Real-Time Microstructure Rendering with MIP-mapped Normal Map Samples

Haowen Tan [1]*, Junqiu Zhu [1]*, Yanning Xu [1]†, Xiangxu Meng [1], Lu Wang [1] and Ling-Qi Yan [2]

[1] School of Software, Shandong University, China
hw_tan@foxmail.com, zhujunqiu@mail.sdu.edu.cn, xyn@sdu.edu.cn, mxx@sdu.edu.cn, luwang_hcivr@sdu.edu.cn
[2] University of California, Santa Barbara, USA
lingqi@cs.ucsb.edu

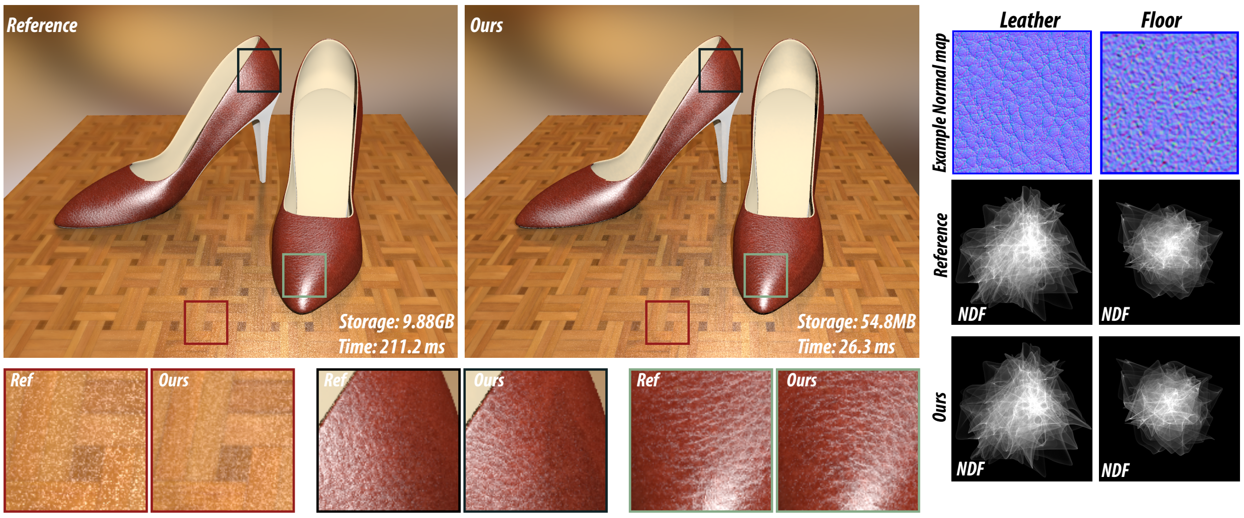


Figure 1: *We render fancy leather shoes and the sparkling wood floor lit by a strong point light and an environment map. We implement the method of Yan et al. [YHMR16] on GPU as the reference. The reference's input normal map resolution is $4K^2$, while our input example normal map resolution is 256^2. Our method's implicitly generated normal maps have the same resolution as the reference. The reference's memory cost of each material is as high as 4.94GB, while ours is only 27.4MB (14.3MB for the MIP-mapped organized 4D lobes and 13.1MB for a lookup table). The rendering quality of our method is similar to the reference. In addition, our method costs 26.3ms for global illumination, with less than 19.2ms for direct illumination, and reaches up to 52 fps for direct illumination with a full high-definition resolution, which is 10.6× faster than the reference on the* ***Leather Shoes and Wooden Floor Scene****.*

Abstract

Normal map-based microstructure rendering can generate both glint and scratch appearance accurately. However, the extra high-resolution normal map that defines every microfacet normal may incur high storage and computation costs. We present an example-based real-time rendering method for arbitrary microstructure materials, which significantly reduces the required storage space. Our method takes a small-size normal map sample as input. We implicitly synthesize a high-resolution normal map from the normal map sample and construct MIP-mapped 4D position-normal Gaussian lobes. Based on the above MIP-mapped 4D lobes and a LUT (lookup table) data structure for the synthesized high-resolution normal map, an efficient Gaussian query method is presented to evaluate $\mathcal{P}$-NDFs (Position-Normal Distribution Functions) for shading. We can render complex scenes with glint and scratch surfaces in real-time ($\geq$ 30 fps) with a full high-definition resolution, and the space required for each microstructure material is decreased to 30MB.

Keywords: Rendering, Surface Microstructure, Glints, Real-time, MIP-map

CCS Concepts

• ***Computing methodologies*** → *Ray tracing; Reflectance modelling;*

*Dual First Authors

1. Introduction

Many glossy materials with complex microstructures show high-frequency highlights, such as glittery, scratched, or brushed metal materials. Efficiently modeling and evaluating the complex microstructures becomes an interesting topic in physically based rendering, especially in the modern Graphics Processing Unit (GPU) context.

Yan et al. [YHJ*14, YHMR16] provided a general approach by using normal maps which can represent both glints and scratches. Since the required high-resolution normal maps may reach the upper bound of GPU storage, the work is not GPU friendly in a scene with many high-frequency materials. Some recent works are designed for real-time microstructure rendering based on approximate representations. Zirr and Kaplanyan [ZK16] introduced a stochastic bi-scale microfacet model to fit glints. Velinov et al. [VWH18] presented a set of analytical pre-integrations and an approximate closed-form solution for iridescent scratches. Wang et al. [WDH20] introduced an approximated separate radiance convolution model for glints. These real-time algorithms focus on special glossy effects and are not suitable for arbitrary complex specular microstructures.

We present a real-time microstructure rendering method based on the microfacet-based BRDF model of Yan et al. [YHMR16]. Our method is physically based and can simulate all types of glossy materials. We greatly decrease the memory cost by implicitly generating microstructures from a normal map sample. We use 4D Gaussian lobes to fit the position-normal distribution. Unlike Zhu et al. [ZXW19] and Wang et al. [WHHY20], our synthesis is implemented on 4D Gaussian lobes instead of the 2D textures, which can give the generated 4D lobes better coherence. By using a MIP-mapped data structure of 4D lobes, we achieve fast range queries on GPU to accurately evaluate $\mathcal{P}$-NDF contained in a ray footprint. Our method has the following contributions:

- We provide a real-time rendering method, which fits all types of high-frequency materials, including anisotropic highlights caused by structured materials, glinty appearance caused by discrete materials, and mixed materials of both discrete and structured materials. Moreover, we implement it in the GPU raytracing rendering pipeline in real-time.
- We present a 4D position-normal lobe generation method for the infinite size microstructure. We generate 4D Gaussian lobes from a fixed-size normal map sample and then synthesize these lobes instead of normal map pixels to describe the microstructure.
- We also introduce a MIP-mapped data structure to organize 4D Gaussian elements on GPU, which supports fast lobe queries for ray footprints to evaluate $\mathcal{P}$-NDF.

2. Related Work

The conventional microfacet BRDF models the detailed surface microstructure with smooth normal distribution functions (NDFs), including Beckmann [BS87] and GGX [WMLT07] distributions. They are suitable for rendering isotropic highlights under distant views but are not enough to deal with high-frequency glint microstructures. Recent works mainly focus on the rendering of structured and discrete microfacet surfaces.

Structured materials modeling: For the simulation of anisotropic highlight distribution caused by structured materials, the nonlinear pre-filtering normal distribution is required to maintain the material model's features and avoid aliasing during rendering. Westin et al. [WAT92] proposed a multi-scale rendering method for structured materials and addressed the necessity of the research on this feature. Fournier [Fou92] discussed how to combine the normal distribution of the structured high-frequency material with the traditional BRDF. Becker and Max [BM93] used multiple BRDF hierarchies to achieve smooth transitions at different scales when rendering structured materials. Kautz and Seidel [KVHS00] intended to parameterize and pre-calculate BRDF models as a set of nonlinear bases, efficiently rendering structured materials on GPU. Toksvig [Tok05] proposed a practical method, which uses an anisotropic Gaussian lobe to fit the normal distribution of structured materials and multi-scale nonlinear filtering for acceleration. Han et al. [HSRG07] used several lobes to simulate the normal distribution functions, which can render a more complex anisotropic highlight. Wu et al. [WDR11] proposed a bi-scale model, which derives a large-scale appearance from the small-scale geometry features designed by users.

Discrete materials modeling: Recent works try to model discrete high-frequency materials such as glinty, scratched, and brushed marks to simulate their complicated reflectance properties.

