Computational Visual Media
DOI 10.1007/s41095-xxx-xxxx-x Vol. x, No. x, mouth year, xx–xx

Review Article

Recent Advances in Glinty Appearance Rendering

Junqiu Zhu[1], Sizhe Zhao[1], Yanning Xu[1] (✉), Xiangxu Meng[1], Lu Wang[1] and Ling-Qi Yan[2]

Abstract The interaction between light and materials is key to physically-based realistic rendering. However, it is also complex to analyze, especially when the materials contain a large number of details and thus exhibit 'glinty' visual effects. Recent methods of producing glinty appearance are expected to be important in next-generation computer graphics. We provide here a comprehensive survey on recent glinty appearance rendering. We start with a definition of glinty appearance based on microfacet theory, then we summarize research works in terms of representation and practical rendering. We have implemented typical methods using our unified platform and compare them in terms of visual effects, rendering speed, and memory consumption. Finally, we briefly discuss limitations and future research directions. We hope our analysis, implementations, and comparisons will provide insight for readers hoping to choose suitable methods for applications, or carry out research.

Keywords glinty appearance; Monte Carlo methods; rendering.

1 Introduction

In the real world, many materials exhibit visual appearances with glinty effects, e.g. car paint under strong sunlight, and tiny scratches on heavily used cutlery. These glinty visual appearances are often complex and unstructured, and they greatly enhance the richness of the visual world. However, in computer graphics, the glinty effects in visual appearances are often ignored, leading to an overly perfect and smooth appearance (see Fig. 1). In order to improve the realism of computer-generated imagery (CGI), it is very important for a photo-realistic renderer to take such glinty appearance into consideration.

However, glinty appearance rendering is difficult. In the real world, tiny bumps and dents on the surfaces of objects can be found everywhere. These geometric details introduce high-frequency variations in appearance and cause glinty effects noticeable to the human eye. Therefore, in order to incorporate glinty appearance in rendering, one must start from the causes of such appearance—complex microgeometry and the way it interacts with light. However, two difficult problems arise naturally. The first is how to represent this complex microgeometry both exhaustively (without overly simplifying or ignoring any geometry) and compactly (without introducing significant storage or memory consumption overheads). The second is how to render such complex microgeometry efficiently, faithfully bringing out the glints without prohibitive computation.

Historically, people have used microfacet models [55] to describe optical properties of surfaces, and more specifically, bidirectional reflectance distribution functions (BRDFs). Microfacet theory assumes that a surface is composed of many microfacets. Each microfacet causes a perfect mirror-like reflection. Traditional methods prefilter glinty appearance and smoothly approximate BRDFs, which results in a smooth appearance which omits glinty effects. It is the distribution of the orientations (normals) of the microfacets that determine appearance. However, microfacet theory uses a statistical approach (e.g. using 2D Gaussians) to describe the microfacets' normal distribution function (NDF). Since statistical functions are usually smooth and only focus on overall distributions rather than details, they inevitably produce smooth appearances lacking glinty reflections.

Recently, various research has been devoted to

1 Shandong University, Jinan, Shandong, China. E-mail: zhujunqiu@mail.sdu.edu.cn, zhaosizhe@mail.sdu.edu.cn, xyn@sdu.edu.cn(✉),mxx@sdu.edu.cn, luwang_hcivr@sdu.edu.cn.
2 University of California, Santa Barbara, USA. E-mail: lingqi@cs.ucsb.edu.
Manuscript received: 2021-xx-xx; accepted: 20xx-xx-xx.

