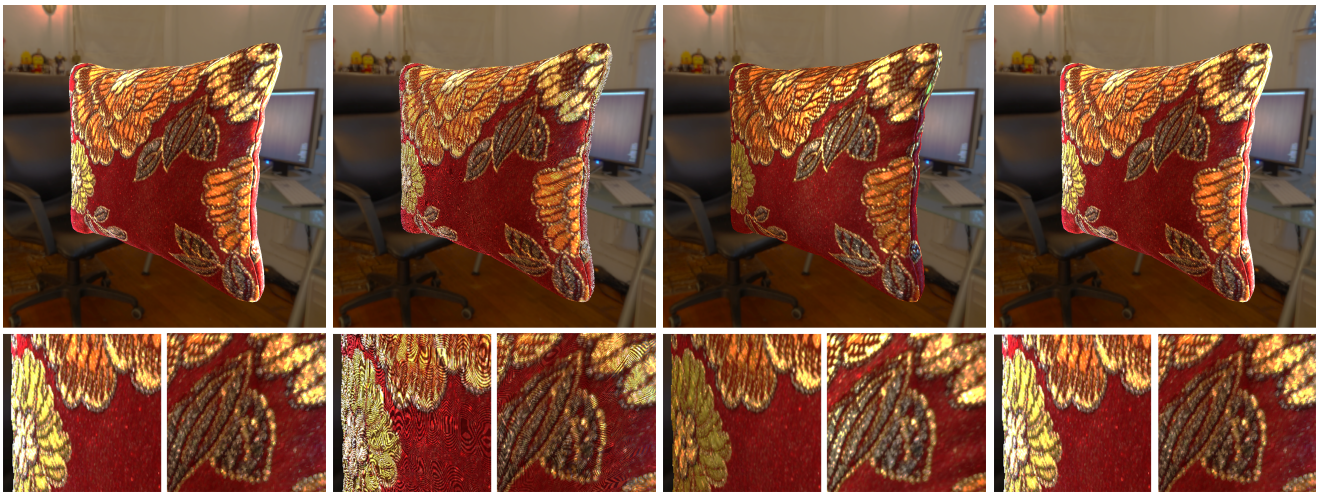
Eurographics Symposium on Rendering 2017
P. Sander and M. Zwicker
(Guest Editors)

Volume 36 (2017), Number 4

Real-Time Linear BRDF MIP-Mapping

Chao Xu[1], Rui Wang[1], Shuang Zhao[2], and Hujun Bao[1]

[1]Zhejiang University
[2]University of California, Irvine



(a) Reference (b) Direct sampling (c) The LEAN mapping (d) Our method

Figure 1: **Images rendered with different BRDF MIP-mapping methods:** (a) the reference generated with 64× supersampling; (b) direct sampling with no MIP-mapping results in severe aliasing; (c) The LEAN mapping [OB10] has difficulty in handling spatially-varying BRDFs; (d) our method well preserves the high-quality reflectance details by MIP-mapping BRDFs and normals jointly.

Abstract

We present a new technique to jointly MIP-map BRDF and normal maps. Starting with generating an instant BRDF map, our technique builds its MIP-mapped versions based on a highly efficient algorithm that interpolates von Mises-Fisher (vMF) distributions. In our BRDF MIP-maps, each pixel stores a vMF mixture approximating the average of all BRDF lobes from the finest level. Our method is capable of jointly MIP-mapping BRDF and normal maps, even with high-frequency variations, at real-time while preserving high-quality reflectance details. Further, it is very fast, easy to implement, and requires no precomputation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Richly detailed surfaces resulting from high-resolution BRDF and normal maps have led to computer generated images with remarkable realism. However, if the resolutions of these maps are significantly higher than those of the final image, the result can be prone to severe aliasing. This is commonly the case when an detailed object is rendered from a distance. The problem can become even worse for shiny BRDFs.

To address this problem, offline rendering techniques usually apply heavy super-sampling which introduces high computational overhead. To reduce aliasing while maintaining real-time performance, texture MIP-mapping [Wil83] has been introduced. This technique builds pre-filtered versions of a texture which then can

be used at render time to reduce aliasing. Unfortunately, unlike color textures which can be easily filtered via linear interpolation, BRDF and normal maps generally cannot be naïvely filtered due to their nonlinear relationship to an object's final appearance.

Due to the significance of this problem, a large body of prior research has been conducted (e.g., [TLQ*05, Tok05, HSRG07, TLQ*08, OB10, DHI*13]). However, some of these methods are limited to normal maps and/or require precomputation, while others offer limited accuracy (see §2).

We introduce a new approach to enable linear MIP-mapping of BRDF and normal maps in a joint manner, offering real-time performance with no precomputation required. In particular, our approach computes at run-time per-pixel effective BRDFs determined by the BRDF and normal maps. Our contributions include:

- A new framework enabling joint MIP-mapping of BRDF and normal maps (§3.1);
- A new scheme to allow multi-lobe BRDF representations (§3.2);
- An efficient GPU implementation of our approach (§4).

2. Related Work

BRDF/normal map filtering. BRDF/normal map filtering has long been an active topic in computer graphics. Please refer to this survey [BN12] for a comprehensive review. From a high level, many previous methods filter BRDF/normal maps based on one [Tok05] or multiple [TLQ*05, HSRG07, TLQ*08] Gaussian-like functions. These methods, however, normally focus on either BRDF or normal map filtering (while assuming the other to be spatially invariant or smooth across the surface). Further, many of them involve precomputations, which can be problematic for highly dynamic scenes.

LEAN mapping [OB10] has previously been introduced to enable real-time linear MIP-mapping of bump-mapped surfaces. The simplicity and effectiveness offered by this method has made it popular in many practical applications. Later, LEADR [DHI*13] mapping generalizes the LEAN framework to support physically-based BRDFs as well as shadowing and masking effects. However, as discussed above, these methods focus on normal maps while leaving high-frequency BRDF variations unhandled. Another major limitation shared by both methods in the context of our work comes from the use of single Beckmann function, which can lead to unsatisfactory results when handling materials with multiple concentrated highlights (e.g., fabrics).