For offline rendering, Jakob et al. [JHY*14] addressed the problem of glinty surfaces using a stochastic approach. They modeled surfaces covered by pixel footprints as several randomly distributed mirror-like flakes. Then they found an effective way to evaluate the percentage of flakes in the specific spatial and directional domains. This method can only be applied to the glinty surface caused by discrete mirror flakes. Wang et al. [WWH18] further extended the method to make the model separable and filterable. This method is more efficient but is still designed for offline rendering.

Yan et al. [YHJ*14] proposed a method to render spatially varying high-frequency discrete materials defined by high-resolution normal maps. Yan et al. [YHMR16] further used the 4D Gaussian mixture elements to simulate the normal distribution of discrete high-frequency materials and achieved better performance than their previous work. Yan et al. [YHW*18] also extended the rendering schemes for discrete materials to deal with wave optics cases. Cheirmain et al. [CCM19] improved the normal mapping-based wave optics rendering. However, all these methods need high-resolution normal maps as input to avoid artifacts.

Since the explicit microstructure is costly to store, a series of methods are designed to reduce storage by dynamically generating large-sized normal maps by small-sized samples. These methods consider the self-similar feature of materials and use the texture synthesis idea. Zhu et al. [ZXW19] synthesized the high-resolution normal maps and stored the corresponding relationship between the Gaussian lobes and the synthesized index position. This work can generate large-scale texture maps while maintaining random patterns. However, lobes at the boundaries between the composite blocks need to be processed to ensure the continuity of normal.

[†]Corresponding Author

Also, the storage space will increase as the size of the synthetic high-resolution normal maps increase. Wang et al. [WHHY20] used the blending method, which ensures a constant storage space. However, since the method does not store lobe information, it generates lobes during the rendering process, which brings a certain amount of additional time overhead. The mixed information blurs the high-frequency features within a specific range, and the spatial continuity of features cannot be well maintained. Our method synthesizes Gaussian lobes instead of normal maps, which can precisely deal with arbitrary microstructures accurately and be adapted to the GPU easily in our practice.

Real-Time high-frequency material rendering: Current real-time research works mainly focus on dealing with certain high-frequency materials. Zirr et al. [ZK16] proposed a stochastic bi-scale microfacet model for real-time rendering of multi-scale glint features, including the discrete glint materials and brushed marks. Wang et al. [WDH20] proposed a pre-filtering method for the stochastic discrete microfacet model to simulate glints under both environment maps and point light sources in real-time. Velinov et al. [VWH18] proposed the treatment of scratches under wave optics. However, no existing real-time solutions can consistently process high-frequency materials explicitly defined by high-resolution normal maps. To our knowledge, this is the first method to render glints, scratches, and their mixtures in one complex scene in real-time. Furthermore, the GPU memory is almost constant, which is 30MB in our practice.

3. Background

The discrete microfacet model [YHMR16] uses high-resolution normal maps to model the microstructure and can be defined as:

$$f_r(\omega_i,\omega_o) = \frac{F(\omega_i,\omega_h)D_{\mathcal{P}}(\omega_h)G(\omega_i,\omega_o,\omega_h)}{4(\omega_i.\mathbf{n})(\omega_o.\mathbf{n})}. \tag{1}$$

Here, ω_h is the half vector computed by normalizing the sum of the light vector ω_i and the view vector ω_o, $\mathbf{n}$ represents the surface normal, F refers to the Fresnel reflection function, and G is the shadowing and masking term. $\mathcal{P}$ is defined on the texture space and is as large as the pixel projection onto the surface. $D_{\mathcal{P}}(\omega_h)$ is also named $\mathcal{P}$-NDF, and represents the normal distribution function over a spatial footprint $\mathcal{P}$ according to the querying direction ω_h.

The evaluation of $\mathcal{P}$-NDF can be written as:

$$D_{\mathcal{P}}(\mathbf{s}) = \int_{\mathbb{R}^2} G_{\mathcal{P}}(\mathbf{u})\mathcal{N}(\mathbf{u},\mathbf{s})d\mathbf{u}. \tag{2}$$

where $\mathbf{s}$ is the query direction, and $\mathbf{u}$ is the query position. $G_{\mathcal{P}}(\mathbf{u})$ is the Gaussian distribution function about position $\mathbf{u}$. $\mathcal{N}(\mathbf{u},\mathbf{s})$ is the position-normal distribution and it can be further expressed as $\mathcal{N}(\mathbf{u},\mathbf{s}) = G_r(\mathbf{n}(\mathbf{u})-\mathbf{s})$. The Gaussian G_r specifies the closeness between the normal in position $\mathbf{u}$ and the query direction $\mathbf{s}$ with an intrinsic roughness parameter σ_r. Yan et al. [YHMR16] traversed each texel in the normal map texture space at the same step size to obtain Gaussian $L(\mathbf{u},\mathbf{s})$ with standard deviation σ_h. They approximated $\mathcal{N}(\mathbf{u},\mathbf{s})$ using many 4D Gaussian lobes:

$$\mathcal{N}(\mathbf{u},\mathbf{s}) \approx \sum_{i=1}^{k} L_i(\mathbf{u},\mathbf{s}). \tag{3}$$

Each lobe is defined as:

$$L_i(\mathbf{u},\mathbf{s}) = c_i e^{-\frac{1}{2}\left((\mathbf{u},\mathbf{s})^T-(\mathbf{u}_i,\mathbf{s}_i)^T\right)^T \Sigma_i^{-1}\left((\mathbf{u},\mathbf{s})^T-(\mathbf{u}_i,\mathbf{s}_i)^T\right)}, \tag{4}$$

where c_i is a constant for normalization, and $\sum_i$ is the covariance matrix computed from the Jacobian $\mathbf{J}$ of the normal $\mathbf{n}(\mathbf{u})$. The inverse of this 4×4 covariance matrix can be expressed as:

$$\Sigma_i^{-1} = \frac{1}{\sigma_h^2}\begin{pmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix} + \frac{1}{\sigma_r^2}\begin{pmatrix}\mathbf{J^T J} & -\mathbf{J^T}\\ -\mathbf{J} & \mathbf{I}\end{pmatrix}. \tag{5}$$

Thus, the evaluation of $\mathcal{P}$-NDF for a given $\mathcal{P}$ and $\mathbf{s}$ becomes:

$$D_{\mathcal{P}}(\mathbf{s}) \approx \sum_{i=1}^{k}\int_{\mathbb{R}^2} G_{\mathcal{P}}(\mathbf{u})L_i(\mathbf{u},\mathbf{s})d\mathbf{u}. \tag{6}$$

In Equation 6, since $\mathbf{s}$ is fixed, the integral inside the sum collapses to a 2D Gaussian, which leads to a closed-form solution. Yan et al. [YHMR16] suggested that converting texels to 4D Gaussian lobes would suffice to produce high-quality results. However, this indicates that millions of Gaussian lobes will be retrieved in each query. In their work, they implement a 4D acceleration hierarchy over both normal and position space to organize these lobes. The query is performed from top to bottom in the hierarchical structure, and only the 4D lobes within the specified range need to be calculated.

4. Overview

Fig. 2 presents a summary of our algorithm. The rendering pipeline includes three stages:

- Preprocessing:
  - Generate 4D lobes from an input sample normal map and synthesize the corresponding LUT (Section 5.1.1).
  - Generate the MIP-mapped structure to accelerate $\mathcal{P}$-NDF evaluation (Section 5.1.2).
- Shading:
  - Locate MIP-mapped levels and query matched 4D lobes (Section 5.2.1).
  - Evaluate $\mathcal{P}$-NDF by interpolating between two MIP-mapped levels (Section 5.2.2).
- Postprocessing:
  - Denoise indirect illumination by applying a spatial filtering pass. Combine the denoised indirect and direct illumination to get a global lighting effect.

In the preprocessing stage (Section 5.1), we implicitly synthesize a fixed-size MIP-mapped normal map with a corresponding LUT instead of explicitly generating a large-size high-resolution normal map. We also build the MIP-mapped structure on spatial domain, just like the MIP-map of the texture maps. The difference is that our texel at each level stores the Gaussian approximation of NDF in the position domain.

