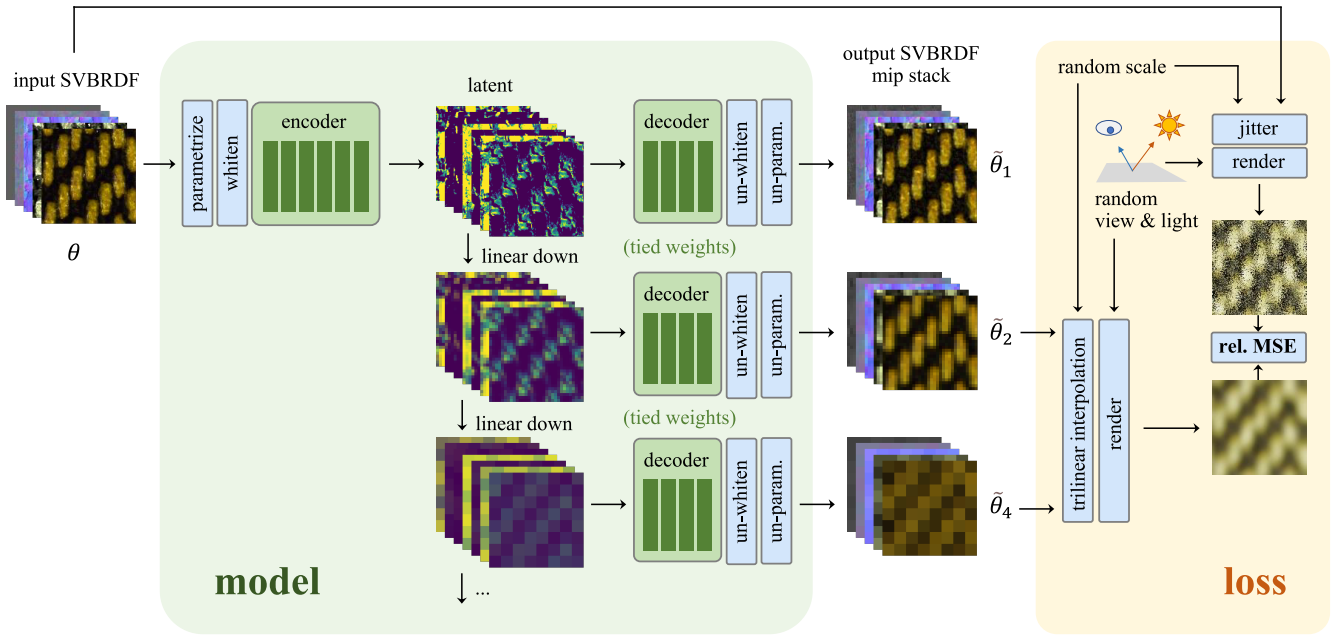


Figure 2: *Overview of our model and training loss. An SVBRDF is encoded into a learned latent intermediate representation, which is linearly downsampled to and decoded at multiple resolutions to form the predicted SVBRDF MIP stack. The encoder-decoder pipeline is wrapped in a conversion to a suitable anisotropy-glossiness-normals parametrization, and whitening. To evaluate the training loss, a predicted filtered SVBRDF is formed by trilinear interpolation of the model outputs using a randomly chosen scale (bandlimit), and differentiably rendered under random viewing and illumination conditions. The training target is formed by jittering according to the same scale, and rendering the ground truth input with the same parameters.*

volutions, since datasets are too large to evaluate the full gradient for each step.

Training a CNN becomes increasingly numerically unstable when more layers are added. Two solutions to this are DenseNets [HLMW17] and ResNets [HZRS16]. DenseNets work by stacking the input of each layer to its output, never removing information but steadily increasing the number of channels in the result images. ResNets add residual connections, where the input to a set of layers is added to its output, so only the difference between the input and desired output has to be learned.

Noise2noise is an approach to CNN training introduced by Lehtinen et al. [LMH*18] based on the interesting observation that neural networks regress to the mean in the presence of noise. This means that in cases where the mean is the desired outcome (such as in denoising), one does not need the standard noise-free targets that can be difficult or impossible to acquire. For all of this to work, the mean of the noise has to be zero; in practice this means we need to use unbiased statistical estimators and the L_2 loss. Our rendering loss is based on Monte Carlo approximations so we adopt this strategy instead of computing noise-free target images.

3. Method

3.1. Goals and Overview

Our goal is to find a pre-filtered multi-scale representation for spatially varying reflectance that reproduces, when shaded only once per pixel, high quality rendering results at any viewing distance. By *high quality*, we mean that the rendered output should be a close approximation to an ideal reference that would result if an infinite-resolution rendering of the full-resolution input SVBRDF was accurately low-pass filtered with a chosen pixel filter.

In order to work seamlessly with existing hardware and rendering systems, we assume the input full-resolution SVBRDF is represented by texture maps that drive a parametric BRDF model (e.g. GGX), and seek to approximate the prefiltered appearance in terms of the same parametric SVBRDF model. Concretely, given texture maps that define the full-resolution input SVBRDF, our system produces a MIP map stack of similar SVBRDF parameter maps that have been optimized so that once-per-pixel shading of the tri-linearly filtered parameters approximates the effects of accurate pixel filtering. For simplicity of exposition, we assume the input and output SVBRDF models are the same, but our architecture is designed so that we can easily output parameter maps for one or more different parametric models from the same input, facilitating conversion.

