Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

A Stationary SVBRDF Material Modeling Method Based on Discrete Microsurface

Junqiu Zhu [1], Yanning Xu [†1] and Lu Wang [‡1]

[1] School of Software, Shandong University, China

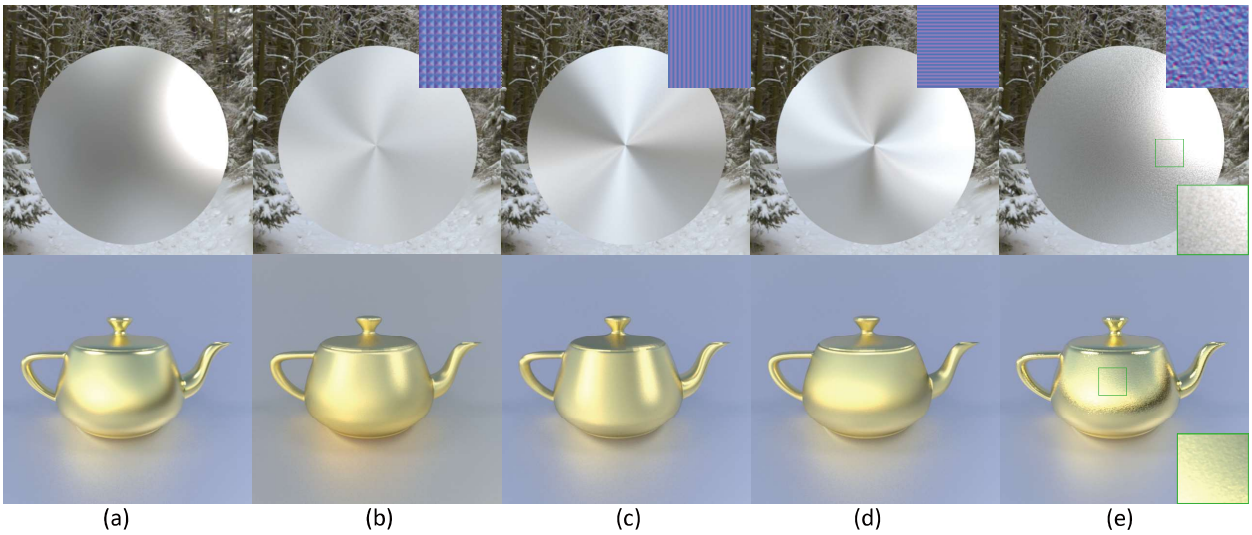

Figure 1: *Our SVBRDF modeling method depicts the discrete microsurface with the sample normal map ((b-e) top right), and achieves precisely specular reflectance in a stationary and efficient way. Rendering with path tracing method, we get continuous effects (a), anisotropic effects with structured normal maps (b-d) and glittery effects with randomized discrete normal map (e).*

Abstract

Microfacet theory is commonly used to build reflectance models for surfaces. While traditional microfacet-based models assume that the distribution of a surface's microstructure is continuous, recent studies indicate that some surfaces with tiny, discrete and stochastic facets exhibit glittering visual effects, while some surfaces with structured features exhibit anisotropic specular reflection. Accordingly, this paper proposes an efficient and stationary method of surface material modeling to process both glittery and non-glittery surfaces in a consistent way. Our method comprises two steps: in the preprocessing step, we take a fixed-size sample normal map as input, then organize 4D microfacet trees in position and normal space for arbitrary-sized surfaces; we also cluster microfacets into 4D K-lobes via the adaptive k-means method. In the rendering step, moreover, surface normals can be efficiently evaluated using pre-clustered microfacets. Our method is able to efficiently render any structured, discrete and continuous micro-surfaces using a precisely reconstructed surface NDF. Our method is both faster and uses less memory compared to the state-of-the-art glittery surface modeling works.

CCS Concepts

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

† Corresponding author: xyn@sdu.edu.cn

‡ Joint corresponding author: luwang_hcivr@sdu.edu.cn

1. Introduction

The simulation of the appearance of specular highlights is a core problem of computer graphics. The microfacet-based BRDF model assumes that a surface is composed of a collection of randomly oriented microfacets, and the aggregate behavior of which determines the surface reflection.

The specular reflection of the surface will appear smooth if its microfacet distribution is continuous. Some BRDF models, such as Torrance-Sparrow [TS67] and Cook-Torrance [CT82], statistically derive closed-form expressions with some user parameters. However, some glint materials (e.g. scratched or brushed metal, metallic paint, snow, etc.) are difficult to be described with expressions, as their microfacet-distributions are discrete and random.

In recent years, a number of discrete micro-surface material modeling methods have appeared. Jakob et al. [JHY*14] propose a stochastic hierarchy method to render glittery surfaces without any textures; although this method does not require additional storage, it is limited to handling point-shaped glints. Yan et al. [YHJ*14, YHMR16] present solutions for rendering arbitrary-shaped surfaces, defined by explicit high-resolution heightfields (or normal maps). While these approaches are successful at simulating both scratches and glints, they also require a large-sized high-resolution normal map to ensure the accuracy of the results on complex surfaces, leading to high additional storage costs. Also, a large-sized high-resolution normal map is difficult to capture from the real world.

Accordingly, we present a SVBRDF modeling method derived from a small-sized sample normal map, which is capable of handling specular reflections (including glint or non-glint phenomena) accurately and efficiently.

An overview of our algorithm is presented in fig.2. In the **preprocessing** step, we take a small-sized sample normal map as input. The sample normal map is supposed to describe the surface normal details in small scales while containing spatial varying patterns. By using normal map synthesis, we can then implicitly generate a large-sized high-resolution normal map and model surfaces with arbitrary size from the small-sized normal map. The normal distribution of the surface is fitted with a small number of discrete lobes and positional correspondence data. Moreover, to accelerate rendering, we build hierarchical searching trees on both position space and normal space. Clustering the neighboring lobes with similar normals is another strategy utilizing for acceleration. During the **rendering** step, we execute importance sampling from the position tree and evaluate $\mathcal{P}$-NDF from the normal tree.

