Journal of Computer Science & Technology

Indexed in: SCIE, EI, INSPEC, CBST, DBLP, etc.

Sponsored by: ICT, CAS & CCF

JCST Papers

Only for academic and non-commercial use

Thanks for reading!

[Survey]

[Computer Architecture and Systems]

[Artificial Intelligence and Pattern Recognition]

[Computer Graphics and Multimedia]

[Data Management and Data Mining]

[Software Systems]

[Computer Networks and Distributed Computing]

[Theory and Algorithms]

[Emerging Areas]

JCST WeChat Subscription Account

JCST URL: https://jcst.ict.ac.cn

SPRINGER URL: https://www.springer.com/journal/11390

E-mail: jcst@ict.ac.cn

Online Submission: https://mc03.manuscriptcentral.com/jcst

Twitter: JCST_Journal

LinkedIn: Journal of Computer Science and Technology

Xing YX, Tan HW, Xu YN *et al.* A tiny example based procedural model for real-time glinty appearance rendering. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 39(4): 771-784 July 2024. DOI: 10.1007/s11390-024-4123-3

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

You-Xin Xing[1] (邢佑鑫), *Member, CCF*, Hao-Wen Tan[2] (谭皓文)
Yan-Ning Xu[1,*] (徐延宁), *Senior Member, CCF*, and Lu Wang[1,*] (王　璐), *Member, CCF*

[1] *School of Software, Shandong University, Jinan 250101, China*
[2] *NetEase (Hangzhou) Network Co., Ltd, Hangzhou 310052, China*

E-mail: youxinxing@mail.sdu.edu.cn; tanhaowen@corp.netease.com; xyn@sdu.edu.cn; luwang_hcivr@sdu.edu.cn

Received January 15, 2024; accepted March 29, 2024.

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 experimental results show that our tiny example based the 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 rendering, reflectance modeling, glinty material, photorealistic rendering, microstructure

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] defined 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, which make it difficult to apply directly in real-time scenarios that require immediate feedback and a lot of computation.

Zhu *et al.*[2] and Wang *et al.*[3] used 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

Regular Paper
Special Section of CVM 2024
This work was supported by the National Key Research and Development Program of China under Grant No. 2022YFB3303203 and the National Natural Science Foundation of China under Grant No. 62272275.
*Corresponding Author (Yan-Ning Xu is responsible for algorithm design and participated in paper writing. Lu Wang is responsible for the overall design and guidance of the paper, and participated in algorithmic optimization.)

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 normal distribution function (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 which 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 the 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. Chermain *et al.*[6] proposed the first practical multiple-scattering glint integrator based on normal maps. Their results are free of artifacts about the black holes on the surface, dark overall appearance, and fireflies. There is also some work[7–9] exploring the glinty appearance under wave optics[10]. However, these approaches have high storage requirements, and the 4D position-normal query is costly[11]. Gamboa *et al.*[12] 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.*[13] 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. Fan *et al.*[14] proposed to render specular glints by a different method: differentiable regularization, which gets significantly reduced noise but cannot guarantee unbiased convergence to the reference. Based on the normal map, explicit representation acquires spatial and directional features directly, providing more flexibility.

Implicit Representation. Jakob *et al.*[15] 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 and Koylazov[16] optimized the sampling process for better overall performance. Wang *et al.*[17] made the rendering process more efficient by deriving the method in a separable and filterable form.

Some methods for the microstructures of specific scratches[18, 19] have also been proposed. Deng *et al.*[20] 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 example normal map infinitely and maintain constant memory overhead. Different from the traditional methods

above, Kuznetsov *et al.*[21] first introduced a deep learning method by generating glinty patches without significant spatial repetition in glinty results. Guo *et al.*[22] proposed an implicit representation method for procedural material parameter estimation. They fitted material parameters to a range of materials, especially for glinty metallic paints through a Bayesian inference approach.

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 and Kaplanyan[23] proposed a method to accelerate the estimation of probability distribution using the MIP hierarchy and achieve real-time performance based on the work of Jakob *et al.*[15]. Deliot *et al.*[24] 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 methods[25, 26] are more physically based and can converge to the Cook-Torrance model[27] when the flake density is high enough. Wang *et al.*[28] simulated randomly discrete microfacet under environment lighting and point light in real time by prefiltering. Furthermore, Velinov *et al.*[29] proposed a scratch appearance method under wave optics based on their previous work[19].

Tan *et al.*[4] introduced a real-time prefiltering approach for microstructures that employs MIP-maps to select microstructures at the suitable level of details (LODs) for storage-stable and efficiency.

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

3 Background

In this section, we first introduce the surface BRDF and then provide the evaluation of the normal distribution function. Table 1 lists the symbols used throughout the paper.

3.1 Surface BRDF

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

$$f_{\mathcal{P}}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}) = \frac{D_{\mathcal{P}}(\boldsymbol{\omega}_{\mathrm{h}}) G(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{h}}) F(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{h}})}{4(\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{n})(\boldsymbol{\omega}_{\mathrm{o}} \cdot \boldsymbol{n})},$$

where $\boldsymbol{\omega}_{\mathrm{i}}$ and $\boldsymbol{\omega}_{\mathrm{o}}$ denote the light and view directions, respectively, the term $\boldsymbol{\omega}_{\mathrm{h}}$ refers to the half vector ($\boldsymbol{\omega}_{\mathrm{h}} = (\boldsymbol{\omega}_{\mathrm{i}} + \boldsymbol{\omega}_{\mathrm{o}})/\|\boldsymbol{\omega}_{\mathrm{i}} + \boldsymbol{\omega}_{\mathrm{o}}\|$), $\boldsymbol{n}$ represents the surface normal, F is the Fresnel term, G is the masking-shadowing function[30]. $D_{\mathcal{P}}(\boldsymbol{\omega}_{\mathrm{h}})$ 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 $\boldsymbol{\omega}_{\mathrm{h}}$. The evaluation of $D_{\mathcal{P}}(\boldsymbol{\omega}_{\mathrm{h}})$ is a major difficulty in microstructure rendering and is the core of the discussion in this paper.

