A Tiny Example-Based Procedural Model for Real-Time Glinty Appearance Rendering

Youxin Xing[1], *Haowen Tan*[2], *Yanning Xu*[1], *Lu Wang*[1]

[1]Shandong University, [2]NetEase (Hangzhou) Network Co., Ltd

Abstract The glinty details from complex microstructures significantly enhance rendering realism. However, the previous methods use high-resolution normal maps to define each micro-geometry, which requires huge memory overhead. This paper observes that many self-similarity materials have independent structural characteristics, which we define as tiny example microstructures. We propose a procedural model to represent microstructures implicitly by performing spatial transformations and spatial distribution on tiny examples. Furthermore, we precompute normal distribution functions (NDFs) by 4D Gaussians for tiny examples and store them in multi-scale NDF maps. Combined with a tiny example-based NDF evaluation method, complex glinty surfaces can be rendered simply by texture sampling. The experiment shows that our tiny example-based microstructure rendering method is GPU-friendly, successfully reproducing high-frequency reflection features of different microstructures in real-time with low memory and computational overhead.

Keywords real-time, reflectance modeling, glinty materials, rendering

1 Introduction

The microstructure of the material surface exhibits sparkling effects in the real world when illuminated by sharp light sources. Traditionally, the microfacet model relies on aggregate statistical distribution to describe these complex and spatial varying micro-geometries, resulting in smooth highlight and loss of high-frequency reflection effect. The microstructure's reflection characteristics are taken into account by highly realistic rendering methods to enhance the visual realism of computer-generated imagery (CGI). However, real-time rendering still lacks a general and efficient microstructure rendering method.

The representation of microstructures is a critical factor in microstructure rendering. First, a complete and thorough representation of different micro-geometries is needed to keep detailed features. Second, it is supposed to be lightweight and introduce no excessive storage overhead. Yan *et al.* [1] define all geometric details of microstructures by the high-resolution normal mapping. However, at the same time, it introduces high memory overhead and an extended performance burden, making it difficult to be applied directly in real-time scenarios that require immediate feedback.

Several recent approaches, such as Zhu *et al.* [2] and Wang *et al.* [3] use example-based approaches to implicitly represent microstructures, which can dramatically decrease the memory cost. However, these methods are still time-consuming since they need complex hierarchies to evaluate the normal distribution. Tan *et al.* [4] optimized it for real-time implementation on GPUs by prefiltering microstructures based on MIP-map. All those methods need to use a proper example, which is still difficult to present varied structures of materials.

We have a key observation that microstructures are generally self-similar and can be abstracted into a tiny example (represented by a tiny normal map). Therefore, this paper assumes that a material's macroscopic surface is composed of many tiny example microstructures, and the global micro-geometry is determined by many tiny examples together after spatial transformation and spatial distribution operations. Based on this assumption, this paper proposes a tiny example-based real-time microstructure representation and rendering method with the following contributions:

- a tiny example-based discretization representation that combines multi-scale NDF maps for materials with self-similarity or independent patch features, which has high expressiveness.
- a tiny example-based spatial transformation and spatial distribution method to enhance the structural diversity of macroscopic surfaces while maintaining only a small amount of data, which significantly reduces memory overhead.
- a tiny example-based NDF evaluation formulation enables real-time rendering of complex microstructures with high performance and low memory cost.

2 Related work

As an essential part of photorealistic rendering, glinty microstructure representation and evaluation have received significant attention from researchers. In this section, we review previous work on glinty microstructure representation for offline rendering and real-time rendering.

2.1 Offline microstructure rendering

There are two prominent families of approaches for microstructure representation. One is the explicit representation based on high-resolution normal mapping, and the other is the implicit representation of microstructure by analytical formulas or stochastic distribution models.

Explicit representation. The explicit representation of microstructure mainly relies on normal mapping or heightfield, which is a methodology inherited from Yan *et al.*'s [5] mathematical framework. Yan *et al.* [1] employed the 4D Gaussian mixture to model the typical distribution of the microstructure, achieving better performance than prior research. They also explored the glinty appearance under wave optics [6]. However, these approaches have high storage requirements, and the 4D position-normal query is costly. Gamboa *et al.* [7] represented microstructures via discrete 2D texture histograms and applied a filtering technique combining environmental lighting and normal mapping, which demands significant memory. Atanasov *et al.* [8] presented the "inverse bin map"–an advanced integral histogram, to speed up the filtering of the bidirectional reflectance distribution function (BRDF) with Beckmann distribution for microstructures in persistent storage. Based on the normal map, explicit representation acquires spatial and directional features directly, providing more flexibility.

Implicit representation. Jakob *et al.* [9] proposed a stochastic approach to simulate temporally stable sparkling effects. In their method, the proportion of spatially randomly distributed metallic flakes is obtained by an efficient evaluation method. Atanasov *et al.* [10] optimized the sampling process for better overall performance. Wang *et al.* [11] made the rendering process more efficient by deriving the method in a separable and filterable form. Some methods for the microstructure of specific scratches [12, 13] have also been proposed. Deng *et al.* [14] developed a prefiltering method relying on precomputation, aggregating data into a 3D NDF tensor to accelerate spatial-angular range queries during rendering. Nevertheless, it incurs the high cost of NDF generation and compression overhead. To address memory issues, Zhu *et al.* [2] and Wang *et al.* [3] extended the work of Yan *et al.* [1]. They generated stationary microstructures through a by-example approach. Zhu *et al.* [2] generated microstructure through texture syntheses and reduced memory overhead by clustering structural elements. Meanwhile, Wang *et al.* [3] employed texture blending to extend the exam-

ple normal map infinitely and maintain constant memory overhead. Different from the traditional methods above, Kuznetsov *et al.* [15] first introduced a deep learning method to generate glinty patches that enable the synthesis of glinty results without significant spatial repetition.

2.2 Real-time microstructure rendering

For real-time rendering, most methods use implicit representations to generate ultra-high-resolution normal maps, and our method is the same. Zirr *et al.* [16] proposed a method to accelerate the estimation of probability distribution using the MIP hierarchy and achieve real-time performance based on Jakob *et al.* [9]. Deliot *et al.* [17] reduced the number of texels falling under a pixel footprint by combining a counting method with an anisotropic parameterization of the texture space to accelerate the runtime performance. In contrast, Chermain *et al.*'s method [18] is more physically based and can converge to the Cook-Torrance model [19] when the flake density is high enough. Wang *et al.* [20] simulated randomly discrete microfacet under environment lighting and point light in real-time by prefiltering. Furthermore, Velinov *et al.* [21] proposed a scratch appearance method under wave optics based on their previous work [13]. Tan *et al.* [4] introduced a real-time prefiltering approach for microstructures that employed MIP-maps to select microstructures at the suitable level of details (LODs) for storage-stable and efficient rendering.

Implicit representation methods are limited in their ability to convey the fine details of micro-geometry accurately. Our approach preserves rich details based on tiny examples and improves the diversity of microstructures through the spatial transformations of tiny examples.