Rendering high-resolution normal maps. Recently, a few techniques have been introduced for physically-based rendering of glossy surface with highly detailed normal distributions [YHJ*14, YHMR16]. Although the resulting renderings have remarkable qualities, these methods are generally too expensive for interactive applications.

Downsampling of volumetric parameters. Pre-filtering has also been studied for downsampling volumetric data in the context of 3D visualization [KB08, SKMH14] and rendering anisotropic participating media [ZWDR16]. These methods, however, are not applicable to our problem of real-time rendering high-resolution BRDF/normal maps.

3. Algorithm

We introduce a new approach to enable linear MIP-mapping of spatially varying BRDFs and normals with no precomputation needed. First, we present a new interpolation method based on the von Mises-Fisher (vMF) distribution to MIP-map BRDF maps (§3.1). Then, we further improve the accuracy of our method by enabling multi-lobe representations via lobe clustering (§3.2).

3.1. BRDF MIP-Mapping

Given a surface point $\boldsymbol{x}$ and direction $\boldsymbol{\omega}_o$, the corresponding exitant radiance L_o is given by the rendering equation [Kaj86] as

$$L_o(\boldsymbol{x},\boldsymbol{\omega}_o) = \int_{\mathbb{S}^2} L_i(\boldsymbol{x},\boldsymbol{\omega}_i)\,\rho(\boldsymbol{x},\boldsymbol{\omega}_i \to \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{\omega}_i, \tag{1}$$

where L_i indicates the incident radiance, and ρ is the BRDF.[†]

Provided a pixel footprint Ω (around some surface point $\boldsymbol{x}$) and direction $\boldsymbol{\omega}_o$, this paper focuses on computing the average exitant radiance L_o given by

$$L_o(\Omega,\boldsymbol{\omega}_o) = \frac{1}{|\Omega|}\int_{\Omega}\int_{\mathbb{S}^2} L_i(\boldsymbol{y},\boldsymbol{\omega}_i)\,\rho(\boldsymbol{y},\boldsymbol{\omega}_i \to \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{\omega}_i\,\mathrm{d}\boldsymbol{y}, \tag{2}$$

where $|\Omega|$ denote the surface area of Ω. When Ω is a small neighborhood around $\boldsymbol{x}$, we have $L_i(\boldsymbol{y},\boldsymbol{\omega}) \approx L_i(\boldsymbol{x},\boldsymbol{\omega})$ for all $\boldsymbol{y} \in \Omega$ and $\boldsymbol{\omega} \in \mathbb{S}^2$. Thus, Eq. (2) becomes

$$L_o(\Omega,\boldsymbol{\omega}_o) = \int_{\mathbb{S}^2} L_i(\boldsymbol{x},\boldsymbol{\omega}_i)\,\rho_{\mathrm{eff}}(\Omega,\boldsymbol{\omega}_i \to \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{\omega}_i, \tag{3}$$

where ρ_{eff} is the *patch-wise effective BRDF* given by

$$\rho_{\mathrm{eff}}(\Omega,\boldsymbol{\omega}_i \to \boldsymbol{\omega}_o) := \frac{1}{|\Omega|}\int_{\Omega} \rho(\boldsymbol{y},\boldsymbol{\omega}_i \to \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{y}. \tag{4}$$

Notice that ρ_{eff} depends on point-wise BRDFs everywhere in Ω. When we assume the pixel footprint Ω to contain N discrete texels with BRDFs $\rho_{\mathrm{tex}}^{(1)}, \rho_{\mathrm{tex}}^{(2)}, \ldots, \rho_{\mathrm{tex}}^{(N)}$, Eq. (4) then becomes

$$\rho_{\mathrm{eff}}(\Omega,\boldsymbol{\omega}_i \to \boldsymbol{\omega}_o) = \frac{1}{N}\sum_{i=1}^{N} \rho_{\mathrm{tex}}^{(i)}(\boldsymbol{\omega}_i \to \boldsymbol{\omega}_o). \tag{5}$$

In this paper, we assume ρ_{tex} to be parameterized with the half-way vector $\boldsymbol{\omega}_h := (\boldsymbol{\omega}_i + \boldsymbol{\omega}_o)/\|\boldsymbol{\omega}_i + \boldsymbol{\omega}_o\|$ and represent ρ_{tex} using the von Mises-Fisher (vMF) distribution:

$$\rho_{\mathrm{tex}}(\boldsymbol{\omega}_i \to \boldsymbol{\omega}_o) = \rho_{\mathrm{tex}}(\boldsymbol{\omega}_h;\,\boldsymbol{\mu},\kappa) = \frac{\kappa}{4\pi\sinh\kappa} e^{\kappa\langle\boldsymbol{\mu},\boldsymbol{\omega}_h\rangle}, \tag{6}$$

with $\boldsymbol{\mu} \in \mathbb{S}^2$ and $\kappa \in \mathbb{R}_+$. When $\kappa \gg 1$, Eq. (6) can be approximated with

$$\rho_{\mathrm{tex}}(\boldsymbol{\omega}_h;\,\boldsymbol{\mu},\kappa) \approx \frac{\kappa}{2\pi} e^{\kappa(\langle\boldsymbol{\mu},\boldsymbol{\omega}_h\rangle - 1)}. \tag{7}$$

Given a pixel footprint Ω containing N texels, obtaining its effective BRDF (5) requires averaging multiple vMF functions (7). Previous work [XWB15] has shown that a vMF distribution with parameters $\boldsymbol{\mu}$ and κ can be characterized using a (unnormalized) vector $\boldsymbol{r} \in \mathbb{R}^3$ satisfying

$$\frac{\boldsymbol{r}}{\|\boldsymbol{r}\|} = \boldsymbol{\mu}, \quad \frac{3\|\boldsymbol{r}\| - \|\boldsymbol{r}\|^3}{1 - \|\boldsymbol{r}\|^2} = \kappa. \tag{8}$$

[†] We assume the cosine of the incident angle to be included in ρ.