Table 1. Notations

Symbol	Definition
$\mathcal{P}$	Footprint
$\boldsymbol{\omega}_{\mathrm{i}}$	Vector of incident light direction
$\boldsymbol{\omega}_{\mathrm{o}}$	Vector of output view direction
$\boldsymbol{\omega}_{\mathrm{h}}$	Half vector of $\boldsymbol{\omega}_{\mathrm{i}}$ and $\boldsymbol{\omega}_{\mathrm{o}}$
$f_{\mathcal{P}(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}})}$	Surface BRDF of $\boldsymbol{\omega}_{\mathrm{i}}$ and $\boldsymbol{\omega}_{\mathrm{o}}$ for $\mathcal{P}$
$\boldsymbol{n}$	Surface normal vector
$\boldsymbol{u}$	2D global texture coordinate vector
$\boldsymbol{u}'$	2D local texture coordinate vector
$\boldsymbol{s}$	Vector of query direction
$D_{\mathcal{P}}(\boldsymbol{s})$	Patch normal distribution function queried by $\boldsymbol{s}$ for $\mathcal{P}$
$\mathcal{N}(\boldsymbol{u}, \boldsymbol{s})$	Position-normal distribution queried by $\boldsymbol{u}$ and $\boldsymbol{s}$
$G_{\mathcal{P}}(\boldsymbol{u})$	Gaussian approximating queried by $\boldsymbol{u}$ for $\mathcal{P}$
$G_i(\boldsymbol{u}, \boldsymbol{s})$	4D Gaussian lobes queried by $\boldsymbol{u}$ and $\boldsymbol{s}$
$n(\boldsymbol{u})$	Normal map function queried by $\boldsymbol{u}$
$\boldsymbol{J}(\boldsymbol{u})$	Jacobian of $n(\boldsymbol{u})$
$\boldsymbol{\Sigma}_i^{-1}$	Inverse of 4×4 covariance matrix
σ_h	Standard deviation of seed Gaussians whose centers are h apart
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 $\mathcal{T}$
$\boldsymbol{t}$	Vector of local texture coordinates in an NDF map
$\boldsymbol{t}'$	Transformed $\boldsymbol{t}$
$\boldsymbol{M}$	Spatial transformation matrix
$\Phi(\boldsymbol{u})$	Tile position in the NDF map space queried by $\boldsymbol{u}$
$B_l(\boldsymbol{u}')$	NDF map function at level l queried by $\boldsymbol{u}'$

3.2 Evaluation of 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:

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

where $\boldsymbol{s}$ is the query direction (also represents the 2D normal with implicit z-coordinate), $G_{\mathcal{P}}$ is Gaussian approximating to a spatial footprint, $\boldsymbol{u}$ 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 (m) of 4D Gaussian lobes G_i obtained by traversing each texel in the normal map to approximate $\mathcal{N}$, i.e.,

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

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

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

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

$$\boldsymbol{\Sigma}_i^{-1} = \frac{1}{\sigma_h^2}\begin{pmatrix}\boldsymbol{I} & 0\\ 0 & 0\end{pmatrix} + \frac{1}{\sigma_r^2}\begin{pmatrix}\boldsymbol{J}(\boldsymbol{u})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{u}) & -\boldsymbol{J}(\boldsymbol{u})^{\mathrm{T}}\\ -\boldsymbol{J}(\boldsymbol{u}) & \boldsymbol{I}\end{pmatrix},$$

where, σ_h is the deviation of seed Gaussians, σ_{r} is the intrinsic roughness.

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

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

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 Subsection 5.1. We then compute multi-scale NDF maps for the tiny example in precomputation (Subsection 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 (Subsection 6.1) for stochastically distributed examples and distribution (Subsection 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 Subsection 6.3. Furthermore, we discuss the multi-microstructure-layer case in Subsection 6.4.

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

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 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 ex-

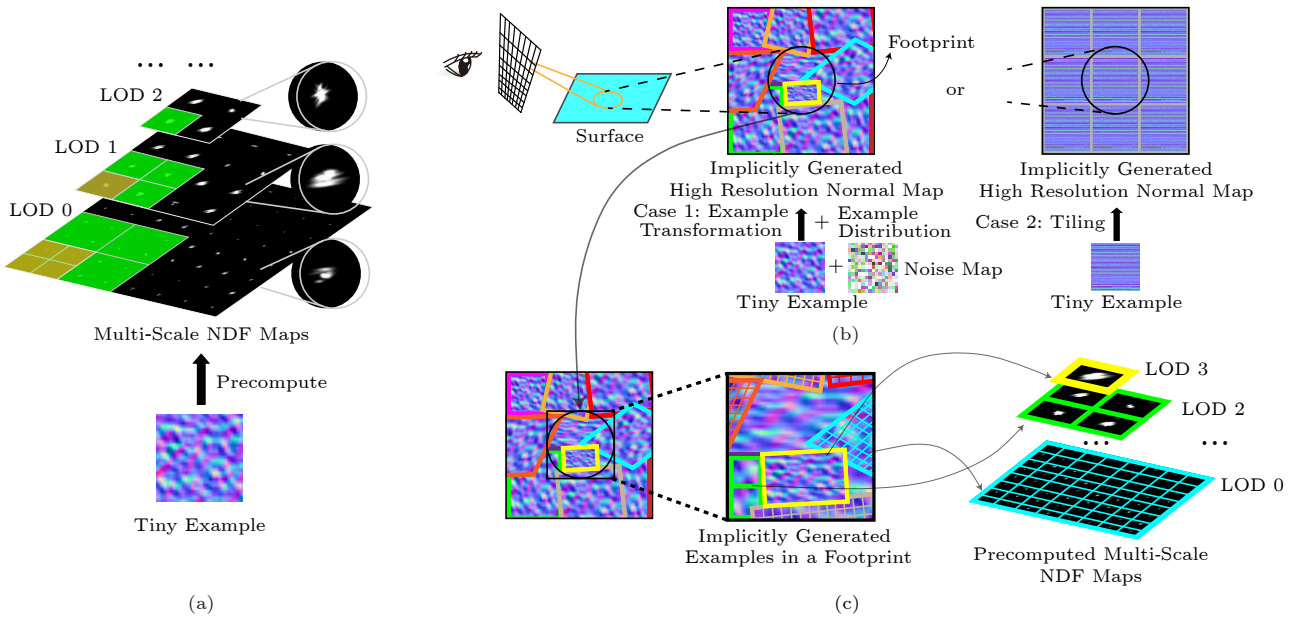


Fig.1. Pipeline of our method including two major parts: (a) precomputation and (b) real-time shading. The query details in a footprint are shown in (c).

hibiting 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 $r \times r$ 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, respectively.

