MIPNet: Neural Normal-to-Anisotropic-Roughness MIP mapping

Alban Gauthier, Robin Faury, Jérémy Levallois, Théo Thonat, Jean-Marc Thiery, Tamy Boubekeur

MIPNet: Neural Normal-to-Anisotropic-Roughness MIP mapping

ALBAN GAUTHIER, LTCI, Télécom Paris, Institut Polytechnique de Paris, France
ROBIN FAURY, Adobe Research, France
JÉRÉMY LEVALLOIS, Adobe Research, France
THÉO THONAT, Adobe Research, France
JEAN-MARC THIERY, Adobe Research, France
TAMY BOUBEKEUR, Adobe Research, France

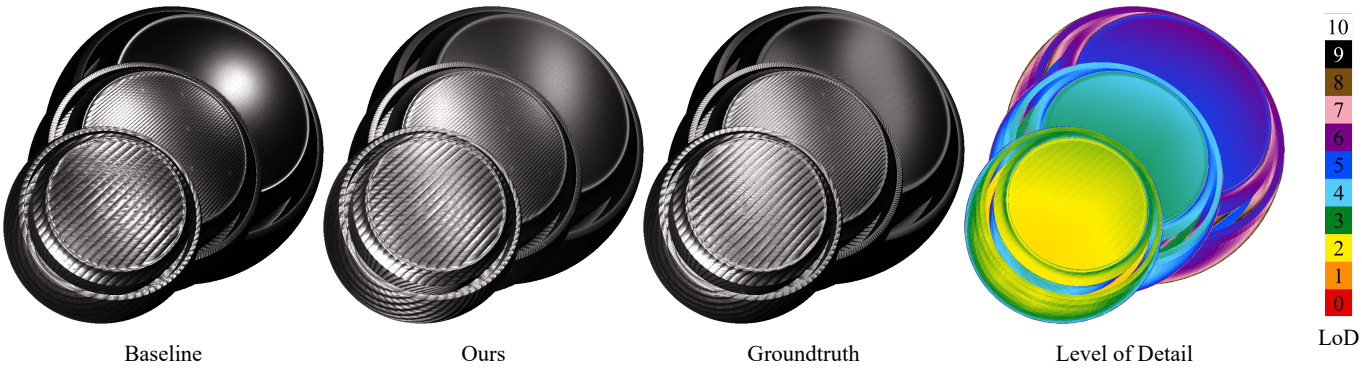


Fig. 1. From left to right: the standard per-channel linear mipmapping (baseline), our method (MIPNet), the groundtruth and the level of detail used for shading. Our learning-based approach allows to preserve the appearance of this anisotropic material unseen during training. Small, medium, and big shaderballs have respectively small, medium, and large uv repetitions.

We present MIPNet, a novel approach for SVBRDF mipmapping which preserves material appearance under varying view distances and lighting conditions. As in classical mipmapping, our method explicitly encodes the multiscale appearance of materials in a SVBRDF mipmap pyramid. To do so, we use a tensor-based representation, coping with gradient-based optimization, for encoding anisotropy which is compatible with existing real-time rendering engines. Instead of relying on a simple texture patch average for each channel independently, we propose a cascaded architecture of multi-layer perceptrons to approximate the material appearance using only the fixed material channels. Our neural model learns simple mipmapping filters using a differentiable rendering pipeline based on a rendering loss and is able to transfer signal from normal to anisotropic roughness. As a result, we obtain a drop-in replacement for standard material mipmapping, offering a significant improvement in appearance preservation while still boiling down to a single per-pixel mipmap texture fetch. We report extensive experiments on two distinct BRDF models.

Authors' addresses: Alban Gauthier, LTCI, Télécom Paris, Institut Polytechnique de Paris, France, albangauthier25@gmail.com; Robin Faury, Adobe Research, France, faury@adobe.com; Jérémy Levallois, Adobe Research, France, levalloi@adobe.com; Théo Thonat, Adobe Research, France, thonat@adobe.com; Jean-Marc Thiery, Adobe Research, France, jthiery@adobe.com; Tamy Boubekeur, Adobe Research, France, boubek@adobe.com.

0730-0301/2022/12-ART246 $15.00
https://doi.org/10.1145/3550454.3555487

CCS Concepts: • **Computing methodologies** → **Reflectance modeling**; **Texturing**.

Additional Key Words and Phrases: Material Appearance, Mipmaps, Machine Learning

ACM Reference Format:
Alban Gauthier, Robin Faury, Jérémy Levallois, Théo Thonat, Jean-Marc Thiery, and Tamy Boubekeur. 2022. MIPNet: Neural Normal-to-Anisotropic-Roughness MIP mapping. *ACM Trans. Graph.* 41, 6, Article 246 (December 2022), 13 pages. https://doi.org/10.1145/3550454.3555487

1 INTRODUCTION

Real-life materials display a diverse set of appearances that continuously vary with the distance to the observer. In computer graphics applications, accurately representing such a variety of appearances has been the focus of physically based shading (PBS), based on the microfacet model. It is considered an industry standard [Burley 2012; Karis 2013] for many real-time engines. The rendering of physically based materials relies on a set of spatially varying bidirectional distribution functions (SVBRDF), queried efficiently at runtime, that allows the reproduction of a wide range of material appearances.

However, efficiently rendering materials in constrained real-time scenarios requires material information to be encoded at a given scale for a given distance of observation. This loss of generality prevents efficient and accurate renderings, especially when zooming out. The pixel-to-texel ratio decreases with the distance, with one pixel covering a larger patch of the surface where textures are

mapped onto. This phenomenon leads to sampling artifacts characterized by temporal flickering and aliasing, illustrated by Moiré patterns or staircase effects.

Traditional anti-aliasing techniques for color textures rely on mipmapping [Williams 1983]. The method's underlying assumption is based on a linear relationship between texels in the textures to prefilter and the final rendered pixel. This prefiltering method is widely available on current graphics APIs and eliminates most of the typical rendering artifacts mentioned above. Unfortunately, using mipmapping independently and on various texture types, e.g., containing geometric information such as the normal and displacement maps, is erroneous. In their survey, Bruneton and Neyret [2012] state that the relationship between these material attributes and the final rendering is non-linear.

When looking at a material from afar, what were considered as macro- or meso-scale details become microscopic details (with respect to the distance of observation), that yield a new source of roughness of the surface. Several methods [Olano and Baker 2010; Hery et al. 2014; Dupuy et al. 2013] address this issue by transferring bump map (i.e., normal or displacement) information into roughness. However, the shading model used for the transfer is based on the Beckmann [1963] normal distribution function (NDF) and assumes a gaussian distribution of slopes at all scales, which is not compatible with the more commonly used Trowbridge-Reitz [1975] (often referred to as GGX [Walter et al. 2007]) distribution. This hypothesis fails for many PBR materials (e.g., highly structured) that exhibit a wide variety of geometric statistics depending on the distance to the observer. In addition, it has been noticed previously [Estevez et al. 2019] that the GGX lobe, being based on a Cauchy distribution, has undefined mean and variance. To circumvent this issue, Patry [2020] uses a heuristic based on the SGGX paper [Heitz et al. 2015] to linearly downsample SGGX matrices, of which the GGX distribution is a special case. This heuristic fails for many in-the-wild SVBRDF which exhibit strong patterns is their normal maps.

To remain fully compatible with existing real-time rendering engines and their PBR materials description, we aim at computing a mipmapping operator that improves upon the de-facto standard 4-to-1-texel linear average, without requiring expensive computations when processing a given input SVBRDF. Specifically, we design this operator to preserve the main characteristics of standard mipmapping, which are (i) low storage cost and (ii) low computation time of the randomly-accessible created mipmaps.

To that end, we propose to replace the simple mipmapping operator with a learned downsampling filter based on a multilayer perceptron cascade, which learns information transfer across level-of-details, and allows for generalization over unseen materials. To do so, we employ a differentiable rendering pipeline along with a rendering loss. Each input texture patch is given as input to the neural architecture which concurrently downsamples each channel of the SVBRDF. The result is fed to the renderer and compared to a multisampled groundtruth computed on the fly.

We focus our efforts on two anisotropic BRDF models to account for appearance changes at varying scales, namely Cook-Torrance [1982] with the GGX and Beckmann variants for the normal distribution functions (NDF), and the Ashikhmin-Shirley [2000] (a.k.a. Anisotropic Phong) model. These models grant a wider variety of highlights, especially for isotropic materials which showcase anisotropic behavior when viewed from afar (see Fig. 1).

Our contributions are **(i)** an efficient pipeline for learning mipmapping filters requiring no data preparation for training, **(ii)** a neural architecture encoding anisotropic appearance and generalizing on unseen materials, and **(iii)** a tensor-based formulation for anisotropic BRDF distributions which is well-suited for differentiable pipelines and trilinear interpolation.

2 PREVIOUS WORK

Texture minification. Mipmapping was first introduced by Williams [1983] and provides an efficient solution to approximate multiple samples at runtime by precomputing a prefiltered image pyramid using Box, Gaussian, Lanczos or Kaiser filters [Akenine-Möller et al. 2018]. At rendering time, this mipmap pyramid is accessed using trilinear interpolation and often considers the anisotropy of the pixel footprint in texture space [Manson and Schaefer 2012]. Our method aims at providing an image pyramid for each SVBRDF map by carefully crafting a downsampling kernel. Image downsampling methods optimize kernels based on image features at multiple scales [Kopf et al. 2013] or using the SSIM metric [Öztireli and Gross 2015]. In our framework, downsampling kernels are learned using a neural network which allows for generalization across multiple materials.