The major contributions of this paper are as follows:

- We construct a stationary microfacet model that is fit not only for the glint surfaces with discrete microfacets, but also for the specular surfaces with continuous and structured microfacets;
- We sample microstructures of varying positions and normals from a small-sized normal map sample, thereby ensuring that the storage space for the normal map and the time required for sampling remains stable even when handling very complicated surfaces;
- An adaptive k-means clustering method is used to cluster lobes in normal space, and to speed up the $\mathcal{P}$-NDF evaluation.

2. Related Work

2.1. Discrete Microfacet Model

There are two kinds of methods used to express discrete microfacet distribution. One involves the random generation of mirror flakes or anisotropic glint features, while the other uses high-resolution normal maps or height maps to represent small-scale details on the surface.

Jakob et al. [JHY*14] use a stochastic approach to simulate glittery effects using a statistical distribution of tiny, mirror-like flakes. This method is memory-efficient and temporally stable in animation. Zirr et al. [ZK16] present a real time method based on the concept of pseudorandom generation, which supports the rendering of brushed surfaces through the generation of anisotropic flakes. Dong et al. [DWMG15] analyze the microgeometry of brushed metal and predict a reflectance model using Kirchhoff scattering theory. Raymond et al. [RGB16] propose an SVBRDF model for scratches, which takes inter-reflections and Fresnel effects inside a scratch into consideration. Velinov et al. [VWH18] render a surface that is modeled by generating randomly oriented scratches and reveals the iridescent colors of the scratches. However, all of these methods are unable to handle arbitrary sub-pixel specular features.

Several high-resolution normal map based reflectance models [YHJ*14, YHMR16, YHW*18] have been proposed to describe glittering effects, such as discrete sparkling, scratched metals and leather. These methods integrate $\mathcal{P}$-NDFs within a single patch rather than evaluating point-based NDFs. Yan et al. [YHJ*14] consider the NDF of a surface patch $\mathcal{P}$ as seen through a single pixel. This method computes BRDF values for a normal map over arbitrary surface regions; this approach is accurate, but slow. Yan et al. [YHMR16] improve this algorithm by approximating the position normal distribution as a mixture of Gaussian elements; however, this method requires a large amount of storage space and a long preprocessing time, and the rendering efficiency would decrease as the number of Gaussian elements increase. Yan et al. [YHW*18] simulate the iridescence glint effects modeled by high-resolution heightfields based on wave optics. They also preprocess the heightfields and fit the NDFs with Gabor kernels, which are more complex but better at matching the complex integrals. By contrast, our method accelerates both rendering and preprocessing by a small-sized normal map, and also reduces storage space with almost no resultant loss of quality.

2.2. Structured Microfacet Model

Structured materials exhibit high-frequency variations in geometry and reflectance. Han et al. [HSRG07] use a small number of lobes to approximate the pixel NDF and simulate appearance at different scales through SVD method. Characteristic point maps [WDR09] are introduced to reduce an initial dense sampling to a minimum number of characteristic sampling points. This method can be effectively applied to structured materials and it considers the shadowing-masking effect. Wu et al. [WDR11] further present an efficient reflectance filtering algorithm to reconstruct micro-scale surfaces by changing micro-scale details. These methods use a preprocessing method to facilitate speedup, but cannot be applied to glittery surface rendering.

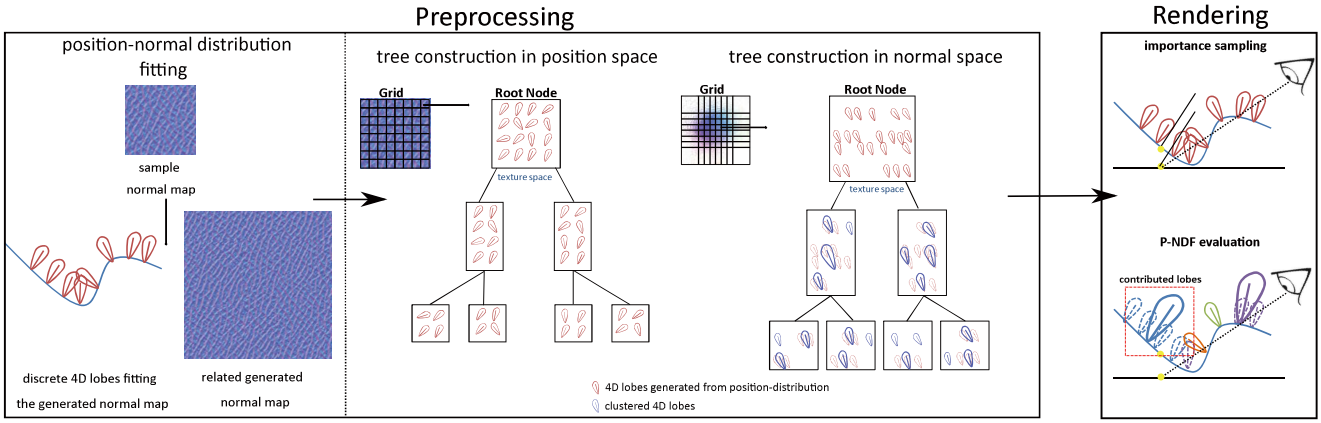

Figure 2: *Our algorithm. In the **preprocessing** step, 4D lobes are constructed from a small-sized sample image (left) to record the normal distribution. We then classify lobes in position and normal space (middle); lobes in normal space are clustered to fit the real normal distribution. During the **rendering** process, moreover, we sample surface normals in position space and evaluate the patch-based BRDF in normal space.*

2.3. Texture Synthesis

It is generally believed that textures have self-similar feature units and are accompanied by a certain random variation. Texture synthesis technology synthesizes large texture images using small sample images. A non-parametric method is proposed by [EL99] to synthesize the normal map point by point. Wei et al. [WL00] accelerate the pixel-based synthesis process by using tree-structured vector quantization. Moreover, classic patch-based methods [EF01] proposes an efficient and effective synthesis method by comparing the similarity among neighboring patches. For a more flexible patch-based synthesis method [KSE*03], the size and shape of a posting patch is not set by default; instead, the optimal size of a patch is calculated according to graph-cut method. Since most normal distributions of glint materials also have self-similar features, it is an effective way to generate high-resolution normal maps with texture synthesis.

3. Preliminaries

A number of important terms and concepts used in this paper are explained in this section. Table 1 above presents an overview of the symbols.