While precomputing the NDF for each level based on the tiny example, we divide the tiny example into $r \times r$ 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.*'s[1], we assume the NDF of a tile is an equally weighted average of these micro-scale normals, which is defined as follows:

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

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

Each discrete NDF is computed per tile using (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 $\boldsymbol{t}$ $(\boldsymbol{t} = (t_x, t_y, \sqrt{1 - {t_x}^2 - {t_y}^2}))$ in the local texture space by a 3×3 matrix $\boldsymbol{M}$ to get transformed local texture coordi-

nates $\boldsymbol{t}'$, which is defined as:

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

The spatial transformation matrix $\boldsymbol{M}$ is defined as follows:

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

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 and y-axis, respectively, 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 $\boldsymbol{M}$, as shown in Fig.2.

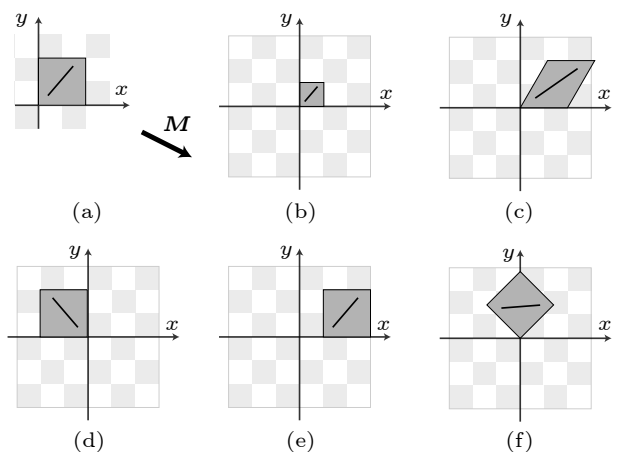


Fig.2. Spatial transformations for a tiny example through different transformation matrices. (a) Example. (b) Transformed example through scaling. (c) Transformed example through shearing. (d) Transformed example through symmetry. (e) Transformed example through offsetting. (f) Transformed example through rotation.

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 $\boldsymbol{M}$ (Subsection 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 $\boldsymbol{M}$, 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) 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 the pixel space, it is essential to determine its footprint $\mathcal{P}$ in the texture space. We use the same Gaussian representation method as Heckbert[31] to approximate the footprint with a parallelogram in the texture space. In this way, we obtain the texture positions $\boldsymbol{u}$ 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 $\boldsymbol{M}$ into a noise map, for a texture position $\boldsymbol{u}$, we can traverse all examples' information in the noise map and determine if $\boldsymbol{u}$ 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 $\boldsymbol{u}$. But for the 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.

Using this strategy, we determine a unique example that covers the position $\boldsymbol{u}$. Because we have dis-

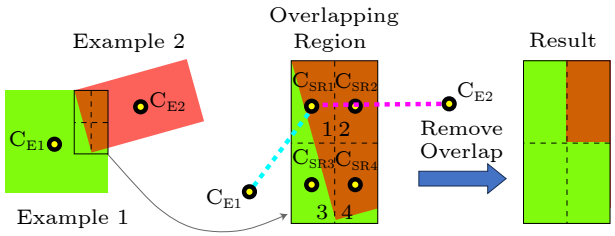


Fig.3. Examples overlapping case. We divide the overlapping region into four sub-regions equally first. C_{E1} and C_{E2} are the center positions of example 1 and 2, respectively. C_{SR1}–C_{SR4} are the center positions of sub-regions. For the sub-region 1 in the upper left part, C_{SR1}, the distance $|C_{SR1} - C_{E1}|$ 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.

tributed and transformed the examples onto the global texture space, we can obtain the local query position $\boldsymbol{u}'$ in NDF maps by back-projecting $\boldsymbol{u}$ based on the center position and the transformation matrix $\boldsymbol{M}$ of the example. We define this computation processing as $\Phi(\boldsymbol{u})$.

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. $B_l(\boldsymbol{u}')$ represents the NDF contribution of each example at LOD level l (under the query direction $\boldsymbol{s}$). 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 (1).

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 subsection discusses the multi-microstructure-layer case (as shown in Fig.4) without explicitly evaluating the complex light transport of interlayer scattering. 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. The

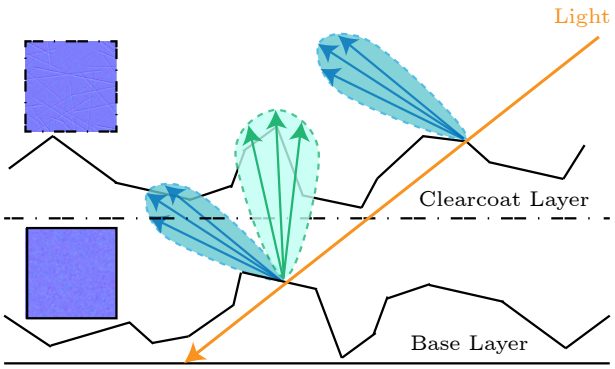


Fig.4. Visualization of the multi-microstructure-layer case.