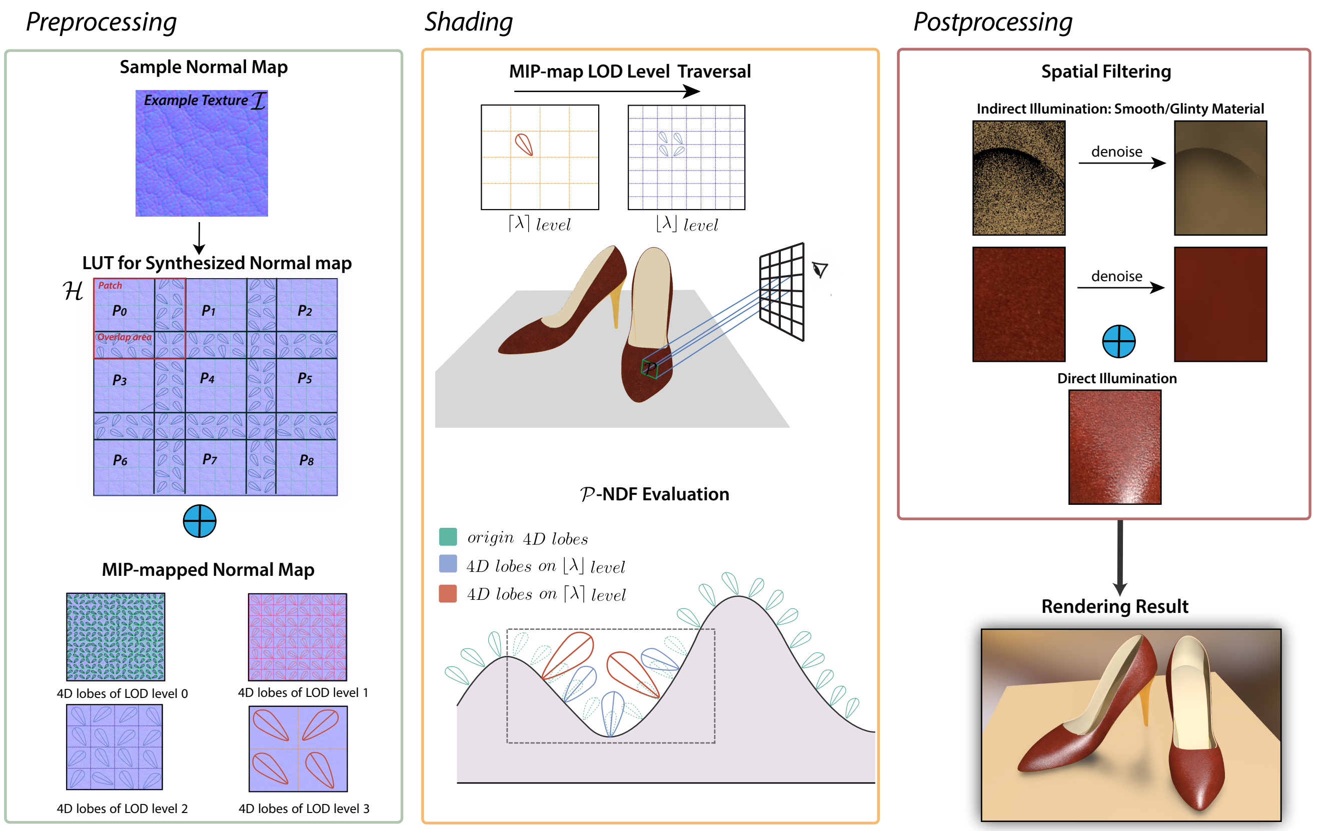

Preprocessing
Shading
Postprocessing
Sample Normal Map
Example Texture $\mathcal{I}$
LUT for Synthesized Normal map
$\mathcal{H}$
Patch
Overlap area
P0 P1 P2 P3 P4 P5 P6 P7 P8
MIP-mapped Normal Map
4D lobes of LOD level 0
4D lobes of LOD level 1
4D lobes of LOD level 2
4D lobes of LOD level 3
MIP-map LOD Level Traversal
$\lceil\lambda\rceil$ level
$\lfloor\lambda\rfloor$ level
$\mathcal{P}$-NDF Evaluation
origin 4D lobes
4D lobes on $\lfloor\lambda\rfloor$ level
4D lobes on $\lceil\lambda\rceil$ level
Spatial Filtering
Indirect Illumination: Smooth/Glinty Material
denoise
denoise
Direct Illumination
Rendering Result

Figure 2: *Pipeline of our algorithms.*

The generated MIP-mapped structure can be reused in the shading stage (Section 5.2). Query time decreases by merging several lobes into a big one and lobe queries can be stopped at a proper MIP-map level. Because the glinty effects mainly come from direct illumination, we separate direct illumination from indirect illumination, and our $\mathcal{P}$-NDF evaluation method is applied only to direct illumination in the shading stage. For indirect illumination, we refer to the work of Yan et al. [YHMR16] and replace the complex microfacet material with traditional material of the same roughness.

Our method is accurate and gives noise-free results in direct illumination, and we use the usual filtering operation to deal with the noise in ray-traced indirect illumination. In the postprocessing stage, we adopt an edge-avoiding wavelet transform filter referred to Dammertz et al. [DSHL10] and a bilateral filter for spatial filtering, while albedo, depth, position normal are collected during ray tracing for keeping illumination features.

5. Real-Time Rendering of High-frequency Materials

5.1. Synthesized MIP-mapped normal map

Some methods [YHMR16,ZXW19] rely on a large number of lobes to fit the microfacet distribution of glinty surfaces. In order to evaluate $\mathcal{P}$-NDF efficiently, they use 4D hierarchy trees to organize the lobes. But this acceleration structure is challenging to port to GPU because the tree pruning operation will consume a long time on GPU, and the hierarchy structure occupies much extra memory. In our work, we synthesize 4D position-normal lobes and organize them with fixed-size MIP-mapped normal maps. In addition, we build a MIP-mapped LUT for GPU-friendly parallel searching.

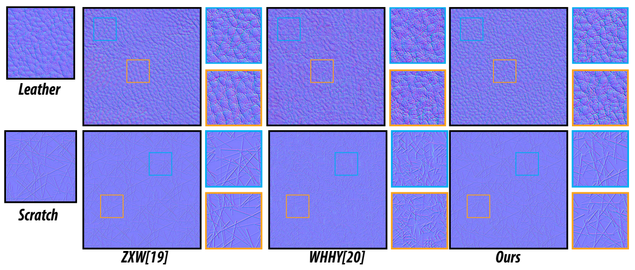


Figure 3: *Comparison with Zhu et al. [ZXW19] and Wang et al. [WHHY20] on the explicitly generated normal map. For structured materials such as leather **(Top row)**, both our method and Zhu et al. [ZXW19] maintain the structural features of the material without visible seams, while Wang et al. [WHHY20] has a significant over-blur at the seams (refer to the inset). For discrete materials such as scratches **(Bottom row)**, our method maintains the local characteristics of the scratches in the normal map sample (refer to the inset) compared to Wang et al. [WHHY20].* α *is set to 0.9.*

5.1.1. Synthesized 4D position-normal lobes

The target large-sized normal map $\mathcal{H}$ is composed of many patches, as illustrated in Fig. 2. A patch can be represented by P_i, which is a square area in the texture space. The size of P_i is user-defined and smaller than the original input sample $\mathcal{I}$. P_i can be understood as a collection of many 4D lobes. The synthesis algorithm of $\mathcal{H}$ can be illustrated as the following steps:

1. Extract 4D lobes from the original sample normal map P_i. The 4D lobes are described by position, direction, Jacobian determinant, and other attributes.
2. Randomly pick a patch P_0 from $\mathcal{I}$ and place it onto the upper left corner of $\mathcal{H}$.
3. For the existing old patch P_i on $\mathcal{H}$, traverse $\mathcal{I}$ to choose some new patches. For each patch, calculate the overlap area error between the 4D lobes contained in the newly selected patch and those contained in the old patches P_i. Randomly pick a new patch that satisfies the error constraints. If no patch satisfies the error constraint, the new patch with the minimum error is selected.
4. Compute the error surface at the overlap region between the chosen new patch and P_i. The boundary between patches is computed as a minimum cost path (refer to Equation 7) through the error surface at the overlap.
5. Position the new patch to P_i's adjacent patch P_j on $\mathcal{H}$, which is fixed along with the minimum error boundary cut.
6. Repeat steps 3-5 until $\mathcal{H}$ is generated.

It has been demonstrated in Yan et al. [YHMR16] that converting a texel in the normal map texture space into four Gaussian lobes with standard deviation $\sigma_h = 1/\sqrt{32log2}$ gives excellent results. We also use the same approach in **step 1**.