Standard BRDF parametrizations are not linear in the sense that the relationship between the parameter values and the resulting shading values is nonlinear. Our main idea is, simply, to learn an SVBRDF encoder-decoder pair with the special property that linear low-pass filtering in the intermediate latent texture space corresponds to pre-filtering the output rendering. The mapping is realized as two convolutional neural networks: an encoder CNN

$E_{\psi_E}(\theta)$ with parameters ψ_E that maps the input SVBRDF texture $\theta(x,y)$ to a latent representation $\mathbf{z}(x,y)$, and a decoder CNN $D_{\psi_D}(\mathbf{z})$ with parameters ψ_D that maps the latent texture back to interpretable SVBRDF parameters. Constructing a MIP map hierarchy is performed by first encoding the full-resolution input; then creating a MIP map pyramid in the latent texture space using standard linear filtering, and finally decoding each latent MIP map level:

$$\tilde{\theta}_r = D(\hat{\phi}_r * E(\theta)) \tag{1}$$

Here r denotes the bandwidth of the desired low-pass filter, and $\hat{\phi}_r$ denotes the joint operation of linearly filtering the input followed by sub-sampling to reduce resolution; in practice, this is a standard antialiased image resizing operation that operates on the latent texture. The encoder and decoder and learned using a multiscale rendering loss, as described in the next subsection.

The emergent latent representation is only used during preprocessing: at render time, only the resulting MIP map in the original parametrization is required. Note that the mappings E and D cannot be pointwise operations; this would not be sufficient to capture the effects of the pixel filter in the original resolution.

Our overall goal is shared with many previous methods, with the exception of our explicit consideration of the pixel filter. Indeed, it is clear that when shading is performed only once per pixel, highly shiny materials easily cause aliasing when illuminated by small light sources, even when the original SVBRDF is viewed at 1:1 pixel-to-texel ratio. Accordingly, we also process the *original* resolution of the SVBRDF map, leading to appropriate smoothing of the specular lobes to approximate proper pixel prefiltering.

3.2. Problem Formulation

The inputs to our method are (1) the desired parametric shading model $f(\theta(x,y),l,v)$ that is a function of the viewing direction v, lighting direction l, and the spatially-varying BRDF parameters $\theta(x,y)$; (2) the desired pixel filter $p(x,y)$, and (3) a training material set $\Theta = \{\theta_1,\ldots,\theta_n\}$ that will be expanded by data augmentation. For simplicity in exposition, the Lambertian cosine is taken to be a part of the shading model.

Assuming that the lighting and viewing conditions can be considered constant over the support of the pixel filter, the ideal prefiltered ground truth solution we seek is, for a single directional light of unit intensity in direction l, the convolution of the full-resolution rendering and the pixel filter $p_r(x,y) = p(x/r, y/r)$ of desired scale r:

$$GT_r(x,y,l,v) = (p_r * f(\theta,l,v))(x,y) \tag{2}$$

$$= \int p_r(x-x', y-y')\, f(\theta(x',y'),l,v)\, \mathrm{d}x'\,\mathrm{d}y'. \tag{3}$$

The parameters ψ_E and ψ_D of the encoder and decoder are optimized to minimize a rendering loss $L(\cdot,\cdot)$ between the ground truth and single-sample-per-pixel renderings of the SVBRDF textures $\tilde{\theta}_r$ produced by the encoder-prefilter-decode pipeline as per Equation (1), taken as an expectation over the training set and the sets of viewing and lighting directions and viewing scales:

$$\underset{\psi_E,\psi_D}{\operatorname{argmin}}\ \mathbb{E}_{\Theta,r,l,v}\left\{L(f(\tilde{\theta}_r,l,v),\, GT_r(x,y,l,v))\right\}. \tag{4}$$

As the ground truth cannot be evaluated exactly, we compute it using Monte Carlo integration, relying on the Noise2Noise principle. To deal with high dynamic range without resorting to bias-inducing tone mapping, we use the relative mean squared loss

$$L(P,GT) = \frac{||P-GT||_2^2}{\text{stop\_grad}\,||P||_2^2 + \varepsilon} \tag{5}$$

between the prediction $P = f(\tilde{\theta}_r,l,v)$ and the ground truth images. The stop_grad operator means that the given term is treated as a constant by the optimization process. As shown by Lehtinen et al. [LMH*18], the optimum for this loss is the expectation of the noisy reference. The loss is evaluated by rendering images and summing over the discrete pixel values. To properly account for the effects of texture filtering that will be eventually applied to the resulting MIP maps at render time, we perform random sub-pixel jittering before rendering, effectively turning the loss into an integral over the (x,y) plane. Note that this process requires considering texture reconstruction filters during the rendering process, but we have omitted the details for notational clarity. In practice, we use bilinear and trilinear filtering for θ and $\tilde{\theta}$, respectively.

Note that this formulation implies an approximation: the material is filtered as if it were viewed directly from above, regardless of the actual viewing direction v. However, this is in line with the isotropic approximation of a trilinear MIP map sample, which is what we use for rendering. The exact match to this optimization formulation would be to perform texture space shading.

3.3. Model Architecture

We use the DenseNet architecture [HLMW17] for our networks, since our task requires only slight modifications to the original signal, and DenseNets propagate the original signal to be always available. Initial testing showed that the choice of architecture is not crucial, and for example ResNets [HZRS16] perform similarly in this task.

The input to the encoder is an SVBRDF texture stack that has been converted to our optimization parameterization and whitened as detailed below. Its output is the latent code $\mathbf{z}$. The input to the decoder is the (potentially resized) latent code. Its output is, after un-whitening and conversion from the optimization parameterization, again an SVBRDF texture stack.