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 (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 the masking-shadowing term for the clearcoat and the 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 differ-

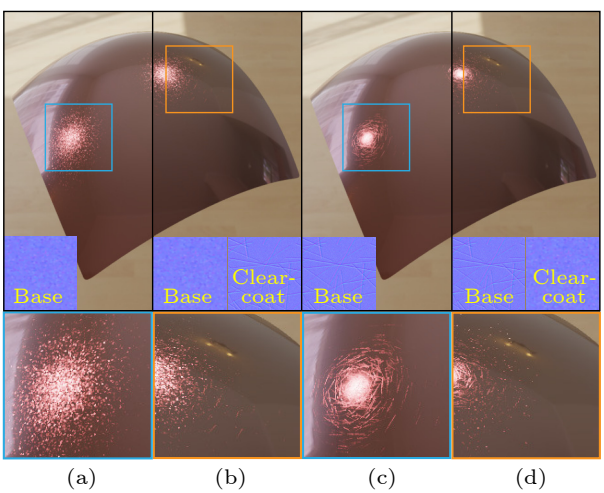


Fig.5. Rendering results of the multi-microstructure-layer case in the Bent quad scene. (a) Single-layer result with an anisotropic noise base (11.2 ms). (b) Multi-layer result with an anisotropic noise base and a scratch clearcoat (14.5 ms). (c) Single-layer result with a scratch base (11.8 ms). (d) Multi-layer result with a scratch base and an anisotropic noise clearcoat (14.5 ms).

ences in the visual characteristics of microstructures.

Two groups of identical microstructures with different layer orders present different visual effects in Figs.5(b) and 5(d), respectively, but the rendering time is consistent. The high-frequency effects in Figs.5(b) and 5(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. Furthermore, the multi-layer microstructures in Figs.5(b) and 5(d) exhibit more detailed features than the single-layer microstructure in Figs.5(a) and 5(c). 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 implement our method in OpenGL 4.6 and compare it with other typical methods (include Yan *et al.*[1], Wang *et al.*[3], Zhu *et al.*[2], and 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® i9-9900K CPU, 32 GB of main memory, and an NVIDIA® TITAN RTX GPU. Measurements are made for FHD (1 920 × 1 080) 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 that of [1]. The scene contains five common types of microstructures, including anisotropic noise, brushed metal, isotropic noise, leather, and structured material. The normal map resolution of each microstructure in the reference method is 2k × 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 consumes 1% of memory overhead compared with the reference method. Furthermore, our method only takes 13.7 ms compared with the reference method

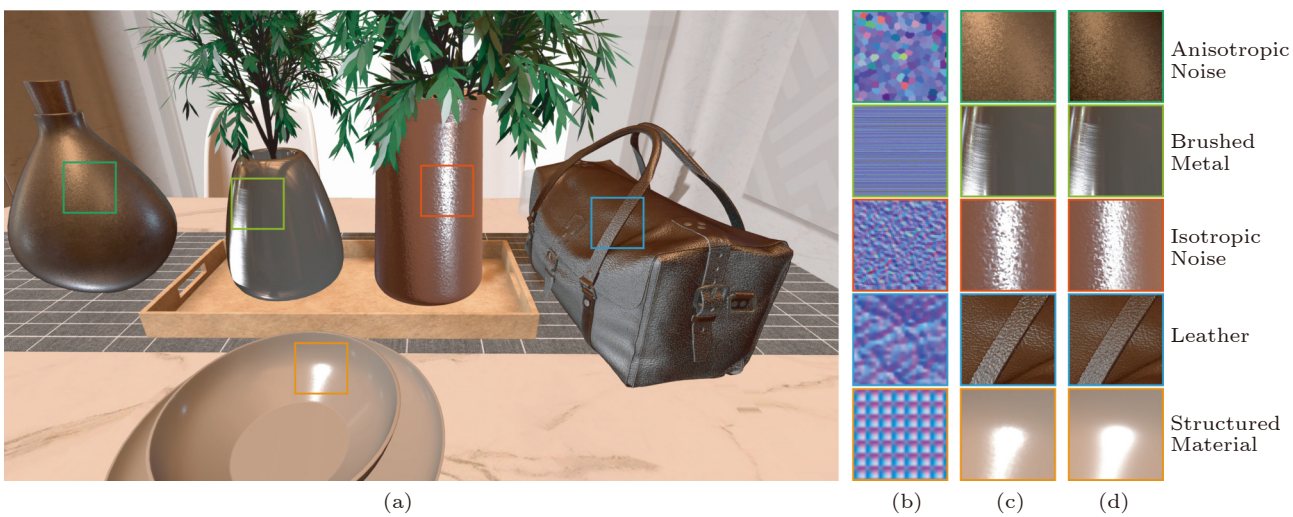


Fig.6. Comparison between our method and the reference method[1] on the Desktop scene with various types of microstructures. (a) Rendering results of our method. (b) Examples. (c) Result details of ours (38.58 MB, 13.7 ms). (d) Result details of the reference method (4 134.4 MB, 91.8 ms).

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. Whether the tiny examples are tiled (brushed metal, structured material, and isotropic noise) or randomly distributed (scratches) 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[2–4]. Therefore, our method is faster and significantly reduces memory overhead while rendering visually identical results.

Other Scenes. In Fig.8, there are three simple scenes: sphere scene (anisotropic noise), shoes scene (leather), and bent quad scene (scratches), corresponding to different microstructure types.

For the leather type in the 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. 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 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.9, we compare our method with that 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 that of [1]. The visualized error between our method and that of [1] is mainly displayed in the overlapping regions of the examples.

Microstructure Representation Range. Our method can simulate a lot of glossy materials under different structural features. In the deer statue scene of Figs.10(b), 10(c) and 10(d), the upper right corner shows the input tiny examples of our method, and