Normal map filtering. Normal distributions can be approximated using a single isotropic or anisotropic lobe. Schilling [1997] identified that roughness information can be derived from the normal map and encodes normal aggregates in a covariance matrix, while Olano and North [1997] use a single 3D Gaussian lobe. Later, Toksvig [2005] proposed to compute the width of the NDF based on the accumulated normals' length. Since a single lobe is often not enough, many previous works proposed to encode the NDF into multiple lobes. Han et al. [2007] propose a framework which generalizes previous works on the topic [Fournier 1992; Tan et al. 2005] and encode averaged information into spherical harmonics (SH) or von Mises-Fisher (vMF) distributions. Such techniques require a custom shading while we provide a drop-in replacement for standard mipmapping.

The most recent and widely adopted techniques, LEAN [Olano and Baker 2010], LEADR [Dupuy et al. 2013] and Pixar's Bump to Roughness [Hery et al. 2014], make use of additional maps encoding geometric statistics, namely the mean and/or variance of the normal map or displacement map, which behave well with classic mipmapping. They use an anisotropic Beckmann distribution to represent the appearance at all scales and assume that the small-scale geometry contained in the normal map follows gaussian statistics. This assumption does not hold for many materials and is incompatible with other BRDF distributions such as Phong or GGX. We propose a comparison between our model learnt on the Beckmann model and LEADR, which limitations are further discussed in Sec 4.

Patry [2020] recently proposed a SGGX-based filtering, compatible with GGX distribution, to filter anisotropic specular maps. The GGX distribution is shown to be a special case of the SGGX distribution, which is encoded using a tensor-based representation. The

original paper [Heitz et al. 2015] provides a parameter estimation algorithm to convert a given NDF into SGGX matrix parameters. However, they show that linear interpolation of matrix parameters is a good approximation of this algorithm. Hence, Patry propose to linearly filter SGGX matrix parameters extracted from an anisotropic GGX distribution at runtime. The eigenvalues and eigenvectors of the filtered matrix are extracted and fed to a classical anisotropic GGX distribution. We take inspiration from their method and use a tensor-based representation which behaves well with linear filtering. However, we avoid the eigenvalue decomposition which leads to artifacts for extreme tensor value configurations. We propose a comparison between our model learnt on the GGX distribution and their work in Sec 4, which we call SSGT (Samurai Shading at Ghost of Tsushima)

Rendering high-resolution normal maps. Efforts have been made to render microstructures such as glints or scratches, since they require higher resolution normal maps which result in highly complex NDFs. Offline as well as real-time [Zirr and Kaplanyany 2016; Chermain et al. 2020; Tan et al. 2022] techniques have been proposed for this task. Such methods focus on a specific appearance problem we are not addressing. Please refer to Zhu et al. [2022] for a complete related work on the topic.

Reflectance filtering. More generally, previous work addressed multiscale appearance preservation by filtering at the shading level. Please refer to Bruneton and Neyret [2012] for a review on the topic. Becker and Max [1993] proposed a method to smoothly transition between displacement, bump mapping and BRDF in a unified framework. Later, efforts were put on the representation of aggregated BRDFs. Claustres et al. [2007] used wavelet encoding, while Tan et al. [2008] generalized the scope of their previous work on Gaussian mixture model. Xu et al. [2017] proposed a framework to encode SVBRDF and normals into BRDF mipmaps, and to filter the latter in real-time. Heitz et al. [2013] proposed to address the non-linear behavior of rendering color textures mapped onto surfaces. More recently, Wu et al. [2019] focused on appearance preservation of displaced surfaces by jointly prefiltering the displacement map and the SVBRDF. These techniques require engine modifications to be applied in a real-time context. Putting SVBRDF aside, several works proposed to tackle volume filtering [Loubet and Neyret 2017; Zhao et al. 2016] or study the effect of filtering when dealing with BTF data [Jarabo et al. 2014].

When rendering materials at different scales, aliasing occurs, especially for highly specular materials. Several works [Tokuyoshi and Kaplanyan 2021; Chermain et al. 2021] proposed to tackle Geometric Specular Antialiasing when rendering highly specular materials. These techniques can be applied as a post-process to our method, but do not focus on SVBRDF prefiltering.

Differentiable rendering. Recent advances in differentiable rendering pipelines allow to optimize rendering data (interpretable or implicit) so that its appearance matches the final pixel colors. This is done by backpropagating gradients of an image-based loss through the rendering pipeline. Several material acquisition methods have been proposed [Deschaintre et al. 2018; Guo et al. 2020; Zhou and Kalantari 2021] which allow for SVBRDF parameters recovery using a rendering loss. These methods allow to create the SVBRDF map at a single scale, and do not solve the challenge of mipmapping the SVBRDF.

Even though SVBRDF maps can be computed explicitly, Kuznetsov et al. [2021] propose to encode material appearance in a level-of-detail pyramid of neural textures, which are fed to a decoder network. By using two multilayer perceptrons (a Neural Offset Module and a Texture Decoder) along with a Neural Texture Pyramid, they encode complex material information which can be queried in real time. However, this technique is not fit for traditional rasterization pipelines. The Neural Texture Pyramid creates a material specific implicit representation which is tied to the MLP Decoder to output radiance information. Rainer et al. [2019] proposed to learn BTF and BRDF [Rainer et al. 2020] encoding using fully-connected multilayer perceptrons. Neural Radiance Fields [Mildenhall et al. 2020] allow to encode spatio-angular radiance information inside a volume using a differentiable framework. This method has been extended to support mipmapping [Barron et al. 2021] and SVBRDF-like parameters [Boss et al. 2021]. However, NeuMIP and Radiance Field-like solutions do not tackle SVBRDF prefiltering, but rather encoding shape and appearance into a neural network, possibly at multiple scales. We stress that we aim at preserving the structure and content of interpretable SVBRDF maps so that they can directly be used inside current renderers without requiring any engine code modification.

Hasselgren et al. [2021] proposed a pipeline to concurrently simplify geometry and textures using an appearance-based framework. Their work automates the task of decimating geometry manually. The method allows to transfer geometric details of the mesh in the normal and displacement maps (via joint shape-appearance simplification) and is able to construct a per material mipmap pyramid. Their method is only fit for a couple composed of a geometry and some maps, hence is incapable of any generalization over a similar dataset. Our work aims at using the learned downsampling kernels for other materials to prevent a lengthy optimization.

3 METHOD

3.1 Training from anisotropic SVBRDF models

To demonstrate our technique, we focus on two commonly used BRDF models: the Cook-Torrance microfacet model along with the anisotropic GGX [Walter et al. 2007] and Beckmann [1963] microfacet distributions, using the Schlick [1994] Fresnel term and the uncorrelated Smith shadowing-masking term [Heitz 2014]; and the Ashikhmin-Shirley model [2000]. Our mipmapping framework takes as input the base color, normal, metallic, height (or displacement) and two roughness maps along with an anisotropy angle map. When the input material is isotropic, the two roughness maps are identical and the angle map contains only 0-degree angles. The base color map contains both diffuse albedo and specular information and is modulated by the metallic map. This results in a linear relationship between diffuse albedo, specular color, and the resulting radiance. On top of influencing the shading for viewpoints above the surface, the height channel impacts the coarse geometric aspect of the shape through the definition of its silhouettes. For this reason, we rely on well-established geometric simplification methods for the height channel and use as baseline the de facto standard linear mipmapping.

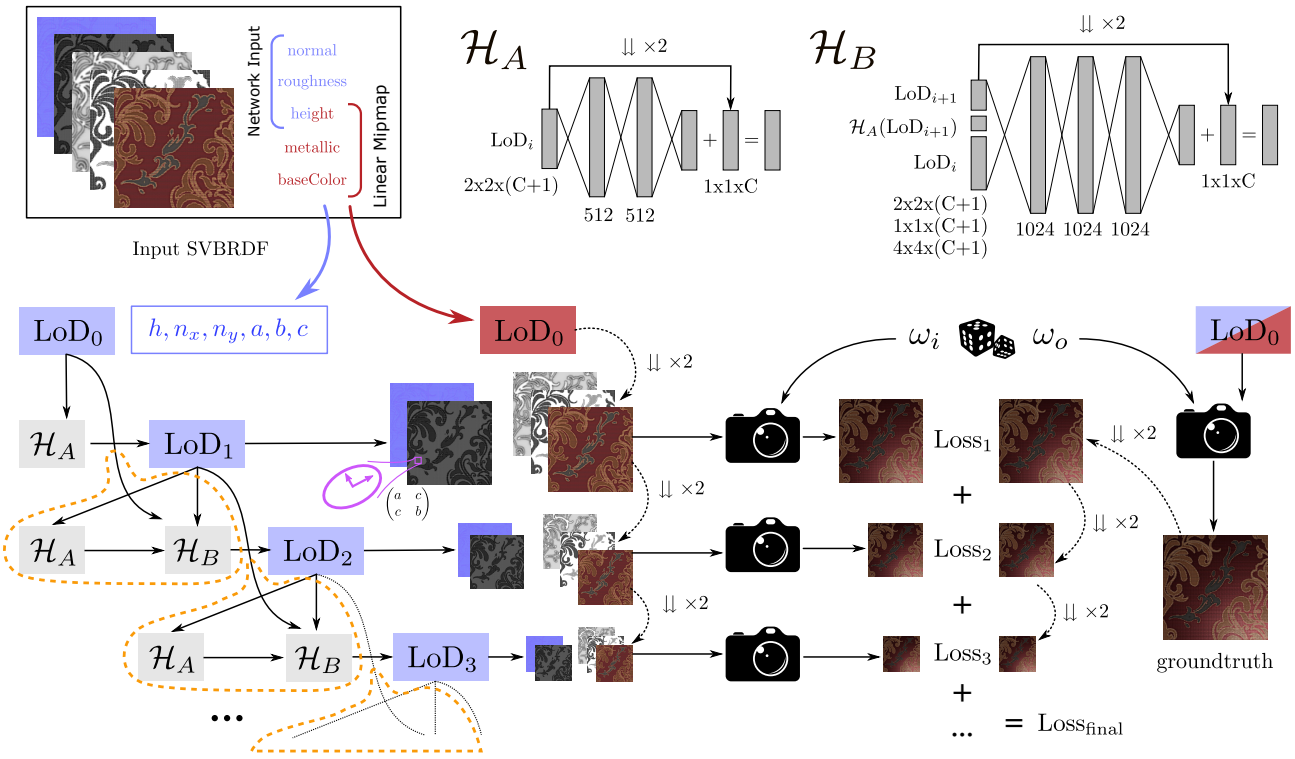


Fig. 2. Overview of our training process. Mipmap levels LoD$_i$ are generated using a cascaded architecture of multilayer perceptrons. The cascade of networks takes as input a height map h, a normal map (n_x, n_y), and the roughness map which provides the initial tensor coefficients (a, b, c). The height, metallic and baseColor maps are linearly downsampled and used along with the outputs of the cascade to be rendered (using random light ω_i and view ω_o directions) and used in the final loss term. This cascade is composed of successive blocks of $\mathcal{H}_A$ and $\mathcal{H}_B$ (circled in dotted orange) which architectures are detailed in the top right corner. $\mathcal{H}_B$ requires a larger capacity than $\mathcal{H}_A$ because of its higher dimensional input. The groundtruth is computed using the full resolution input SVBRDF.