The key step of the synthesizing method is to find P_i's proper neighbouring patch P_j from $\mathcal{I}$ (**step 3**). We first randomly choose P_i from $\mathcal{I}$. Then from $\mathcal{I}$, we test many equally sized areas with P_i, find one area that satisfies the overlap area error constraint, and name it P_j. Finally, the lobes near the boundary of patches P_i and P_j are fixed along with the minimum error boundary cut (**step 4**).

When synthesizing high-resolution normal maps from samples, we basically follow the method of Zhu et al. [ZXW19], but different from theirs, we use the Jacobian similarity as part of the error constraint instead. By using this error constraint, we can generate continuous position-normal distributions.

Both the overlap error constraint and the minimum error boundary cut are evaluated by the similarity of lobes, which can give the position-normal distribution of lobes on $\mathcal{H}$ good coherence. The similarity of two neighbouring 4D lobes L_i and L_j is defined in Equation 7, in which α balances the contribution of the normal similarity and the Jacobian similarity.

$$Dist_{ij} = \alpha||\mathbf{n}_i - \mathbf{n}_j||^2 + (1-\alpha)||\mathbf{J}_i - \mathbf{J}_j||^2, \tag{7}$$

where $\mathbf{n}_i$ and $\mathbf{n}_j$ are normals for L_i and L_j. $\mathbf{J}_i$ and $\mathbf{J}_j$ are Jacobian matrix for L_i and L_j. Considering the Jacobian as an error constraint, the synthesized lobes near the boundary of patches on the large-sized map are of good normal and Jacobian similarity.

We compare our texture synthesis method to Zhu et al. [ZXW19] and Wang et al. [WHHY20] in Fig. 3. Our method keeps more details than Wang et al. [WHHY20] and can maintain better coherence of 4D lobes compared to Zhu et al. [ZXW19].

We further define the LUT to save the mapping relationship between $\mathcal{H}$ and $\mathcal{I}$, including the patch index and the lobe index on patch boundaries. By only porting the basic 4D lobe data of the input sample and the LUT for $\mathcal{H}$, the memory cost is mainly determined by the resolution of the input sample, which is suitable for GPU storage.

5.1.2. MIP-mapped acceleration structure

Since lots of nearby 4D lobes face a similar direction, we make use of a LOD MIP-mapped structure to organize original 4D lobes, which is in line with GPU data fetching at the same time. First, the lobes on $\mathcal{I}$ are divided into n spatial grids. Then, the k-means clustering method is used for each grid to aggregate the lobes into k new lobes. During clustering, both position and normal distribution are considered. We use the same k for different grids, making the storage steady.

For $\mathcal{H}$, we always use the bottom level of lobes to process the boundary case for a patch (usually 2 to 4 lobes) when rendering. Furthermore, the patch index of LUT is used to find inner lobes on different MIP-mapped levels.

5.2. Real-Time $\mathcal{P}$-NDF evaluation

5.2.1. MIP-mapped based lobe query

We can directly locate the corresponding LOD level λ according to the ray footprint $\mathcal{P}$ by Equation 8. The ray footprint [Ige99] defines the size of the query area in the texture space.

$$\lambda = C_{\mathcal{P}} log_2 \mathcal{P}, \tag{8}$$

where $C_{\mathcal{P}}$ is the scale factor which controls the ray footprint.

If only lobes on one MIP-mapped level are used to evaluate $\mathcal{P}$-NDF, discontinuities between adjacent pixels will be obvious. In our practice, the strategy of interpolating sampled lobes between two adjacent MIP-mapped levels can resolve the problem perfectly.

For a shading point, the computed λ by Equation 8 is a floating-point number, which means it may be located between two MIP-mapped levels. We set the lower MIP-mapped level as $\lfloor\lambda\rfloor$ and the higher one as $\lceil\lambda\rceil$. $L_i^{\lfloor\lambda\rfloor}$ and $L_i^{\lceil\lambda\rceil}$ are the queried 4D lobes on these two levels.

As shown in Fig. 4, when querying on a MIP-mapped level, we can efficiently compute the four corners of the rectangle defined by pixel footprint. By testing all the lobes located in the axis-aligned bounding box (AABB) of this rectangle on GPU, we can easily cull the lobes that have no intersection with the half-vector in planar space and find the lobes that fit within the rectangle. Lobes on a MIP-mapped level can be tested simultaneously, which is very fast.

5.2.2. $\mathcal{P}$-NDF evaluation

Assuming that a ray footprint $\mathcal{P}$ is given on the microstructure, we need to remove non-contributing lobes and get the candidate lobes using the MIP-map. Wang et al. [WHHY20] used Range Minimum

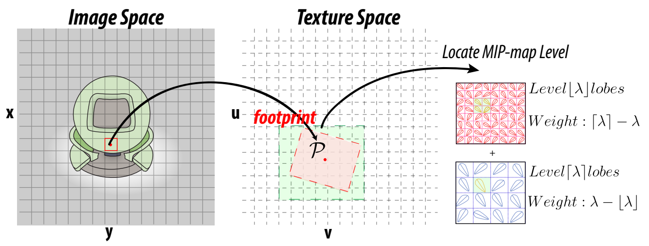
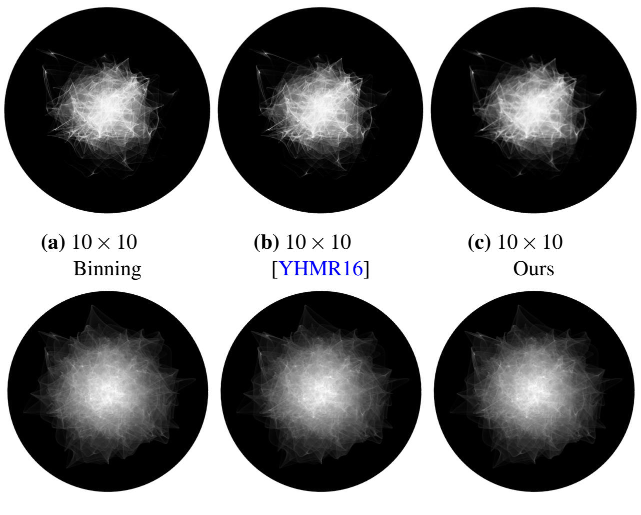


Image Space
Texture Space
Locate MIP-map Level
x
y
u
v
footprint
$\mathcal{P}$
$Level\lfloor\lambda\rfloor lobes$
$Weight : \lceil\lambda\rceil - \lambda$
+
$Level\lceil\lambda\rceil lobes$
$Weight : \lambda - \lfloor\lambda\rfloor$

Figure 4: *For a shading point in image space, we get its footprint in texture space and then locate the MIP-mapped level to query matching lobes (Section 5.2.1). The lobes of adjacent levels are interpolated according to the weights to complete the evaluation of $\mathcal{P}$-NDF (Section 5.2.2).*



(a) 10×10 Binning **(b)** 10×10 [YHMR16] **(c)** 10×10 Ours

(d) 30×30 Binning **(e)** 30×30 [YHMR16] **(f)** 30×30 Ours

Figure 5: *Comparison of $\mathcal{P}$-NDF evaluated by our approach to Yan et al. [YHMR16] and $\mathcal{P}$-NDF computed by binning.* ***Top row:*** *a small pixel footprint covering 10×10 texels.* ***Bottom row:*** *a large pixel footprint covering 30×30 texels.*

Query (RMQ) as the pruning scheme. Yan et al. [YHMR16] retrieved the contribution lobes from top to bottom on the quadtree based on normal and position space. If we implement the above hierarchy structures directly on the GPU, a large number of lobes will be separated into grids. As the lobe number of a single grid increases, too many conditional branches will increase the evaluation time and will bring excessive storage overhead. We use a MIP-mapped acceleration structure, apply the k-means clustering method to generate different levels of the MIP-mapped position normal distribution, and remove the low contribution or non-contribution lobes at the same time in the clustering process.

We query 4D lobes on two MIP-mapped levels (with τ lobes located on level $\lfloor\lambda\rfloor$ and ϕ lobes located on level $\lceil\lambda\rceil$), and the $\mathcal{P}$-NDF evaluation function in Equation 6 can be written as:

$$\begin{aligned} D_{\mathcal{P}}(\mathbf{s}) \approx (\lceil\lambda\rceil - \lambda) \sum_{i=1}^{\tau} \int_{\mathbb{R}^2} G_{\mathcal{P}}(\mathbf{u}) L_i^{\lfloor\lambda\rfloor} d\mathbf{u} \\ + (\lambda - \lfloor\lambda\rfloor) \sum_{j=1}^{\phi} \int_{\mathbb{R}^2} G_{\mathcal{P}}(\mathbf{u}) L_j^{\lceil\lambda\rceil} d\mathbf{u}. \end{aligned} \tag{9}$$

Our method can maintain the continuity between different patches. One footprint may cover two or more patches during $\mathcal{P}$-NDF evaluation. In order to obtain a continuous position-normal distribution and solve the problem that the position-normal distribution will be discontinuous around the patch boundaries, we treat it separately. We subdivide a footprint into four sub-footprints when the footprint crosses more than one patch. We will calculate $\mathcal{P}$-NDF value for each sub-footprint and add them up. However, some sub-footprints may still cross several patches. For those sub-footprints, we will search the non-clustered lobes.