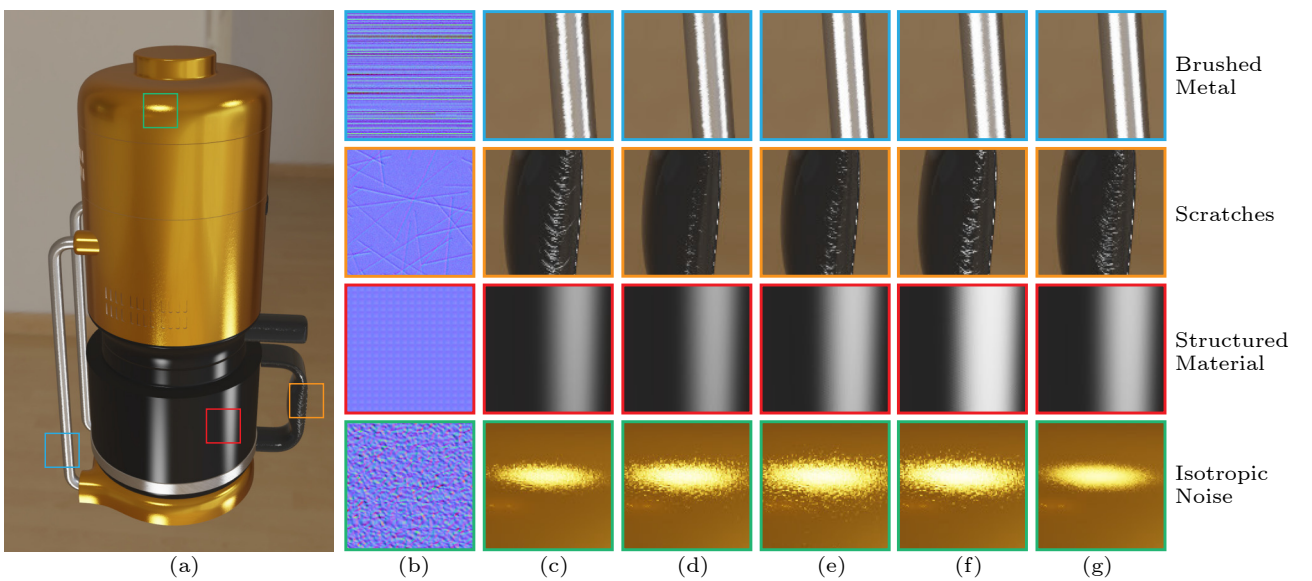


Fig.7. Comparison of our method with other microstructure rendering methods[1–4] on the coffee machine scene containing four types of microstructure. (a) Rendering results of our method. (b) Examples. (c) Result details of ours (113.2 MB, 14.5 ms). (d) Result details of Wang *et al.*[3] (592.0 MB, 142.7 ms). (e) Result details of Zhu *et al.*[2] (1 464.5 MB, 142.8 ms). (f) Result details of Tan *et al.*[4] (149.2 MB, 16.3 ms). (g) Result details of the reference method[1] (17 364.4 MB, 142.8 ms).

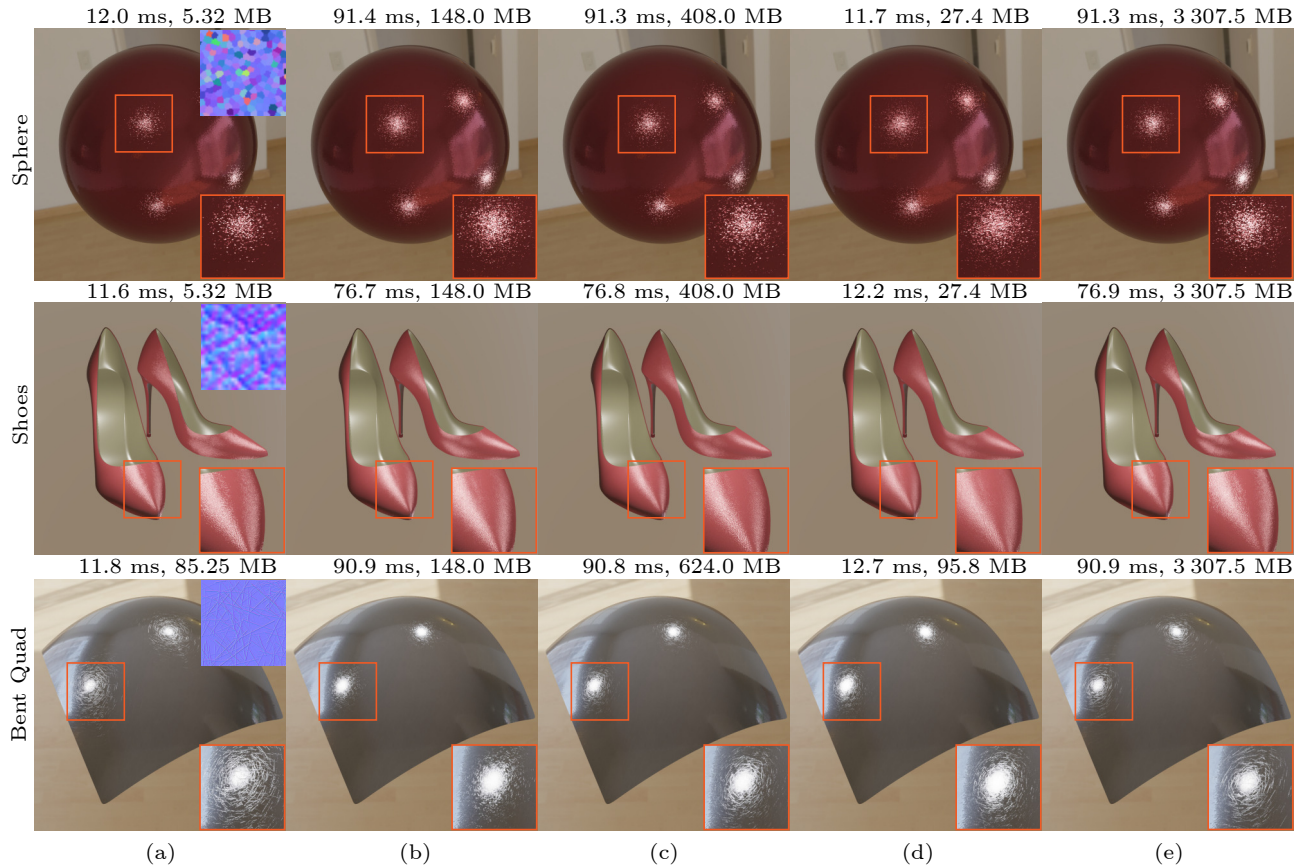


Fig.8. Comparison of rendering results using our method and other microstructure rendering methods[1–4] on three different scenes. We compare the glint effect generated by anisotropic noise microstructures using different methods in sphere scene, the continuous highlight of leather type microstructures in the shoes scene, and the scratch effect generated in the bent quad scene. (a) Ours. (b) Wang *et al.*[3] (c) Zhu *et al.*[2] (d) Tan *et al.*[4] (e) Reference method.[1]