$f_{\mathcal{P}(\omega_i,\omega_o)}$	Surface BRDF
$\mathcal{P}$	Footprint
$\boldsymbol{\omega_h}$	Half vector of reflection
$\boldsymbol{\omega_i}, \boldsymbol{\omega_o}$	Light and view directions
$\mathbf{n}$	Surface normal
$\mathbf{u}$	2D global texture coordinate
$\mathbf{u}'$	2D local texture coordinate
$\mathbf{s}$	Query direction
$D_{\mathcal{P}}(\mathbf{s})$	Patch normal distribution function
$\mathcal{N}(\mathbf{u},\mathbf{s})$	Position-normal distribution
$G_{\mathcal{P}}(\mathbf{u})$	Gaussian approximating to a footprint
$G_i(\mathbf{u},\mathbf{s})$	4D Gaussian lobes
$n(\mathbf{u})$	Normal map function
$\mathbf{J}$	Jacobian of $n(\mathbf{u})$
Σ_i^{-1}	Inverse of 4×4 covariance matrix
σ_h	Std. deviation of seed Gaussians
σ_r	Intrinsic roughness
l	Level of multi-scale NDF maps
$\mathcal{T}$	Example tile
$N_{\mathcal{T}}$	Number of texels in an example tile $\mathcal{T}$
$\mathbf{t}$	Local texture coordinates in a NDF map
$\mathbf{t}'$	Transformed $\mathbf{t}$
M	Spatial transformation matrix
$\Phi(\mathbf{u})$	Tile position in NDF map space
$B_l(\mathbf{u}')$	NDF map function at level l

Table 1. Notations

3 Background

In this section, we first introduce the surface BRDF and then provide the evaluation of the normal distribution function.

3.1 Surface BRDF

During rendering, the surface BRDF $f_{\mathcal{P}}$ for the footprint $\mathcal{P}$ is defined as:

$$f_{\mathcal{P}}(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{D_{\mathcal{P}}(\boldsymbol{\omega_h}) G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{\omega_h}) F(\boldsymbol{\omega_i}, \boldsymbol{\omega_h})}{4(\boldsymbol{\omega_i} \cdot \mathbf{n})(\boldsymbol{\omega_o} \cdot \mathbf{n})},$$

where $\boldsymbol{\omega_i}$ and $\boldsymbol{\omega_o}$ denote the light and view directions, the term $\boldsymbol{\omega_h}$ refers to the half vector of reflection, $\mathbf{n}$ represents the surface normal, F is the Fresnel term, G is the masking-shadowing function [22]. $D_{\mathcal{P}}(\boldsymbol{\omega_h})$ is the patch normal distribution function ($\mathcal{P}$-NDF) over a spatial footprint $\mathcal{P}$ according to the querying direction, also known as the half vector $\boldsymbol{\omega_h}$. The evaluation of

$D_{\mathcal{P}}(\boldsymbol{\omega_h})$ is a major difficulty in microstructure rendering and is the core of the discussion in this paper.

3.2 Evaluation of the normal distribution function

Our $\mathcal{P}$-NDF evaluation builds upon the method of Yan *et al.* [1], where the evaluation of $D_{\mathcal{P}}$ can be written as follows:

$$D_{\mathcal{P}}(\mathbf{s}) = \int G_{\mathcal{P}}(\mathbf{u})\mathcal{N}(\mathbf{u},\mathbf{s})\mathrm{d}\mathbf{u},$$

where $\mathbf{s}$ is the query direction (also represents the 2D normal with implicit z-coordinate), $G_{\mathcal{P}}$ is Gaussian approximating to a spatial footprint, $\mathbf{u}$ represents the 2D texture coordinate. $\mathcal{N}$ is the position-normal distribution which can be approximated by Gaussian lobes. Yan *et al.* [1] used a large number (k) of 4D Gaussian lobes G_i obtained by traversing each texel in the normal map to approximate $\mathcal{N}$, i.e.,

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

After defining $\delta\mathbf{u}_i = (\mathbf{u}-\mathbf{u}_i)^T$, $\delta\mathbf{s}_i = (\mathbf{s}-\mathbf{s}_i)^T$, each Gaussian lobe G_i is represented as:

$$G_i(\mathbf{u},\mathbf{s}) = c_i \mathrm{e}^{\left(-\frac{1}{2}(\delta\mathbf{u}_i,\delta\mathbf{s}_i)^T\Sigma_i^{-1}(\delta\mathbf{u}_i,\delta\mathbf{s}_i)\right)},$$

where c_i is a constant for normalization. With the Jacobian $\mathbf{J}$ of the 2D normal $n(\mathbf{u})$ (sampled from a normal map, $n(\mathbf{u}) = (n_x, n_y)$), the inverse of 4×4 covariance matrix Σ_i^{-1} is expressed as:

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

where, σ_h is the deviation of seed Gaussians, σ_r is the intrinsic roughness.

Therefore, for the given $\mathcal{P}$ and $\mathbf{s}$, the $\mathcal{P}$-NDF is defined as:

$$D_{\mathcal{P}}(\mathbf{s}) \approx \sum_{i=1}^{k}\int G_{\mathcal{P}}(\mathbf{u})G_i(\mathbf{u},\mathbf{s})\mathrm{d}\mathbf{u}. \quad (1)$$

4 Overview

The representation of microstructure (Section 5) and the corresponding shading process, particularly the example transformation, distribution, and NDF evaluation (Section 6), are crucial for depicting high-frequency details.

For the issue of microstructure representation, we propose to use a tiny normal map to represent the overall characteristics of the microstructure and define it as an example in Section 5.1. We then compute multi-scale NDF maps for the tiny example in precomputation (Section 5.2) so that real-time applications can process complex specular surfaces through simple texture sampling at different LODs.

The core issues related to microstructure rendering are example transformation, distribution, and the NDF evaluation. Based on the geometry characteristics of materials, we classify the tiny examples into stochastically distributed and tiled examples. In order to enrich the diversity of materials, we determine the corresponding spatial transformation (Section 6.1) for stochastically distributed examples and distribution (Section 6.2), thereby generating a large-scale microstructure that describes the overall material. We evaluate NDF based on stochastically distributed and tiled tiny examples and reproduce the glinty appearance in Section 6.3. Furthermore, we also discuss the multi-microstructure-layer case in Section 6.4.

We illustrate the overall pipeline of our method in Fig. 1, which mainly consists of the precomputation and real-time shading stage.

5 Tiny example-based microstructure representation and precomputation

A normal map that stores the tangent space normals of objects is often used to describe the geometric structure characteristics of materials. The explicit method