Thus we focus our efforts on mipmapping the remaining maps that most influence the shading at different scales, namely normal and roughness.

We follow those well-established common good practices:

(1) We interpolate (and average trilinearly when rendering from the created mipmaps) the linearly-perceptual roughness α instead of the roughness r. Burley [2012] suggests indeed that $\alpha = r^2$ remaps correctly the roughness in $[0, 1]$, resulting in a progressive linear behavior for the remapped roughness.
(2) We forbid too small values for the roughness, to avoid numerical instabilities when using GGX or Beckmann distributions with point lights. Following Lagarde and de Rousiers [2014], we enforce $r \geq \epsilon_r = 0.045$, resulting in $\alpha \geq \epsilon_\alpha = \epsilon_r^2 \simeq 0.002$.
(3) Because we aim at mipmapping (and averaging) non-axis-aligned roughness values, we use a tensor-based representation. Since the angle-based representation is not unique (adding π to the angle results in the same anisotropy, and isotropic tensors have undefined angles by definition), averaging 4 of those representations during mipmapping (or rendering using the mipmaps) is an ill-defined operation, while averaging the tensors is well-defined. This common observation motivated among others the SGGX-based filtering [Patry 2020], to which we compare our technique in this paper.

Following these last observations, we encode the anisotropic linearly-perceptual roughness as a symmetric 2×2 tensor:

$$A := R_\gamma \begin{pmatrix} \alpha_b & 0 \\ 0 & \alpha_t \end{pmatrix} R_\gamma^T := \begin{pmatrix} a & c \\ c & b \end{pmatrix}. \tag{1}$$

Given this definition, note that A^2 has the same eigenvectors as A, and its eigenvalues are the squared eigenvalues of A:

$$A^2 = R_\gamma \begin{pmatrix} \alpha_b^2 & 0 \\ 0 & \alpha_t^2 \end{pmatrix} R_\gamma^T \tag{2}$$

We provide in Appendix A the expressions for the anisotropic GGX, Beckmann and Ashikhmin-Shirley models using this representation, which we use in our renderer and learning architecture.

When learning (or mipmapping, or averaging) the (a, b, c) channels of A, we have to ensure that these values correspond to physically-plausible roughness values (i.e., the eigenvalues of A): $\epsilon_\alpha \leq \alpha \leq 1$. We derive the following constraints on A's trace, determinant, and

eigenvalues:

$$\det(A) = ab - c^2 = \alpha_b\alpha_t \geq \epsilon_\alpha^2$$
$$\text{trace}(A) = a + b = \alpha_b + \alpha_t \geq 2\epsilon_\alpha$$
$$\epsilon_\alpha \leq a = \cos^2(\gamma)\alpha_b + \sin^2(\gamma)\alpha_t \leq 1$$
$$\epsilon_\alpha \leq b = \sin^2(\gamma)\alpha_b + \cos^2(\gamma)\alpha_t \leq 1$$

Constraining the largest eigenvalue to be less than 1 leads to:

$$\frac{a + b + \sqrt{(a-b)^2 + 4c^2}}{2} \leq 1 \Leftrightarrow$$
$$c^2 \leq (1-a)(1-b)$$

Similarly, constraining the smallest one to be larger than ϵ_α leads to:

$$\frac{a + b - \sqrt{(a-b)^2 + 4c^2}}{2} \geq \epsilon_\alpha \Leftrightarrow$$
$$c^2 \leq (a-\epsilon_\alpha)(b-\epsilon_\alpha)$$

In summary, the following constraints need to be met:

$$\epsilon_\alpha \leq a \leq 1 \tag{3}$$
$$\epsilon_\alpha \leq b \leq 1 \tag{4}$$
$$c^2 \leq \min(ab - \epsilon_\alpha^2, (1-a)(1-b), (a-\epsilon_\alpha)(b-\epsilon_\alpha)) =: c_{max}^2 \tag{5}$$

To ensure these, we propose a simple projection procedure, which leads to good results in practice in our observations. We first clamp a and b within $[\epsilon_\alpha, 1]$, before clamping c within $\left[-\sqrt{c_{max}^2}, \sqrt{c_{max}^2}\right]$. Note that this projection is not an orthogonal projection onto the space of *valid* (a, b, c) tensor channels, but it is close enough in practice for our application scenarios. Note also that the first two conditions on (a, b) ensure that c_{max}^2 is positive, which makes our procedure valid for any input (a, b, c) channels. These bounds ensure physically-plausible roughness values, and result in increased stability of gradient-based optimizations, as they prevent the BRDFs we consider (see Appendix A) from taking negative or infinite (undefined) values.

3.2 Mipmapping

We define the BRDF as a function $f : \mathbf{x}, \omega_i, \omega_o \in \mathbb{R}^d \times \Omega^2 \rightarrow \mathbb{R}^3$, where $\mathbf{x}$ is a vector of d material parameters. For a texel at position $p \in \mathbb{R}^2$, the k^{th} level-of-detail of a SVBRDF pyramid is expressed as follows:

$$\text{LoD}_k : p \in \mathbb{R}^2 \rightarrow \text{LoD}_k(p) \in \mathbb{R}^d \tag{6}$$

Classic mipmapping relies on a simple texel average per channel. This supposes that there is a linear relationship between the SVBRDF maps and the BRDF. However, as described by Bruneton and Neyret [2012], it is not the case in general. We aim at finding a translation-invariant kernel $\mathcal{H}$, which computes any texel at LoD_k from LoD_{k-2} and LoD_{k-1}, such that for any level-of-detail k, and position p:

$$\text{LoD}_k(p) \coloneqq \mathcal{H}(\{\text{LoD}_{k-2}(\mathcal{P}_{k-2}(p)), \text{LoD}_{k-1}(\mathcal{P}_{k-1}(p))\}) \tag{7}$$

where $\mathcal{P}_k(p) \subset \mathbb{R}^2$ covers the footprint of texel p in LoD_k.

We can construct the mipmap by applying the kernel over all patches of the $i - 2^{th}$ and $i - 1^{th}$ levels and get the level i. To do so, we learn a kernel model to minimize the difference between rendered values. Given a SVBRDF at base level LoD_0, we minimize the L_1 distance over all positions p between the ground truth radiance of the material computed from averaged radiance values corresponding to the footprint of p, called GT, and the rendering of a single sample per pixel radiance at position p. We define the rendering loss L_{total} to minimize as follows:

$$L_{\text{total}} \coloneqq \sum_{k \geq 1} \sum_{p \in \text{LoD}_k} \sum_{\omega_i, \omega_o \in \Omega} L_k(p, \omega_i, \omega_o), \text{ s.t.} \tag{8}$$
$$L_k(p, \omega_i, \omega_o) = \|f(\text{LoD}_k(p), \omega_i, \omega_o) - \text{GT}(p, \omega_i, \omega_o)\|_1, \tag{9}$$
$$\text{GT}(p, \omega_i, \omega_o) = \frac{1}{\#\{\mathcal{P}_0(p)\}} \sum_{x \in \mathcal{P}_0(p)} f(\text{LoD}_0(x), \omega_i, \omega_o) \tag{10}$$

For the computation of LoD_1, only LoD_0 is required. Note that our loss depends on the actual BRDF model used for rendering, but any model which supports differentiation can be used beyond the ones we chose to illustrate our approach.

3.3 Overview

Neural architecture. Our normal-roughness downsampling kernels are implemented as a cascade of multiple fully-connected MLPs (see Fig.2): $\mathcal{H}_A$ and $\mathcal{H}_B$, which downsample at half and quarter resolution, respectively. In all of our experiments, we use four occurrences of the networks $\mathcal{H}_A$ and $\mathcal{H}_B$ (i.e., the orange block in Fig. 2 is repeated four times). To better capture the anisotropic behavior of certain materials, each network is required to process anisotropic data, even when these are missing from the base LoD. We here follow the common assumption that isotropic materials may appear as anisotropic when seen from afar. Compared to a single-level architecture (Sec. 3.4), our cascade grants each network visibility over complex multiscale phenomena which only appear progressively in the LoD, while being efficient even if shallow (Sec. 3.5). Enforcing network compactness – by limiting the size and the number of the hidden layers of the MLPs to a minimum – leads to better generalization [Goodfellow et al. 2016], prevents material overfitting and offers faster training and inference. Hence, we apply the networks on small local texel footprints to keep the number of weights to a minimum. $\mathcal{H}_A$ processes 2x2 patches of material parameters, where $\mathcal{H}_B$ takes as inputs three patches of size 4x4, 2x2 and 1x1. Also, we noticed faster convergence when learning differences from the averaged maps. Hence, we use the linearly downsampled versions of LoD_0 and LoD_1 in addition with the network output to compute LoD_1 and LoD_2, respectively.