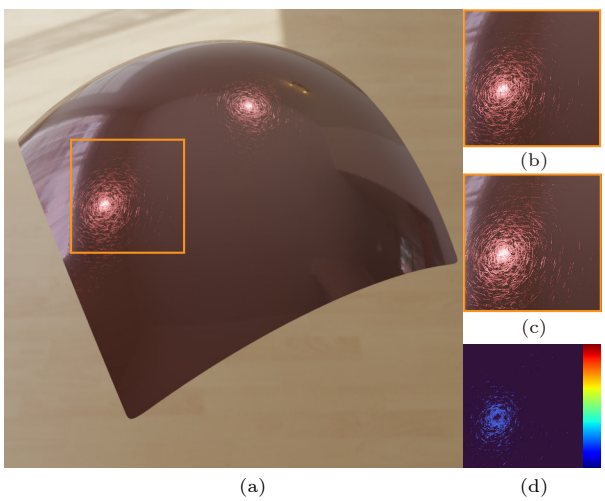


Fig.9. Comparison between our method and the reference method[1] on the bent quad scene under the same scratch microstructure. (a) Rendering results of our method. (b) Result details of ours (85.25 MB, 11.8 ms). (c) Result details of the reference method (3 307.5 MB, 90.9 ms). (d) Difference visualization between ours and the reference method.

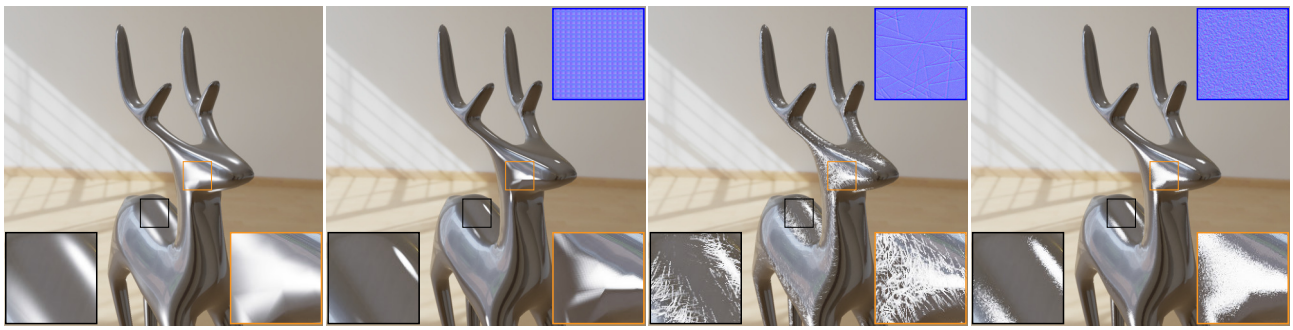
Fig.10(a) shows the result under the GGX statistical distribution[30].

For tiled examples, we obtain structural 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] traversed 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 method[1] on different types of microstructures in



(a) (b) (c) (d)

Fig.10. Comparison between our results and the reference result under the GGX statistical distribution[30] on the deer statue scene. (a) Reference result (9.3 ms). (b) Our result with structured materials (13.3 ms). (c) Our result with scratches (13.9 ms). (d) Our result with isotropic noise (12.0 ms).

Table 2. Comparison of Precomputation Time (s)

Microstructure	Reference[1]	Ours
Leather	1 565.7	6.2
Structure	3 531.1	2.3
Brushed metal	1 569.7	49.7
Isotropic noise	1 568.9	6.2
Anisotropic noise	1 559.8	6.1
Scratch	1 567.3	72.4

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 method 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 1.4%) of memory cost compared with the reference method on different scenes. 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 characteris-

Table 3. Performance and Memory Cost of Our Method and Reference Method[1]

Scene	Material	Input Resolution		Rendering Time (ms)			Memory (MB)	
		Ours	Reference	Ours	Reference	Speedup (x)	Ours	Reference
Deer statue	Scratches	32×32	$4k \times 4k$	13.9	71.4	5.1	85.25	3 307.5
	Anisotropic	16×16	$4k \times 4k$	12.3	62.5	5.1	5.32	3 307.5
	Isotropic	16×16	$4k \times 4k$	12.0	71.2	5.9	5.32	3 307.5
	Structure	8×8	$6k \times 6k$	13.3	80.7	6.1	1.31	7 441.9
	Brushed metal	32×32	$4k \times 4k$	13.2	76.9	5.8	21.31	3 307.5
Coffee machine	Scratches	32×32	$4k \times 4k$	14.5	142.8	9.8	85.25	3 307.5
	Isotropic	16×16	$4k \times 4k$	14.5	142.8	9.8	5.32	3 307.5
	Structure	8×8	$6k \times 6k$	14.5	142.8	9.8	1.31	7 441.9
	Brushed metal	32×32	$4k \times 4k$	14.5	142.8	9.8	21.31	3 307.5
Bent quad	Scratches	32×32	$4k \times 4k$	11.8	90.9	7.7	85.25	3 307.5
	Anisotropic	8×8	$4k \times 4k$	11.2	76.9	6.9	5.32	3 307.5
	Scratches & anisotropic	32×32 & 16×16	$4k \times 4k$	14.5	166.7	11.5	90.57	6 615.0
Sphere	Anisotropic	256×256	$4k \times 4k$	12.0	91.3	7.6	5.32	3 307.5
Shoes	Leather	16×16	$4k \times 4k$	11.6	76.9	6.6	5.32	3 307.5
Desktop	Isotropic	16×16	$2k \times 2k$	13.7	91.8	6.7	5.32	826.8
	Anisotropic	16×16	$2k \times 2k$	13.7	91.8	6.7	5.32	826.9
	Leather	16×16	$2k \times 2k$	13.7	91.8	6.7	5.32	826.9
	Structure	8×8	$2k \times 2k$	13.7	91.8	6.7	1.31	826.9
	Brushed metal	32×32	$2k \times 2k$	13.7	91.8	6.7	21.31	826.9

Note: Input resolution denotes the resolution of the input normal maps (i.e., tiny examples in our method) used to represent the microstructure. For the coffee machine and desktop scenes, we show multiple materials in the same scene and count their overall rendering time.