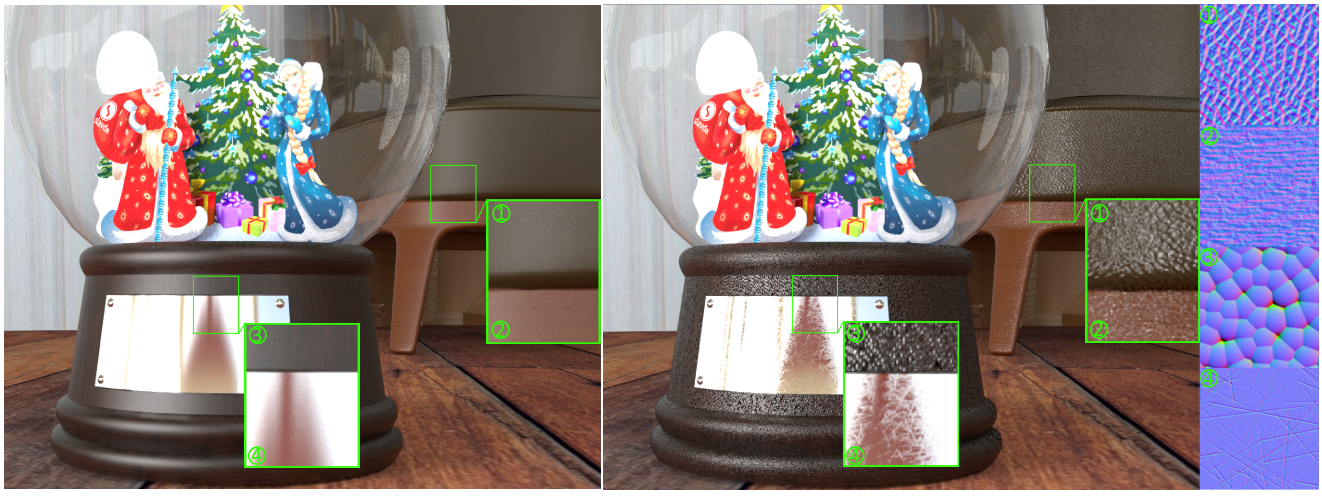


Fig. 1 A complex indoor scene with four kinds of glinty appearance. Left: rendering with traditional microfacet theory. Right: Rendering using method from [66]. The glinty features on the wooden sofa legs, the leather sofa cushion, the bumpy base of the ornament and scratched metals dramatically improve the image's realism.

extending microfacet theory, especially using actual NDFs, to produce glinty details on surfaces. We call this line of research *glinty appearance rendering*. As noted earlier, the key problems, which we focus on in this survey, are the *representation* and *rendering* of the complex microgeometry. We provide the first thorough summary of state-of-the-art glinty appearance rendering research.

Most related works concern offline methods, so we focus on offline pipelines, and discuss real-time methods as an extension to them. We begin by introducing background knowledge, and overview solutions in Sec. 2. We then divide the complex glinty appearance problem into representation and rendering issues, and discuss commonalities and differences of previous research work in Secs. 3 and 4. In Sec. 5, we present our unified platform, which implements typical offline approaches and use it to compare these approaches in terms of visual effects, rendering efficiency, and memory consumption. Finally, we introduce extended works in Sec. 6, covering *wave optics*, *machine learning* and *real-time rendering*. We believe our survey will not only help readers new to this area to quickly understand the high-level concepts and solutions, but also benefit experienced researchers in choosing suitable methods for different application scenarios, and discovering future research directions in this area. z

2 Background and Overview

In this section, we first introduce the microfacet-based BRDF, then, we define what glinty appearance is and analyze deficiencies of naïve rendering methods.

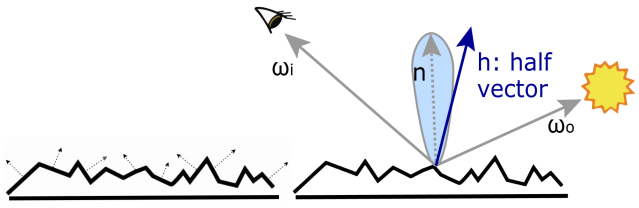


Fig. 2 Microfacet theory. Left: each surface microfacet has a normal, indicating the direction it faces. Right: in practice, statistical functions are used to describe the distributions of the normals of a collection of microfacets (blue lobe); this is an important term of a microfacet-based BRDF. h is the half vector of a specific ω_i and ω_o, used in the BRDF to calculate the reflectance.