SVBRDF processing. Our pipeline aims at concurrently downsampling normal and anisotropic roughness maps through our learned kernels. We provide patches of the full resolution maps once to $\mathcal{H}_A$, which outputs a 2x downsample. This output is fed a second time to $\mathcal{H}_A$, which results in a 4x downsample. $\mathcal{H}_B$ gets the original, 2x and 4x downsampled maps as input and produces a 4x downsample. The metallic and albedo maps are linearly downsampled at half and quarter resolution and provided to the renderer, along with the downsampled normal and roughness maps from the first output of $\mathcal{H}_A$ and the ones from $\mathcal{H}_B$. This allows to render a texture patch

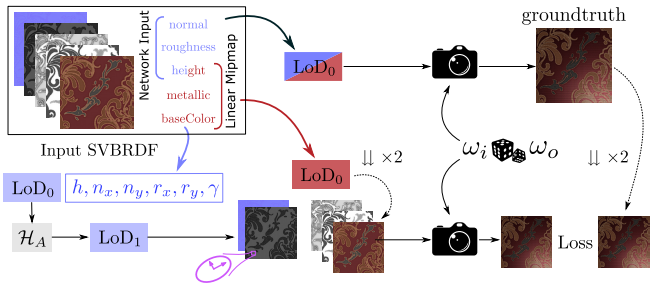


Fig. 3. Simplified, single resolution training pipeline.

at half and quarter of the original resolution. The height is downsampled linearly and is used to compute the light position in the rendering pass. Alternatively, height downsampling can optimize for error quadrics [Garland and Heckbert 1997; Trettner and Kobbelt 2020].

Physically-based renderer. We implement a differentiable physically-based renderer in PyTorch. As input, we provide the downsampled material maps, as well as a lighting setup. We render materials on a plane viewed perpendicularly which eases UV (texels)/screen (rendered pixels) mapping. Tonemapping the radiance is common to prevent numerical instabilities for gradient descent optimization methods [Deschaintre et al. 2018; Kuznetsov et al. 2021; Hasselgren et al. 2021]. We compute the rendering loss after tonemapping because our aim is to optimize mipmaps according to the perceived appearance, rather than the raw responses. The operator is interchangeable in the training, and can adapt to the downstream implementation to which the mipmaps will be fed. We tested multiple tonemappers for learning our mipmapping filters and found out a simple Reinhard [2002] worked better than e.g., a logarithmic tonemapper as used by Kuznetsov et al. [2021].

Training loss. We divide our loss in a sum of terms computed on the tonemapped radiance values of our renders, where Loss_i corresponds to the computation of level-of-detail LoD_i. Each loss is computed as the L_1 per-pixel distance between ground truth and mipmapped renderings. To compute the groundtruth, we render materials with a 1:1 pixel to texel ratio at the original SVBRDF resolution. The rendering is then bilinearly downsampled until it reaches the target resolution for the corresponding loss.

3.4 Single-resolution neural architecture

To warm up, let us first discuss the use of a single 2x downsampling kernel $\mathcal{H}_A$ (see Fig. 3). This pipeline learns LoD_1 from LoD_0 only. The inference for creating the mipmap pyramid is straightforward: each level of detail LoD_{i+1} is computed from the previously given level LoD_i using $\mathcal{H}_A$.

This model already provides improvements over a simple average of all maps in the SVBRDF mipmap pyramid, even when using a small number of network parameters (2 fully connected hidden layers, both of size 16). The network is able to generalize over unseen materials when trained with a sufficient number of materials. Similarly to LEAN and LEADR, we observe a transfer from normal to roughness where the variance among normals over a patch is high. However, this simplified model suffers from several artifacts. When applied successively at each level of the pyramid, the network tends to accumulate errors across previously unseen LoDs. Additionally, this architecture cannot reliably create effects in the mipmap, which were not present in the input, such as anisotropy. Indeed, such effets are not present in the training set. This motivates our *cascaded* architecture, where $\mathcal{H}_A$ learns to handle such data as input on top of outputting it.

3.5 Multi-resolution neural architecture

Our full architecture (Fig. 2) is designed to make $\mathcal{H}_A$ learn to output complex effects such as anisotropy as well as treat it robustly as input, even when these effects are not present in the first level-of-detail, but emerge progressively from the mesostructures convolved at higher mip levels. Rather than simply learning jointly several levels of the mipmap (which we evaluate in Sec. 4), we introduce a two-level network $\mathcal{H}_B$ jointly trained with $\mathcal{H}_A$ (orange dotted group in Fig. 2) that learns to best account for two-scale descriptors and offers a robust cascaded inference mechanism.

Our final multi-resolution architecture requires a 4×4 footprint for the computation of a single texel in the coarser mipmap levels, which makes the model compact enough to allow for robust generalization capabilities, and fast enough at mipmap inference (less than a second for the whole pyramid).

Architecture hyper-parameters. In our experiments, we found that setting $\mathcal{H}_A$ to be composed of 2 hidden layers of size 512 and $\mathcal{H}_B$ of 3 layers of size 1024 resulted in high-quality results and a good generalization behavior, indicating that our model is compact enough to avoid overfitting.

3.6 Training setup

Ground truth training data. We train our model with a ground truth computed just-in-time, in the sense that no precomputation is required prior to training. In our implementation, we use the same rendering pipeline for training and computing the ground truth, since differentiation can simply be disabled for the latter.

Training details. We noticed that training the network using only a few light directions leads to an unstable training. To improve convergence, we use up to 32 sample points of the Hammersley sequence distributed on the hemisphere. We use batches of size 16 using 32×32 texture patches to compute the loss. We adopt a learning rate of 0.0001 along with an Adam [2015] optimizer.

4 RESULTS

In this section we compare our technique with previous work, both with methods performing generic on-the-fly SVBRDF mipmapping, and with methods relying on per-material optimizations. We report quantitative comparisons in Table 1 and illustrate our main results in Figure 4 for the Ashikhmin-Shirley model, and in Figures 5 and 6 for the Cook-Torrance model with the GGX and Beckmann NDFs. These figures feature results of the materials from several point of views to showcase the main effects appearing in the created

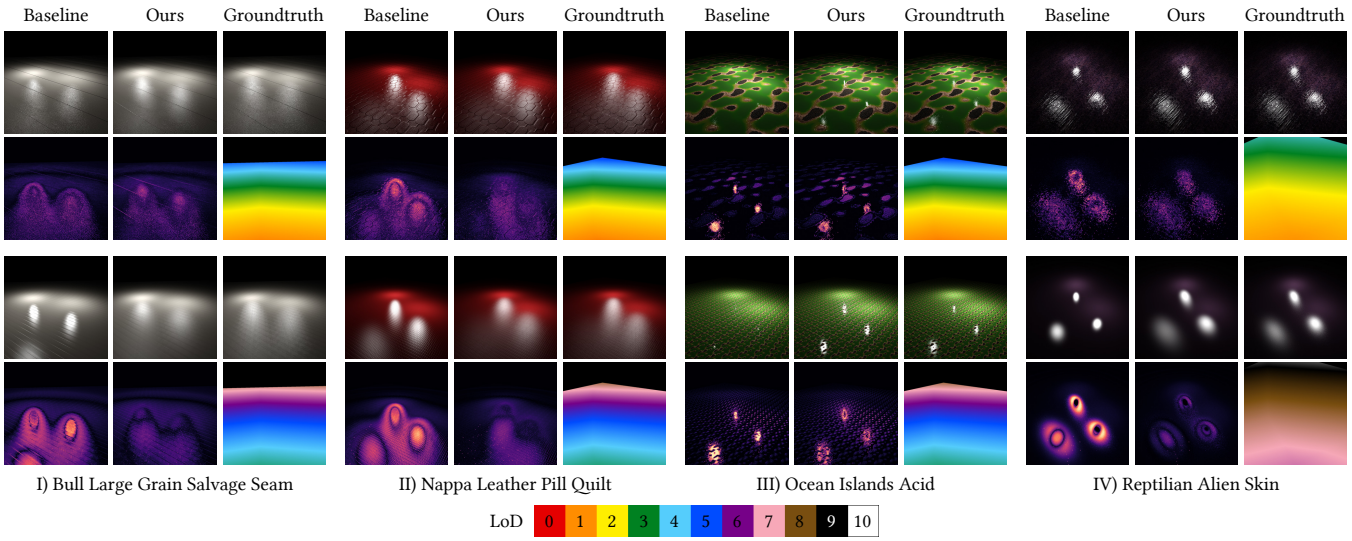


Fig. 4. Renderings of materials using the Ashikhmin-Shirley model under varying view conditions. We compare our method to the standard linear mipmapping (baseline) on four challenging materials *(more in additional material)*. Every second row shows the pixel-wise ꟻLIP deviation to the groundtruth, as well as a false color image depicting the mipmap LoD used for rendering (with trilinear filtering).

Table 1. Quantitative comparisons between the baseline, our method and the competition on each BRDF variant. The train set is composed of 1104 different materials from 14 categories. The test set is composed of 100 materials, identified as the most difficult (given by baseline error). The best result for each comparison is highlighted in bold.