To find $\boldsymbol{r}$, one can first compute its magnitude $\|\boldsymbol{r}\|$ by solving the cubic equation

$$\|\boldsymbol{r}\|^3 - \kappa\|\boldsymbol{r}\|^2 - 3\|\boldsymbol{r}\| + \kappa = 0, \tag{9}$$

and then letting $\boldsymbol{r} = \|\boldsymbol{r}\|\boldsymbol{\mu}$. Notice that although Eq. (9) has three real roots, only one of them will be valid in general (see Xu et al.'s work [XWB15] for more details).

When every vMF distribution i characterized by an vector $\boldsymbol{r}^{(i)}$, we can then approximate their average with one vMF distribution characterized by

$$\boldsymbol{r}_M = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{r}^{(i)}. \tag{10}$$

Lastly, the original parameters (i.e., $\boldsymbol{\mu}$ and κ) can be obtained by applying Eq. (8) to $\boldsymbol{r}_M$.

Handling normal maps. Our BRDF MIP-mapping method can also be integrated to render objects with normal maps. At the finest level of a BRDF map, we assume each texel to be associated with a normal distribution function (NDF) γ and a base BRDF ρ parameterized with $\langle\boldsymbol{\omega}_h, \boldsymbol{n}\rangle$. Then, the effective BRDF at each texel is known to be a convolution between γ and ρ [HSRG07]:

$$\rho_{\text{tex}}(\boldsymbol{\omega}_h) = \int_{\mathbb{S}^2}\rho(\langle\boldsymbol{\omega}_h, \boldsymbol{n}\rangle)\,\gamma(\boldsymbol{n})\,\mathrm{d}\boldsymbol{n}. \tag{11}$$

We also represent the NDF γ with the von Mises-Fisher (vMF) distribution with mean $\boldsymbol{\mu}_N$ and concentration factor κ_N as:

$$\gamma(\boldsymbol{n};\,\boldsymbol{\mu}_N, \kappa_N) = \frac{\kappa_N}{2\pi}e^{\kappa_N(\langle\boldsymbol{\mu}_N, \boldsymbol{n}\rangle - 1)}, \tag{12}$$

As stated in prior work [HSRG07], the per-texel effective BRDF (11) simplifies for certain base BRDFs ρ such as Blinn-Phong [Bli77] and Torrance-Sparrow [TS67]. In the former case, for instance, $\rho(\langle\boldsymbol{\omega}_h, \boldsymbol{n}\rangle) = \frac{s+1}{2\pi}\langle\boldsymbol{\omega}_h, \boldsymbol{n}\rangle^s$ for some fixed $s \in \mathbb{R}_+$. Given an NDF determined by $\boldsymbol{\mu}_N$ and κ_N (12), let $s' := \kappa_N \cdot s/(\kappa_N + s)$. Then, it holds that

$$\rho_{\text{tex}}(\boldsymbol{\omega}_h) = \frac{s'+1}{2\pi}\langle\boldsymbol{\omega}_h, \boldsymbol{\mu}_N\rangle^{s'}. \tag{13}$$

This Blinn-Phong BRDF can be further approximated using an vMF function with parameters $\boldsymbol{\mu}_N$ and κ [WRG*09] as:

$$\rho_{\text{tex}}(\boldsymbol{\omega}_h;\,\boldsymbol{\mu}_N, \kappa) = \frac{\kappa}{2\pi}e^{\kappa(\langle\boldsymbol{\mu}_N, \boldsymbol{\omega}_h\rangle - 1)}. \tag{14}$$

where $\kappa = s' + 1$.

3.2. Enabling vMF Mixtures

Eq. (10) approximates the average of multiple vMF distributions using one vMF function. This, unfortunately, can be quite inaccurate for input with complex distributions (Figure 2-ac). To address this problem, previous methods [HSRG07] have proposed to fit vMF mixtures (i.e., linear combinations of vMF lobes). Unfortunately, the previously developed fitting methods are too expensive for real-time applications.

We introduce a new method to fit vMF mixtures (Figure 2-d), offering a good balance between resulting accuracy and performance. We start with partitioning the angular domain $\mathbb{S}^2$ into J components

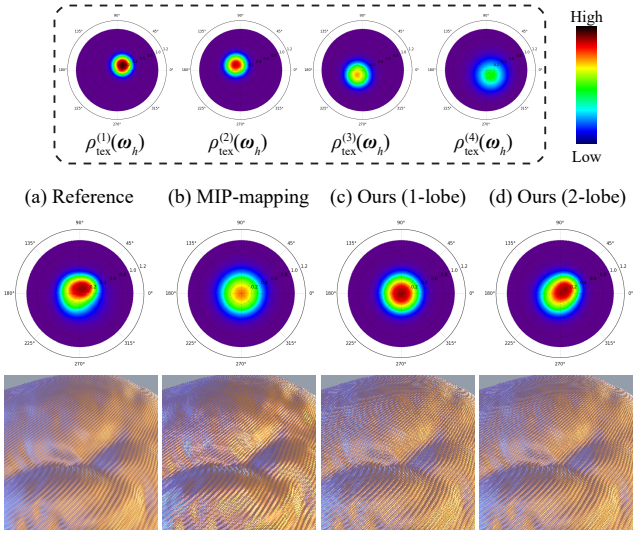


Figure 2: **Necessity of using multiple lobes.** For input with complex BRDF/normal variations (a), linear MIP-mapping of BRDF parameters (b) is largely invalid. Using only one vMF lobe per pixel footprint can lead to unsatisfactory results (c). Our method offers much better accuracy by using vMF mixtures (d). For example, to accurately resemble the average of the four input BRDFs shown on the top, at least two vMF lobes are needed.