Next, we recap pre-filtering techniques and analyze their limitations for glinty appearance rendering. Finally, we overview the glinty appearance methods that are discussed in detail in the following sections.

2.1 Statistical Appearance Models

In physically-based rendering, the microfacet BRDF is widely used to model surfaces with many tiny facets that reflect rays as perfect mirrors (see Fig. 2). The distribution of microfacet normals is generally defined by a normal distribution function (NDF) [10]. With the NDF, we can determine how many microfacets reflect light from the incident direction $\omega_\mathbf{i}$ to the outgoing direction $\omega_\mathbf{o}$, or how many microfacets normals point exactly along the half vector direction $\mathbf{h}$ between the camera and light directions.

The microfacet BRDF can be defined as:

$$f_r(\omega_i, \omega_o) = \frac{F(\omega_i, \omega_h)\, G(\omega_i, \omega_\mathbf{o}, \mathbf{h})\, D(\mathbf{h})}{4(\mathbf{n}\cdot\omega_\mathbf{i})(\mathbf{n}\cdot\omega_\mathbf{o})}, \quad (1)$$

where F is the Fresnel term, G is the shadowing-

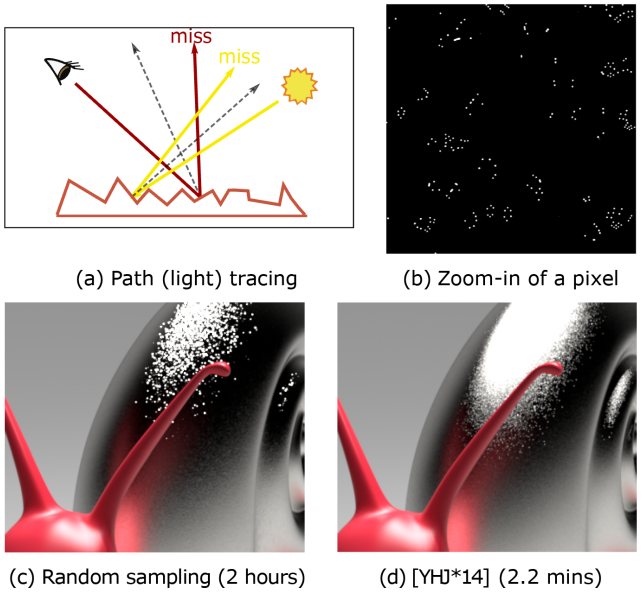


Fig. 3 (a, b): Path tracing with random pixel sampling has little chance to find a valid path because only a tiny minority of microfacets will contribute to the reflectance. (b, c): Brute force sampling will consume much more time but may still fail as the small proportion of contributing microfacets provide a large amount of energy, and brightness will be greatly reduce if some are missed. (d): Recent advances in glinty appearance rendering generate better results in less time. (b–d) are taken from Yan et al. [65]

masking term, and D is the NDF term. We focus on the NDF term throughout this survey because it is the dominant factor in glinty appearance rendering.

The microfacet BRDF [55] is successfully used in practice. However, traditional methods work by statistically modeling the aggregate behavior of a collection of microfacets using a smooth NDF. This smooth NDF tends to eliminate glint features and result in smooth appearance (see Fig. 1). Kurt [41] reviews BRDF measurement and representation methods, and overviews the microfacet-based model very well. These methods focus on the naïve microfacet BRDF which can not effectively render complex glinty appearance.

2.2 Glinty Appearance

2.2.1 What is Glinty Appearance?

Many surfaces in reality have rich, high-frequency variable reflections under intense light sources. Random, tiny facets on surfaces, with size from a few to hundreds of microns, can produce a glinty appearance. Human eyes are very sensitive to shiny reflections, under slight changes in lighting or viewing direction. In this survey, we identify materials that have microfacets and generate glinty visual effects as having *glinty appearance*.