BRDF	method	train set (error $\times 10^{-3}$)			test set (error $\times 10^{-3}$)		
		ꟻLIP	L1	MSE	ꟻLIP	L1	MSE
GGX	baseline	44.19	**9.76**	1	67	15.3	1.93
	SSGT	52.66	12.47	1.4	73.32	17.59	2.07
	ours	**43.33**	9.77	**0.8**	**64.55**	**14.69**	**1.54**
Beck.	baseline	49.76	11.57	1.49	75.23	18.37	2.94
	LEADR	108.19	25.56	3.12	154.23	44.36	8.95
	ours	**49.4**	**11.5**	**1.11**	**72.07**	**17.19**	**2.12**
A-S	baseline	53.24	11.64	1.6	78.64	18.32	2.78
	ours	**52.32**	**11.42**	**1.28**	**76.33**	**17.52**	**2.3**

mipmaps. ***Note that we provide all these scenes along with our viewer as additional material.***

Our method relies on an offline preprocessing step, which optimizes the weights of our neural networks. The training of this model was performed on 1104 materials using 40 epochs, taking approximately 3h per epoch using a Nvidia V100 GPU, and requires less than 2Go of VRAM. The test set is composed of 100 materials unseen during training. Once trained, our model generates the prefiltered version of any given material with linear complexity in the number of texels. Our (unoptimized) python implementation takes less than a second to generate the full mipmap for 4096×4096 SVBRDF maps.

4.1 Comparison to generic SVBRDF mipmapping

We compare our method with state-of-the-art techniques which generate materials mipmaps consumed by standard shading integrators and report quantitative comparisons in Table 1. We compare between a baseline, MIPNet and two competitor techniques for the Cook-Torrance model with GGX and Beckmann NDFs. For the baseline, we use the common 4-to-1 texel averaging implemented by `glGenerateMipmap` in OpenGL [Khronos Group 2022]. For the competitors, we compare with SSGT [Patry 2020] for the GGX model and LEADR [Dupuy et al. 2013] for the Beckmann model. As our method, these techniques can process a material instantaneously. To the best of our knowledge, only the baseline has been used so far to produce a mipmap pyramid for the Ashikhmin-Shirley model. In the next paragraphs, we evaluate the ability of our method to preserve the material appearance at different scales, under various point-of-views and lighting directions compared with their respective competitors.

Comparison with the baseline. We present renderings of our approach compared to the baseline in Figures 4, 5 and 6. The baseline tends to produce wrongly concentrated highlights due to the underestimated roughness (see materials I, II, III and IV in Fig. 4, materials II and VI in Fig. 5, and materials I, II, III, IV and V in Fig. 6). In addition, the baseline creates bright isotropic specular spots which do not capture anisotropy (see material IV in Fig. 4, materials II and VI in Fig. 5, and material VI in Fig. 6). Note that our method performs similarly to the baseline on simple materials which showcase relatively flat normal maps, since the transfer from normal to anisotropic roughness is then limited. In this case, our method outputs flat normal maps, which do not affect the roughness filtering, similarly to the baseline. Finally, note that our approach improves **on average** over the baseline, but some specific regions can still be

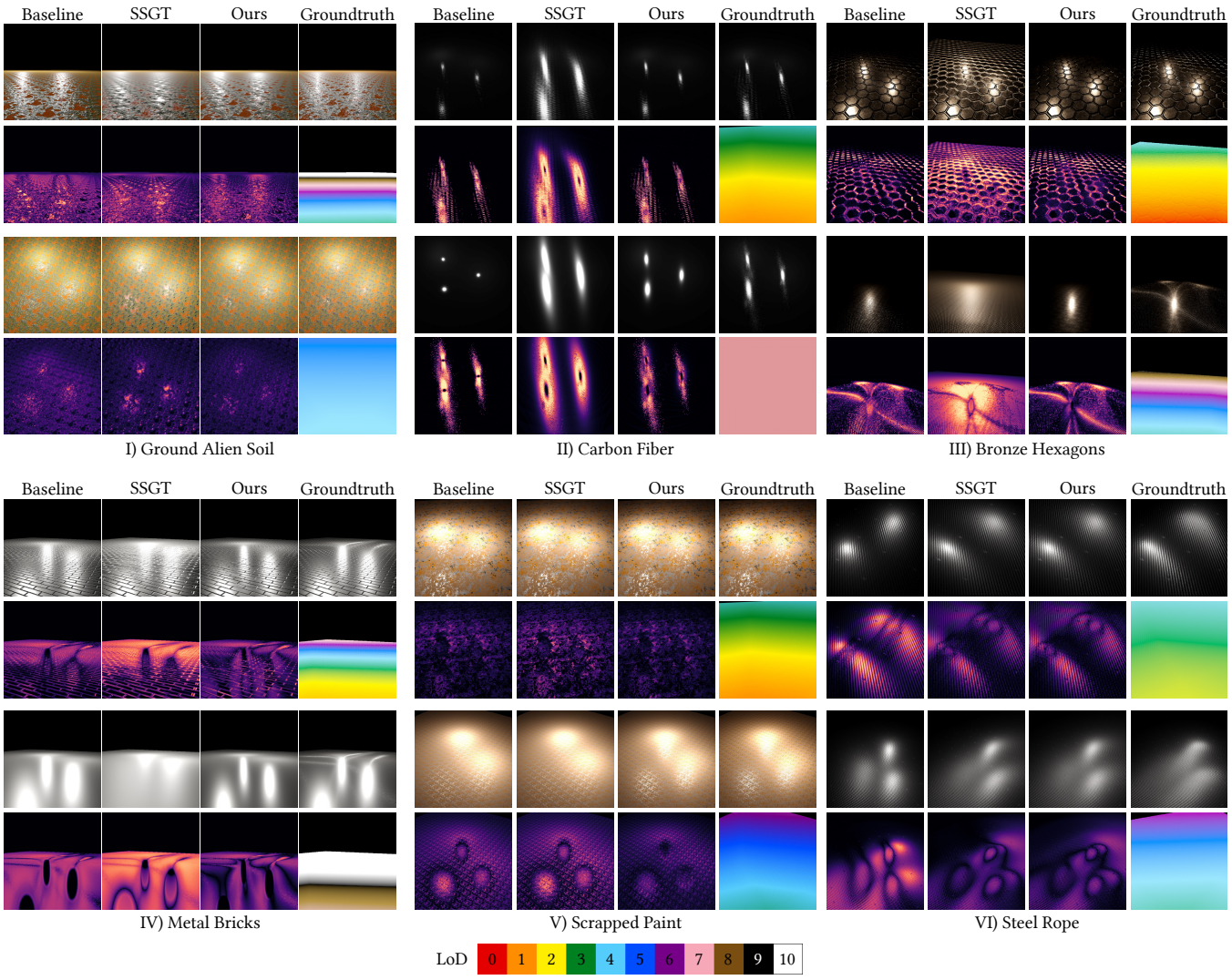


Fig. 5. Renderings of materials using the GGX model under varying view conditions. We compare our method to the standard linear mipmapping (baseline) and main competitor specialized in GGX material mipmapping (SSGT) on six challenging materials *(more in additional material)*. Every second row shows the pixel-wise ꟻLIP deviation to the groundtruth, as well as a false color image depicting the mipmap LoD used for rendering (with trilinear filtering).

better treated by the baseline (see Fig. 1, bottom part of the front shaderball, for a flagrant example). We provide renders from multiple viewpoints (both in terms of view angles as well as distance to the scene) in the supplementary materials.

GGX: comparison with SSGT. We provide renderings using our technique for the GGX distribution in Fig. 5, in which we compare our results with the baseline and with SSGT [Patry 2020] which is specialized in GGX distributions mipmapping.

We remind that with SSGT, the anisotropic roughness and the normal are packed into a 3×3 symmetric tensor (sharing similarities with our roughness encoding on that point), and this representation is mipmapped. At shading time, the local normal is extracted from the mipmapped linearly-interpolated tensor as its largest eigenvector, and the 2×2 roughness tensor as the orthogonal part to the normal.

While we observe in practice a behavior closer to the groundtruth than the baseline on many examples, this technique can extract strongly biased lobes on challenging materials, where the normal map features strong discontinuities (materials II, III, and IV in Fig. 5), and is often further away from the groundtruth than the baseline on this type of examples. Our results consistently feature the overall anisotropy effects appearing when the scene is rendered from further away on these challenging examples. Note that we still observe cases where our approach leads to incorrect results on some viewpoints. We analyze this phenomenon in the limitations section (see Section 5). *More results can be found in the supplementary material.*