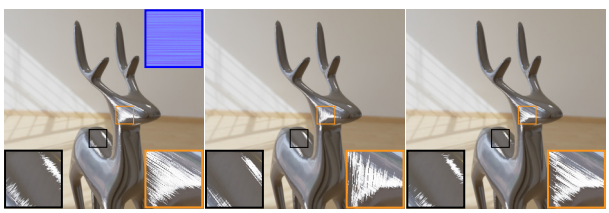
tics such as brushed metal and scratches. For scratches, a 32×32 example is enough. Its corresponding NDF maps has the resolutions of 16×16, 8×8, 4×4, 2×2, and 1×1, which result in higher memory usage than the 16×16 example's NDF maps.

7.4 Parameter Analysis

We analyze the impact of examples' spatial transformation on high-frequency appearance for tiled brushed metal in Fig.11. Compared with 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.

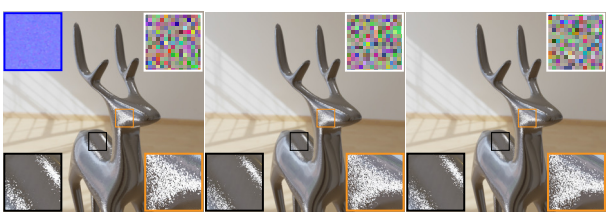
We also discuss the macroscopic visual effects of the different spatial distributions of examples by different noise maps for randomly placed anisotropic noise in Fig.12. Although the noise maps differ, the effects and time costs of different example distributions are highly similar.

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



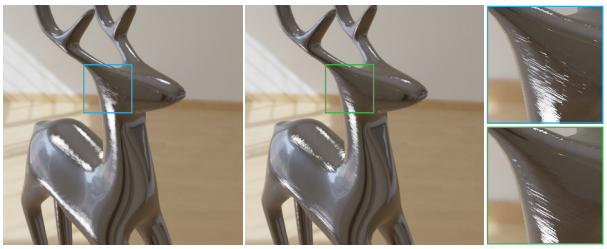
(a) (b) (c)

Fig.11. Glinty results of brushed metal microstructure exhibiting different visual effects in the deer statue scene. (a) Tiling result. (b) Result with a rotation transformation matrix. (c) Result with a scaling transformation matrix.



(a) (b) (c)

Fig.12. Shiny glints of anisotropic noise microstructure with different spatial distribution noise maps (shown in the upper right corner) in the deer statue scene. (a)–(c) Results with different example spatial distributions determined by the three different noise maps separately.



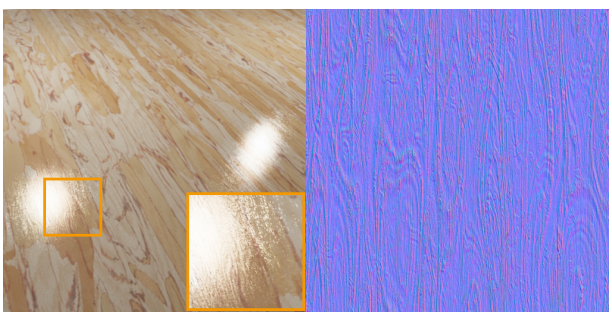
(a) (b) (c)

Fig.13. Comparison of different example resolutions on the deer statue scene. (a) Result with an 8×8 tiny example (1.33 MB, 13.8 ms). (b) Result with a 32×32 tiny example (21.31 MB, 13.2 ms). (c) Details of our results.

7.5 Discussions 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. The rendering result for wood grain in Fig.14(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 Fig.14(b).



(a) (b)

Fig.14. Limitation of our method. (a) Result of the wooden floor. (b) High-resolution normal map.

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

8 Conclusions

We presented a practical real-time method that efficiently renders a glinty appearance in a GPU-

friendly manner with lower memory and computational overhead. The structural microstructures of microgeometry is defined 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. Besides, the extension of our method for wave optics could lead to valuable advancements. Additionally, the combination of glinty appearance and 3D visual optical arts[32] is also promising.

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 Trans. Graphics*, 2016, 35(4): Article No. 56. DOI: 10.1145/2897824.2925915.

[2] Zhu J Q, Xu Y N, Wang L. A stationary SVBRDF material modeling method based on discrete microsurface. *Computer Graphics Forum*, 2019, 38(7): 745–754. DOI: 10.1111/cgf.13876.

[3] Wang B B, Hašan M, Holzschuch N, Yan L Q. Example-based microstructure rendering with constant storage. *ACM Trans. Graphics*, 2020, 39(5): Article No. 162. DOI: 10.1145/3406836.

[4] Tan H W, Zhu J Q, Xu Y N, Meng X X, Wang L, Yan L Q. Real-time microstructure rendering with MIP-mapped normal map samples. *Computer Graphics Forum*, 2022, 41(1): 495–506. DOI: 10.1111/cgf.14448.

[5] Yan L Q, Hašan M, Jakob W, Lawrence J, Marschner S, Ramamoorthi R. Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Trans. Graphics*, 2014, 33(4): 116. DOI: 10.1145/2601097.2601155.

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

[7] Yan L Q, Hašan M, Walter B, Marschner S, Ramamoorthi R. Rendering specular microgeometry with wave optics. *ACM Trans. Graphics*, 2018, 37(4): 75. DOI: 10.1145/3197517.3201351.

[8] Guo J, Chen Y J, Guo Y W, Pan J G. A physically-based appearance model for special effect pigments. *Computer Graphics Forum*, 2018, 37(4): 67–76. DOI: 10.1111/cgf.13476.

[9] Xia M Q, Walter B, Hery C, Maury O, Michielssen E, Marschner S. A practical wave optics reflection model for hair and fur. *ACM Trans. Graphics*, 2023, 42(4): Article No. 39. DOI: 10.1145/3592446.

[10] Yu Y C, Xia M Q, Walter B, Michielssen E, Marschner S. A full-wave reference simulator for computing surface reflectance. *ACM Trans. Graphics*, 2023, 42(4): 109. DOI: 10.1145/3592414.