According to Equation 9, we do not need to apply a top-down traversal, and the complexity of the query is irrelevant to the distance from the shading point to the camera. In Fig. 5, we validate the correctness of our $\mathcal{P}$-NDF evaluation by comparing it with Yan et al. [YHMR16], which is treated as the reference. The ground truth is computed by the binning method. Results show a good match between our $\mathcal{P}$-NDF evaluation result and the reference. When the footprint size is expanded from 10×10 texels to 30×30 texels, our method always keeps $\mathcal{P}$-NDF information in the low-frequency domain with only a slight difference in the high-frequency domain.

6. Results

We implement our algorithm inside Optix Renderer and compare our algorithm against: (i) Yan et al. [YHMR16], which we consider as the reference for quality validation, (ii) Wang et al. [WHHY20], and (iii) Zhu et al. [ZXW19]. We implement algorithms of Yan et al. [YHMR16] on GPU by using quadtree of position and normal.

All results in this section are measured on a PC with 3.6GHz Intel (R) i9-9900K CPU, 32GB of main memory, and NVIDIA TITAN RTX GPU. The relevant information of the scene and its materials are listed in Table 1. Unless specified, the rendered images are FHD resolution (1920 × 1080) with two ray tracing samples

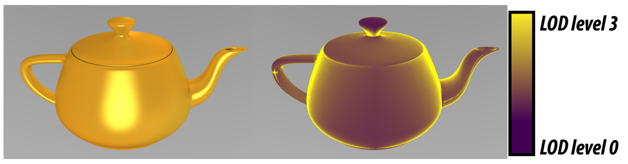


Figure 6: ***(Left)*** *the rendering result of* ***Teapot Scene*** *with isotropic noise metal material under direct illumination.* ***(Right)*** *corresponding level-of-detail (LOD) displayed with color coding. Higher MIP-mapped levels are used for those pixels with larger footprints.*

Table 1: *Performance for the scenes used in this paper. Polys. denotes the number of polygons of the scene; Input Res. denotes the resolution of the input normal map. Most scenes are with global illumination, except the* ***Car Scene*** *with direct illumination.*

			Input Res.		*$\mathcal{P}$-NDF Ev. Time (ms)*			*Rendering Time (ms)*	
Scene	*Polys.*	*Material*	*Ours*	*[YHMR16]*	*Ours*	*[YHMR16]*	*Speedup*	*Ours*	*[YHMR16]*
Leather Shoes & Wooden Floor	272.9K	leather	256^2	$4K^2$	7.9	192.8	24.4×	26.3	211.2
		isotropic	256^2	$4K^2$					
Shoes	272.7K	anisotropic	256^2	$4K^2$	3.6	71.5	19.9×	15.2	83.4
		isotropic	256^2	$4K^2$	3.4	72.2	21.2×	15.6	84.5
Teapot	254.1K	isotropic	256^2	$4K^2$	4.8	60.5	12.6×	15.6	71.4
		brushed metal	256^2	$4K^2$	4.8	60.4	12.6×	15.6	71.2
		structure	256^2	$2K^2$	3.4	46.8	13.8×	16.1	59.6
Door Handle	80.3K	scratch	512^2	$7K^2$	6.1	114.3	18.8×	18.6	126.4
		structure	256^2	$4K^2$	4.9	71.8	14.7×	20.1	76.9
		mixture	512^2	$7K^2$	5.8	107.9	18.6×	24.2	126.3
Material Ball	457.2K	scratch	512^2	$7K^2$	4.8	26.6	5.5×	26.3	47.6
		anisotropic	256^2	$4K^2$	3.5	23.6	6.7×	18.6	38.5
Bunny	144.1K	isotropic	256^2	$4K^2$	3.7	77.8	21.0×	16.7	90.9
Car	1023.1K	isotropic	256^2	$4K^2$	9.4	94.8	10.1×	32.2	116.3
		scratch	512^2	$7K^2$					
		structure	256^2	$4K^2$					

per pixel. The global illumination only contains one extra bounce, which means that the max depth of light tracing is two in default.

6.1. Quality analysis

We first compare our method with the reference method by Yan et al. [YHMR16] in Fig. 1. Qualitatively, our approach produces results that are very similar to the reference. In terms of computation time, our method costs 26.3ms for global illumination, with less than 19.2ms for direct illumination and only 7.9ms for $\mathcal{P}$-NDF evaluation at FHD resolution.

In Fig. 7, we compare our method with Yan et al. [YHMR16], Zhu et al. [ZXW19], and Wang et al. [WHHY20]. The method of Wang et al. [WHHY20] is based on texture blending. For normal maps with weak structural features, their method can get a good result (Fig. 7 (c)), but it is difficult for them to maintain the high structural features, such as scratches (Fig. 7 (g)). Zhu et al. [ZXW19] and our method can get good results for materials with both structured and random discrete features, and our method achieves better coherence (Fig. 7 (d) (h)).

Our method can handle different glossy effects based on different normal maps. In Fig. 8, anisotropic highlights (Fig. 8 (b)), glints (Fig. 8 (c)) and brushed marks (Fig. 8 (d)) are rendered by using different microstructures on the sample normal maps. Furthermore, our method can render mixed materials and display the anisotropic highlights precisely caused by structured and discrete microfacet distribution (Fig. 9).

Our method can process different microfacets well, and the rendering results are visually identical to the reference [YHMR16]. Fig. 10 presents the rendering results using different microfacet distributions in a complex scene. Two kinds of glint materials are integrated with the car body and the front mask. The front mask is a scratched material, and the car wheel hub is a structured one.

6.2. Performance analysis

We report the computation times of the reference method by Yan et al. [YHMR16] and our method in Table 1 and compare the timings for all our test scenes. We also provide the temporal cost associated with $\mathcal{P}$-NDF computation. In all the test scenes, the entire rendering time of our method is less than 33 ms, in which only less than 10 ms is spent on $\mathcal{P}$-NDF evaluation.

Fig.15 displays the temporal costs for the main stages in our Optix ray tracing-based implementations, including Ray tracing, Diffuse Shading, $\mathcal{P}$-NDF Evaluation, Indirect Illumination, and Post-processing. In all of our test scenes, $\mathcal{P}$-NDF evaluation costs less than 32% of the total cost. Ray tracing and indirect illumination are the most expensive components, while diffuse shading and post-processing are relatively cheap.

In Fig. 11, we show the impact of image resolution on the rendering time and $\mathcal{P}$-NDF evaluation time over the *Shoes Scene*. The cost of rendering increases linearly with the number of pixels in the screen space, and $\mathcal{P}$-NDF evaluation time also increases. Our method can meet the real-time requirement in large screen space.

6.3. Storage analysis

The memory cost of our method is mainly on LUT and 4D lobes of the input normal map, which are listed in Table 2. The storage