The encoder E consists of 6 layers with 3x3 convolutions, and the decoder D consists of 4 layers with 1x1 convolutions. Both introduce 16 new feature maps per layer. The latent SVBRDF $\mathbf{z}$ contains 64 feature maps. Note that neither the encoder nor decoder internally change the resolution of the feature maps by up- or downsampling. This has the effect that each output texel only depends on a limited pixel neighborhood in the input SVBRDF. Thanks to this fully convolutional nature, the models can be run on arbitrarily-sized inputs.

3.4. Training Details

We now provide details related to optimization, including parameterization and data augmentations.

Optimizer. We use the gradient-based Adam optimizer [KB14]

Figure 3: *Example data augmentations. Top left is* `fabric_zigzag`*, the rest are random augmentations of it.*

with learning rate 5×10^{-5}, $(\beta_1, \beta_2) = (0.95, 0.95)$ and $\varepsilon = 1 \times 10^{-6}$. We train our models for 310k iterations or 24 hours on a V100 using 64 lights per batch, 32x32 crops, and 5 MIP map levels. Both light and camera directions are cosine distributed, and the MIP level is chosen at uniform random from -0.5 to 5.5 (half more than the highest MIP level). The learning rate is ramped up during the first 100 iterations and halved every 50 thousand iterations. Beyond this we do not use dropout, weight decay, batch normalization, or other additional modifications to the optimization procedure.

Batching. For each iteration, a single material is chosen, augmentations are performed, and a number of view and light directions are chosen for the shading; a minibatch contains multiple identical instances of a material patch, each with a single distinct distant lighting and viewing direction. We use patches of resolution 32x32 and 64 view/light configurations per mini batch. The same setup is rendered at 5 different levels of detail by interpolating between the predicted MIP levels. The integrals in the rendering loss are computed with Monte Carlo integration at one sample per pixel by jittering the locations where the shading is evaluated: a jittered pixel grid is used for the integral of the norm, and a further offset is chosen for the ground truth, distributed according to the pixel filter. As our shading estimates are unbiased, the networks will regress towards the correct mean.

Parameterization. Parameterization plays an important role in optimizing parametric (SV)BRDF fits. Naive parameterizations, such as using angles for anisotropy, suffer from discontinuities that are difficult to model by typical neural networks and optimizers. Simple remedies, such as parameterizing anisotropy through the entries of a covariance matrix, may introduce constraints that are difficult to enforce in optimization: not all tuples of four real numbers form a valid 2x2 positive definite covariance matrix. We adopt a simplified version of the anisotropy/glossiness parameterization of Aittala et al. [AWL15] that directly ensures that the tensor will be positive definite. The network outputs three unconstrained scalars (a, b, c) that are uniquely and smoothly mapped to a positive definite 2x2 matrix A by the matrix exponential:

$$A = \exp \begin{pmatrix} a & c \\ c & b \end{pmatrix}. \tag{6}$$

If required by the following renderer, principal axes and their scales may be extracted by eigendecomposion.

Similarly, surface normals are required to be unit vectors. We choose to simply drop their height coordinates before feeding them to the encoder, and lift the 2D vectors output by the decoder back to the hemisphere before rendering. If the network predicts a normal outside of the unit disc, we normalize it before lifting.

Finally, even in our smooth parameterization, the different SVBRDF parameters have different dynamic ranges. As large differences between the scales of input (resp. output) channels are known to be problematic for CNNs, we approximately whiten the parameters by means and covariance matrices that are precomputed from the training set using 16 augmented samples. The whitening is applied as the last step before feeding to the encoder, and undone as the first step after reading the output from the decoder (cf. Figure 2).

Data Augmentation. We use data augmentations to increase the range of materials covered by the input set, see Figure 3 for examples. Augmentations are a natural fit, as the different parameters can be modified independently and relatively easily. We apply random permutations of colors, rotations and scalings of the roughness/glossiness tensor, and modify the eccentricity of normals. Additionally, we combine two differently augmented instances with a binary mask that is formed using a random iso-level of a Perlin noise [Per85] texture with several octaves. This is to make sure that a single example patch can contain drastically different values for each parameter even if such materials do not exist in the training set.

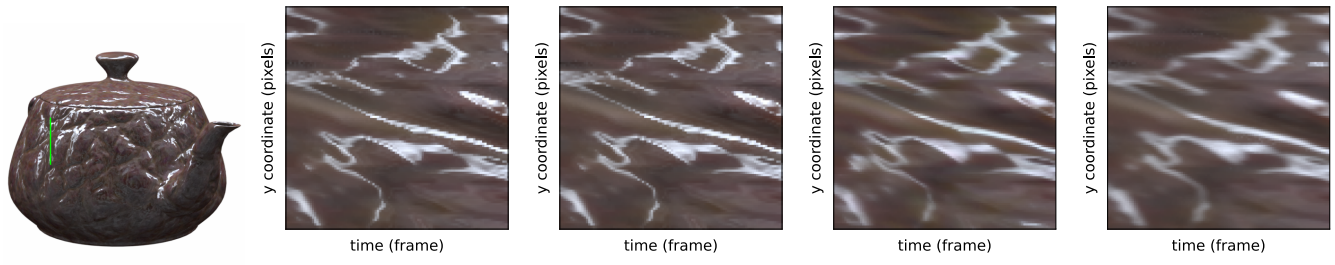
Implementation. We implement our method in PyTorch [PGM*19] with a custom CUDA kernel for the shading models for improved performance. The gains are due to data locality; fusing all of the operations of a complex shading model into a single kernel reduces global memory traffic significantly. The kernel uses checkpointing and local forward mode automatic differentiation to provide gradients.