Normal Map Textures: To reduce the space and time costs required, we present a normal map synthesis method. We take the small-sized sample normal map $\mathcal{I}$ as input, then pick a number of blocks $\mathcal{B}_i$ from it to synthesize arbitrary-sized high-resolution normal maps $\mathcal{H}$ implicitly. For each block $\mathcal{B}_i$ in $\mathcal{H}$, we need only to store its corresponding block index in $\mathcal{I}$. We define the unit of a block $\mathcal{B}_i$ to be a square area of user-defined size on the normal map and record its left corner coordinate.

Normal Mapping: In this paper, mapping functions $n(\mathbf{p})$ and $m(\mathbf{u})$ are used to map points in the texture space to normals on a planar domain. Normals are represented as two-dimensional vectors on the projected hemisphere (unit disk). The normal map function can be expressed as $n(\mathbf{p})$: from point $\mathbf{p} = (p,q)$ in texture space (on $\mathcal{I}$) to $\mathbf{s} = (s,t)$ on the extended unit disk. We also set a position map function $m(\mathbf{u})$ in texture space, from point $\mathbf{u} = (u,v)$ on $\mathcal{H}$ to point $\mathbf{p} = (p,q)$ on $\mathcal{I}$.

Table 1: *Summary of notations used in this paper*

Symbol	Meaning
$\mathcal{I}$	input small-sized sample normal map
$\mathcal{H}$	synthetic high-resolution normal map
$\mathcal{B}_i$	i-th block on the normal map
$\mathbf{p} = (p,q)$	texture space parameters of $\mathcal{I}$
$\mathbf{u} = (u,v)$	texture space parameters of $\mathcal{H}$
$\mathbf{s} = (s,t)$	normal (on proj. hemisphere)
$m(\mathbf{u})$	position mapping function
$n(\mathbf{p})$	normal mapping function
$\mathcal{P}$	ray footprint
G_i	i-th 4D lobe
K_j	j-th 4D K-lobe clustered by adaptive k-means
δ_k	clustering coefficient
h	step size of lobes
σ_h	std. deviation of lobes
σ_r	intrinsic roughness

Lobes for Normal Distribution: For structured or non-structured surfaces, the integration of SVBRDF on a patch is determined by $\mathcal{P}$-NDF; i.e the contribution of normals over a patch. We thus fit the normal distribution with 4D lobes G (including Gaussian expression, Beckmann and GGX) from texture $\mathcal{I}$. Moreover, those lobes with similar normals can be clustered as a new 4D lobe K_j by means of an adaptive k-means method. We use cluster coefficient δ_k to evaluate the k-means effect of K_j in normal space, then set the coefficient as the mean square error between the center normal of a K-lobe K_j and the center normals of the component lobes.

The density of these lobes is defined in the texture domain on $\mathcal{I}$ by step size h. We stationarily distribute the lobes with step size h, i.e. the distance of a lobe to their neighboring lobes. σ_h denotes the standard deviation of the lobes in the texture domain. We set h to half a texel and $\sigma_h = h/\sqrt{8\log 2}$ as in [YHMR16].

Intrinsic Roughness: For glint materials, we introduce a limited and tiny amount of intrinsic roughness σ_r to avoid the singularities that may arise on a perfectly specular surface. For structured materials, the normal distribution can be considered as few certain lobes, and the shape of lobes in vector space is the same as that of the standard analytical BRDF models. We use the intrinsic roughness σ_r of structured surface to simulate real structured-surfaces.

Ray Footprint: Our model simulates light reflection through a ray footprint patch $\mathcal{P}$. The ray footprint [Ige99] represents the entire area defined by the intersection of the central light and the following lights on the surface of the object (a total of three lights). In this paper, we define this sampling patch as the ray footprint, and the size of $\mathcal{P}$ is related to the sampling rate.

4. Position-normal Distribution Construction

4.1. High-Resolution Normal Maps Generation

To generate high-resolution normal maps $\mathcal{H}$, we pick blocks $\mathcal{B}$ from $\mathcal{I}$ and patch chosen blocks, considering the influence of boundaries (see Fig.3(a)). Some ideas of the generation method are borrowed from the texture synthesis algorithm:
1. Randomly pick a block $\mathcal{B}_0$ from $\mathcal{I}$ and place it onto the upper left corner of $\mathcal{H}$.
2. For the existing old block $\mathcal{B}_i$ on $\mathcal{H}$, traverse $\mathcal{I}$ to choose some new blocks. For each block, calculate the overlap area error between the newly chosen block and the old blocks $\mathcal{B}_i$. Randomly pick a new block that satisfies the error constraints. If there is no block satisfies the error constraints, pick a new block with the minimum error.
3. Compute the error surface at the overlap region between the chosen new block and $\mathcal{B}_i$. The boundary between blocks is computed as a minimum cost path through the error surface at the overlap.
4. Locate the new block to $\mathcal{B}_i$'s neighbouring block $\mathcal{B}_j$ on $\mathcal{H}$, which is fixed along the minimum error boundary cut.
5. Repeat step 2-4 until the larger-size normal map is generated.

We can implicitly generate an arbitrarily sized, high-resolution normal map $\mathcal{H}$ from a given input sample normal map $\mathcal{I}$ by using the index of appropriate blocks of $\mathcal{I}$. The boundary cut between two overlapping blocks can be used to ensure that the two neighboring blocks are matched in the best possible way, which can ensure that $\mathcal{H}$ is adequately smooth. By using the mapping relations between $\mathcal{H}$ and $\mathcal{I}$, we create the position mapping function $m(\mathbf{u})$.

4.2. Position-normal Distribution Fitting

The position-normal distribution is defined in the 4D cross space of the surface position and the normal. Continuous normal distribution can be easily fit with 4D lobes. Yan et al. [YHMR16] present an appropriate method for glittery surfaces; accordingly, we focus here on how to describe structured surfaces with 4D lobes.