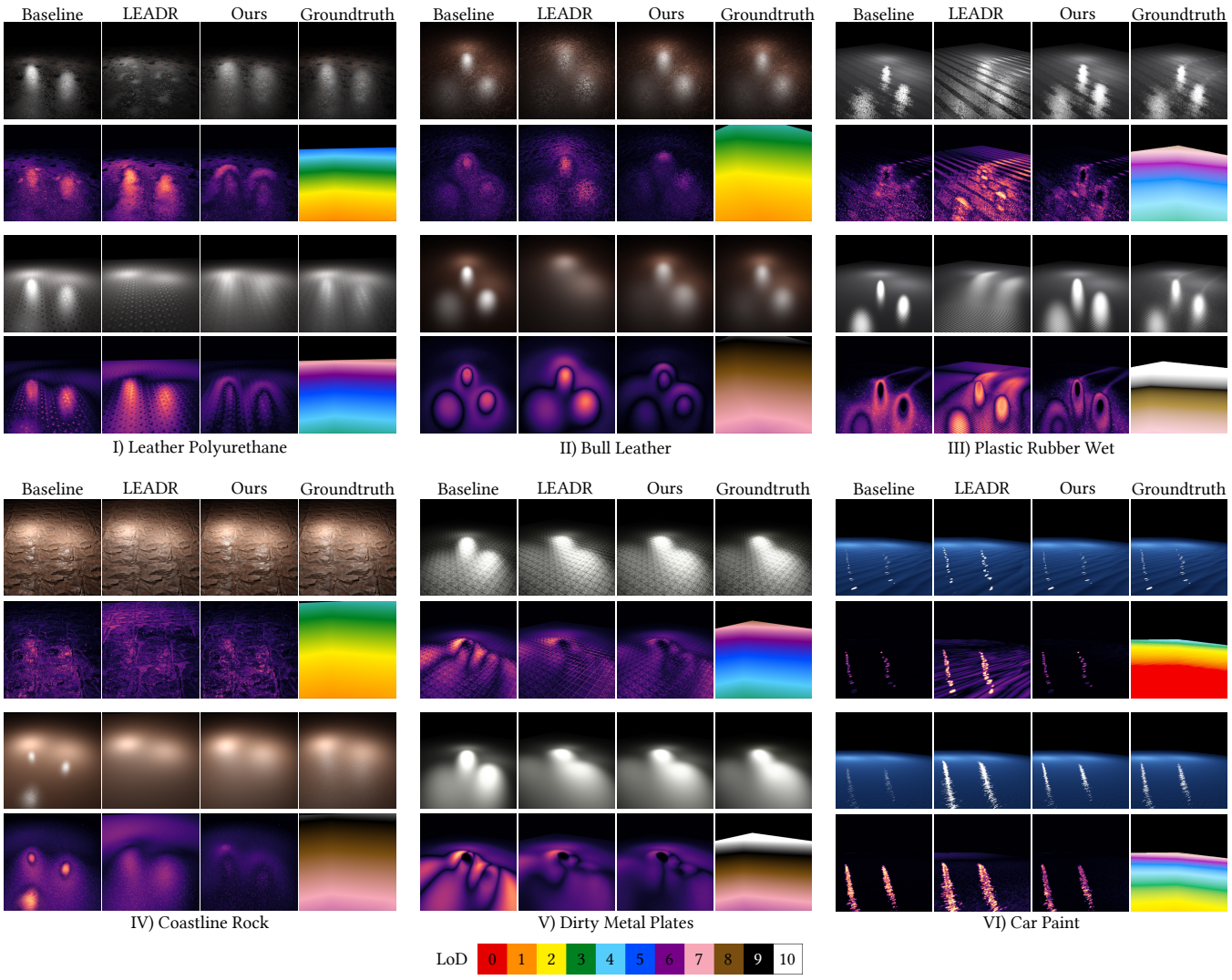


Fig. 6. Renderings of materials using the Beckmann model under varying view conditions. We compare our method to the standard linear mipmapping (baseline) and main competitor specialized in Beckmann distributions mipmapping (LEADR) on six challenging materials *(more in additional material)*. Every second row shows the pixel-wise ꟻLIP deviation to the groundtruth, as well as a false color image depicting the mipmap LoD used for rendering (with trilinear filtering).

Beckmann: comparison with LEADR. We present our results on the Beckmann distribution in Fig. 6, in which we compare our results to the baseline and to the LEADR technique [Dupuy et al. 2013], which is the current state-of-the-art in Beckmann distributions mipmapping. This method uses two maps encoding the statistics of the normal (or displacement) map. Note that this model is not well-suited for the use of both normal and displacement maps (which often encode complementary signals): the filtering is performed on either one of them, but not both.

As can be seen, LEADR exhibits biased anisotropic effects on some examples (most noticeable on materials I, II, III, VI). As expected, the examples on which LEADR fails to capture the anisotropy (and performs worse than the baseline) are the ones where the normal map features geometric statistics that strongly deviate from Gaussian distributions. This is indeed the main hypothesis made by LEADR. *More results can be found in the supplementary material.*

4.2 Comparison to per-material optimization methods

We also compare our method with techniques that compute materials level of details using per-material optimizations. We report in Figure 7 qualitative comparisons and timings for processing a single material. With respect to these two methods, we achieve reasonable material appearance preservation while being 3 to 4 orders of magnitude faster for the mipmap generation. More visual comparisons are available in the supplemental material.

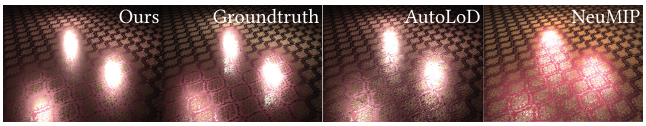


Fig. 7. Qualitative comparison, using the GGX model, with offline material preprocessing methods NeuMIP [Kuznetsov et al. 2021], and AutoLoD [Hasselgren et al. 2021]. For this example, preprocess takes less than a second for our method while requiring 90 minutes for NeuMIP and 25 minutes for AutoLoD (10k iterations).

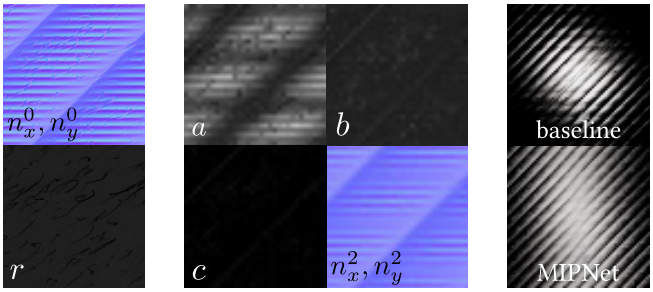


Fig. 8. Transfer between maps: From isotropic SVBRDF parameters (left, normal map and roughness from base LoD), our method is able to generate mip levels which reveal information transfer from normal to roughness and anisotropy creation (middle: generated maps from LoD_2), responsible for the elongated highlight in the rendering (right, top linear mipmapping, bottom ours) in accordance with the groundtruth (not shown).

NeuMIP. We compare renderings of Kuznetsov et al. [2021] using the trained weights and rendering code kindly provided by the authors, with the same materials they trained their model on. Note that a quantitative comparison is impractical since their MBTF has been learned on path-traced rendering and using an unspecified shading model. NeuMIP must train a different model for each material, taking approximately 45 minutes per material with neural textures of resolution 512×512. NeuMIP also requires a neural module at material evaluation time, making its integration in a path-tracing engine not straightforward.

AutoLoD. Hasselgren et al. [2021] propose a differentiable framework to modify the geometry and shading models by optimizing a rendering loss for a given collection of view and light conditions. We modify their publicly available implementation to better match our context, by adding an anisotropic GGX model and disabling the optimization of the geometry, the albedo map, and the metallic map. AutoLoD uses an expensive per-object optimization, requiring around 1h and 11GB of VRAM on a Quadro RTX 6000, for a material with resolution of 1024×1024 (20k iterations with a learning rate of 0.003). The main drawback of AutoLoD resides in its heavy optimization, which is strongly tied to a single material and does not allow generalization.

5 DISCUSSION & FUTURE WORK

Normal-to-roughness transfer. Fig. 8 showcases an example of normal-to-roughness transfer between level-of-details. The base level maps (LoD_0) are shown on the left. In the middle, the tensor coefficients show anisotropy – a is stronger than b which denotes stronger axis-aligned roughness. The resulting renderings (right) produce a natural vertical highlight in our result, compared to the artificially round highlight in the baseline (which originates from independently averaging normal and roughness).

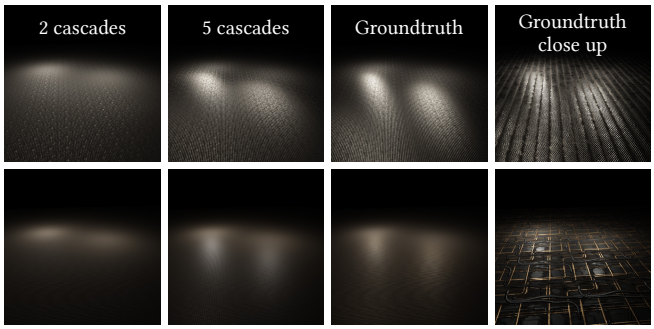


Fig. 9. Qualitative comparison of the number of occurrences of $\mathcal{H}_\mathcal{A}$ and $\mathcal{H}_\mathcal{B}$ blocks used during training using the same batch size for the two experiments. With only a single occurrence (i.e., no repetition of the orange block in Fig. 2), the network fails to learn anisotropy which appears at various level-of-details (see groundtruth), often after the first level. With four occurrences (four repetitions of the orange block in Fig. 2), anisotropy is better learned, by providing gradients of losses computed with further level-of-details.

Training strategies. Multiple strategies for material mipmapping are made possible with our method. First, we consider overfitting the training on a single material SVBRDF. This results in better appearance than standard mipmapping in general but sometimes fails to preserve the appearance for the unseen LoDs of the pyramid. Another strategy consists in pretraining the architecture on multiple materials and fine-tuning the weights of the network on each material separately. We observe a consistent improvement over the material overfitting which supposes a regularization induced by optimizing over diversified examples. The choice of the dataset is critical for a good generalization, and an alternative to a broad, all material generalization consists in specializing networks by materials categories.

Choice of Loss. Our experiments demonstrate a better convergence using a per-pixel L1 loss as well as better appearance preservation across LoDs according to the ꟻLIP metric. We tried an optimization based directly on ꟻLIP, but this led to poorer convergence and quality in the final output.

Ablation. While designing the cascaded architecture, we focused on the quality of the visual appearance as well as the loss value during training. Fig. 9 showcases the quality of mipmapped materials for a single occurrence and four occurrences of $\mathcal{H}_\mathcal{A}$ and $\mathcal{H}_\mathcal{B}$ blocks in the cascaded architecture, which provides better generalization and visual quality. We also provide the loss graph of multiple trainings with varying network parameters (width and depth) in the supplementary material.