3.5. Cross-training multiple shading models

Our high-level architecture allows for encoding and decoding parameters for any shading model to a shared latent representation. This can be done relatively freely in any desired combination, as we can treat any set of shading models as inputs and outputs. For this, we need the shading models and a set of example materials for each desired model. The training process is only minimally affected: We pick a material from each of the given sets in turn and encode it using the encoder of its respective shading model. We then decode and shade the same latent with all of the chosen decoders and respective shading models. This way each step we train one encoder and all of the decoders, and the same latent code is jointly used for all shading models.

4. Results

Training data and shading models. We test our model on two shading models: a generalized Beckmann model from [AWL15] and anisotropic GGX. We use the dataset generated by [AWL15] for the



(a) Scene **(b)** MIPNet **(c)** LEAN **(d)** Ours **(e)** Reference

Figure 4: *A temporal slice over an animation. An animation is rendered where the object rotates 0.1 degrees per frame. The same column of pixels is taken from each frame and stacked horizontally to form a new image. (a) scene configuration, green line indicates the column of pixels, (b)-(d) animation stacks, (e) reference. Due to explicitly modeling the pixel filter, our result avoids most of the temporal aliasing present in others. The aliasing is visible here as overly sharp horizontal features.*

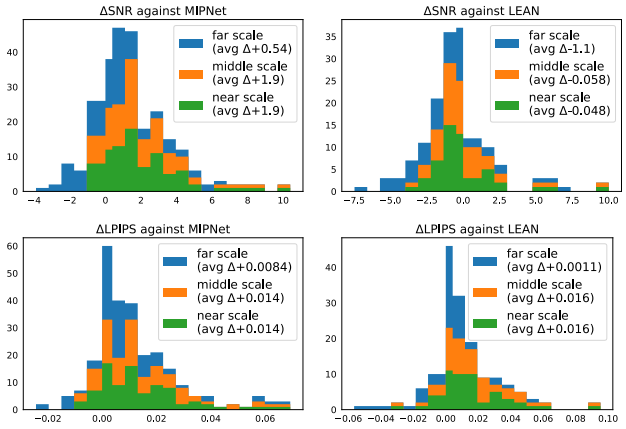


Figure 5: *Rendering quality over our datasets as deltas in signal to noise ratios and LPIPS scores. The renderings use the scene from Figure 1 with only the closer object. The differently colored areas in the histograms match different texture repetition scales. The near, middle and far scales correspond roughly to MIP levels 0, 3, and 7, respectively. Both our model and MIPNet are only trained on MIP levels 0-5, making the far case a test on generalization. Positive values indicate that our method outperforms the respective comparison method. The histograms contain both the train and test set due to the small number of materials in the test set. The main conclusions remain the same when only considering the test set.*

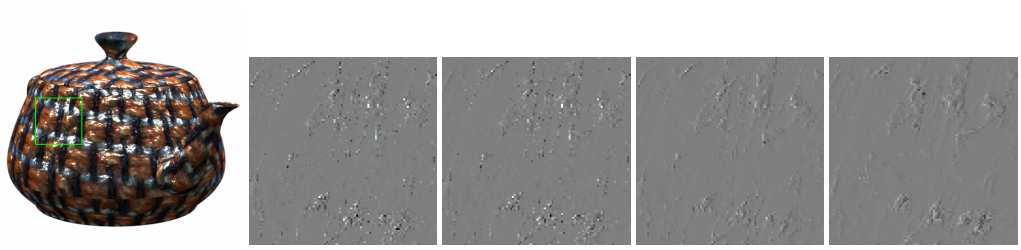
Beckmann variant and a dataset curated from [PH23] for GGX. The latter contains only isotropic materials, and both contain mainly stationary materials without stark jumps or large differences in material parameters. As neither dataset contains extremely sharp specularities or stark normal maps, we boost both properties slightly in the training and test data. The resulting materials are still reasonable, the goal of these modifications is to make the differences between methods more apparent.

Comparison methods. We compare our results against our implementations of the concurrent MIPNet [GFL*22] and LEAN mapping [OB10]. For the generalized Beckmann material, we use linear prefiltering for the additional kurtosis parameter for both MIPNet and LEAN. This violates LEAN's underlying distribution assumption, but works well in practice.

Performance metrics. We quantitatively compare our method to previous work by computing the signal to noise ratio and the LPIPS perceptual distance [ZIE*18] of renderings under a set of environment maps, see Figure 5. Neither of these error metrics are directly optimized by any of the methods. The histograms show LEAN having better average SNR but worse LPIPS than our method, while MIPNet performs slightly worse than our method on both metrics. Initial testing showed that the MIPNet architecture is unstable for our loss, and our Noise2Noise training produces unusable results with their tonemapped L_1 loss, so comparing the networks with the same loss is not feasible. LEAN only works for the Beckmann model due to how it is derived, so GGX materials are omitted from its results.

We begin the results by showing a series of test set materials rendered on a plane from two different distances to illustrate a standard surface reflectance prefiltering setting (Figure 9 and Figure 10). As can be seen, LEAN mapping generally fixes the issues in naive linear filtering. The differences between MIPNet and our result are mostly subtle. Again, refer to Figure 5 for numerical results.