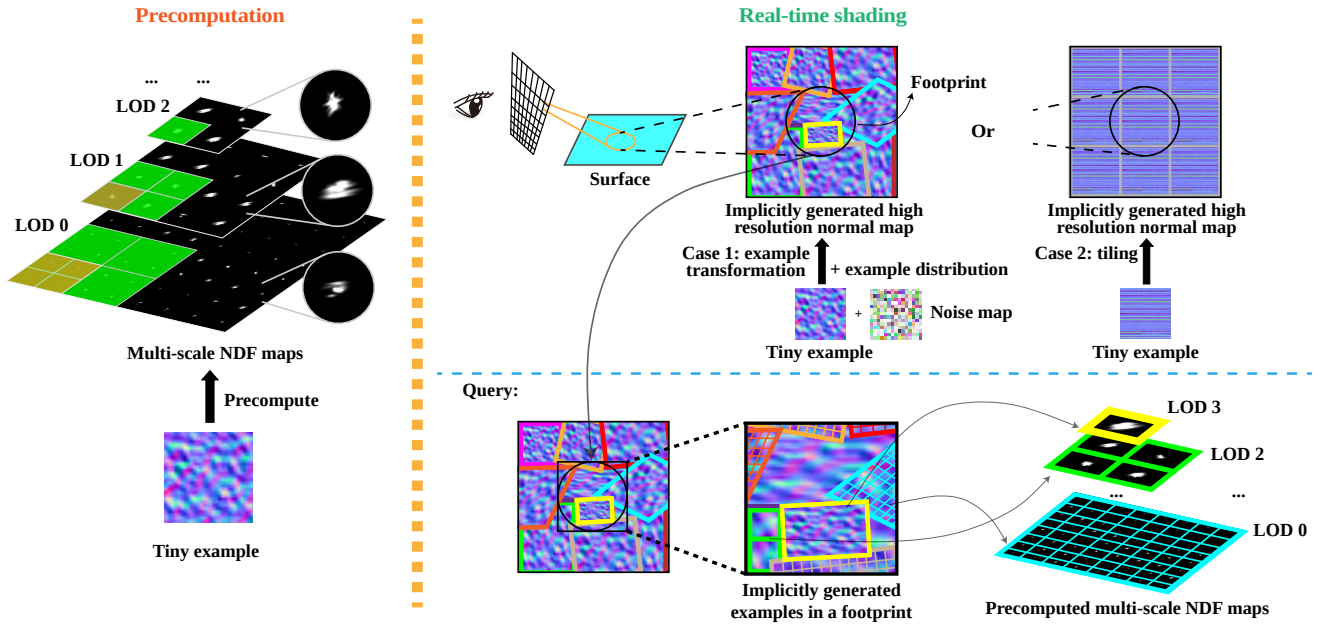


Fig.1. The pipeline of our method includes two major parts: precomputation and real-time shading. During precomputation, we compute NDF at various LOD levels for the input tiny example and save them into multi-scale NDF maps. In the real-time shading stage, we use a procedural model to implicitly generate large-scale microstructures by performing spatial transformations and distribution on the tiny example with a noise map. While shading, we identify examples partly covered by the footprint and divide them into tiles of varying LODs. For the examples entirely covered by the footprint, we select the top LOD. Finally, we employ a tiny example-based NDF evaluation method to enable a fast and accurate approximation of the NDF by summing up the tiles of precomputed NDF maps. The entire shading process is carried out in real-time and is compatible with GPU to ensure the method's efficiency.

utilizes an arbitrary high-resolution normal map to specify the microstructure with heavy storage overhead. In our method, the global microstructure is implicitly generated by tiny examples that are defined by the example normal map.

5.1 Discrete representation

Most of the materials exhibit similar structural characteristics and corresponding light transport. Therefore, we assume that a structural element exhibiting self-similarity and its variants form a macroscopic surface under a spatial distribution function. We define a small scale of microstructures with the same characteristics in the spatial domain as a tiny example.

We use an example normal map (usually smaller than 32×32), called the tiny example, to represent the example microstructure of a specific material. Given a tiny example, we precompute approximate NDFs and store them in multi-scale textures.

5.2 Multi-scale NDF maps precomputation

For a tiny example with $n \times n$ resolution, we compute the highest level of LOD for its corresponding multi-scale NDF maps by $log_2 n - 1$, and the NDF map resolution $m \times m$ at each level l by $n/(2^{l+1}) \times n/(2^{l+1})$. For instance, the input tiny example in Fig. 1 has a resolution of 16×16. Therefore it can build NDF maps with 4 layers of LOD, and their resolutions are 8×8, 4×4, 2×2, and 1×1.

While precomputing the NDF for each level based on the tiny example, we divide the tiny example into $m \times m$ tiles equally. Each tile in the example contains a set of explicitly specified normals which are used to compute its NDF value. In a similar way to Yan *et al.* [1], we assume the NDF of a tile is an equally weighted average of these micro-scale normals, which is

defined as follows:

$$D_{\mathcal{P}}(\mathbf{s}) = \frac{1}{N_{\mathcal{T}}} \sum_{\mathbf{u} \in \mathcal{T}} G_i(\mathbf{u}, \mathbf{s}), \tag{2}$$

where $N_{\mathcal{T}}$ is the number of texels in an example tile $\mathcal{T}$.

Each discrete NDF is computed per tile using Eq. 2 and encoded into NDF maps at each level. Besides, we also save the Gaussian lobes used to describe the microstructure more precisely if needed. Attributes include the position, normal, Jacobian matrix, etc..

6 Real-time shading

In this section, based on the representation of a single discrete tiny example presented above, we provide the corresponding spatial transformations, distributions, and NDF evaluation methods. Moreover, we apply these methods to explore the visual effects of a multi-layer microstructure with a simplified multi-layer microstructure material model.

6.1 Spatial transformation

Distributing the same tiny example-based microstructure on the surface directly by tiling may result in noticeable visual errors, such as repetitive features. We obtain more examples by spatial transformations during real-time shading.

For texels in the multi-scale NDF maps, we transform their local texture coordinates $\mathbf{t}$ ($\mathbf{t} = (t_x, t_y, \sqrt{1 - {t_x}^2 - {t_y}^2})$) in local texture space by a 3×3 matrix M to get transformed local texture coordinates $\mathbf{t}'$, which is defined as:

$$\mathbf{t}' = M\mathbf{t}.$$

The spatial transformation matrix M is defined as follows:

$$M = \begin{pmatrix} s_x & h_x & t_x \\ h_y & s_y & t_y \\ 0 & 0 & 1 \end{pmatrix},$$

where, s_x and s_y are scaling coefficients that control the transformation at the microstructure scale, allowing us to obtain isotropic or independent scale transformations in different directions. h_x and h_y represent the stretching coefficients along the x-axis or y-axis, which allows us to control the macroscopic diffusion direction of the highlights by sheer transformations. Besides, rotation, symmetry, and other spatial transformations are also achieved through M, as shown in Fig. 2.

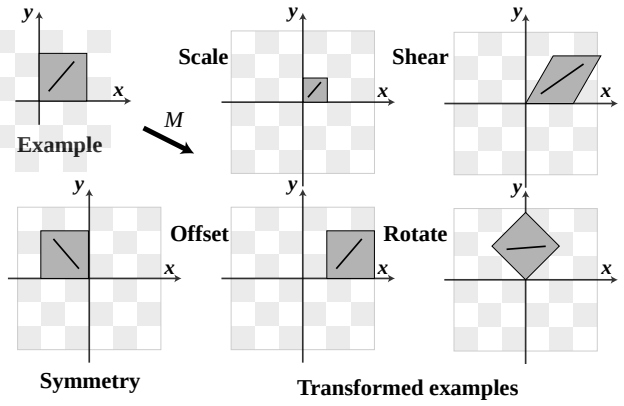


Fig.2. Spatial transformations for tiny examples through different transformation matrices.

6.2 Spatial distribution

Based on microstructure characteristics, we classify tiny examples into stochastically distributed type (*e.g.*, scratches) and tiled type (*e.g.*, brushed metal), and distribute them in different ways.

Stochastically distributed type. For stochastically distributed microstructures, their overall structural characteristics are not prominent. Furthermore, their appearance characteristics can be enriched through examples of spatial transformations during distribution. Therefore, we encode examples' random center positions and coefficients of transform matrices M (Section 6.1) into a noise map first. During the shading stage, for each example, the noise map implicitly depicts the example area in the global texture space by its

global center position and coefficients of M, thereby enriching the overall appearance diversity of macroscopic materials.

Tiled type. For tiled microstructures with self-similarity and inapparent seams after tiling, such as brushed metal, the random transformation of a single tiny example leads to the loss of the original structural characteristics at a macroscopic scale. Thus we use specific transformations (such as isometric scaling, etc.) to maintain macro-consistency and control the macroscopic high-frequency appearance features. Then we tile the examples directly in the global texture space. The tiled examples are closely arranged, and there is no overlap between them.