Failure cases. Fig. 10 (right column) showcases two examples that stand out as failure cases of our approach on the GGX model, where

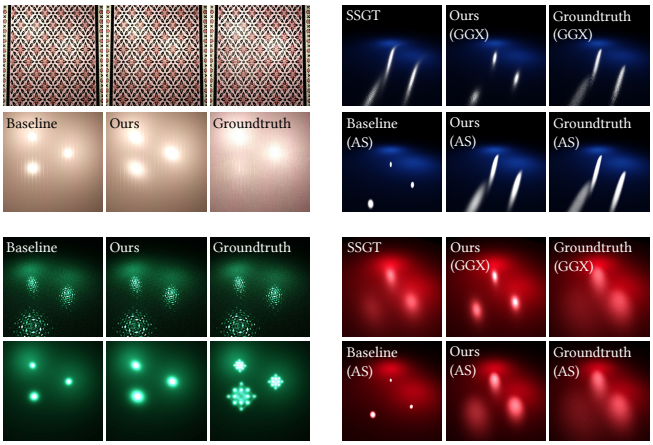


Fig. 10. Limitations of our approach. *Top left:* While we couple the mipmapping of roughness and normals in our method, mipmapping the albedo channel separately does not allow reproducing the base color shift observed in the groundtruth. *Bottom left:* Some very challenging materials might require several lobes to accurately depict the behavior seen from afar. *Right:* We observe some failure cases for GGX.

we clearly see that the SSGT approach produces better results than ours. These results are probably due to the low number of samples in the learned groundtruth for such highly specular materials (roughness smaller than 1e-2 in the original maps), which creates fireflies and prevents smooth gradients to be computed.

Dedicated architectures for material models. Our cascaded neural model is designed to efficiently mipmap materials based on an anisotropic GGX/Beckmann NDF or the Ashikhmin-Shirley model. The case of sheen [Burley 2012], layered [Weier and Belcour 2020] and iridescent [Belcour and Barla 2017] materials would be interesting to study.

Mipmapping the albedo and metallic channels. We focused on the mipmapping of the normal and roughness values for several SVBRDF models. While it is commonly accepted in the literature to mipmap the albedo channel separately – there is no real consensus on the case of the metallic channel – we believe that coupling the mipmapping of all channels might help mimicking more complex behaviors appearing at different scales (see Fig. 10, top left).

Single lobe vs many lobes. As showcased in Fig. 10 (bottom left), fitting a single lobe can prove insufficient to adequately represent the complex radiance distributions emerging at coarse scale (see also Fig. 5, examples III and IV). The use of a single anisotropic lobe is motivated by our choice of remaining compatible with most rendering systems, and by the low storage requirements resulting in high performance renderings. Still, our method is stable under the expressivity power of a single anisotropic lobe on complex setups such as cross-shaped micro-structures (see specific examples in the supplementary materials). We plan to investigate the fitting of several distributions (akin to layered materials), similar to Tan et al. [2022] who proposed to tackle the MIPmapping of normal map-based microstructures in the context of real-time rendering. However, we anticipate that optimizing for the precise number of lobes, for a given material, in a gradient-based optimization framework as well as interpolating between the levels of such representations at runtime will prove particularly challenging.

Multiresolution expression power. We noticed during our experiments that restricting the computation footprint to a standard 2×2 footprint (i.e., using combination of 4 texels to compute one texel in the next mipmap level) was insufficient to efficiently mipmap complex materials for their use with rich lighting. In our work, we have considered a 2-resolution computation architecture using a 4×4 texel footprint to compute the mipmaps after the second LoD. This choice led to efficient mipmap synthesis for large 4K SVBRDFs. It would be interesting to study whether increasing the texel computation footprint and the number of resolutions used helps mipmapping SVBRDFs even more precisely (using for example a 8×8 texel footprint, corresponding to a 3-resolution computation architecture, or a 16×16 texel footprint corresponding to 4-level computation architecture). We envision that training a heavy architecture followed by simplification of the optimized network could allow for efficient synthesis of the mipmap levels while making use of geometric descriptors optimized at many resolutions.

Conclusion. We presented a cascaded neural model to learn downsampling kernels for computing a SVBRDF mipmap pyramid. Our method generalizes over unseen materials and better preserves the appearance of the materials at multiple distances of observation, with the particular behavior of giving rise to anisotropic roughness in the mip levels, which captures salient mesostructures. Overall, our approach is a drop-in replacement for standard MIP mapping, requiring no modification to the host rendering engine.

ACKNOWLEDGMENTS

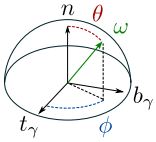
The authors would like to thank Chloé Paliard for proofreading and helping with the experiments, and Élie Michel for discussing about the mathematical formulation of the problem.

REFERENCES

Brent Burley. Physically-based shading at disney. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, New York, NY, USA, 2012. Association for Computing Machinery.
Brian Karis. Real shading in unreal engine 4. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, New York, NY, USA, 2013. Association for Computing Machinery.
Lance Williams. Pyramidal Parametrics. *SIGGRAPH Comput. Graph.*, 1983.
Eric Bruneton and Fabrice Neyret. A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics*, 18(2):242–260, 2012.
Marc Olano and Dan Baker. Lean mapping. I3D '10, pages 181–188, New York, NY, USA, 2010. Association for Computing Machinery.
Christophe Hery, Michael Kass, Junyi Ling, and Pixar Animation Studios. Geometry into shading. In *Pixar Technical Memo 14-04*. 2014.
Jonathan Dupuy, Eric Heitz, Jean Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. Linear efficient antialiased displacement and reflectance mapping. *ACM Transactions on Graphics*, 32(6), 2013.
Petr Beckmann and Andre Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Pergamon Press, 1963.
TS Trowbridge and Karl P Reitz. Average irregularity representation of a rough surface for ray reflection. *JOSA*, 65(5):531–536, 1975.
Bruce Walter, Sr Marschner, Hongsong Li, and Ke Torrance. Microfacet models for refraction through rough surfaces. *Eurographics*, pages 195–206, 2007.

Alejandro Conty Estevez, Pascal Lecocq, and Clifford Stein. *A Microfacet-Based Shadowing Function to Solve the Bump Terminator Problem*, pages 149–158. Apress, Berkeley, CA, 2019.
Jasmin Patry. Samurai shading in ghost of tsushima. In *ACM SIGGRAPH 2020 Courses*, New York, NY, USA, 2020. Association for Computing Machinery.
Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. The sggx microflake distribution. *ACM Trans. Graph.*, 34(4), 2015.
R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, jan 1982. ISSN 0730-0301.
Michael Ashikhmin and Peter Shirley. An anisotropic phong brdf model. *Journal of graphics tools*, 5(2):25–32, 2000.
Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michał Iwanicki, and Sébastien Hillaire. *Real-Time Rendering 4th Edition*. A K Peters/CRC Press, Boca Raton, FL, USA, 2018.
Josiah Manson and Scott Schaefer. Parameterization-Aware MIP-Mapping. *Eurographics Symposium on Rendering 2012*, 31(4), 2012.
Johannes Kopf, Ariel Shamir, and Pieter Peers. Content-adaptive image downscaling. *ACM Transactions on Graphics*, 32(6), 2013.
A. Cengiz Öztireli and Markus Gross. Perceptually based downscaling of images. *ACM Transactions on Graphics*, 34(4), 2015.
Andreas Schilling. Towards real-time photorealistic rendering: challenges and solutions. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 7–15, 1997.
Marc Olano and M North. Normal distribution mapping. *Univ. of North Carolina Computer Science Technical Report*, pages 1–7, 1997.
Michael Toksvig. Mipmapping Normal Maps. *Journal of Graphics Tools*, 10(3):65–71, 2005.
Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. Frequency domain normal map filtering. *ACM Transactions on Graphics*, 26(3), 2007.
Alain Fournier. Filtering normal maps and creating multiple surfaces. Technical Report TR-92-41, Department of Computer Science, University of British Columbia, Vancouver, BC, Canada, 1992.
Ping Tan, Stephen Lin, L Quan, B Guo, and HY Shum. Multiresolution Reflectance Filtering. *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, pages 111–116, 2005.
Tobias Zirr and Anton S. Kaplanyany. Real-time rendering of procedural multiscale materials. *Proceedings - 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D 2016*, pages 139–148, 2016.
X. Chermain, B. Sauvage, J. M. Dischler, and C. Dachsbacher. Procedural Physically based BRDF for Real-Time Rendering of Glints. *Computer Graphics Forum*, 39(7): 243–253, 2020.
Haowen Tan, Junqiu Zhu, Yanning Xu, Xiangxu Meng, Lu Wang, and Ling-Qi Yan. Real-time microstructure rendering with mip-mapped normal map samples. *Computer Graphics Forum*, 41(1):495–506, 2022.
Junqiu Zhu, Sizhe Zhao, Yanning Xu, Xiangxu Meng, Lu Wang, and Ling-Qi Yan. Recent Advances in Glinty Appearance Rendering. *Computational Visual Media*, 2022.
Barry G. Becker and Nelson L. Max. Smooth transitions between bump rendering algorithms. *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1993*, pages 183–189, 1993.
Luc Claustres, Loïc Barthe, and Mathias Paulin. Wavelet encoding of BRDFs for real-time rendering. *Proceedings - Graphics Interface*, pages 169–176, 2007.
Ping Tan, Stephen Lin, Long Quan, Baining Guo, and Heung Yeung Shum. Filtering and rendering of resolution-dependent reflectance models. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):412–425, 2008.
Chao Xu, Rui Wang, Shuang Zhao, and Hujun Bao. Real-Time Linear BRDF MIP-Mapping. *Computer Graphics Forum*, 36(4):27–34, 2017.
Eric Heitz, Derek Nowrouzezahrai, Pierre Poulin, and Fabrice Neyret. Filtering color mapped textures and surfaces. *Proceedings of the Symposium on Interactive 3D Graphics*, pages 129–136, 2013.
Lifan Wu, Shuang Zhao, Ling Qi Yan, and Ravi Ramamoorthi. Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces. *ACM Transactions on Graphics*, 38(4), 2019.
Guillaume Loubet and Fabrice Neyret. Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets. *Computer Graphics Forum*, 36(2):431–442, 2017.
Shuang Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi. Downsampling scattering parameters for rendering anisotropic media. *ACM Transactions on Graphics*, 35(6):1–11, 2016.
Adrian Jarabo, Hongzhi Wu, Julie Dorsey, Holly Rushmeier, and Diego Gutierrez. Effects of approximate filtering on the appearance of bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics*, 20(6):880–892, 2014.
Yusuke Tokuyoshi and Anton S Kaplanyan. Stable Geometric Specular Antialiasing with Projected-Space NDF Filtering. *Journal of Computer Graphics Techniques*, 10(2): 31–58, 2021.
Xavier Chermain, Simon Lucas, Basile Sauvage, Jean-Michel Dischler, and Carsten Dachsbacher. Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(1):1–16, 2021.
Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image SVBRDF capture with a rendering-aware deep network. *ACM Transactions on Graphics*, 37(4), 2018.
Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. MaterialGAN: Reflectance Capture using a Generative SVBRDF Model. *ACM Transactions on Graphics*, 39(6), 2020.
Xilong Zhou and Nima Khademi Kalantari. Adversarial single-image svbrdf estimation with hybrid training. *Computer Graphics Forum*, 2021.
Alexandr Kuznetsov, Krishna Mullia, Zexiang Xu, Miloš Hašan, and Ravi Ramamoorthi. Neumip: Multi-resolution neural materials. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 40(4), July 2021.
Gilles Rainer, Wenzel Jakob, Abhijeet Ghosh, and Tim Weyrich. Neural BTF Compression and Interpolation. *Computer Graphics Forum*, 38(2):235–244, 2019.
Gilles Rainer, Abhijeet Ghosh, Wenzel Jakob, and Tim Weyrich. Unified Neural Encoding of BTFs. *Computer Graphics Forum*, 39(2):167–178, 2020.
Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.
Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.
Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. Appearance-driven automatic 3d model simplification. In *Eurographics Symposium on Rendering*, 2021.
Christophe Schlick. An inexpensive brdf model for physically-based rendering. In *Computer graphics forum*, volume 13, pages 233–246. Wiley Online Library, 1994.
Eric Heitz. Understanding the masking-shadowing function in microfacet-based brdfs. *Journal of Computer Graphics Techniques (JCGT)*, 3(2):48–107, June 2014.
Sébastien Lagarde and Charles de Rousiers. Moving frostbite to pbr. In *ACM SIGGRAPH 2014 Courses*, SIGGRAPH '14, New York, NY, USA, 2014. Association for Computing Machinery.
Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Conference on Computer Graphics and Interactive Techniques*, pages 209–216, New York, NY, USA, 1997. ACM.
P Trettner and L Kobbelt. Fast and robust qef minimization using probabilistic quadrics. *Computer Graphics Forum*, 39:325–334, 2020.
Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. Graph.*, 21(3):267–276, jul 2002.
Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.
Khronos Group. Opengl 4.6 core profile. https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf, 2022. [Online; accessed 13-May-2022].
Philippe Weier and Laurent Belcour. Rendering layered materials with anisotropic interfaces. *Journal of Computer Graphics Techniques (JCGT)*, 9(2):37–57, June 2020.
Laurent Belcour and Pascal Barla. A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence. *ACM Transactions on Graphics*, 36(4):65, July 2017.