$\mathcal{S}^{(1)}, \mathcal{S}^{(2)}, \ldots, \mathcal{S}^{(J)}$ (for some fixed J). Then, we use this partitioning to cluster the input vMF lobes as follows. Given a set of vMF distributions characterized by $\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(2)}, \ldots$, we compute for each $j \in \{1, 2, \ldots, J\}$ a vector $\boldsymbol{r}_M^{(j)}$ by applying Eq. (10) to all $\boldsymbol{r}^{(i)}$ satisfying $\boldsymbol{r}^{(i)}/\|\boldsymbol{r}^{(i)}\| \in \mathcal{S}^{(j)}$. The outcome of this process is then a vMF mixture with J lobes:

$$\rho_{\text{eff}}(\Omega, \boldsymbol{\omega}_i \to \boldsymbol{\omega}_o) = \sum_{j=1}^{J} w^{(j)}\rho_{\text{tex}}\left(\boldsymbol{\omega}_h;\,\boldsymbol{r}_M^{(j)}\right), \tag{15}$$

where $w^{(j)}$ denotes the fraction of characteristic vectors $\boldsymbol{r}^{(i)}$ contained in $\mathcal{S}^{(j)}$.

To implement this idea, we scale-up the BRDF textures and use the added resolution to store multiple vMF lobes. Precisely, we pack multiple textures each of which stores one vMF lobe at every MIP-mapping level. The second column of Figure 3 illustrates an example where the angular domain $\mathbb{S}^2$ is subdivided into 4 components (i.e., $J = 4$), resulting in four vMF lobes per pixel.

When J is large, evaluating Eq. (15) can be time consuming. We tackle this problem by generating multiple versions of MIP-mapped BRDF textures with varying levels of subdivision. At runtime, we then adaptively select the level of subdivision to achieve a good balance between performance and quality. Please see §4 for more details.

4. Implementation

We now describe our implementation of the technique described in §3. With no precomputation, our method renders the input scene from scratch for each frame. Our run-time rendering pipeline consists of three main steps outlined as follows.

1. We start with creating an instant BRDF map for each object based on its geometry as well as, base BRDF and normal maps.
2. Then, we create a hierarchy of MIP-mapped versions of the instant BRDF map using our vMF filtering technique (§3).
3. Lastly, these BRDF MIP-maps are used to reconstruct the effective BRDF at each pixel, which is in turn integrated with the lighting to compute the final shading.

We now provide more details for each of these steps.

Generation of instant BRDF map. Our method first creates an instant BRDF map that encodes local BRDF and normal information. This map changes per frame for dynamic scenes.

We first create a frame buffer with the same resolution as the base BRDF map. Then, triangles of the object are rasterized into this buffer using their texture coordinates. For each triangle, the vertex normals are interpolated to obtain the shading normal at each texel. During this process, all shading normals are transformed into the camera space with the back-facing ones culled. Then, at each texel of this instant BRDF map, we compute an effective BRDF in the form of Eq. (7). Given per-texel effective BRDFs (represented as vMF distributions), we solve the cubic equation (9) to compute the length $\|\boldsymbol{r}\|$ of the corresponding characteristic vector $\boldsymbol{r}$ (which only depends on κ). To accelerate this process, we discretize κ and store the solution $\|\boldsymbol{r}\|$ for each κ in a lookup table (represented as a 1×1024 texture in our implementation) during preprocessing. At render time, we then fetch this texture to quickly determine $\boldsymbol{r}$ for each texel.

BRDF MIP-mapping. Given the instant BRDF map, we perform an extra rendering pass to build its MIP-mapped versions. First, a set of textures are created to store different levels of MIP-maps. As described in §3.2, we partition the angular domain (which is now a hemisphere toward the viewport since all shading normals have been transformed into the camera space) into a few components. Our implementation uses three levels of subdivision: 1×1, 1×4, and 2×4 where $x \times y$ denotes the partitioning of longitude and latitude into x and y components, respectively. For a subdivision level $x \times y$, each texel contains $(x \cdot y)$ vMF lobes each of which represented using an unnormalized vector. To reduce the number of textures, we pack all these vectors for all texels into a single MIP-mapped texture.

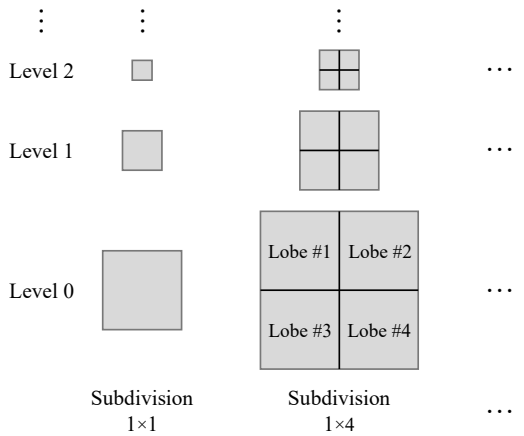


Figure 3: Storing effective BRDFs ρ_{eff} as vMF mixtures (Eq. (15)) at varying subdivision levels.

Algorithm 1 Pseudocode of the pixel shader in final shading

```
1:  procedure PixelShader(uv)
2:      {Setup: calculate half angle ω_h, spatial MIP-map level l}
3:      ρ = 0        ▷ get effective vMF BRDF from the level-0 map
4:      r^(0) = FetchLevel0Texture(l, uv)
5:      if ||r^(0)|| < ε_0 then    ▷ check if lobes are not concentrated
6:          for i = 0 to 3 do
7:              r^(1i) = FetchLevel1Texture(l, i, uv)
8:              if ||r^(1i)|| < ε_1 then ▷ compute using the next level
9:                  for i = 0 to 3 do
10:                     r^(2i) = FetchLevel2Texture(l, i, uv)
11:                     ρ += ComputeShading(r^(2i), ω_h)
12:                 end for
13:             else
14:                 ρ += ComputeShading(r^(1i), ω_h)
15:             end if
16:         end for
17:     else
18:         ρ += ComputeShading(r^(0), ω_h)
19:     end if
20:     return ρ · L                ▷ L denotes the light intensity
21: end procedure
22:
23: procedure ComputeShading(r, ω_h)
24:     μ = r/||r||, κ = (3||r|| − ||r||^3)/(1 − ||r||^2)
25:     ρ = (κ/2π) e^(κ(⟨μ, ω_h⟩ − 1))
26:     return ρ
27: end procedure
```

Figure 3 illustrates BRDF MIP-maps with angular subdivisions of 1×1 (i.e., no subdivision) and 1×4. For the latter, each BRDF map contains four sub-textures each of which stores one of the four vMF lobes for all texels. For each level of subdivision, we build a full spatial MIP-map using Eq. (10) to merge lower-level texels when building higher-level maps. All these computations are performed in the pixel shader with results outputted to two rendering targets (which corresponds to 1×4 and 2×4 subdivisions respectively). After obtaining the vMF lobe information (i.e., characteristic vectors) for all three subdivision levels, we use hardware API to build MIP-maps upon them.