For structured surface, 4D lobes can be expressed as a product of two linearly independent 2D functions:

$$G_i(u,s) = G_{pi}(u)G_{ni}(s). \tag{1}$$

$G_{pi}(u)$ represents a two-dimensional Gaussian function in the position space:

$$G_{pi}(u) = \frac{1}{2\pi|\sum_{pi}|^{-0.5}} e^{(-\frac{1}{2}(u-u_i)^T \sum_{pi}^{-1}(u-u_i))}. \tag{2}$$

$$\sum_{pi}^{-1} = \begin{pmatrix} {\sigma_h}^2 & 0 \\ 0 & {\sigma_h}^2 \end{pmatrix}. \tag{3}$$

Eq. 2 is the position Gaussian function, which represents a Gaussian region in the position space around u_i.

G_{ni} represents the region in normal space around the value $n(u_i)$. For glittery surfaces, we compute G_{ni} in a similar way to [YHMR16]. For structured surfaces, traditional distribution functions such as Beckmann and GGX can also be used here. Eq. 4 is derived from the Beckman-Spizzichino model:

$$G_{ni}(s) = \frac{e^{-\tan^2(s-s_i)/\sigma_r^2}}{\pi\sigma_r^2\cos^4(s-s_i)}. \tag{4}$$

Moreover, Eq. 5 is derived from the GGX model:

$$G_{ni}(s) = \frac{\sigma_r^2\chi^+(s_i\cdot s)}{\pi\cos^4(s-s_i)\left(\sigma_r^2+\tan^2(s-s_i)\right)^2}. \tag{5}$$

We go on to replace the Beckman model and GGX model with other traditional NDF models in our experiments (Fig.7).

4.3. Acceleration Hierarchy Building

In order to speed up the $\mathcal{P}$-NDF evaluation, we organize the 4D lobes generated from the sample normal map $\mathcal{I}$ into two hierarchies in normal space and position space separately in our preprocessing stage(see Fig.2).

First, lobes are organized separately to the position space and normal space; the position space is the same as the (p,q) domain. The normals of lobes are sorted on the projected hemisphere and re-projected to a 2D texture space (i.e. the normal space). Subsequently, 4D lobes on $\mathcal{I}$ are divided into grids in both the position and normal space. The number of lobes in each grid is relatively balanced. We then simply use a 2D hierarchy to organize the lobes in each grid. By using the 4D bounding boxes of the lobes, a 2D hierarchy is built in a top-down manner, using median splits in the texture space. Furthermore, we cluster lobes for each non-leaf node in the normal space hierarchy; these will be discussed in more detail in Section 4.4.

For glint materials, lobes should be also generated from the overlapping regions in $\mathcal{H}$. We use the same hierarchy building strategy to organize them into a position tree and normal tree; however, we do this without clustering them. For structured materials, there is no need to consider the influence of boundaries.

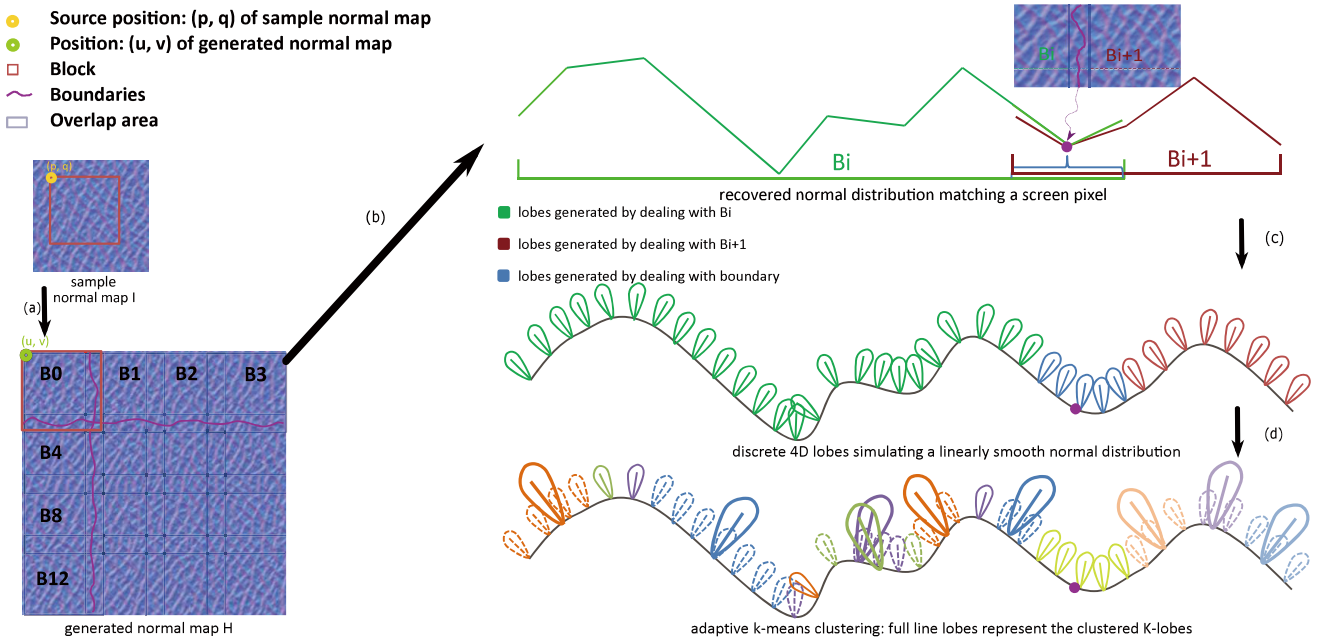

Figure 3: *Our modeling pipeline: Starting from the sample normal map $\mathcal{I}$, we synthesize the large-sized high-resolution normal map $\mathcal{H}$ of any size from the input sample normal map (a); next, we recover the micro-structure of the surface from the normal map $\mathcal{H}$ (b); subsequently, we interpolate the original micro-structure to obtain a smooth surface normal distribution and construct 4D lobes (c); finally we use the adaptive k-means method to cluster the lobes and to fit the normal distribution function with fewer K-lobes (d).*

4.4. Adaptive k-means Clustering

To further accelerate rendering, we cluster similar lobes into a new 4D K-lobe for the inner node in the normal tree hierarchy. The similarity of lobes with their positions and normals are defined as follows:

$$D(G_i, G_j) = \alpha \times \|P_{G_i} - P_{G_j}\|^2 + \|N_{G_i} - N_{G_j}\|^2. \quad (6)$$

with G_i and G_j being two lobes, P being their position, and N being their normal on the projected hemisphere. Furthermore, the weight α trades cluster flatness for spatial extent which is related to the size of a node.