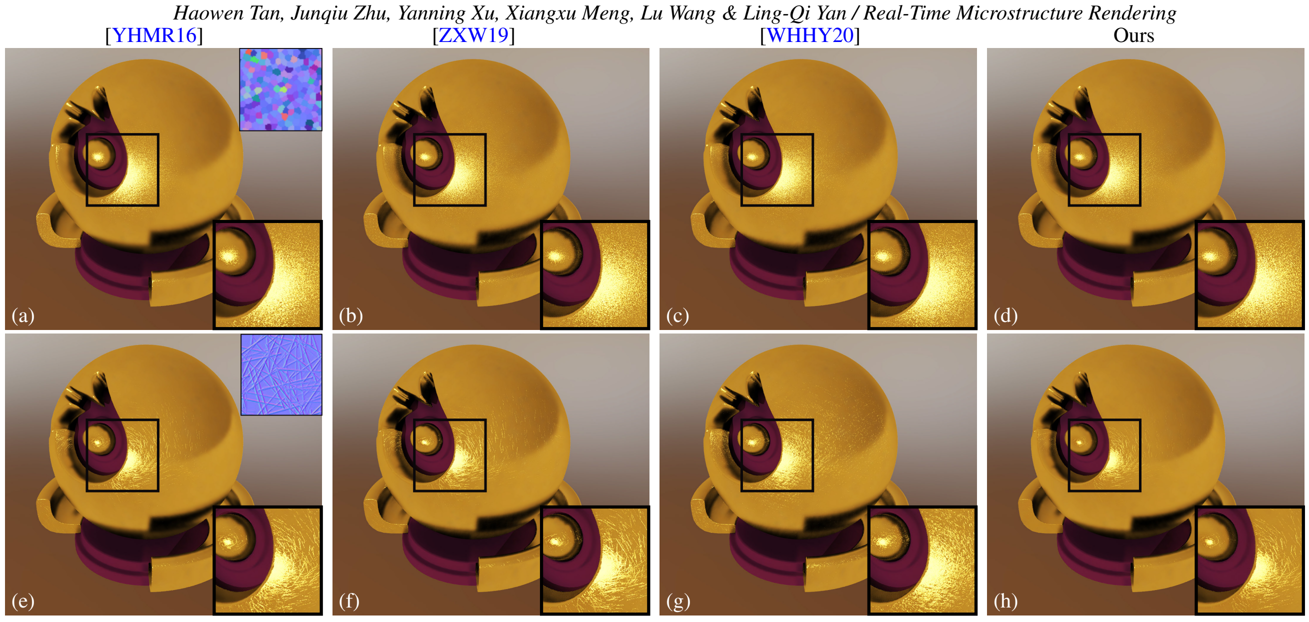


Figure 7: *Quality Comparison with Yan et al. [YHMR16], Zhu et al. [ZXW19] and Wang et al. [WHHY20] on the* ***Material Ball Scene*** *highlighted by directional light and environment light.* ***Top row****: rendering results of isotropic noise metal material.* ***Bottom row****: scratched metal material.*

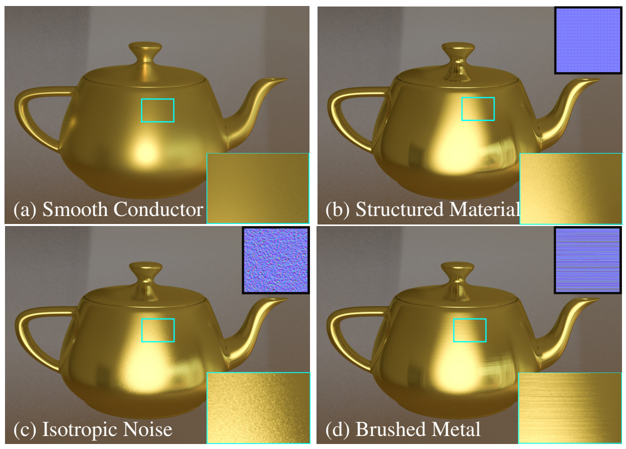


Figure 8: *Rendering results of our method by using different normal maps for the* ***Teapot Scene****. We get anisotropic highlights with structural normal map (b), glinty effects with randomized discrete normal map (c), brushed metal effects (d). The continous effect of smooth conductor is shown in (a).*

Table 2: *The storage of different materials in our test scenes. We use normal map samples in size of 256^2 to 512^2 to generate high-resolution normal maps. The patch blocks are in a resolution of 64^2 or 128^2. For materials with strong discrete features such as scratches, we set patch blocks resolution to 128^2 and overlap size of 4. For other materials in the scene, patch size is set to 64^2 and overlap size of 2. For each high-frequency material, the preprocessing process takes about 10-35 minutes in our implementation, depending on the size of the synthesized normal maps and the size of the patch.*

	Res.		MIP-map	LUT
Material	*Sample*	*Generation*	(MB)	(MB)
Leather	256^2	$4K^2$	14.3	13.1
Isotropic	256^2	$4K^2$	14.3	13.1
Anisotropic	256^2	$4K^2$	14.3	13.1
Scratch	512^2	$7K^2$	37.3	38.5
Structure	256^2	$2K^2$	14.3	4.3
Brushed Metal	256^2	$4K^2$	14.3	13.1
Mixture	512^2	$7K^2$	37.3	38.5

is constant for different scenes if we use the exact resolution of input samples. Our method need storage space ranging from 27.4MB (256×256) to 75.8MB (512×512) for different input samples in this paper.

6.4. Parameter analysis

Fig. 6 visualizes the level of MIP-map used for each pixel (footprint) with gradient colors. Fig. 14 illustrates that results have better coherence if we interpolate between two MIP-mapped levels.

In Fig. 12, we analyze our approach's impact of intrinsic roughness σ_r on material's appearance. For all roughness values, our method provides similar results to the reference. We find that computation time remains almost the same for all parameters' values because lobes can be queried on GPU in parallel.

We also analyze the influence of clustering parameter k of our lobe clustering method in Fig. 13. For glints, our result with $k = 32$ (Fig. 13 (c) (g)) closely matches the reference [YHMR16] (Fig. 13 (a) (e)), with a speedup of 5.4× compared to Yan et al. [YHMR16].

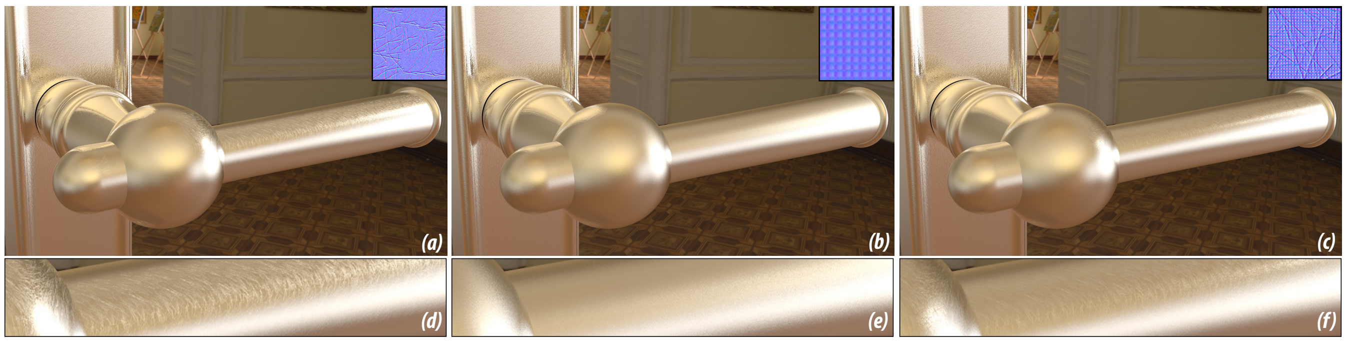


Figure 9: ***Door Handle Scene**: (a) the scratched appearance caused by discrete material. (b) the anisotropic highlights caused by structured material. (c) a mixture of structured and discrete material. The closer details of the microsurface of (a) (b) (c) can be seen from (d) (e) (f).*

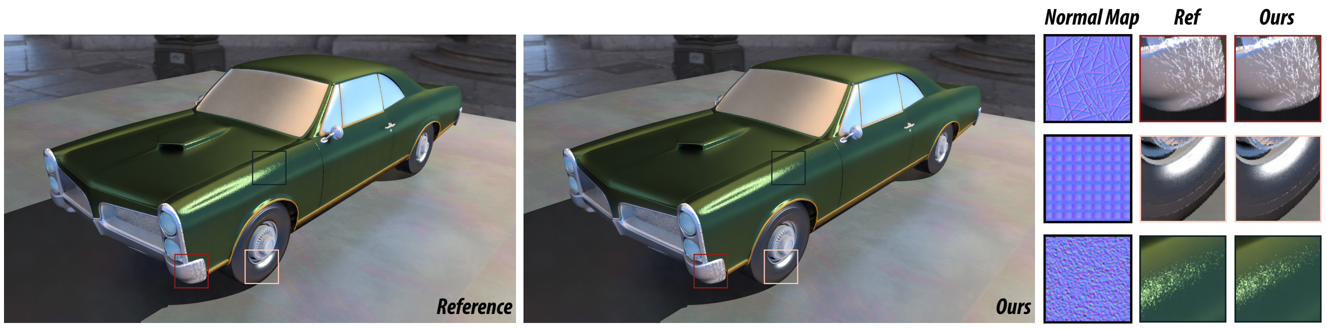


Figure 10: *Comparison with reference [YHMR16] on the **Car Scene** highlighted by point light and environment light. The scene contains three high-frequency materials: isotropic metal material, scratched material, and structural material. For reference, we use synthesized normal maps as input to compare results with ours under the same condition.*