A TENSOR-BASED ANISOTROPIC MODELS



In this section, we note (t, b) the axis-aligned tangent and bitangent vectors (whereas t_γ, b_γ denote the rotated tangent and bitangent vectors), n the normal, ω_i the unit vector pointing to the light, ω_o the unit vector pointing to the camera, ω_h the half vector ($\omega_h = (\omega_i + \omega_o)/\|\omega_i + \omega_o\|$), and (θ, ϕ) the spherical coordinates of unit vectors ω in the frame (n, t_γ, b_γ) (see inset), as well as $\chi^+(x) = 1$ for $x \geq 0$ and 0 otherwise the Heavyside function.

The height-correlated *masking and shadowing* function G for the GGX and Beckmann distributions is given by

$$G := \frac{\chi^+(\omega_i \cdot \omega_h)\chi^+(\omega_o \cdot \omega_h)}{1 + \Lambda_{\omega_i} + \Lambda_{\omega_o}} \tag{11}$$

A.1 Anisotropic GGX distributions

The normal distribution D^{GGX} is given by:

$$\begin{aligned} D^{GGX} :=& \frac{\chi^+(\omega_h \cdot n)}{\pi\alpha_t\alpha_b \left(\frac{(t_\gamma \cdot \omega_h)^2}{\alpha_t^2} + \frac{(b_\gamma \cdot \omega_h)^2}{\alpha_b^2} + (n \cdot \omega_h)^2 \right)^2} \\ =& \frac{\chi^+(\omega_h \cdot n)}{\pi \det(A) \left(\frac{\omega_h^T \cdot B^T \cdot A^2 \cdot B \cdot \omega_h}{\det(A)^2} + (n \cdot \omega_h)^2 \right)^2} \\ =& \frac{\chi^+(\omega_h \cdot n)}{\pi \det(A) \left(\left\| \frac{A \cdot B \cdot \omega_h}{\det(A)} \right\|_2^2 + (n \cdot \omega_h)^2 \right)^2}, \end{aligned} \tag{12}$$

where we use $\det(A) = \alpha_b\alpha_t$, and $B^T = (t, b) \in \mathcal{R}^{3\times 2}$.

We rewrite the masking-shadowing for the GGX distribution:

$$G^{GGX} = \frac{2(\omega_i \cdot \omega_h)(\omega_o \cdot \omega_h)}{(\omega_i \cdot \omega_h)\hat{\Lambda}^{GGX}_{\omega_o} + (\omega_o \cdot \omega_h)\hat{\Lambda}^{GGX}_{\omega_i}} \tag{13}$$

$$\begin{aligned} \hat{\Lambda}^{GGX}_{\omega_x} =& \sqrt{\omega_x{}^T \cdot B^T \cdot J^T \cdot A^2 \cdot J \cdot B \cdot \omega_x + (n \cdot \omega_x)^2} \\ =& \sqrt{\|A \cdot J \cdot B \cdot \omega_x\|_2^2 + (n \cdot \omega_x)^2} \end{aligned} \tag{14}$$

where $x = i$ or o in the previous equation, and $J := (0, 1; 1, 0) \in \mathcal{R}^{2\times 2}$ swaps both lines of the matrix at its right side.

A.2 Anisotropic Beckmann distributions

We use the following common approximation [Walter et al. 2007; Heitz 2014] (where $x = i$ or o in the following):

$$\Lambda^B_{\omega_x} \simeq \frac{1 - G_1(\omega_x)}{G_1(\omega_x)} \simeq \begin{cases} \frac{1-1.259\xi_x+0.396{\xi_x}^2}{3.535\xi_x+2.181{\xi_x}^2} & \text{if } \xi_x < 1.6 \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \alpha_x :=& \sqrt{\cos(\phi_x)^2\alpha_t^2 + \sin(\phi_x)^2\alpha_b^2} \\ =& \frac{1}{\sin(\theta_x)} \|A \cdot J \cdot B \cdot \omega_x\|_2 \\ \xi_x :=& \frac{1}{\alpha_x \tan(\theta_x)} = \frac{\cos(\theta_x)}{\|A \cdot J \cdot B \cdot \omega_x\|_2} \\ D^B :=& \frac{\chi^+(\omega_h \cdot n)}{\pi\alpha_t\alpha_b \cos^4(\theta_h)} \exp\left(-\tan^2(\theta_h) \frac{\cos^2(\phi_h)\alpha_b^2 + \sin^2(\phi_h)\alpha_t^2}{\alpha_b^2\alpha_t^2} \right) \\ =& \frac{\chi^+(\omega_h \cdot n)}{\pi \det(A) \cos^4(\theta_h)} \exp\left(-\left\| \frac{A \cdot B \cdot \omega_h}{\cos(\theta_h) \det(A)} \right\|_2^2 \right) \end{aligned}$$

A.3 Ashikhmin-Shirley distributions

We use $s_t = \frac{1}{\alpha_t^2}$ and $s_b = \frac{1}{\alpha_b^2}$. This equivalence is motivated in [Olano and Baker 2010] to best match the anisotropy profiles of Phong and Beckmann distributions. The specular term of Ashikhmin and Shirley [2000] is then given by:

$$\begin{aligned} \rho_s(\omega_i, \omega_o) :=& \frac{\sqrt{(s_t+1)(s_b+1)}}{8\pi} \frac{(n \cdot \omega_h)^{\frac{s_t(t_\gamma \cdot \omega_h)^2 + s_b(b_\gamma \cdot \omega_h)^2}{1-(n\cdot\omega_h)^2}}}{(\omega_h \cdot \omega_i) \max((n \cdot \omega_i), (n \cdot \omega_o))} \\ =& \frac{\sqrt{\det(A^2) + \mathrm{tr}(A^2) + 1}}{8\pi \det(A)} \frac{(n \cdot \omega_h)^{\frac{\omega_h^T \cdot B^T \cdot A^2 \cdot B \cdot \omega_h}{(1-\cos^2(\theta_h))\det(A^2)}}}{(\omega_h \cdot \omega_i) \max((n \cdot \omega_i), (n \cdot \omega_o))} \\ =& \frac{\sqrt{\det(A^2) + \mathrm{tr}(A^2) + 1}}{8\pi \det(A)} \frac{(n \cdot \omega_h)^{\left\| \frac{A \cdot B \cdot \omega_h}{\sin(\theta_h)\det(A)} \right\|_2^2}}{(\omega_h \cdot \omega_i) \max((n \cdot \omega_i), (n \cdot \omega_o))} \end{aligned}$$