Final shading. The pseudocode of our pixel shader implementing the final shading step is shown in Algorithm 1. To shade one pixel, we first calculate the size of its footprint (on the object surface) to determine the spatial MIP-map level. Then, we fetch the parameters of effective BRDF from the corresponding spatial level with subdivision level 0, i.e., $J = 1$ (line 4). Assuming the fetched BRDF to be characterized by $\boldsymbol{r} \in \mathbb{R}^3$, the length of this vector $\|\boldsymbol{r}\|$ then indicates how concentrated the mean directions of constituent BRDFs are. For example, if all these BRDFs have approximately the same mean direction (i.e., $\boldsymbol{\mu}$), $\|\boldsymbol{r}\|$ will be close to 1. On the other hand,

when the constituent BRDFs have widely varying $\boldsymbol{\mu}$, $\|\boldsymbol{r}\|$ will be close to 0. Based on this observation, we use $\|\boldsymbol{r}\|$ as a metric to determine whether a finer subdivision (i.e., greater J) should be used. In particular, we iteratively increase the level of angular subdivision until $\|\boldsymbol{r}\|$ exceeds some threshold (i.e., ε_0 and ε_1 in lines 5–19).[‡] Once the characteristic vector $\boldsymbol{r}$ is determined for each pixel, we use Eq. (15) to construct the corresponding effective BRDFs (lines 23–27).

We use two types of light sources, point and environmental, to generate all rendered results. For point sources, the effective BRDF is directly used to evaluate the final shading. For environmental lighting, we leverage pre-filtered environment maps [KVHS00] for efficient shading computation.

5. Results

We implement our algorithm in Microsoft DirectX 11 on a PC with Intel Core i7 CPU and NVIDIA Quadro M4000M graphics card. Since our method does not rely on any scene-specific precomputation (and computes everything from scratch per frame), it supports highly dynamic scenes with changing lighting, viewpoint, geometry, and even BRDFs.

5.1. Evaluations

To evaluate our new BRDF MIP-mapping technique, we compare images rendered with our approach with those generated with several other methods including direct sampling, LEAN, and LEADR.

We start with two simple scenes, as shown in Figure 4, each of which contains a planar base geometry with spatially varying BRDFs. On the top of Figure 4, we show a plane with two different base BRDFs in a checkerboard pattern and fixed normals (i.e., no shading normal). Direct sampling, which computes only one shading sample per pixel, suffers from aliasing artifacts. The LEAN mapping provides a non-linear filtering on normals but lacks the ability to handle spatially-varying or multi-lobe BRDFs, leading to over-blurred regions on the top of column (c). Our method manages to closely resemble the reference.

On the bottom, we show another plane with a complex micro-geometry represented with high-resolution normals and micro-facet BRDFs. Direct sampling fails to resolve this micro-geometry and produces serve aliasing. LEADR extends LEAN's normal map filtering framework by handling variations in the Fresnel and shadowing terms. Unfortunately, it has difficulties handling rapidly changing BRDFs, yielding an overly soft result, as seen on the bottom of column (c). Our method handles the Fresnel and shadowing terms similarly as LEADR and better preserves the detailed surface appearance.

5.2. Main Results

We further demonstrate the effectiveness of our approach by comparing rendered images for more complex virtual scenes, as shown in Figures 1, 5, and 6. Details on the scene configurations are as follows.

- The Pillow scene (Figure 1) uses spatially-varying BRDFs obtained from the work by Wang et al. [WRG*09]. These BRDFs are represented with one diffuse color map, one specular color map and one specular coefficient map. Since the two color maps can be linearly MIP-mapped, we only apply our BRDF MIP-mapping to the specular coefficients.
- In the Tablecloth scene (Figure 5-top), the cloth has a two-lobe BRDF with ($\boldsymbol{\mu}_0 = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0)$, $\kappa = 128$) and ($\boldsymbol{\mu}_1 = (-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0)$, $\kappa = 128$) respectively in the local coordinate of $\boldsymbol{n} = (0,0,1)$. The normal map used on the cloth has a resolution of 1024×1024 and is generated from a noise function.
- In the Bunny scene (Figure 5-middle), we use a 1024×1024 normal map with hemispherical bump patterns. The BRDF on bunny is a material map has a resolution of 1024×1024.
- The Dragon scene (Figure 5-bottom) is modeled using a 2048×2048 normal map and a 671×457 BRDF map tiled across the surface.
- The Sponza scene (Figure 6) is created after Crytek's Sponza model with the fabric and floor textures replaced using high-resolution BRDF and normal maps with resolutions up to 2048×2048.

When generating the images, the direct sampling method does not use any MIP-mapping or super-sampling and directly takes the BRDF parameter and normal stored in the base BRDF and normal maps to compute the shading. The naïve MIP-mapping method uses standard MIP-mapping strategy that directly averages $\boldsymbol{\mu}$ and κ stored in the instant BRDF map. The LEAN mapping [OB10] is a widely used linear normal map filtering technique with its limitations inherited by the LEADR mapping [DHI*13]. Therefore, we only compare to the LEAN mapping to show the benefits provided by our technique.

Similar to the examples shown in §5.1, direct sampling suffers from severe aliasing, and LEAN/LEADR has difficulties preserving detailed appearance variations arising from high-frequency shading changes. Our method, in contrast, provides a good balance between performance and accuracy. Please see animated versions of the results in Figures 4 and 5 in the accompanying video.