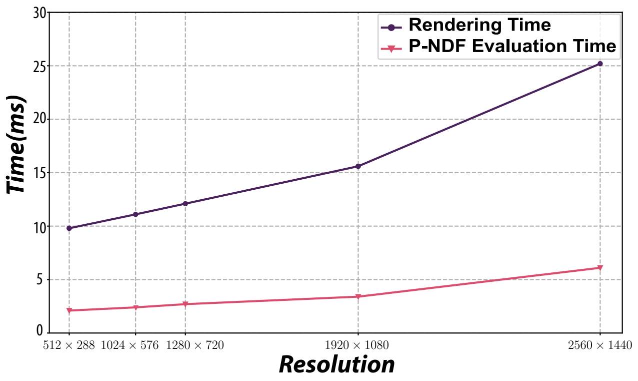
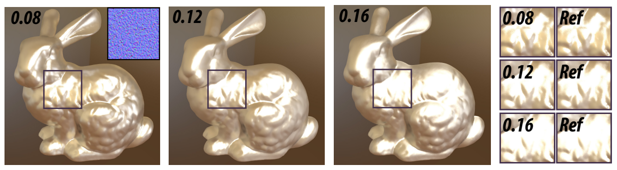


Figure 11: *Rendering time and $\mathcal{P}$-NDF evaluation time as a function of image resolution (number of pixels) on the **Shoes Scene**.*

For scratches, a larger $k = 64$ is preferred to keep the scratching details (Fig. 13 (j)), which is 1.8× faster than Yan et al. [YHMR16] (Fig. 13 (i)).



(a) 16.7ms (b) 16.5ms (c) 16.5ms

Figure 12: *Influence of intrinsic roughness σ_r parameter on **Bunny Scene** lighten by a strong point light and environment light. **(Left)** the general appearance of the scene. **(Right)** the detailed microsurface can be seen by zooming in.*

6.5. Limitations

While our method can fit most high-frequency materials and maintains reflectance characteristics while keeping a constant storage space, we identify scenarios in which our algorithm can be improved.

Failure due to improper patch size. When implicitly generating a large-scale normal map, the size of patch P_i is usually set to 64 or 128 in practice. If the patch size is set too small, the patch P_i may not cover enough features to produce plausible results (Fig. 16 (a)). On the other hand, when the patch size is too large, it means that

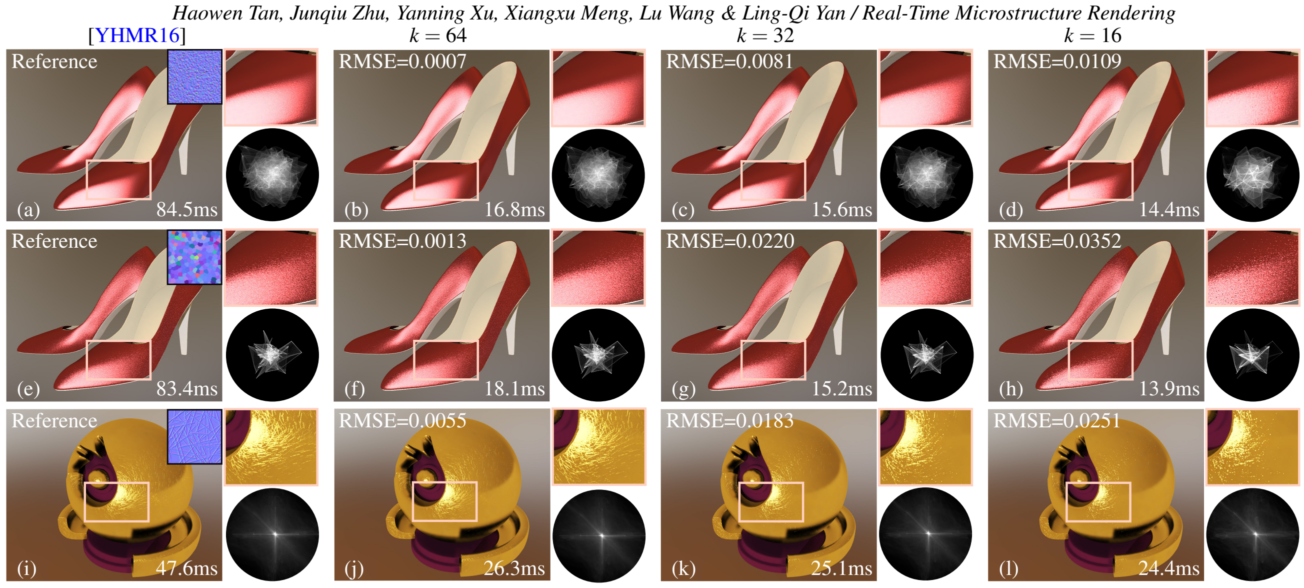
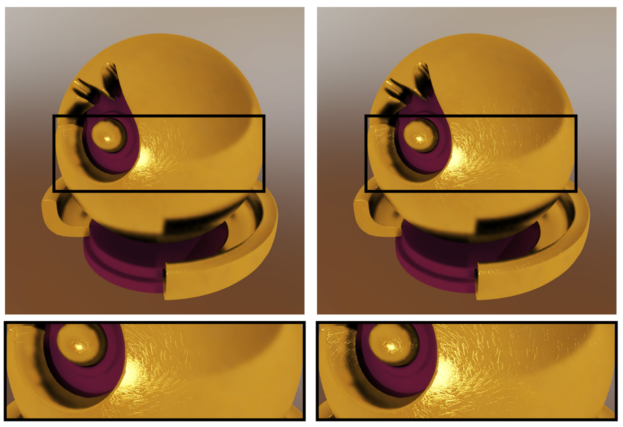


Figure 13: *Comparison of clustering coefficient k on* ***Shoes Scene*** *and* ***Material Ball Scene*** *by our method. We compare three different materials, and each row corresponds to one. The first column in each row is the reference, and the following three columns are from different clustering coefficients. We compare four aspects with the reference in terms of RMSE, time, local zoom results, and NDF visualization. For reference, we also use synthesized normal maps as input to compare results with ours under the same condition.*



(a) *Without MIP-mapped interpolation* **(b)** *With MIP-mapped interpolation*

Figure 14: *Quality comparison of the rendering results without MIP-mapped interpolation* ***(a)*** *and with interpolation* ***(b)****.*

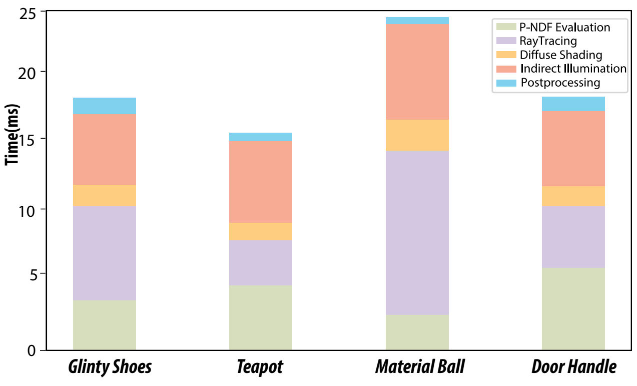


Figure 15: *Computation time for each component ($\mathcal{P}$-NDF evaluation, Ray tracing, Diffuse shading, Indirect illumination and Post-processing) in our algorithm for four scenes.*

the diversity of the patches is reduced, and thus, repetitive patterns appear in the results (Fig. 16 (b)). Our approach may fail when a suitable neighboring patch can not be found. In this case, although the overlap area is handled separately, the method still produces a seam between the two patches (Fig. 16 (c)). The essential reason is that the general texture synthesis methods assume that materials usually have self-similar features. It is not easy to notice the apparent boundary seams in our practice.

MIP-mapped level misalignment. The material's microstructure may be lost when located to a misaligned level. In Fig. 17, we analyze the effects of different $C_{\mathcal{P}}$ of Equation 8 on the rendering results and the overhead performance. Higher $C_{\mathcal{P}}$ value (Fig. 17 (c)) tends to locate to a higher MIP-mapped level, lower time overhead, but brings more over-blurred rendering results. In our practice we set $C_{\mathcal{P}}$ to 0.5.

Indirect illumination of high-frequency materials. We assume that sparkling effects mainly come from direct illumination and integrate single scattering of high-frequency materials only in our raytracing implementation. In the case of glints objects in a mirror, our work may fail (Fig. 18 (a)). We can settle the problem by evaluating $\mathcal{P}$-NDF for each bounce of ray without filtering in global illumination (Fig. 18 (b)), but it undoubtedly results in a significant overhead performance.