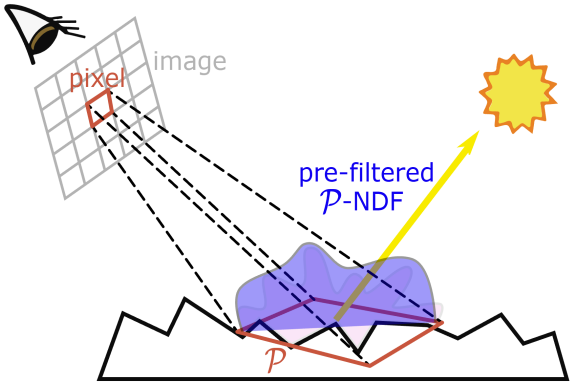


Fig. 4 Pre-filtering methods use a smooth distribution function to approximating the complex $\mathcal{P}$-NDF.

2.2.2 Difficulties in Glinty Appearance Rendering

The most straightforward approach to glinty appearance rendering is the path tracing [33] technique. However, neither the naïve path tracing approach nor similar bidirectional approaches [31, 57] can efficiently process glinty surfaces. The main problem is that the perfect mirror-like reflection behavior of individual facets prevents us from sampling the correct facets and finding valid light paths. As Fig. 3 shows, even if we spend more time to process more samples, and introduce a little roughness to make the facets not so smooth, it is still a challenge to shade glinty effects properly. Among tens of thousands of discrete tiny facets, only tens of them might contribute to a given pixel's highlight. We need to find them all to obtain a noise-free image containing glinty features. Doing so is extremely expensive and quite intractable. Recently, researchers have focused on these problems and provided various solutions.

2.3 Solutions: Overview

To solve the glinty appearance rendering problem, the basic idea is to process a surface patch $\mathcal{P}$ seen through a screen pixel all at once. Some researchers pre-filter patch $\mathcal{P}$ to approximate the contribution over a surface patch (see Fig. 4). However, these methods tend to reduce variations and produce smooth results. To render glinty effects, some researchers take the actual patch as input and calculate the discrete NDF over $\mathcal{P}$ accurately (see Fig. 6). We briefly consider the former and discuss the latter in detail.

2.3.1 Pre-filtering Methods

Pre-filtering methods have been widely used for decades and provide partial solutions to glinty appearance rendering. These methods evaluate the

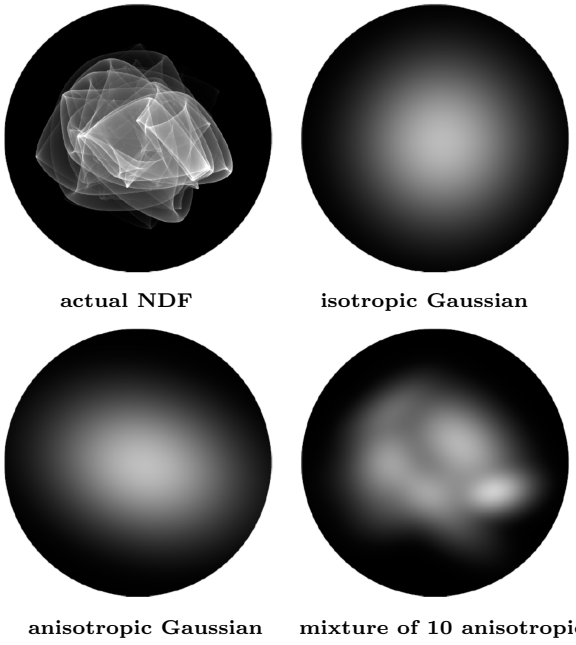


Fig. 5 Approximating the actual $\mathcal{P}$-NDF by an isotropic lobe ([54]), an anisotropic lobe ([45]) or several lobes ([23]) loses the sharp features that cause glints. Image from Yan et al. [65]

outgoing radiance over a patch $\mathcal{P}$ and try to average the outgoing radiance from $\mathcal{P}$ (see Fig. 4). Brusneton et al. [3] present a thorough overview of such pre-filtering methods. Here we briefly introduce some typical pre-filtering methods and explain reasons why they cannot preserve glinty effects.