Thanks to our consideration of the pixel filter, differences become clearly more pronounced when we consider closer texture scales. We demonstrate this with two experiments: changes under a small translation in Figure 6 and a slow rotation in Figure 4. The goal in both of these is to emphasize the stability of our results under motion when the texel to pixel ratio is close to 1:1 (the renderings use values between the original resolution and the first MIP level). In Figure 6, we show the change in the rendered results when an object is moved by half a pixel. The comparison methods produce noisy differences, corresponding to temporal aliasing under animations. In Figure 4 we stack the same column of pixels from a number of animation frames to form an image. The animation is a slowly rotating object. This setup makes temporal changes apparent: our result changes smoothly over time, like the reference, while the comparison methods produce popping and sparkling images, corresponding to abrupt changes in pixel brightness. Also see the accompanying videos.



(a) Scene **(b)** MIPNet **(c)** LEAN **(d)** Ours **(e)** Reference

Figure 6: *Changes under a small translation. We compute how offsetting the object by 0.1 pixel changes the resulting rendering. (a) scene configuration, (b)-(e) change in pixel intensity reference. Change is positive for bright pixels and negative for dark ones. Our rendering is relatively stable under this slight perturbation, like the reference.*

Figure 7: *Qualitative results of our cross-training setup. In this experiment we train one encoder and two decoders for each input model (GGX and Beckmann). Here the reference is rendered using the respective input model, and the predicted output for both models is shown.*

Figure 11 shows a material that exhibits strong correlations between albedo, anisotropy and normal channels, violating the assumptions in MIPNet and LEAN. As our model takes in all the SVBRDF parameters at once, it has the means to capture the result better.

Our model architecture makes it possible to decode the same latent code into the parameters of multiple material models. We demonstrate this in Figure 7, where we jointly trained one encoder for a target input material model, and two decoders for two output material models. Decoding to a different model is mostly successful, but sometimes fails due to the inherent difference in specular lobe shape (for example, top right). Perhaps unsurprisingly, the decoder that corresponds to the input material model generally produces more accurate results.

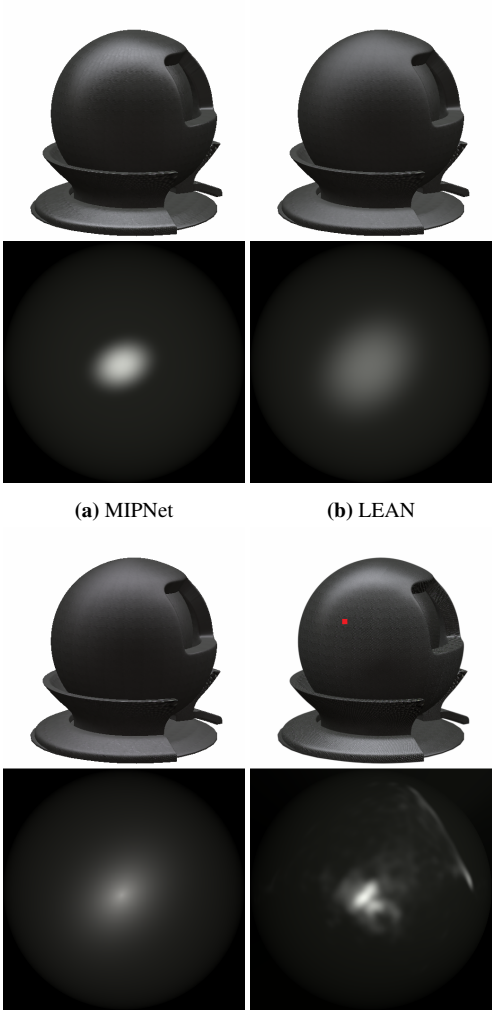
4.1. Limitations

As is apparent from Figure 8, the single-lobe shading model is a shared limitation of all of the methods considered in this work. It is hopeless to try to match the appearance well when the normal map causes the apparent BRDF to have multiple distinct lobes, or a lobe that is of a distinctly different shape than implied by the shading model. For a dramatic example of how this complexity may create nontrivial patterns in the image, see the reference image of the red material in the lower right corner of Figure 9.

Our assumption of constant viewing and lighting directions over the support of the pixel filter means we do not properly model the effects of curvature. This will lead to incorrect results in sharp edges and creases if no post-hoc correction like [KHPL16] is used.

5. Conclusion

We have presented a way to prefilter general surface appearance as represented by an SVBRDF, while retaining the exact desired pixel filter and allowing to automatically convert between materials. Interesting extensions include incorporating anisotropic filters and surface curvature, handling multiple pixel filters with a shared latent, and treating the multi-lobe case where more complicated apparent BRDFs could be achieved.



(a) MIPNet **(b)** LEAN

(c) Ours **(d)** Reference

Figure 8: *Apparent BRDF for a complex normal map. Top row: renderings, Bottom row: the apparent BRDF visualized at the point shown with the red dot in the reference. The apparent BRDF is the pixel filter weighted average of the BRDF for one pixel. The material has a normal map that isn't possible to fit with a single lobe with the chosen BRDFs. All methods struggle.*

Acknowledgements

The Mitsuba knob used in some of the renderings has been reproduced with kind permission.

References

[AWL15] AITTALA, MIIKA, WEYRICH, TIM, and LEHTINEN, JAAKKO. Two-shot SVBRDF Capture for Stationary Materials. *ACM Trans. Graph.* 34, 4 (2015). DOI: 10.1145/2766967.

[Bli78] BLINN, JIM. Simulation of Wrinkled Surfaces. *Siggraph 1978*. 1978. DOI: 10.1145/965139.507101.

[BM93] BECKER, BARRY G and MAX, NELSON L. Smooth transitions between bump rendering algorithms. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. 1993. DOI: 10.1145/166117.166141.