Performance. Table 1 presents the performance numbers (in computation time and FPS) for all results in Figures 1, 5, and 6. Our method takes about 1 ms to build the BRDF MIP-maps which involves the construction of MIP-mapped vMF characteristic vectors at multiple subdivision levels. As discussed in §4, our implementation uses three levels, which is only slightly slower than building a single level due to the constant CPU and GPU overhead introduced by multiple rendering stages. With our BRDF MIP-maps, the final shading process usually takes 0.6–0.9 ms per frame. This level of efficiency makes our method suitable for many real-time applications. Compared with linear-filtering based techniques, our method better preserves the reflectance details. Compared with heavy super-sampling, on the other hand, our method is significantly faster (by one order of magnitude in our examples) while offering similar result quality. Lastly, our method is more memory efficient than LEAN as the latter stores and filters the first and second mo-

[‡] In practice, we pick $\varepsilon_0 = 0.95$ and $\varepsilon_1 = 0.85$.

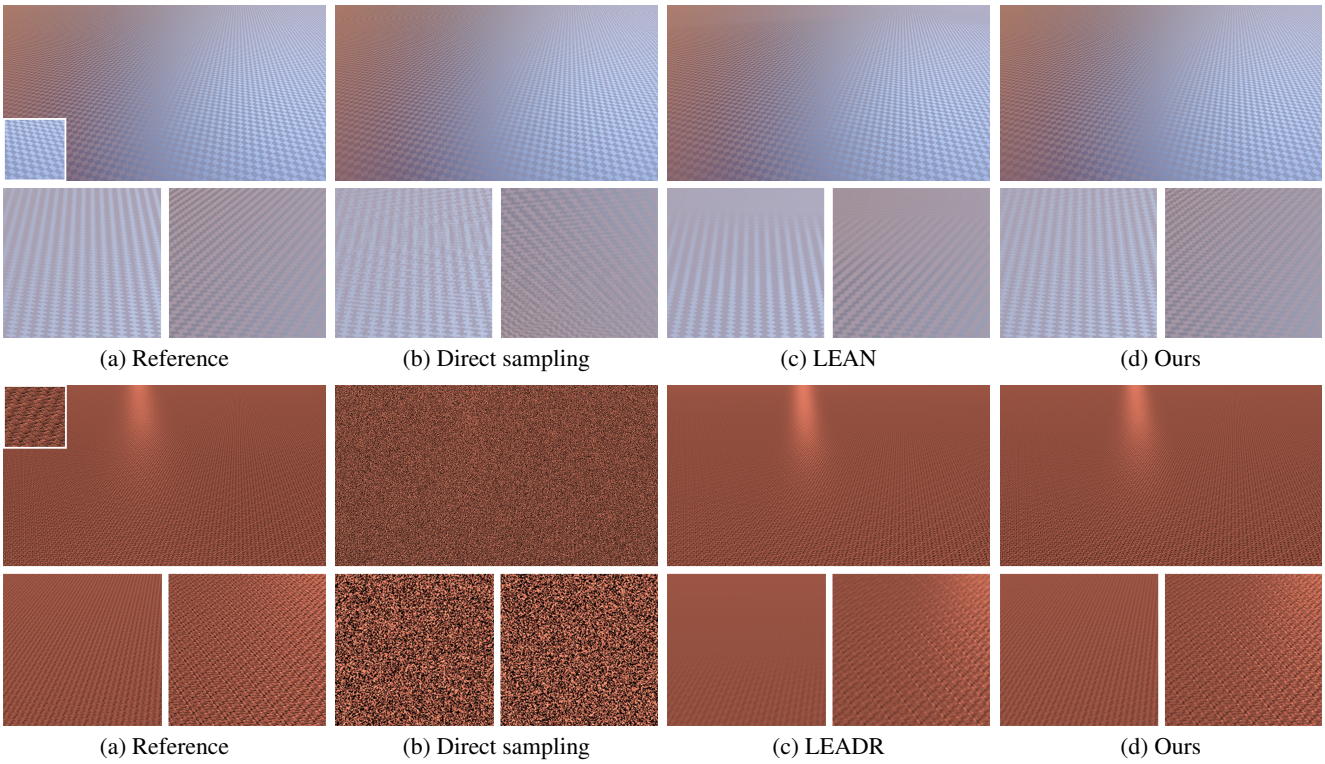


Figure 4: **Comparison of renderings** of two planar surfaces generated with direct sampling (b), LEAN/LEADR (c), and our method (d). The insets in white rectangles show close-up views of the two planes.