The clustering algorithm (Alg. 1) takes a set of lobes in a tree node as input. We first set the max clustering number MCN (depend on the normal distribution of the materials, we usually set $MCN = 4$) in our practice. Moreover, for each clustering scheme, we try ECN (we usually set $ECN = 5$) times to evaluate the clustering effect and find the best number C of clusters ($C \leq MCN$). Next, we apply the k-means algorithm to cluster lobes. Finally, we obtain the C clusters and consider each cluster as a 4D K-lobe:

$$K_j(u,s) = N \cdot G_{pj}(u) G_{nj}(s). (1 \leq j \leq C) \quad (7)$$

Here, N denotes the number of lobes in a cluster, $G_{pj}(u)$ refers to the 2D Gaussian in position space, and $G_{nj}(s)$ defines the normal around the center normal n_{uj} in vector space.

Algorithm 1 Adaptive K-means clustering

```
for i = 0 to MCN do
    clusterdist[i] ← 0
    for j = 0 to ECN do
        assignment(G,j)
        update(G,j)
    end for
    clusterdist[i] ← evaluatedist(G)
end for
C ← findbestclusternumber(clusterdist)
while stop condition is not met do
    assignment(G,C)
    update(G,C)
end while
K ← clusteringresult(G)
return K
```

5. SVBRDF Evaluation

We use the following SVBRDF model in our work:

$$f_r(\omega_i,\omega_o) = \frac{F(\omega_i,\omega_h)D_p(\omega_h)G(\omega_i,\omega_o,\omega_h)}{4(\omega_i.n)(\omega_o.n)}. \quad (8)$$

Here, ω_h is the half vector, n represents the surface normal, F refers to the Fresnel reflection coefficient, G is the shadowing and masking term, and D_p represents the normal distribution function over a patch.

5.1. Glittery Surfaces Simulation

$\mathcal{P}$-NDF is an integral function, that can be expressed as in [YHJ*14]:

$$D_p(s) = \int_{-\infty}^{\infty} G_p(u)G_r(n(u)-s)du. \quad (9)$$

Here, s is the normal (half-vector) of interest. G_p represents the Gaussian coefficient in the position space, whose center value is the position of the sampling point and whose standard deviation is related to the size of $\mathcal{P}$. G_r denotes the inner roughness Gaussian with zero mean, and the standard deviation is related to the intrinsic roughness coefficient.

Next, we reveal the normal distribution with discrete 4D lobes, then convert the integral function into an accumulation of the contribution of the K-lobes (K_j) generated from the sample normal map and the lobes (G_i) located on the overlapping regions:

$$G_r(n(u)-s) = \sum_{i=1}^{\tau} G_i(u,s) + \sum_{j=1}^{z} K_j(u,s). \quad (10)$$

Thus, the $\mathcal{P}$-NDF query (for a given $\mathcal{P}$ and s) can be written as follows:

$$D_p(s) = \sum_{i=1}^{\tau}\int_{-\infty}^{\infty} G_p(u)G_i(u,s)du + \sum_{j=1}^{z}\int_{-\infty}^{\infty} G_p(u)K_j(u,s)du. \quad (11)$$

5.2. Structured Surfaces Simulation

The surface normal distribution could be fitted by clustered 4D lobes:

$$G_r(n(u)-s) = \sum_{j=1}^{z} K_j(u,s). \quad (12)$$

As the structured materials don't show spatial varing features around boundaries, there is no need to deal with the lobes around the boundary. Moreover, the $\mathcal{P}$-NDF of the structured surface is represented as follows:

$$D_p(s) = \sum_{j=1}^{z}\int_{-\infty}^{\infty} G_p(u)K_j(u,s)du. \quad (13)$$

5.3. Hierarchy Traversal

In order to evaluate the $\mathcal{P}$-NDF for glittery surfaces, we should search the two normal trees for sample normal map and the overlapping regions; this is done in order to obtain the contributed K-lobes and lobes over the patch $\mathcal{P}$. Experiments show that the probability of searching the lobe trees near the boundary is less than 5%, which only has a little impact on evaluation efficiency. Different from the glittery surface, the acceleration structures for the lobes generated from the sample normal map are sufficient to evaluate $\mathcal{P}$-NDF. We therefore focus on the K-lobe searching here.

A $\mathcal{P}$-NDF evaluation query in the normal tree is given by a rectangle in (u,v) space bounding the footprint Gaussian $G_p(u)$, and a point in (s,t) space specifying the half-vector of interest. The (u,v) space rectangle is first projected into (p,q) space by the function $m(\mathbf{u})$. The contributing lobes can be found by a top-down traversal. If the bounding box of a node does not intersect with the query, the node should be pruned. Otherwise, the K-lobes inside this node should be detected to determine whether they intersect. If an intersected K-lobe is found, the querying stops and the K-lobes' contribution to the $\mathcal{P}$-NDF is computed. If no K-lobes are found in a node, we go on to search its children until we find the intersected K-lobes or lobes on the leaf.

To integrate this SVBRDF model with a Monte Carlo multiple importance-sampling framework [Vea97], we also need to sample the $\mathcal{P}$-NDF on a given footprint P. In the present work, for the sake of computational efficiency and accuracy, we sample the discrete lobes G instead of the K-lobes; this is achieved by searching the position tree of lobes by a top-down traversal. We provide the peso code of $\mathcal{P}$-NDF evaluation in supplementary materials.

6. Results

We integrate our SVBRDF model in Mitsuba framework and compared our algorithm against [YHMR16], which can be considered as the reference for quality validation. All rendering results are with global illumination by path tracing. Normal map contrast was enhanced for visualization purposes in all figures. Table 2 shows the related settings of materials for all our test scenes. All timings in this section are measured on a 2.20GHz Intel Xeon (22 cores) with 128 GB of memory.

6.1. Rendering Correctness

Fig.4 and Fig.5 compare the rendering results of our $\mathcal{P}$-NDFs computation method with those of [YHMR16].