6.3 NDF evaluation of tiny example-based microstructure

For a shading point in pixel space, it is essential to determine its footprint $\mathcal{P}$ in texture space. We use the same Gaussian representation method as Heckbert [23] to approximate the footprint with a parallelogram in texture space. In this way, we obtain the texture positions $\mathbf{u}$ covered by $\mathcal{P}$ in the global texture space.

Because we have already encoded examples' center positions in the global texture space and corresponding transformation matrix M into a noise map. Therefore, for a texture position $\mathbf{u}$, we can traverse all examples' information in the noise map and determine if $\mathbf{u}$ is inside the examples. Because the number of examples covered by $\mathcal{P}$ is small and the parallelism of the GPU is fully utilized, the processing is very fast.

For the tiled example type, we get only one example at $\mathbf{u}$. But for stochastically distributed example type, we get several examples due to the multiple examples that may overlap with each other. It is necessary to define the occlusion relationship of the overlapping region first. We divide the overlapping region into four sub-regions of equal size. For a single sub-region, we compare the Euclidean distance between the center of the sub-region and the centers of each example, and always use the nearest example to define the occlusion relationship. The details are shown in Fig. 3.

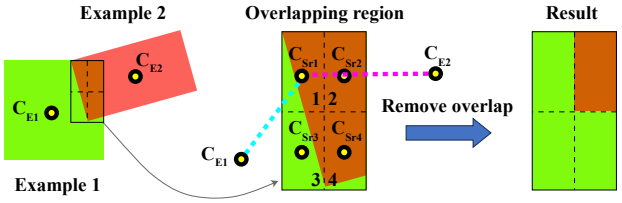


Fig.3. Examples overlapping case. We divide the overlapping region into four sub-regions equally first. C_{E1} and C_{E2} are the center positions of example 1 and 2. $C_{Sr1\text{-}4}$ are the center positions of sub-regions. For the sub-region 1 in the upper left part, the distance $|C_{Sr1} - C_{E1}|$ is smaller than $|C_{Sr1} - C_{E2}|$. Therefore we consider example 1 to override sub-region 1. We use the same strategy to deal with the other sub-regions.

Using this strategy, we determine a unique example that covers the position $\mathbf{u}$. Because we have distributed and transformed the examples onto the global texture space, we can obtain the local query position $\mathbf{u}'$ in NDF maps by back-projecting $\mathbf{u}$ based on the center position and the transformation matrix M of the example. We define this computation processing as $\Phi(\mathbf{u})$.

The $\mathcal{P}$-NDF is accumulated by the NDF contributions of examples covered by the footprint $\mathcal{P}$, which can be queried from the precomputed multi-scale NDF maps. $B_l(\mathbf{u}')$ represents the NDF contribution of each example at LOD level l (under the query direction $\mathbf{s}$). When a tiny example is entirely covered by a footprint, we sample and accumulate its normal distribution term from the highest LOD NDF map. When it is partially covered by $\mathcal{P}$, the LOD of the NDF map reduces gradually from the highest level to determine whether the current tiles in the NDF map are totally covered by $\mathcal{P}$ until the NDF map is subdivided to the lowest LOD level. This spatial subdivision is shown in Fig. 1.

At the same time, when the footprint is smaller than the lowest LOD tile, we directly compute NDF with the precomputed Gaussian lobes instead of sampling from

the precomputed NDF maps, as shown in Eq. 1.

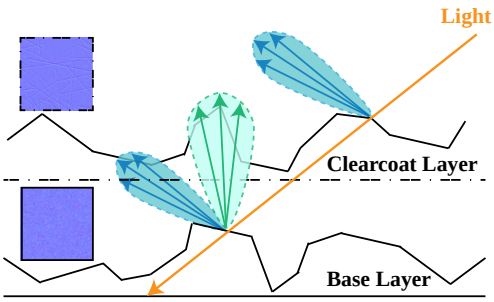


Fig.4. The visualization of the multi-microstructure-layer case. The microstructure of the material surface includes a base layer and a clearcoat layer above it. We simulate complex reflection effects by combining the lobes of two layers.

6.4 Multi-microstructure-layer rendering

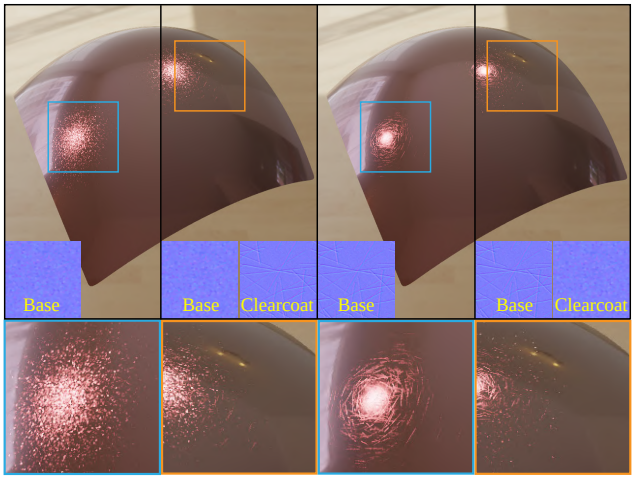
To gain efficiency in real-time rendering, we approximate the resulting light transport by two BRDF lobes at the shading point. This section discusses the multi-microstructure-layer case (as shown in Fig. 4) without explicitly evaluating the complex light transport of interlayer scattering. The first BRDF lobe f_c accounts for the top layer (or a clearcoat layer), and the second lobe f_b accounts for the bottom layer (or a base layer).

The light transport passing through the clearcoat and reaching the base layer is approximately proportional to $1 - F_c$, where F_c represents the Fresnel term of the clearcoat. The resulting BRDF f_s of the multi-layer model is expressed as Eq. 3:

$$f_s = f_b\,(1 - F_c) + f_c. \tag{3}$$

At the same time, different normals are used to evaluate the Fresnel term and masking-shadowing term for the clearcoat and base layer, which is typically done in production and works with our method. NDFs at different layers are also computed independently.

Fig. 5 compares the results of two different microstructures on the Bent quad scene under different layer orders, which lead to significant differences in the visual characteristics of microstructures.



(a) 11.2 ms (b) 14.5 ms (c) 11.8 ms (d) 14.5 ms

Fig.5. Rendering results of the multi-microstructure-layer case. In the **Bent quad scene**, two groups of identical microstructures with different layer orders present different visual effects (b, d), but the rendering time is consistent. At the same time, the multi-layer microstructures (b, d) exhibit more detailed features than the single-layer microstructure (a, c).

The high-frequency effects in Fig. 5 (b, d) are mixed with scratches and anisotropic noise, but they look different. The high-frequency effects in Fig. 5 (b) are more visually shown as bright flakes, and the less energy distribution on the clearcoat leads to discontinuous scratches. In contrast, scratches in Fig. 5 (d) are more obvious and continuous. In addition, computing the NDF of the microstructure for each layer in a single shading point also incurs a certain degree of additional performance burden, but it is still affordable for real-time rendering.