[CLS*21] CHERMAIN, XAVIER, LUCAS, SIMON, SAUVAGE, BASILE, et al. Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering. *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* 4, 1 (2021). DOI: 10.1145/3451257.

[CMRS98] CIGNONI, P., MONTANI, C., ROCCHINI, C., and SCOPIGNO, R. A general method for preserving attribute values on simplified meshes. *Proceedings Visualization '98 (Cat. No.98CB36276)*. 1998. DOI: 10.1109/VISUAL.1998.745285.

[COM98] COHEN, JONATHAN, OLANO, MARC, and MANOCHA, DINESH. Appearance-Preserving Simplification. *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '98. 1998. DOI: 10.1145/280814.280832.

[DHI*13] DUPUY, JONATHAN, HEITZ, ERIC, IEHL, JEAN-CLAUDE, et al. Linear Efficient Antialiased Displacement and Reflectance Mapping. *ACM Trans. Graph.* 32, 6 (2013). DOI: 10.1145/2508363.2508422.

[DSS78] DUNGAN, WILLIAM, STENGER, ANTHONY, and SUTTY, GEORGE. Texture Tile Considerations for Raster Graphics. *SIGGRAPH Comput. Graph.* 12, 3 (1978). DOI: 10.1145/965139.807383.

[GFL*22] GAUTHIER, ALBAN, FAURY, ROBIN, LEVALLOIS, JÉRÉMY, et al. MIPNet: Neural Normal-to-Anisotropic-Roughness MIP Mapping. *ACM Trans. Graph.* 41, 6 (2022). DOI: 10.1145/3550454.3555487.

[HLMW17] HUANG, G., LIU, Z., MAATEN, L. VAN DER, and WEINBERGER, K. Q. Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017. DOI: 10.1109/CVPR.2017.243.

[HSRG07] HAN, CHARLES, SUN, BO, RAMAMOORTHI, RAVI, and GRINSPUN, EITAN. Frequency Domain Normal Map Filtering. *ACM Trans. Graph.* 26, 3 (2007). DOI: 10.1145/1275808.1276412.

[HZRS16] HE, K., ZHANG, X., REN, S., and SUN, J. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. DOI: 10.1109/CVPR.2016.90.

[KB14] KINGMA, DIEDERIK P. and BA, JIMMY. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). DOI: 11245/1.505367.

[KD13] KAPLANYAN, ANTON and DACHSBACHER, CARSTEN. Path Space Regularization for Holistic and Robust Light Transport. *Computer Graphics Forum* 32 (2013). DOI: 10.1111/cgf.12026.

[KHPL16] KAPLANYAN, ANTON S., HILL, STEPHEN, PATNEY, ANJUL, and LEFOHN, AARON. Filtering Distributions of Normals for Shading Antialiasing. *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics*. 2016. DOI: 10.2312/hpg.20161201.

[LBH15] LECUN, YANN, BENGIO, Y., and HINTON, GEOFFREY. Deep Learning. *Nature* 521 (2015). DOI: 10.1038/nature14539.

[LMH*18] LEHTINEN, JAAKKO, MUNKBERG, JACOB, HASSELGREN, JON, et al. Noise2Noise: Learning image restoration without clean data. *35th International Conference on Machine Learning, ICML 2018*. Vol. 7. Proceedings of Machine Learning Research 80. 2018.

[MN88] MITCHELL, DON P. and NETRAVALI, ARUN N. Reconstruction Filters in Computer-Graphics. *SIGGRAPH Comput. Graph.* 22, 4 (1988). DOI: 10.1145/378456.378514.

[MPFJ99] MCCORMACK, JOEL, PERRY, RONALD, FARKAS, KEITH I, and JOUPPI, NORMAN P. Feline: fast elliptical lines for anisotropic texture mapping. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 1999. DOI: 10.1145/311535.311562.

[OB10] OLANO, MARC and BAKER, DAN. LEAN Mapping. *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10. 2010. DOI: 10.1145/1730804.1730834.

[Per85] PERLIN, KEN. An Image Synthesizer. *SIGGRAPH Comput. Graph.* 19, 3 (1985). DOI: 10.1145/325165.325247.

[PGM*19] PASZKE, ADAM, GROSS, SAM, MASSA, FRANCISCO, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32*. 2019. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[PH23] Poly Haven. 2023. URL: https://polyhaven.com/.

[Sha49] SHANNON, C.E. Communication in the Presence of Noise. *Proceedings of the IRE* 37, 1 (1949). DOI: 10.1109/jrproc.1949.232969.

[TK21] TOKUYOSHI, YUSUKE and KAPLANYAN, ANTON S. Stable Geometric Specular Antialiasing with Projected-Space NDF Filtering. *Journal of Computer Graphics Techniques (JCGT)* 10, 2 (2021). URL: http://jcgt.org/published/0010/02/02/.

[Tok05] TOKSVIG, MICHAEL. Mipmapping Normal Maps. *J. Graphics Tools* 10 (2005). DOI: 10.1080/2151237X.2005.10129203.

[TZX*22] TAN, HAOWEN, ZHU, JUNQIU, XU, YANNING, et al. Real-Time Microstructure Rendering with MIP-Mapped Normal Map Samples. *Computer Graphics Forum* 41, 1 (2022). DOI: 10.1111/cgf.14448.

[WDH*21] WEIER, PHILIPPE, DROSKE, MARC, HANIKA, JOHANNES, et al. Optimised Path Space Regularisation. *Computer Graphics Forum* 40, 4 (2021). DOI: 10.1111/cgf.14347.