[11] Zhu J Q, Zhao S Z, Xu Y N, Meng X X, Wang L, Yan L Q. Recent advances in glinty appearance rendering. *Computational Visual Media*, 2022, 8(4): 535–552. DOI: 10.1007/s41095-022-0280-x.

[12] Gamboa L E, Guertin J P, Nowrouzezahrai D. Scalable appearance filtering for complex lighting effects. *ACM Trans. Graphics*, 2018, 37(6): Article No. 277. DOI: 10.1145/3272127.3275058.

[13] Atanasov A, Wilkie A, Koylazov V, Křivánek J. A multi-scale microfacet model based on inverse bin mapping. *Computer Graphics Forum*, 2021, 40(2): 103–113. DOI: 10.1111/cgf.142618.

[14] Fan J H, Wang B B, Wu W S, Hašan M, Yang J, Yan L Q. Efficient specular glints rendering with differentiable regularization. *IEEE Trans. Visualization and Computer Graphics*, 2023, 29(6): 2940–2949. DOI: 10.1109/TVCG.2022.3144479.

[15] Jakob W, Hašan M, Yan L Q, Lawrence J, Ramamoorthi R, Marschner S. Discrete stochastic microfacet models. *ACM Trans. Graphics*, 2014, 33(4): Article No. 115. DOI: 10.1145/2601097.2601186.

[16] Atanasov A, Koylazov V. A practical stochastic algorithm for rendering mirror-like flakes. In *Proc. the 2016 ACM SIGGRAPH Talks*, Jul. 2016, Article No. 67. DOI: 10.1145/2897839.2927391.

[17] Wang B B, Wang L, Holzschuch N. Fast global illumination with discrete stochastic microfacets using a filterable model. *Computer Graphics Forum*, 2018, 37(7): 55–64. DOI: 10.1111/cgf.13547.

[18] Raymond B, Guennebaud G, Barla P. Multi-scale rendering of scratched materials using a structured SV-BRDF model. *ACM Trans. Graphics*, 2016, 35(4): Article No. 57. DOI: 10.1145/2897824.2925945.

[19] Werner S, Velinov Z, Jakob W, Hullin M B. Scratch iridescence: Wave-optical rendering of diffractive surface structure. *ACM Trans. Graphics*, 2017, 36(6): Article No. 207. DOI: 10.1145/3130800.3130840.

[20] Deng H, Liu Y, Wang B B, Yang J, Ma L, Holzschuch N, Yan L Q. Constant-cost spatio-angular prefiltering of glinty appearance using tensor decomposition. *ACM Trans. Graphics*, 2022, 41(2): Article No. 22. DOI: 10.1145/3507915.

[21] Kuznetsov A, Hašan M, Xu Z X, Yan L Q, Walter B, Kalantari N K, Marschner S, Ramamoorthi R. Learning generative models for rendering specular microgeometry. *ACM Trans. Graphics*, 2019, 38(6): Article No. 225. DOI: 10.1145/3355089.3356525.

[22] Guo Y, Hašan M, Yan L, Zhao S. A Bayesian inference framework for procedural material parameter estimation. *Computer Graphics Forum*, 2020, 39(7): 255–266. DOI:

10.1111/cgf.14142.
[23] Zirr T, Kaplanyan A S. Real-time rendering of procedural multiscale materials. In *Proc. the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, Feb. 2016, pp.139–148. DOI: 10.1145/2856400.2856409.
[24] Deliot T, Belcour L. Real-time rendering of glinty appearances using distributed binomial laws on anisotropic grids. *Computer Graphics Forum*, 2023, 42(8): e14866. DOI: 10.1111/cgf.14866.
[25] 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. DOI: 10.1111/cgf.14141.
[26] Chermain X, Lucas S, Sauvage B, Dischler J M, Dachsbacher C. Real-time geometric glint anti-aliasing with normal map filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2021, 4(1): Article No. 1. DOI: 10.1145/3451257.
[27] Cook R L, Torrance K E. A reflectance model for computer graphics. *ACM Trans. Graphics*, 1982, 1(1): 7–24. DOI: 10.1145/357290.357293.
[28] Wang B B, Deng H, Holzschuch N. Real-time glints rendering with pre-filtered discrete stochastic microfacets. *Computer Graphics Forum*, 2020, 39(6): 144–154. DOI: 10.1111/cgf.14007.
[29] 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. DOI: 10.1111/cgf.13347.
[30] Walter B, Marschner S R, Li H S, Torrance K E. Microfacet models for refraction through rough surfaces. In *Proc. the 18th Eurographics Conference on Rendering Techniques*, Jun. 2007, pp.195–206. DOI: 10.5555/2383847.2383874.
[31] Heckbert P S. Fundamentals of texture mapping and image warping. Technical Report UCB/CSD-89-516, University of California, 1989.
[32] Shen P F, Li R Z, Wang B B, Liu L G. Scratch-based reflection art via differentiable rendering. *ACM Trans. Graphics*, 2023, 42(4): 65. DOI: 10.1145/3592142.

You-Xin Xing received his B.S. degree in digital media technology from the College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, in 2020. He is currently a Ph.D. candidate at the School of Software, Shandong University, Jinan. His research interests include real-time rendering, material appearance modeling, and game development.

Hao-Wen Tan received his M.S. degree in software engineering from the School of Software, Shandong University, Jinan, in 2023. He is a senior game engine developer at NetEase (Hangzhou) Network Co., Ltd, Hangzhou. His research interest is mainly in real-time rendering.

Yan-Ning Xu received his Ph.D. degree in computer science from the Department of Computer Science and Technology, Shandong University, Jinan, in 2006. He is an associate professor at the School of Software, Shandong University, Jinan. His research interests are computer-aided design (CAD), graphics, virtual reality, etc.

Lu Wang received her Ph.D. degree in computer science and technology from the Department of Computer Science and Technology, Shandong University, Jinan, in 2009. She is a professor at the School of Software, Shandong University, Jinan. Her research interests include photorealistic rendering, real-time rendering, material appearance modeling, and high-performance rendering.