Steel Kettle: Fig.4 presents images of a scratched steel kettle. The scratched feature of the reference algorithm [YHMR16] exhibits strong continuity and randomness (Fig.4(b)). Our method (Fig.4(c)) can simulate this randomness well by using a small-sized sample normal map as input. Compared with the simply tiled textures (which have lost the randomness of the scratched materials (Fig.4(d))), we found that our method can generate continuous result. Fig.4(e) also illustrates that our method works well for the brushed metal, where cracks can be easily seen when the normal maps are randomly tiled.

Indoor Scene : Fig.5 (top) presents the rendering results using

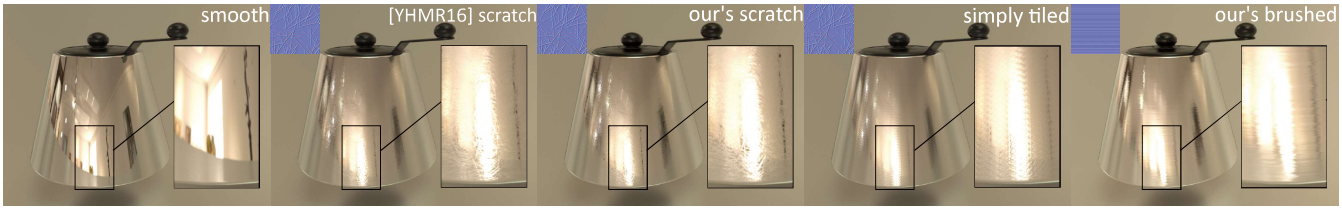

Figure 4: *Comparison with reference [YHMR16] and tiled normal maps on the Steel Kettle Scene. (a) displays a smooth steel kettle rendered using a standard microfacet BRDF model. (b-d) present a scratched steel kettle rendered with the same texture pattern. (b) is based on the randomly distributed high-resolution normal map (2048×2048) using [YHMR16]. (c) is rendered by our SVBRDF model with a sample map (350×350). (d) is rendered using a simply tiled small-sized normal map. (e) displays a steel kettle with brushed metal using our method.*

Figure 5: *Comparison with reference [YHMR16] of an Indoor scene (top) and a Leather Shoes scene (Bottom). The left column is rendered by [YHMR16], while the right column is rendered by our method.*

our stationary SVBRDF model for different microfacet distributions in a complex scene. Two kinds of glint materials are utilized: varnish wooden material and leather. The anisotropic crystal ball metal plate is a structured material. Our approach can process different microfacets well, and the rendering result is visually identical to the reference [YHMR16].

Leather Shoes: Fig.5 (bottom) shows shoes with a very fine leather grain. Our result (right) has no seams, and we are able to capture the randomly distributed glint features as well as Yan's method [YHMR16] (left).

Fig.1 also illustrated the results of rendering using our stationary SVBRDF model. The Disk and Teapot exhibit a different specular highlight appearance due to the use of normal maps with regular patterned structures, as well as the glinty effect, achieved using a normal map with random flakes. It should also be noted that our algorithm is good at preserving temporal coherence; please refer to the companion video.

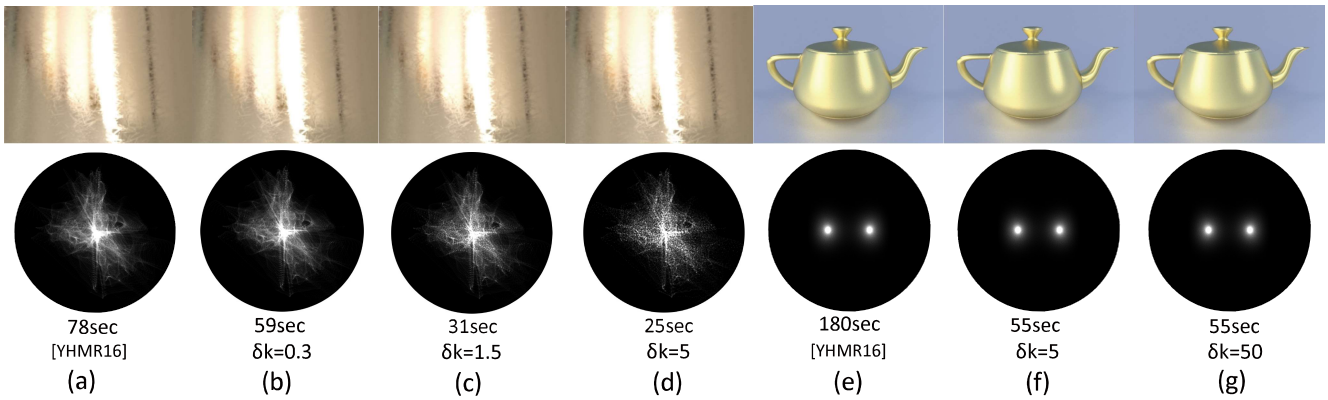

Figure 6: *Comparison of $\mathcal{P}$-NDFs evaluated by our method (with different clustering coefficient, b-d, f-g) to the $\mathcal{P}$-NDFs computed by [YHMR16] (a,e) for the scratched steel kettle scene and the teapot with structured surface.*

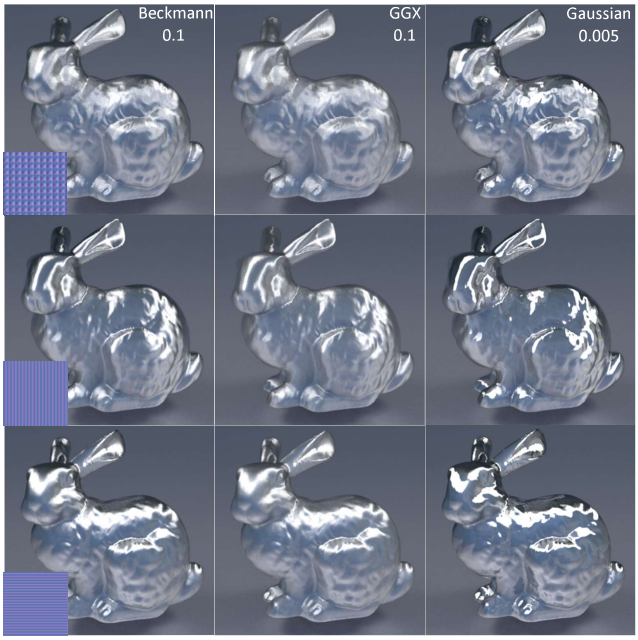

Figure 7: *A Metal Bunny rendered with structured normal maps (top to bottom: pyramidal structures, lateral grooved structures, lengthways structures) and different types of lobes (left to right: Beckmann, σ_r =0.1 ; GGX, σ_r =0.1 and Gaussian, σ_r =0.005).*

6.2. Structured surface rendering

Our SVBRDF model can be efficiently applied to structured surfaces. Fig.7 presents a metallic bunny rendered using different normal distribution expression methods. From the figure, we can see that various specular reflection effects can be simulated using different shapes of lobes.