Anisotropic texture fetching near object edges. We assume

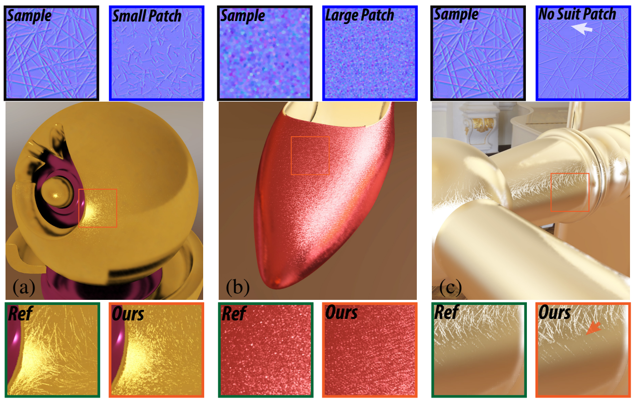


Figure 16: *Limitations of our approach. (a) shows loss of scratch structure due to small-sized patches. (b) shows visible repeating patterns caused by oversized patches, which is due to the fact that generated normal map with fewer patches involved would present a strong pattern-like organization. (c) shows obvious seams of our method when no suitable neighboring patch is found. We take the zoomed-in results when our method is correctly rendering as the reference (green inset) and compare it to when our method goes wrong (orange inset). We also show the corresponding generated normal map (blue inset) and the normal map sample (black inset).*

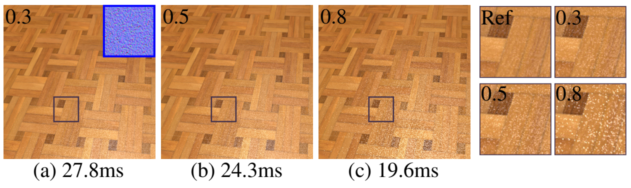
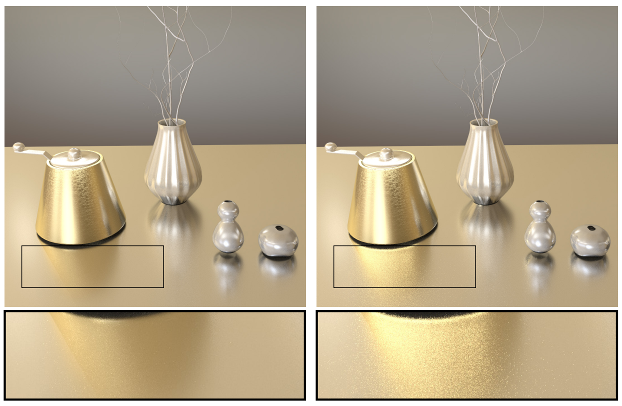


Figure 17: *Influence of different scale factor $C_{\mathcal{P}}$ on results. The results may have over blur with improper $C_{\mathcal{P}}$* ***(c)****.*



(a) *Direct illumination of high-frequency material* **(b)** *Global illumination of high-frequency material*

Figure 18: *Reflections of high-frequency materials. In our direct illumination* ***(a)****, we can not notice the glints and scratches as in global illumination* ***(b)****. The slight difference can be noticed from the* **Bottom row** *by zooming in. We set the max depth of light tracing to four to show the reflection effect more clearly.*

that the ray footprint $\mathcal{P}$ in Equation 8 is an isotropic variable. However, it is anisotropic and thus leads to over-blurring at the edges of objects of anisotropic materials. All MIP-mapped-based methods also share this limitation.

7. Conclusion and Discussion

This paper proposes a practical real-time rendering method for high-frequency materials. We reduce the storage by two orders of magnitude and increase the speed by one order of magnitude while maintaining the same quality as the reference. We use MIP-map as our acceleration structure adapted to the GPU requirement. Additionally, multi-level models generated using our technique with varying cluster scales can be combined to form a MIP-map, allowing level-of-detail rendering of detailed surfaces in an efficient and consistent manner.

Future works will focus on paralleling the preprocessing stage on GPU and modeling the specular light transport within the microstructure.

Acknowledgement. This work has been partially supported by the National Key R&D Program of China under grant No. 2020YFB1708900, the National Natural Science Foundation of China under grant No.61872223, the Shandong Provincial Natural Science Foundation of China under grant No.ZR2020LZH016.

References

[BM93] Becker B. G., Max N. L.: Smooth transitions between bump rendering algorithms. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA, 1993), pp. 183–190. doi:10.1145/166117.166141. 2

[BS87] Beckmann P., Spizzichino A.: The scattering of electromagnetic waves from rough surfaces. *Norwood* (1987). 2

[CCM19] Chermain X., Claux F., Mérillou S.: Glint rendering based on a multiple-scattering patch brdf. *Computer Graphics Forum 38*, 4 (2019), 27–37. doi:10.1111/cgf.13767. 2

[DSHL10] Dammertz H., Sewtz D., Hanika J., Lensch H. P.: Edge-avoiding À-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, Switzerland, 2010), pp. 67–75. doi:10.5555/1921479.1921491. 4

[Fou92] Fournier A.: Normal distribution functions and multiple surfaces. In *Graphics Interface '92 Workshop on Local Illumination* (Vancouver, BC, Canada, 1992), pp. 45–52. 2

[HSRG07] Han C., Sun B., Ramamoorthi R., Grinspun E.: Frequency domain normal map filtering. In *ACM SIGGRAPH* (New York, NY, USA, 2007), pp. 28–es. doi:10.1145/1275808.1276412. 2

[Ige99] Igehy H.: Tracing ray differentials. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 1999), pp. 179–186. doi:10.1145/311535.311555. 5

[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), 1–10. doi:10.1145/2601097.2601186. 2

[KVHS00] Kautz J., Vázquez P.-P., Heidrich W., Seidel H.-P.: A unified approach to prefiltered environment maps. In *Eurographics*

Workshop on Rendering Techniques (Vienna, Austria, 2000), pp. 185–196. doi:10.1007/978-3-7091-6303-0_17. 2

[Tok05] TOKSVIG M.: Mipmapping normal maps. *Journal of Graphics Tools 10*, 3 (2005), 65–71. doi:10.1080/2151237X.2005.10129203. 2

[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. doi:10.1111/cgf.13347. 2, 3

[WAT92] WESTIN S. H., ARVO J. R., TORRANCE K. E.: Predicting reflectance functions from complex surfaces. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1992), pp. 255–264. doi:10.1145/133994.134075. 2

[WDH20] WANG B., DENG H., HOLZSCHUCH N.: Real-time glints rendering with pre-filtered discrete stochastic microfacets. *Computer Graphics Forum 39*, 6 (2020), 144–154. doi:10.1111/cgf.14007. 2, 3

[WDR11] WU H., DORSEY J., RUSHMEIER H.: Physically-based interactive bi-scale material design. *ACM Transactions on Graphics (TOG) 30*, 6 (2011), 1–10. doi:10.1145/2024156.2024179. 2

[WHHY20] WANG B., HAŠAN M., HOLZSCHUCH N., YAN L.-Q.: Example-based microstructure rendering with constant storage. *ACM Transactions on Graphics (TOG) 39*, 5 (2020), 1–12. doi:10.1145/3406836. 2, 3, 4, 5, 6, 7, 8

[WMLT07] WALTER B., MARSCHNER S. R., LI H., TORRANCE K. E.: Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (2007), pp. 195–206. doi:10.2312/EGWR/EGSR07/195-206. 2

[WWH18] WANG B., WANG L., HOLZSCHUCH N.: Fast global illumination with discrete stochastic microfacets using a filterable model. *Computer Graphics Forum 37*, 7 (2018), 55–64. doi:10.1111/cgf.13547. 2

[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 (TOG) 33*, 4 (2014), 1–9. doi:10.1145/2601097.2601155. 2

[YHMR16] YAN L.-Q., HAŠAN M., MARSCHNER S., RAMAMOORTHI R.: Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–9. doi:10.1145/2897824.2925915. 1, 2, 3, 4, 5, 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), 1–10. doi:10.1145/3197517.3201351. 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* (New York, NY, USA, 2016), pp. 139–148. doi:10.1145/2856400.2856409. 2, 3

[ZXW19] ZHU J., XU Y., WANG L.: A stationary svbrdf material modeling method based on discrete microsurface. *Computer Graphics Forum 38*, 7 (2019), 745–754. doi:10.1111/cgf.13876. 2, 4, 5, 6, 7, 8