7 Results

We have implemented our method in OpenGL 4.6 and compared it with other typical methods (include Yan *et al.* [1], Wang *et al.* [3], Zhu *et al.* [2], Tan *et al.* [4]) using our unified platform in terms of visual effects, memory consumption, and rendering speed. Besides, We treat the result of Yan *et al.* [1] as the correct value and use their method as the reference. All statistics in this section are performed on a PC with a 3.6-GHz Intel (R) i9-9900K CPU, 32 GB of main memory, and an NVIDIA TITAN RTX GPU. Measurements are

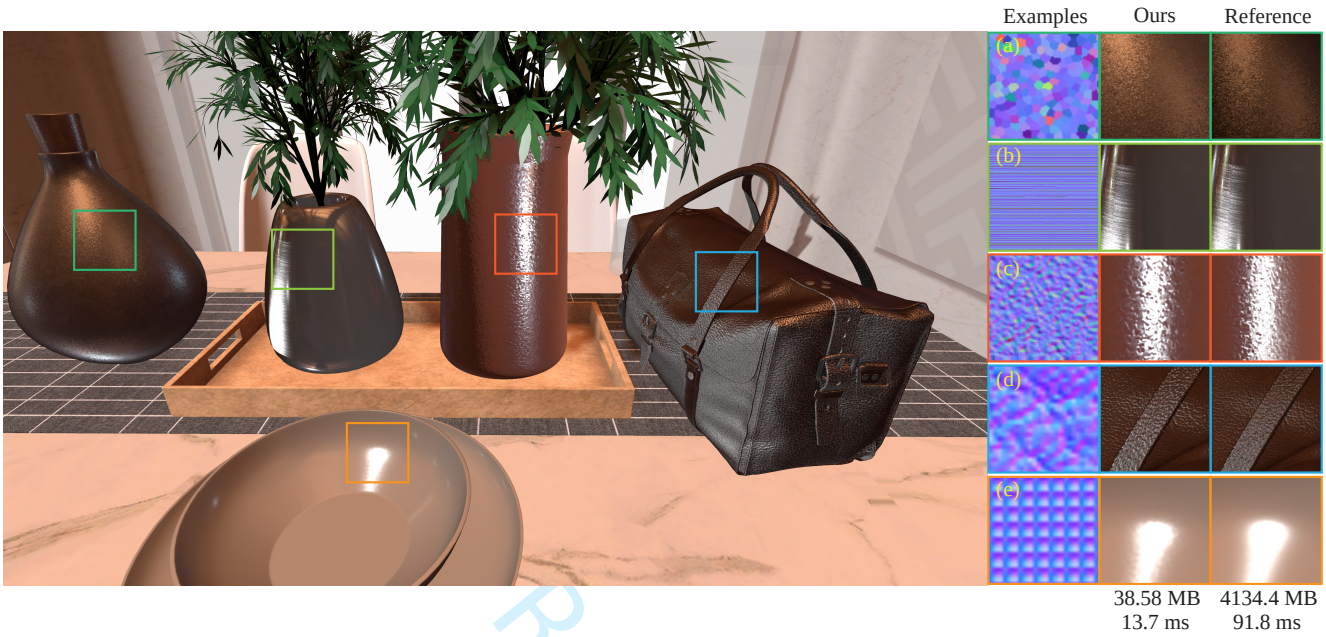


Fig.6. The comparison between our method and the reference method [1] on the **Desktop scene** with various types of microstructures. The high-frequency effects are almost identical. Our method only consumes 1% of memory overhead and significantly improves the rendering speed compared to the reference.

made for FHD (1920 × 1080) image resolution using forward rendering without postprocessing.

We use four sharp point lights and an environment light to illuminate the microstructure and the entire scene. The environment light is encoded through distance light probes, which is an image-based lighting method that simulates the diffuse reflection of ambient light by precomputing the irradiance into a cube map and approximating the specular reflection part by prefiltering the environment light map and combining it with a BRDF lookup table.

In our experiments, we use tiled tiny examples for structured materials, brushed metal, leather, and other microstructures with unique geometric features. Stochastically distributed tiny examples are applied to microstructures of scratches and noise.

7.1 Comparison with previous work

Desktop scene. In this scene of Fig. 6, we compare the result of our method with the reference [1]. The scene contains five common types of microstructures, including (a) anisotropic noise, (b) brushed metal, (c) isotropic noise, (d) leather, and (e) structured materials. The normal map resolution of each microstructure in the reference method is $2K \times 2K$. The two methods are almost equivalent regarding high-frequency detail features, but our method significantly reduces memory and computational overhead. For this complex scene composed of multiple microstructures, our method only takes 13.7 ms, which reduces memory overhead by tens of times compared to the reference method (91.8 ms). This makes our method more suitable for the real-time rendering pipeline.

Coffee machine scene. We compare our method with other microstructure rendering methods in a more complex scene, as shown in Fig. 7. For zoomed-in local details, our method is consistent with other microstructure representation methods and can well represent the characteristics of different types of microstructures.

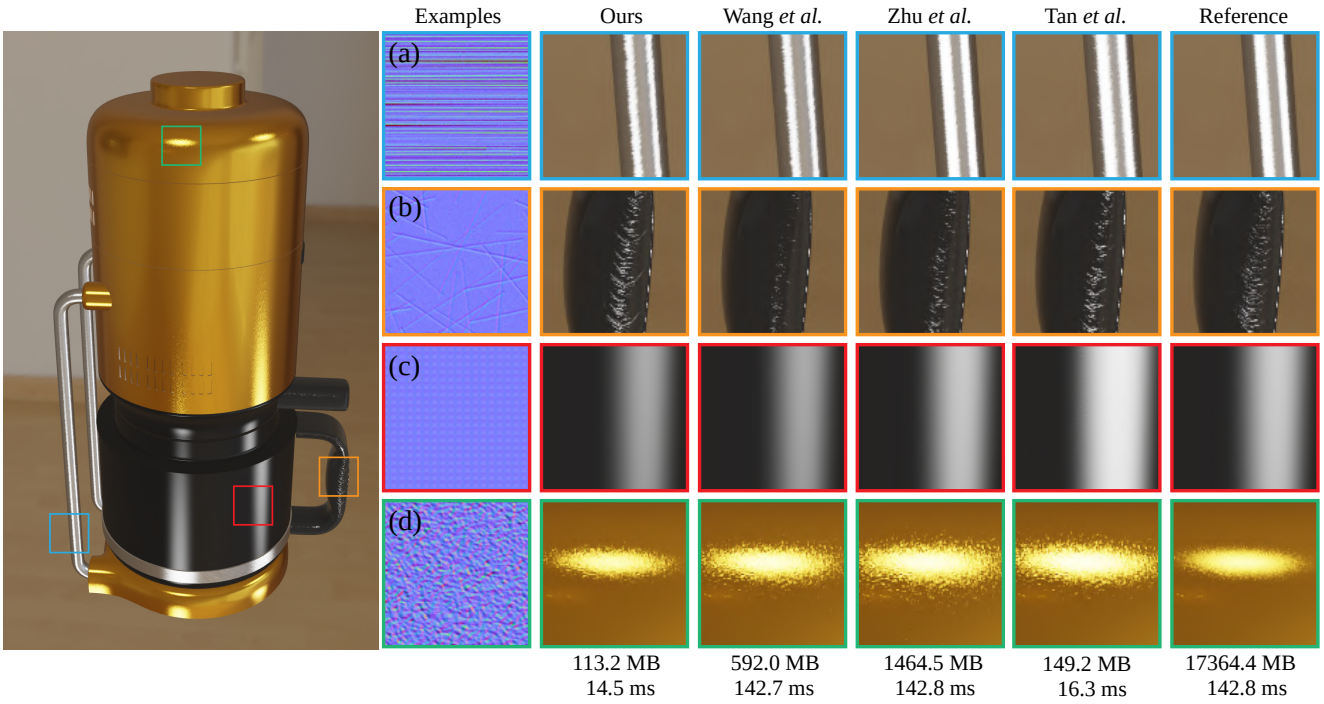


Fig.7. The comparison of our method with other microstructure rendering methods [3, 2, 4, 1] on the **Coffee machine scene** containing four types of microstructure, including (a) brushed metal, (b) scratch, (c) structured materials, and (d) isotropic noise. Compared to these methods, our method is faster and significantly reduces memory overhead while rendering visually identical results.