6.3. Performance and Timings

Table 3 displays the timings of a typical frame for all our test scenes. We report the computation times of our method and the reference method [YHMR16]. We report both the total computation time, the preprocessing time required for 4D microfacets tree construction, and the rendering time for SVBRDF evaluation. Our method provides an overall speedup 5-9 times faster compared to [YHMR16].

During the preprocessing stage, our method can achieve an acceleration of a factor of 300 times for a structured surface (e.g. Teapot). For general glint materials (e.g. Leather Shoes), moreover, our method is more than 100 times faster than the reference [YHMR16]. For scratched materials with strong randomness (e.g. Steel Kettle), we use a larger-sized sample normal map as input; the preprocessing time is more than 50 times faster than [YHMR16]. This preprocessing speedup occurs because the number of lobes is reduced by using a sampled normal map as input.

During the rendering process, we achieve a speedup of roughly 1.2× relative to [YHMR16] for glittery surfaces. For structured surfaces (e.g. teapot), the speedup is more significant, i.e. about 2.7× faster. The speedup is achieved primarily due to the adaptive k-means clustering method, which simplifies the process of searching for contributed lobes.

6.4. Storage space

The storage required by our method is determined by the acceleration hierarchies of lobes, which are closely related to the size of the input normal map. Table 3 also compares the storage of our method and [YHMR16]. Our method only requires storage of the lobes generated from the small-sized sample normal maps and the lobes near the boundary. By contrast, the reference method [YHMR16] requires storage of the entire lobes generated from high-resolution normal maps. Accordingly, the use of our method can result in dramatic savings in terms of storage.

For general glint materials like the Leather Shoes, a small-sized normal map is used as input; our storage requirements are about

Table 2: *Parameters used for different materials. Res. denotes the resolution of the input normal map. δ_k is the clustering coefficient used in our method.*

Scene	Teapot	Steel Kettle	Leather Shoes	Indoor Scene		
Objects	teapot	kettle	shoes	sofa legs crystal ball base	sofa seat	crystal ball plate
Material	anisotropic metal	scratched metal	leather	varnish wooden	leather	anisotropic metal
Res.([YHMR16])	8192×8192	2048×2048	2048×2048	2048×2048	2048×2048	8192×8192
Res.(Ours)	100×100	350×350	240×240	240×240	240×240	100×100
δ_k(Ours)	30	0.3	0.4	0.4	0.3	30

Table 3: *Computation time and memory costs for our test scenes.*

Scene	Preprocessing Time (min)			Rendering Time (min)			Total Time(min)			Memory Costs(MB)	
	[YHMR16]	Ours	Speedup	[YHMR16]	Ours	Speedup	[YHMR16]	Ours	Speedup	[YHMR16]	Ours
Teapot	62	0.2	310×	37.5	13.5	2.7×	99.5	13.7	7.3×	11537	23.5
Steel Kettle	36.7	0.7	52.4×	9.3	7.8	1.19×	46	8.5	5.4×	1203	156
Leather Shoes	33.2	0.3	110.6×	10.7	8.7	1.22×	43.9	9	4.9×	1208	102
Indoor Scene	156	1.2	120×	21.5	18.6	1.15×	177.5	19.8	8.9×	13300	273

8% of Yan's [YHMR16]. For the scratched Steel Kettle, we use a larger-sized sample normal map to guarantee the scratched effect; the storage space required by our method is about 13% of that required by [YHMR16]. Moreover, we only need a very small-sized normal map for the anisotropic metal, using only 0.2% of the storage required by [YHMR16]. For the complex Indoor Scene, the overall storage required by our method is about 2% of Yan's [YHMR16].

6.5. Analysis of the *k*-clustering

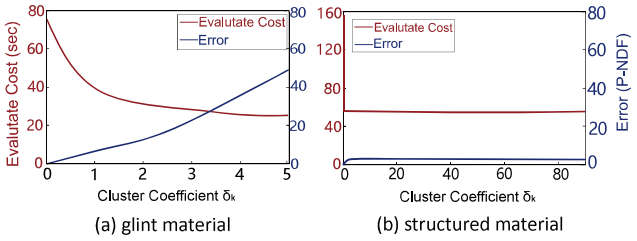

Figure 8: *Error and rendering cost ($\mathcal{P}$-NDF evaluation time) as a function of clustering coefficient δ_k for the Scenes in Fig.6.*

Fig.6 presents the results of rendering using different clustering coefficients δ_k for our *k*-means lobe clustering method. We also evaluate the $\mathcal{P}$-NDFs in Fig.6 with a pixel footprint at current rendering scale.

We can see that scratched details cannot be maintained when more lobes are clustered (Fig.6(a-d)). Our result with $\delta_k = 0.3$ (Fig.6(b)) closely matches the reference [YHMR16] (Fig.6(a)), with a speedup of 1.3× compared to [YHMR16]. Our result with $\delta_k = 5$ (Fig.6(c)) exhibits an obvious differences from the baseline image, but is 3.1× faster than [YHMR16].