[Wil83] WILLIAMS, LANCE. Pyramidal Parametrics. *SIGGRAPH Comput. Graph.* 17, 3 (1983). DOI: 10.1145/964967.801126.

[XWZB17] XU, CHAO, WANG, RUI, ZHAO, SHUANG, and BAO, HUJUN. Real-Time Linear BRDF MIP-Mapping. *Computer Graphics Forum* 36, 4 (2017). DOI: 10.1111/cgf.13221.

[YHJ*14] YAN, LING-QI, HAŠAN, MILOŠ, JAKOB, WENZEL, et al. Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces. *ACM Trans. Graph.* 33, 4 (2014). DOI: 10.1145/2601097.2601155.

[ZIE*18] ZHANG, RICHARD, ISOLA, PHILLIP, EFROS, ALEXEI A, et al. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *CVPR*. 2018. DOI: 10.1109/CVPR.2018.00068.

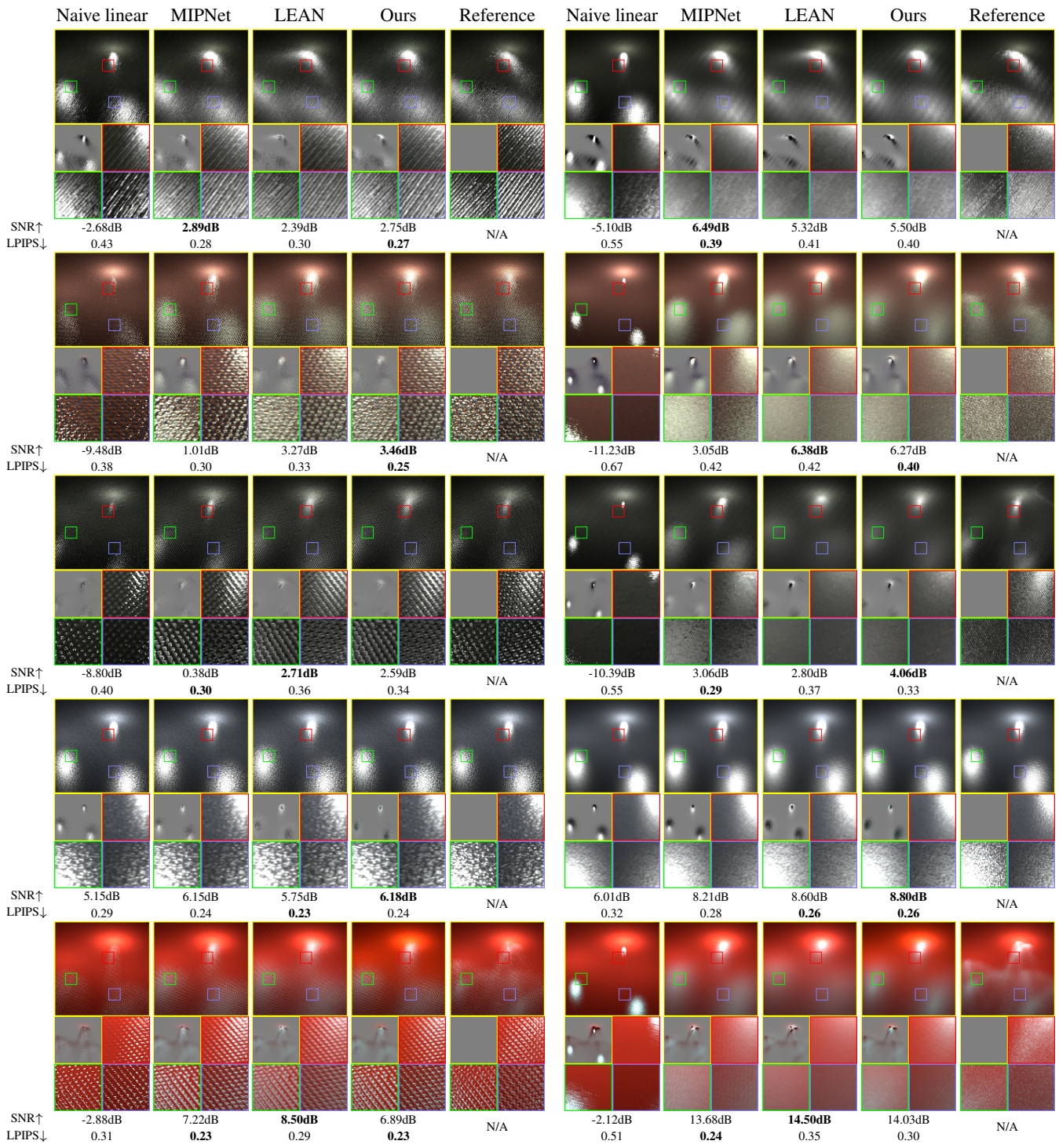


Figure 9: *Overview of the generalized Beckmann test set. Each row contains two views of the same material. The left column is viewed from slightly closer, the right column from further away. At both of these levels of detail, the methods are mostly comparable in quality.*