Whether the tiny examples are tiled in Fig. 7 (a, c, d) or randomly distributed in Fig. 7 (b), our method maintains the continuous characteristics of the structure without visually significant duplication while providing a good approximation of the reference method [1]. Our method directly focuses on structural patterns, avoiding the generation of redundant microstructure unrelated to micro-geometry features, which is common in example-based approaches [3, 2, 4]. Our method significantly reduces the memory cost and enhances performance.

Other scenes. In Fig. 9, there are three simple scenes: Sphere scene (noise), Shoes scene (leather), and Bent quad scene (scratch), corresponding to different microstructure types. For the leather type in Shoes scene, we generate the results using our method of tiled examples, which is comparable to the visual effect of other methods without any structural discontinuity.

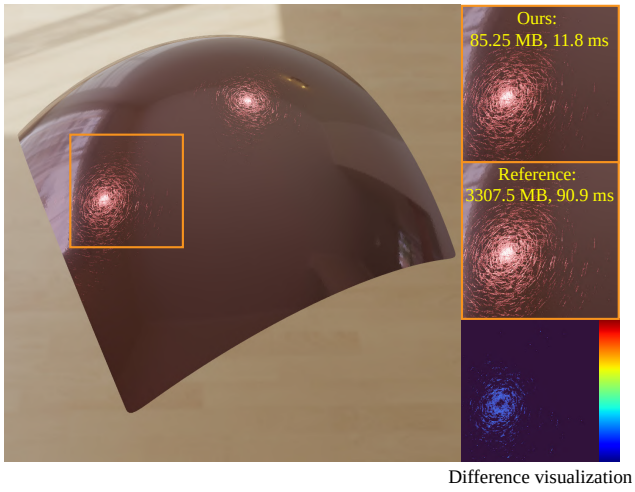


Fig.8. Comparison between our method and reference [1] on the **Bent quad scene** under the same scratch microstructure.

For noise and scratch materials in the Sphere scene and the Bent quad scene, we adopt stochastically distributed examples and increase the diversity of microstructure through spatial transformations. The microstructure presents continuous high-frequency effects, specifically showing continuous scratches in the Bent

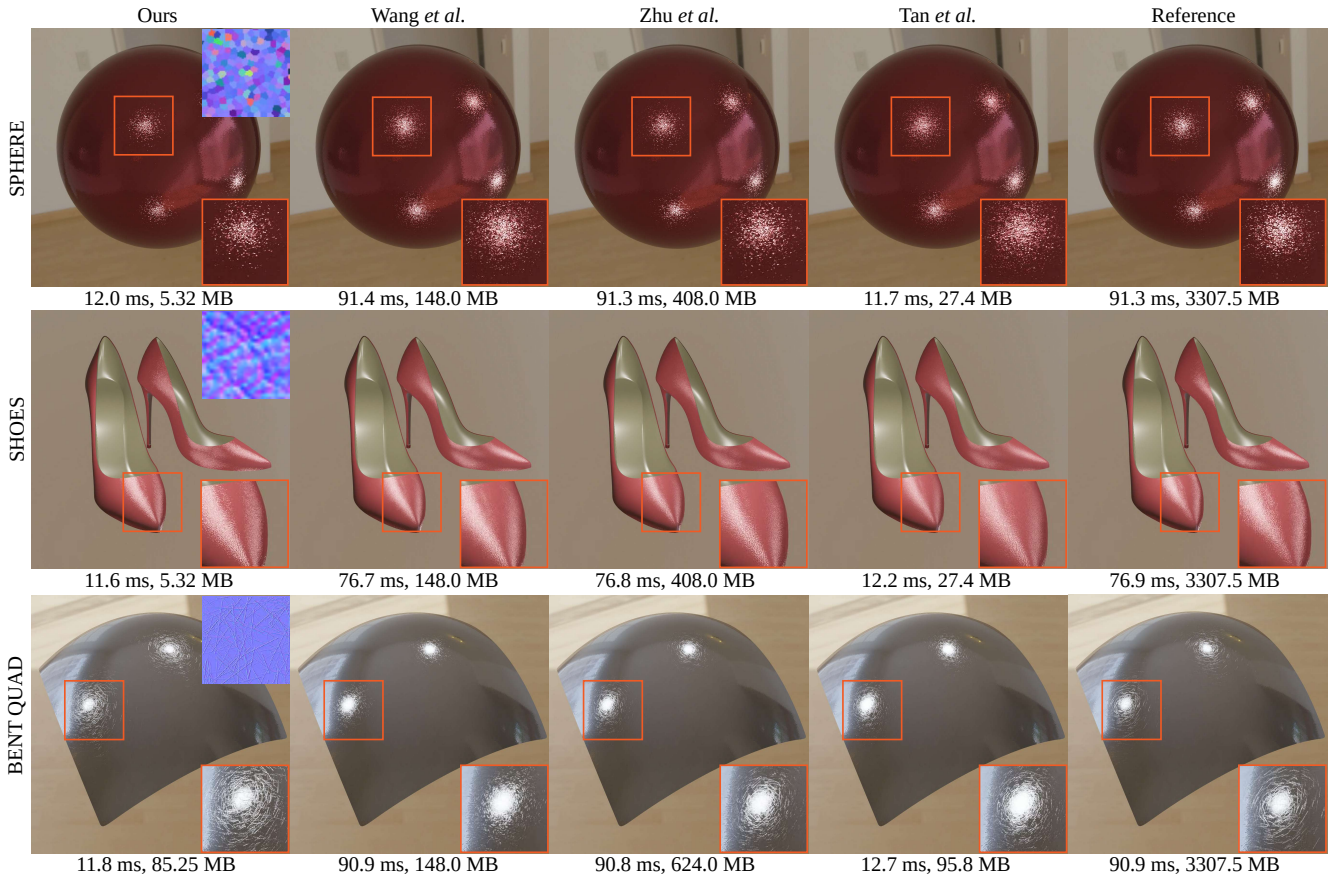


Fig.9. The comparison of rendering results using our method and other microstructure rendering methods [3, 2, 4, 1] on three different scenes. We compared the glint effect generated by anisotropic noise microstructures using different methods in **Sphere scene**, the continuous highlight of leather type microstructures in **Shoes scene**, and the scratch effect generated in **Bent quad scene**.

quad scene and disorderly distribution of noise in the Sphere scene. Compared with the method of Wang *et al.* [3] based on texture blending, our method does not cause the blur of highlight details.