However, the specular objects with structured surfaces could achieve a 3.2× speedup compared with [YHMR16] through the use

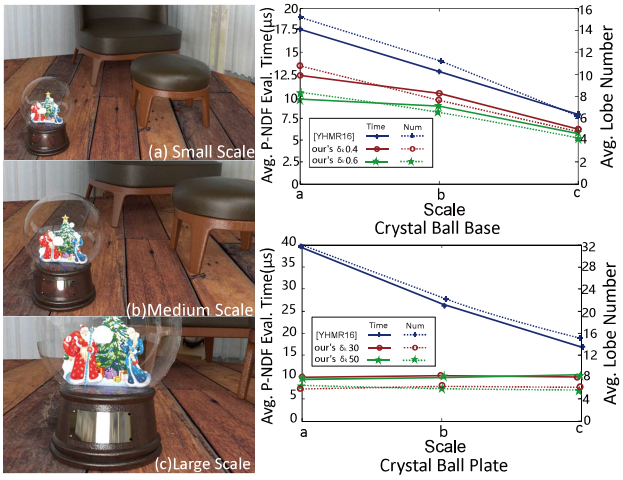

Figure 9: *Average $\mathcal{P}$-NDF evaluation time and average number of contributed lobes of a shading point for the reference [YHMR16] and our method (with different clustering coefficient) on the Indoor Scene for different levels of zoom during rendering.*

of our clustered method with a high quality specular phenomena (Fig.6(f,g)). The quality and speed of our method remains stable once the clustering is used.

Fig.8 displays the behavior of our algorithm as we change the parameter δ_k, using the Scenes in Fig.6. We compute both the error (evaluated by comparing the $\mathcal{P}$-NDFs of our algorithm with [YHMR16]) and the rendering cost (measured as the SVBRDF evaluation time). For glinty materials, increasing δ_k decreases the rendering time, at the expense of increasing error. $0.3 < \delta_k < 1$ appears to be a reasonable compromise for glinty surfaces(Fig.8(a)). For structured surfaces (Fig.8(b)), since the feature of the struc-

tured material is simple, δ_k doesn't affect the rendering efficiency and accuracy.

Fig.9 compares the $\mathcal{P}$-NDF evaluation time and average number of contributed lobes of our method with varying δ_k when zooming in on the Indoor Scene. Compared to [YHMR16], the speedup for both glinty and structured surfaces is much larger, and fewer related lobes are detected in the normal tree when zooming out. For glinty surfaces (e.g. Ball Base), we obtain a higher speedup and fewer number of lobes when we use a larger δ_k, especially when zooming out. For structured surfaces, the $\mathcal{P}$-NDF evaluation time and related lobe numbers are not influenced by δ_k or the view scales.

6.6. Limitations

Figure 10: *Comparison between our algorithm and Yan et al. [YHMR16] for scenes using wood material. The input normal map size of [YHMR16] is* 1024×1024*; the input texture size of our method is* 350×350*.*

As the high-resolution normal map is generated by the texture synthesis strategy in our method, our method doesn't support the simulation of materials without self-similar features (see Fig.10). The wood texture and normal map are correlated but independent for physically accurate rendering of wood materials. However, our method can not imitate the annual ring structure on large scale, since the sample normal map is unable to contain spatial varying patterns.

7. Conclusion and Future Work

This paper presents a stationary spatial varying appearance modeling method for both discrete and structured micro-structure surfaces. A texture synthesis-based implicit high-resolution normal map generation method is proposed to decrease the memory cost, while a *k*-means clustering method is also applied to improve our method's overall efficiency. Our work provides engineers with a stationary method to model complex microfacet-based surfaces that are difficult to be described using analytic expressions. Our method requires significantly less pre-processing time and memory storage, and also enables a faster rendering than the state-of-the-art method presented in [YHMR16]. In future work, it will be interesting to consider the influence of the shadowing-masking function. We also plan to port the model for interactive rendering on the GPU.

Acknowledgments. We thank the reviewers for their valuable comments. This work has been partially supported by the National Key R&D Program of China (under grant No. 2017YFB0203000), the National Natural Science Foundation of China (under grants No. 61872223, 61802187, 61702311), the Young Scholars Program of Shandong University (under grant No. 2015WLJH41).

References

[CT82] COOK R. L., TORRANCE K. E.: A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG) 1*, 1 (1982), 7–24. 2

[DWMG15] DONG Z., WALTER B., MARSCHNER S., GREENBERG D. P.: Predicting appearance from measured microgeometry of metal surfaces. *Acm Transactions on Graphics 35*, 1 (2015), 1–13. 2

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 341–346. 3

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision* (1999), vol. 2, IEEE, pp. 1033–1038. 3

[HSRG07] HAN C., SUN B., RAMAMOORTHI R., GRINSPUN E.: Frequency domain normal map filtering. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), 28. 2

[Ige99] IGEHY H.: Tracing ray differentials. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 179–186. 4

[JHY*14] JAKOB W., HAŠAN M., YAN L.-Q., LAWRENCE J., RAMAMOORTHI R., MARSCHNER S.: Discrete stochastic microfacet models. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 115. 2

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (ToG) 22*, 3 (2003), 277–286. 3

[RGB16] RAYMOND B., GUENNEBAUD G., BARLA P.: Multi-scale rendering of scratched materials using a structured sv-brdf model. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 57. 2

[TS67] TORRANCE K. E., SPARROW E. M.: Theory for off-specular reflection from roughened surfaces. *Josa 57*, 9 (1967), 1105–1114. 2

[Vea97] VEACH E.: *Robust Monte Carlo methods for light transport simulation*, vol. 1610. Stanford University PhD thesis, 1997. 6

[VWH18] VELINOV Z., WERNER S., HULLIN M. B.: Real-time rendering of wave-optical effects on scratched surfaces. *Computer Graphics Forum 37*, 2 (2018), 123–134. 2

[WDR09] WU H., DORSEY J., RUSHMEIER H.: Characteristic point maps. In *Twentieth Eurographics Conference on Rendering* (2009), pp. 1227–1236. 2

[WDR11] WU H., DORSEY J., RUSHMEIER H.: Physically-based interactive bi-scale material design. *ACM Transactions on Graphics (TOG) 30*, 6 (2011), 145. 2

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 479–488. 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 Transactions on Graphics 33*, 4 (2014), 1–9. 2, 6

[YHMR16] YAN L. Q., HAŠAN M., MARSCHNER S., RAMAMOORTHI R.: Position-normal distributions for efficient rendering of specular microstructure. *Acm Transactions on Graphics 35*, 4 (2016), 56. 2, 4, 6, 7, 8, 9, 10

[YHW*18] YAN L.-Q., HAŠAN M., WALTER B., MARSCHNER S., RAMAMOORTHI R.: Rendering specular microgeometry with wave optics. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 75. 2

[ZK16] ZIRR T., KAPLANYAN A. S.: Real-time rendering of procedural multiscale materials. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2016), ACM, pp. 139–148. 2