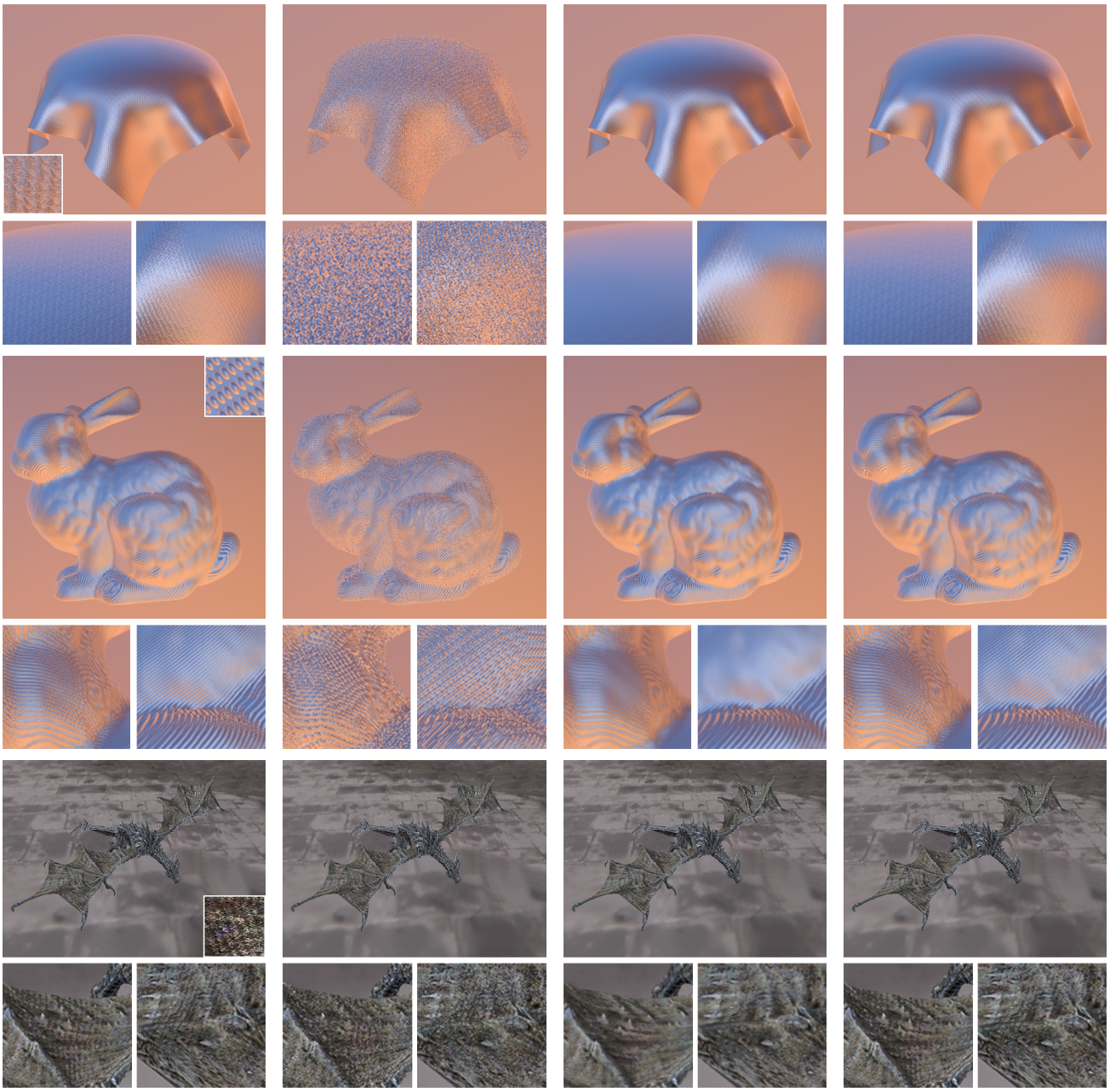
Table 1: **Statistics of example scenes.** From left to right: scene, types of BRDFs, resolution of BRDF map, and the performance costs of Direct Sample, naïve MIP-mapping, reference and our method. The reference is rendered using 64× supersampling. We report the averaged milliseconds and corresponding FPS.

Scene	Map Res.	Performance (in millisecond/FPS)						
		Direct	MIP-map	LEAN	Ref.	Our Method		
						MIP-mapping	Shading	Total
Pillow (Fig 1)	2048 × 1024	0.6/1228	0.6/1231	0.9/826	59/17	0.9	0.9	2.0/489
Tablecloth (Fig 5)	1024 × 1024	0.4/1637	0.4/1621	0.7/1037	50/20	0.8	0.7	1.6/623
Bunny (Fig 5)	1024 × 1024	0.5/1397	0.5/1384	0.8/967	51/19	0.9	0.8	1.8/553
Dragon (Fig 5)	2048 × 2024	0.5/1289	0.6/1296	0.8/898	54/18	1.1	0.6	1.7/643
Sponza (Fig 6)	2048 × 2048	1.6/589	1.6/596	1.9/454	97/10	1.3	1.2	2.5/371

ments of normal distributions while ours only store a unnormalized vector $\boldsymbol{r}$ for each vMF distribution.

5.3. Limitations and Future Work

Our method does have some limitations that can encourage further research in this direction. Firstly, even with the filtering of effective BRDFs achieved using linear summations of their characteristic vectors, the computational cost of our approach is still higher than hardware-accelerated naïve texture MIP-mapping due to the overhead incurred by multiple stages of the rendering process. Thus, faster computation and summation of characteristic vectors for vMFs are worthy exploring in future. Secondly, our method uses the length of characteristic vectors as a metric to determine whether the underlying lobes are concentrated. This metric, however, may lead to unnecessary subdivisions and reduced performance for highly diffuse materials as the base BRDFs already have short characteristic vectors. One way to address this problem is to store additional information, such as the variance, of the vectors. Thirdly, our method does not optimize temporal coherence, which can lead to animated results with small amounts of temporal noise (as seen in the accompanying video). Extending our technique to pre-filter not only spatially but also temporally varying contents is an interesting topic for future research. Lastly, Our method will need to be extended to handle BRDFs that cannot be well approximated with a small number of vMF distributions (e.g., highly anisotropic BRDFs).