7.2 Quality analysis

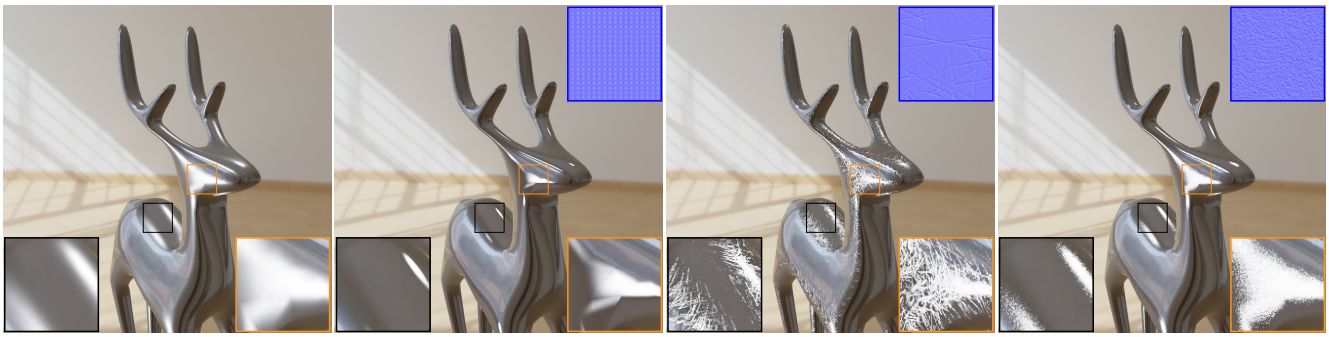
Algorithm validation. In Fig. 8, we compare the method of Yan *et al.* [1] to verify the correctness of our method and show the visualization of the results errors. To ensure that the geometric information of the two methods on the Bent quad scene is consistent, we explicitly export the microstructure generated by our procedural method and apply it to the reference method. Our proposed method is a fast approximate estimate of the correct value with low memory cost and is visually comparable to the reference [1]. The visualized error between our method and the reference [1] is mainly displayed in the overlapping regions of the examples.

Table 2. Comparison of Precomputation Time

Microstructure	Pre. time (s) Reference [1]	Pre. time (s) Ours
Leather	1565.7	6.2
Structure	3531.1	2.3
Brushed metal	1569.7	49.7
Isotropic noise	1568.9	6.2
Anisotropic noise	1559.8	6.1
Scratch	1567.3	72.4

Note: **Pre. time** is an abbreviation for precomputation time.

Microstructure representation range. Our method can simulate a lot of glossy materials under



(a) 9.3 ms (b) 13.3 ms (c) 13.9 ms (d) 12.0 ms

Fig. 10. Comparison between our results (b-d) and the result (a) under the GGX statistical distribution [22] on the **Deer statue scene**. Our method depicts the microstructure with the example normal map ((b-d) top right) and gets (b) structural highlight, (c) scratched effects, and (d) glittery effects.

Table 3. Performance and memory cost for the scenes of our method and reference [1]

Scene	Material	Input res.		Rendering time (ms)			Memory (MB)	
		Ours	Reference	Ours	Reference	Speedup	Ours	Reference
Deer statue	Scratch	32^2	$4K^2$	13.9	71.4	5.1×	85.25	3307.5
	Anisotropic	16^2	$4K^2$	12.3	62.5	5.1×	5.32	3307.5
	Isotropic	16^2	$4K^2$	12.0	71.2	5.9×	5.32	3307.5
	Structure	8^2	$6K^2$	13.3	80.7	6.1×	1.31	7441.9
	Brushed metal	16^2	$4K^2$	13.2	76.9	5.8×	21.31	3307.5
Coffee machine	Scratch	32^2	$4K^2$	14.5	142.8	9.8×	85.25	3307.5
	Isotropic	16^2	$4K^2$				5.32	3307.5
	Structure	8^2	$6K^2$				1.31	7441.9
	Brushed metal	16^2	$4K^2$				21.31	3307.5
Bent quad	Scratch	32^2	$4K^2$	11.8	90.9	7.7×	85.25	3307.5
	Anisotropic	8^2	$4K^2$	11.2	76.9	6.9×	5.32	3307.5
	Scratch & anisotropic	32^2 & 16^2	$4K^2$	14.5	166.7	11.5×	90.57	6615.0
Sphere	Anisotropic	256^2	$4K^2$	12.0	91.3	7.6×	5.32	3307.5
Shoes	Leather	16^2	$4K^2$	11.6	76.9	6.6×	5.32	3307.5
Desktop	Isotropic	16^2	$2K^2$	13.7	91.8	6.7×	38.58	4134.4
	Anisotropic	16^2	$2K^2$					
	Leather	16^2	$2K^2$					
	Structure	8^2	$2K^2$					
	Brushed metal	16^2	$2K^2$					

Note: **Input res.** denotes the resolution of the input normal maps (i.e. tiny examples in our method) used to represent the microstructure.

different structural features. In the Deer statue scene of Fig. 10 (b, c, d), the upper right corner shows the input tiny examples of our method, and Fig. 10 (a) shows the result under the GGX statistical distribution [22]. For tiled examples, we obtain structured highlights in Fig.10 (b). For stochastically distributed examples, we get disordered scratch effects in Fig. 10 (c) and shiny flakes under isotropic noise in Fig. 10 (d).

7.3 Performance and storage analysis

Precomputation time. Yan *et al.* [1] traverse a high-resolution normal map with the fixed texture sampling rate to obtain the Gaussian lobes. The preprocessing time depends on the resolution of the input normal map. Unlike the time-consuming high-resolution normal map, the time for sampling the tiny example is

negligible. Moreover, the precomputation time of our method mainly depends on the resolution of NDF images. Compared with previous work, we only need to precompute the multi-scale NDF approximation of a single tiny example. Thus the preprocessing speed has been significantly improved, as shown in Table 2.

Rendering time. Table 3 shows the rendering time comparison between our method and the reference [1] on different types of microstructures in test scenes. Our method reduces memory overhead and efficiently completes the originally time-consuming NDF evaluation, significantly improving the frame rate and meeting real-time rendering requirements. For the multi-microstructure-layer case, separately evaluating NDF for each layer results in almost double the performance cost.

Memory cost. In Table 3, we report the memory cost of our method and the reference during real-time shading. The memory cost of our method mainly depends on the resolution and LOD level of multi-scale NDF maps. The additional saved Gaussian lobes of the example require negligible memory consumption. Our method only uses a fraction (up to 0.05%) of memory cost compared to the reference. To describe leather, noise, and structured materials, the resolutions of tiny examples are usually smaller than 16×16, which are used along with NDF maps at resolutions of 8×8, 4×4, 2×2, and 1×1. Larger example sizes are used for microstructures with strong discrete characteristics such as brushed metal and scratches. For scratches, a 32×32 example is enough, corresponding NDF maps with resolutions of 16×16, 8×8, 4×4, 2×2, and 1×1, which result in higher memory usage.

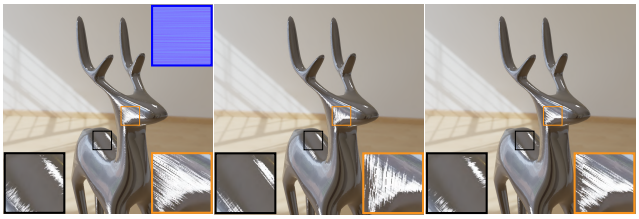


Fig.11. Results of the different example spatial transformations. The brushed metal microstructure exhibits different visual effects under different spatial transformation matrices in **Deer statue scene**.

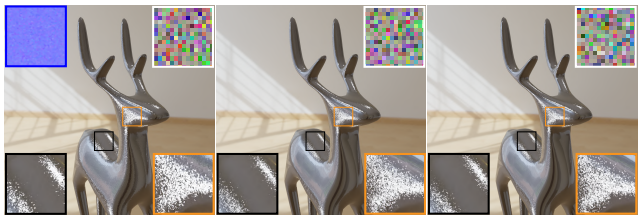


Fig.12. Results of the different example spatial distribution. The shiny glints of anisotropic noise microstructure are visually similar under different spatial distribution noise maps (shown in the upper right corner) in **Deer statue scene**.