One pre-filtering strategy is to store a large number of pre-computed or measured reflectances for different viewing and lighting directions, and organize them according to $\mathcal{P}$ covering a coarse mesh. The values can be stored in a 6D table called a bidirectional texture function, or BTF [11]. Suykens et al. [50] create a BTF for Monte-Carlo simulation on a geometric surface model. Ma et al. [39] enable interactive BTF rendering by compressing the BTF into a manageable representation.

Assuming that surface colors, NDF, visibility and shadow-masking are uncorrelated, another strategy pre-filters these properties separately; NDF pre-filtering issues have been studied in many works.

The simplest approach is to pre-filter the NDF as a single lobe in $\mathcal{P}$. Neyret [43] models the NDF with an ellipsoid function. Olano and North [45] model the NDF with a Gaussian-lobe defined by its mean normal and a covariance matrix. Their method supports anisotropic highlights efficiently. Heitz et al. [25] introduce the SGGX function to represent the spatially varying properties of anisotropic microflakes.

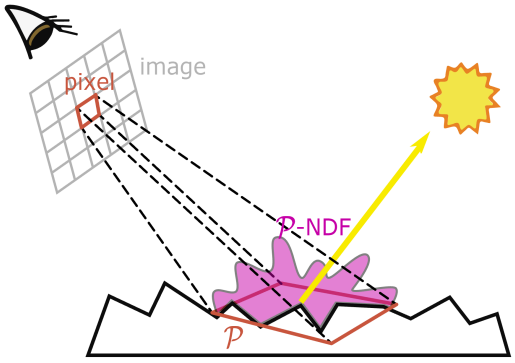


Fig. 6 Rendering glinty appearance needs accurate evaluation of the $\mathcal{P}$-NDF of the corresponding patch $\mathcal{P}$ on the surface which covers one pixel's projection.

However, these methods cannot deal with surfaces that have structured microfacets or ones organized into a pattern.

Multiple lobe pre-filtering methods provide the ability to fit a more complex NDF. Fournier [17] represents the NDF with a sum of Blinn-Phong lobes [47]. His method supports up to 7 lobes and 28 parameters in his implementations. Tan et al. [51, 52] extend the approach by using a mixture of isotropic Gaussian lobes to represent the average NDF. Han et al. [23] use a convolution method to model the macroscopic NDF with its decomposition into spherical harmonics and VMFs. Wu et al. [63] define characteristic point maps and present a principal component analysis method to find principal lobes based on their data structure. They [64] further present an efficient filtering algorithm to reconstruct bi-scale surfaces that contain both macro-scale and micro-scale information.

The above pre-filtering methods are helpful for some kinds of glinty appearance but have limitations in rendering glinty materials, as the NDF for a glinty appearance can be too complex to be fitted by several lobes. Fig. 5 shows the effect of replacing one NDF by a single lobe or multiple lobes: note the loss of sharp features.

2.3.2 Glinty Appearance Rendering Methods

Recently, researchers have taken actual microfacets as input and accurately describe the microfacets' distribution (mainly focusing on the NDF) in a surface patch $\mathcal{P}$, which preserves glinty effects. However, the distribution can be extremely complex, so the core problem of glinty appearance rendering is how to efficiently and accurately evaluate the NDF over the patch ($\mathcal{P}$-NDF). We consider glinty appearance rendering methods in terms of *representation* and

rendering.

To render the glinty appearance, the first step is represent the actual microstructures. This is because glinty appearances are generally complex and varied, and representing them is a prerequisite for rendering them. Furthermore, the representation determines the method of evaluation of the NDF, which affects overall performance and storage. The main issues to be considered include memory costs, computation costs, accuracy, and supported material types. In Sec. 3, we discuss existing representation methods and compare them in terms of the above factors.