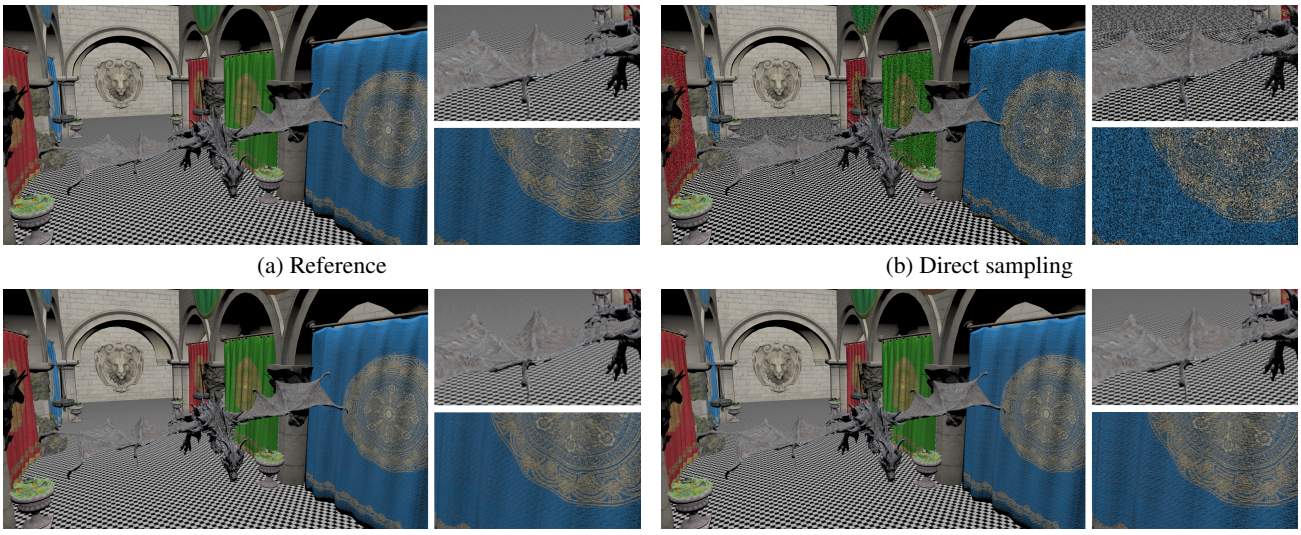


(a) Reference (b) Direct sampling (c) LEAN (d) Ours

Figure 5: **Comparison of renderings** generated with direct sampling (b), LEAN (c), and our method (d). The insets in white rectangles show highly zoomed views of each reference materials. Please see Table 1 for the performance numbers.

6. Conclusion

In this paper, we present a novel technique to linearly MIP-map BRDF and normal maps in a joint manner. The key of our approach is to represent effective BRDFs as von Mises-Fisher (vMF) distributions characterized with unnormalized vectors. Under this representation, averaging vMFs can be performed efficiently and at real-time through a linear summation of their characteristic vectors. To handle complex distributions, we introduce a new method to fit vMF mixtures by subdividing the angular domain, offering a good balance between resulting accuracy and performance. We implement this technique on the GPU in three steps. Given the base BRDF and normal maps, we start with generating an instant BRDF map. Its MIP-mapped versions are then computed based on linearly mixed vMF distributions. Lastly, at each pixel, the effective BRDF is constructed before obtaining the final shading color. Our method is very fast, easy to implement, and requires no precom-



(a) Reference (b) Direct sampling

(c) LEAN (d) Ours

Figure 6: **Comparison of renderings** of a modified Sponza scene with direct sampling (b), LEAN (c), and our method (d). The insets show highly zoomed views of different materials.

putation. Experiments demonstrate that our technique is capable of MIP-mapping BRDF and normal maps, even with high-frequency variations, at real-time while preserving high-quality reflectance details.

Acknowledgement

This work was partially supported by National Key R&D Program of China (No. 2016YFB1001503), NSFC (No. 61472350), the Fundamental Research Funds for the Central Universities (No. 2017FZA5012).

References

[Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph. 11*, 2 (1977), 192–198. 3

[BN12] BRUNETON E., NEYRET F.: A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics 18*, 2 (2012), 242–260. 2

[DHI*13] DUPUY J., HEITZ E., IEHL J.-C., POULIN P., NEYRET F., OSTROMOUKHOV V.: Linear efficient antialiased displacement and reflectance mapping. *ACM Trans. Graph. 32*, 6 (2013), 211:1–211:11. 2, 5

[HSRG07] HAN C., SUN B., RAMAMOORTHI R., GRINSPUN E.: Frequency domain normal map filtering. *ACM Trans. Graph. 26*, 3 (2007), 28:1–28:12. 2, 3

[Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH Comput. Graph. 20*, 4 (1986), 143–150. 2

[KB08] KRAUS M., BÜRGER K.: Interpolating and downsampling RGBA volume data. In *VMV* (2008), pp. 323–332. 2

[KVHS00] KAUTZ J., VÁZQUEZ P.-P., HEIDRICH W., SEIDEL H.-P.: Unified approach to prefiltered environment maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (London, UK, UK, 2000), Springer-Verlag, pp. 185–196. 5

[OB10] OLANO M., BAKER D.: Lean mapping. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2010), pp. 181–188. 1, 2, 5

[SKMH14] SICAT R., KRUGER J., MOLLER T., HADWIGER M.: Sparse PDF volumes for consistent multi-resolution volume rendering. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2417–2426. 2

[TLQ*05] TAN P., LIN S., QUAN L., GUO B., SHUM H.-Y.: Multiresolution reflectance filtering. In *EuroGraphics Symposium on Rendering* (2005), pp. 111–116. 2

[TLQ*08] TAN P., LIN S., QUAN L., GUO B., SHUM H.: Filtering and rendering of resolution-dependent reflectance models. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 412–425. 2

[Tok05] TOKSVIG M.: Mipmapping normal maps. *Journal of graphics, GPU, and game tools 10*, 3 (2005), 65–71. 2

[TS67] TORRANCE K. E., SPARROW E. M.: Theory for off-specular reflection from roughened surfaces. *JOSA 57*, 9 (1967), 1105–1112. 3

[Wil83] WILLIAMS L.: Pyramidal parametrics. *SIGGRAPH Comput. Graph. 17*, 3 (1983), 1–11. 1

[WRG*09] WANG J., REN P., GONG M., SNYDER J., GUO B.: All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 133:1–133:10. 3, 5

[XWB15] XU C., WANG R., BAO H.: Realtime rendering glossy to glossy reflections in screen space. *Computer Graphics Forum 34*, 7 (2015), 57–66. 2, 3

[YHJ*14] YAN L.-Q., HAŠAN M., JAKOB W., LAWRENCE J., MARSCHNER S., RAMAMOORTHI R.: Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Trans. Graph. 33*, 4 (2014), 116:1–116:9. 2

[YHMR16] YAN L.-Q., HAŠAN M., MARSCHNER S., RAMAMOORTHI R.: Position-normal distributions for efficient rendering of specular microstructure. *ACM Trans. Graph. 35*, 4 (2016), 56:1–56:9. 2

[ZWDR16] ZHAO S., WU L., DURAND F., RAMAMOORTHI R.: Downsampling scattering parameters for rendering anisotropic media. *ACM Trans. Graph. 35*, 6 (2016). 2