7.4 Parameter analysis

We analyze the impact of examples' spatial transformation on high-frequency appearance for tiled brushed metal in Fig. 11. Compared to Fig. 11 (a), we obtain the effect of different highlight diffusion directions in Fig. 11 (b) through a unified rotation transformation matrix for each example. Fig. 11 (c) shows the coarser highlights under the unified scaling matrix.

We also discuss the macroscopic visual effects of the different spatial distributions of examples by different noise maps for randomly placed anisotropic noise in Fig. 12. Although the noise maps differ, the effects and time costs of different example distributions are highly similar.

We analyze the impact of the example resolution on rendering results and performance in Fig. 13. For brushed metal, examples with low resolution (Fig. 13 (a)) can not represent the overall geometric information of the material, and the results show obvious visual seam patterns after tiling.

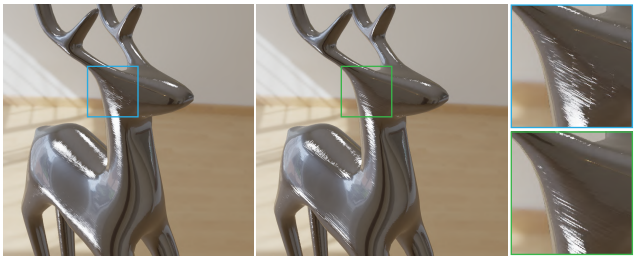


Fig.13. Comparison of different example resolutions on **Deer statue scene**. Tiling different resolutions of brushed microstructure examples yields different effects. The low-resolution setting in (a) causes visual errors.

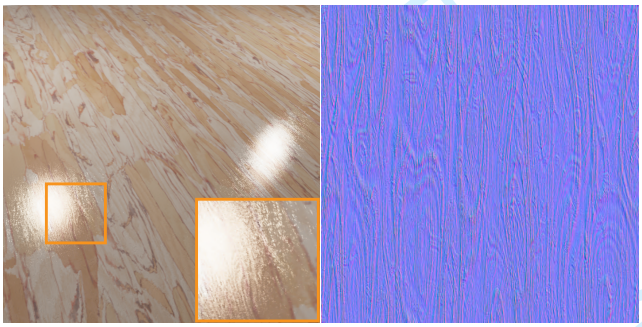


Fig.14. Limitation of our method. The rendering result for wood grain in (a) exhibits a loss of macro features due to the challenge of extracting structural examples from the strongly spatial correlated patterns in the continuous microstructure shown in (b).

7.5 Discussion and limitations

Our proposed method has several limitations due to our assumptions. We identify scenarios in which our method can be improved.

Absence of macroscopic features. We assume that the micro-geometry is composed of numerous microstructures described by the input tiny example. Therefore, our method does not support the simulation of global features in macro-geometry, such as the growth patterns of wood, as shown in Fig. 14.

Absence of generality of the representation. We assume that the examples have tiled and stochastically distributed types based on prior knowledge. Our method can not handle all microstructures uniformly, resulting in a lack of generality.

8 Conclusion and future work

We have presented a practical real-time method that efficiently renders a glinty appearance in a GPU-friendly manner with lower memory and computational overhead. We define the structural microstructures of micro-geometry as a tiny example, encoding its multi-scale NDF maps in a precomputed manner. Additionally, the overall geometry of the microstructure is obtained from example spatial distributions, and the structural diversity is enhanced through example spatial transformations. Eventually, high-frequency reflection features such as glints are reproduced in real-time with low memory cost using our method.

In the future, optimizing our method by integrating a more efficient example representation without the normal map, would be interesting. Additionally, the extension of our method for wave optics could lead to valuable advancements.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] Yan L Q, Hašan M, Marschner S, Ramamoorthi R. Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics (TOG)*, 2016, 35(4):1–9.

[2] Zhu J, Xu Y, Wang L. A stationary svbrdf material modeling method based on discrete microsurface. *Computer Graphics Forum*, 2019, 38(7):745–754.

[3] Wang B, Hašan M, Holzschuch N, Yan L Q. Example-based microstructure rendering with constant storage. *ACM Transactions on Graphics (TOG)*, 2020, 39(5):1–12.

[4] Tan H, Zhu J, Xu Y, Meng X, Wang L, Yan L Q. Real-time microstructure rendering with mip-mapped normal map samples. *Computer Graphics Forum*, 2022, 41(1):495–506.

[5] 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)*, 2014, 33(4):1–9.

[6] Yan L Q, Hašan M, Walter B, Marschner S, Ramamoorthi R. Rendering specular microgeometry with wave optics. *ACM Transactions on Graphics (TOG)*, 2018, 37(4):1–10.

[7] Gamboa L E, Guertin J P, Nowrouzezahrai D. Scalable appearance filtering for complex lighting effects. *ACM Transactions on Graphics (TOG)*, 2018, 37(6).

[8] Atanasov A, Wilkie A, Koylazov V, Křivánek J. A multiscale microfacet model based on inverse bin mapping. *Computer Graphics Forum*, 2021, 40.

[9] Jakob W, Hašan M, Yan L Q, Lawrence J, Ramamoorthi R, Marschner S. Discrete stochastic microfacet models. *ACM Transactions on Graphics (TOG)*, 2014, 33(4):1–10.

[10] Atanasov A, Koylazov V. A practical stochastic algorithm for rendering mirror-like flakes. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '16, Anaheim, CA, USA, July 24-28, 2016, Talks*, 2016, pp. 67:1–67:2.

[11] Wang B, Wang L, Holzschuch N. Fast global illumination with discrete stochastic microfacets using a filterable model. *Computer Graphics Forum*, 2018, 37(7):55–64.

[12] Raymond B, Guennebaud G, Barla P. Multi-scale rendering of scratched materials using a structured sv-brdf model. *ACM Transactions on Graphics*, 07 2016, 35.

[13] Werner S, Velinov Z, Jakob W, Hullin M B. Scratch iridescence: Wave-optical rendering of diffractive surface structure. *ACM Transactions on Graphics (TOG)*, 2017, 36(6):1–14.

[14] Deng H, Liu Y, Wang B, Yang J, Ma L, Holzschuch N, Yan L Q. Constant-cost spatio-angular prefiltering of glinty appearance using tensor decomposition. *ACM Transactions on Graphics (TOG)*, 2022, 41(2):1–17.

[15] Kuznetsov A, Hašan M, Xu Z, Yan L Q, Walter B, Kalantari N K, Marschner S, Ramamoorthi R. Learning generative models for rendering specular microgeometry. *ACM Trans. Graph.*, nov 2019, 38(6).

[16] 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, pp. 139–148.

[17] Deliot, Thomas, Belcour, Laurent. Real-time rendering of glinty appearances using distributed binomial laws on anisotropic grids, 2023.

[18] Chermain X, Sauvage B, Dischler J M, Dachsbacher C. Procedural physically based brdf for real-time rendering of glints. *Computer Graphics Forum*, 2020, 39(7):243–253.

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

[20] Wang B, Deng H, Holzschuch N. Real-time glints rendering with pre-filtered discrete stochastic microfacets. *Computer Graphics Forum*, 2020, 39(6):144–154.

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

[22] 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.

[23] Heckbert P S. Fundamentals of texture mapping and image warping. Technical Report UCB/CSD-89-516, EECS Department, University of California, Berkeley, Jun 1989.