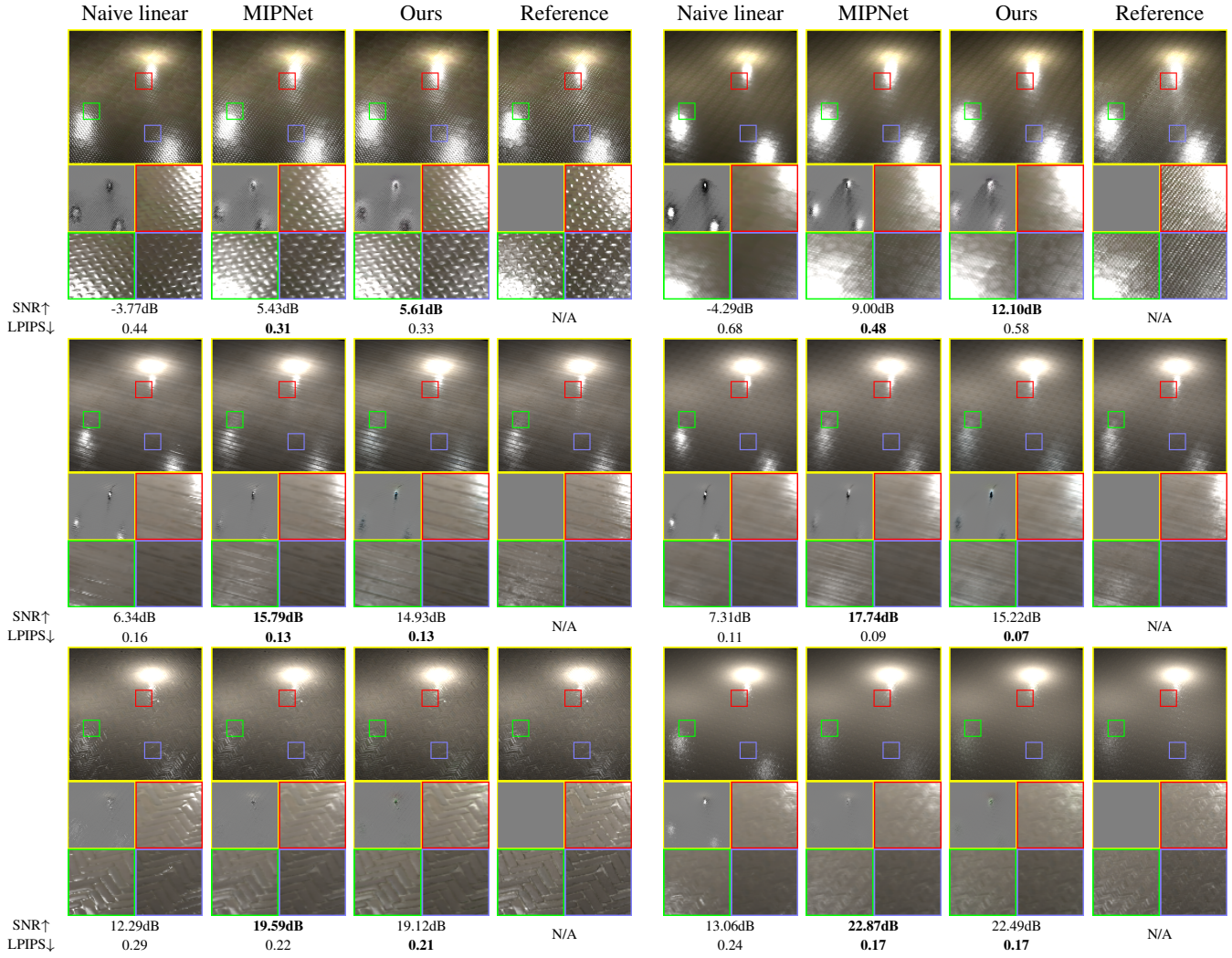


Figure 10: *Overview of the GGX test set. When viewed far enough so that most detail is gone, MIPNet and our method are comparable in quality. Slight color variations are visible in some of our highlights.*

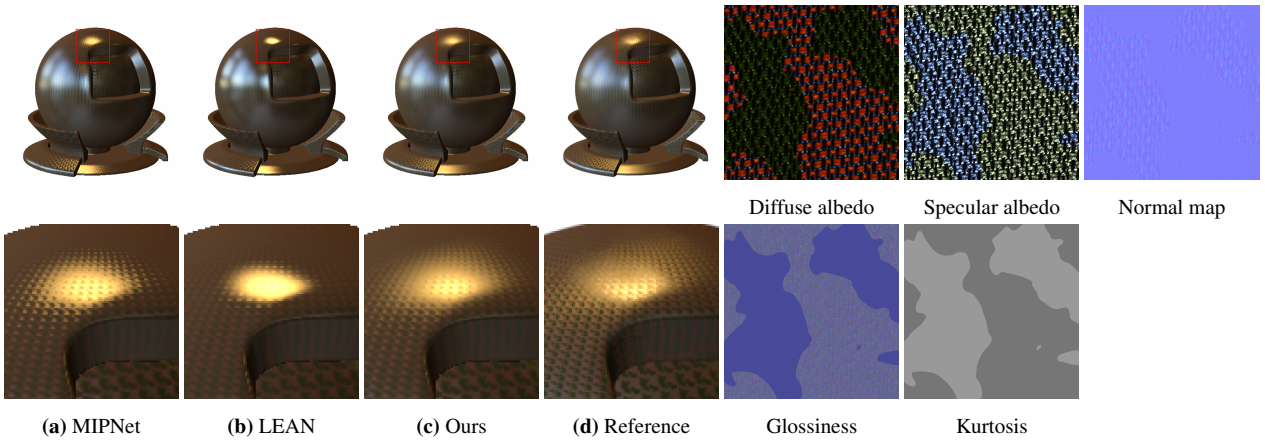


Figure 11: *A material with strong correlations between different channels. Our method is the only one that takes these into account.*