In practice, how to render the glinty appearance is another challenge. In Sec. 4, we discuss different evaluation technologies including acceleration data structures. We also discuss the importance sampling and multiple scattering which have great impacts on the final results.

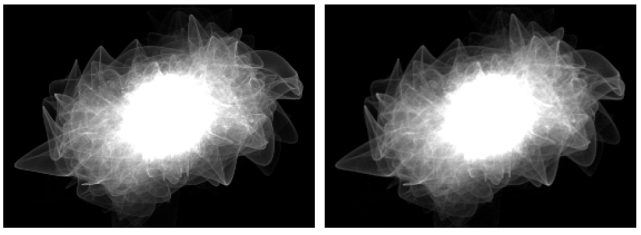
3 Microstructure Representation

In the real world, most shiny surfaces have glinty appearance, which is noticeable under strong light. Typical high-frequency materials like bumps, flakes, scratches, leather grain, dimples etc. contain various microstructures. Representing such features is not an easy task. First, the glint features are small and complex, and representing all details would require an extremely large amount of storage. Another difficulty is that we need area-integration methods to accurately evaluate the $\mathcal{P}$-NDF of each patch, and the representation must effectively support evaluation. Different representations have their own advantages and limitations. In this section, we classify recent representations as *explicit* or *implicit*, and further discuss each kind in turn.

3.1 Explicit Representation

Explicit representations store the original microstructure of a surface in various forms. The naïve form is a normal map. This well-known representation records surface details in terms of normals instead of geometry [8]. A high-resolution normal map allows the recreation of microstructure details. Unfortunately, in practice, a normal map based representation does not work well for high-frequency rendering. The reason is that we can hardly evaluate the $\mathcal{P}$-NDF from normals that are sampled from the normal map with methods such as importance sampling.

To support area-integration based $\mathcal{P}$-NDF evaluation, existing methods represent microstructure as piece-wise elements E such as discrete triangles, Gaussian lobes or spherical histograms. Each form has its own advantages and drawbacks.



[YHJ*14] binning

Fig. 7 Triangle representation compared to binning as a reference. The triangle representation keeps features very well. From Yan et al. [65]

Yan et al. [65] discretize the high-resolution position-normal distribution as a large number of triangles. Each triangle contains position and normal information. They then evaluate the $\mathcal{P}$-NDF by accumulating the contributions of triangles located in $\mathcal{P}$ (see Fig. 7). This representation is very accurate. However, since integration must be performed for each triangle element, evaluation is quite expensive.

To simplify evaluation, Yan et al. [66] define 4D Gaussian elements to describe the distribution of normals in one tiny area. They describe a glinty surface with a 4D position-normal distribution function, which is approximated by a mixture of millions of 4D Gaussians. The method requires a large amount of memory (1.7–1.9 GB for 2K×2K maps) to organize the Gaussians, but calculating Gaussians is much faster than integration over triangles. Many works [5, 59, 68] use similar elements to represent the microstructure.

Gamboa et al. [19] use a spherical histogram to represent microstructure. They discretize the microsurface as a 2D texture histogram. Each texel stores an element, which is an accumulated spherical histogram of normals. This method also requires a large amount of memory (2.3-2.7 GB for 2K×2K maps). In their implementation, they equally divide the longitudinal and azimuthal space according to texture size. At run-time, they introduce a summed area table (SAT) to compute the $\mathcal{P}$-NDF for an arbitrary range (see Fig. 9). Atanasov et al. [2] further introduce inverse bin maps (IBMs) which use constant memory (36MB) to store the inverses of histograms.

An issue of concern is that, in order to define the microstructure details producing glinty appearance, we usually need extremely high-resolution normal maps. For example, the normal map on the snail's shell in Fig. 3 has a resolution of 200K×200K. One suggestion is to generate the high-resolution microstructure on

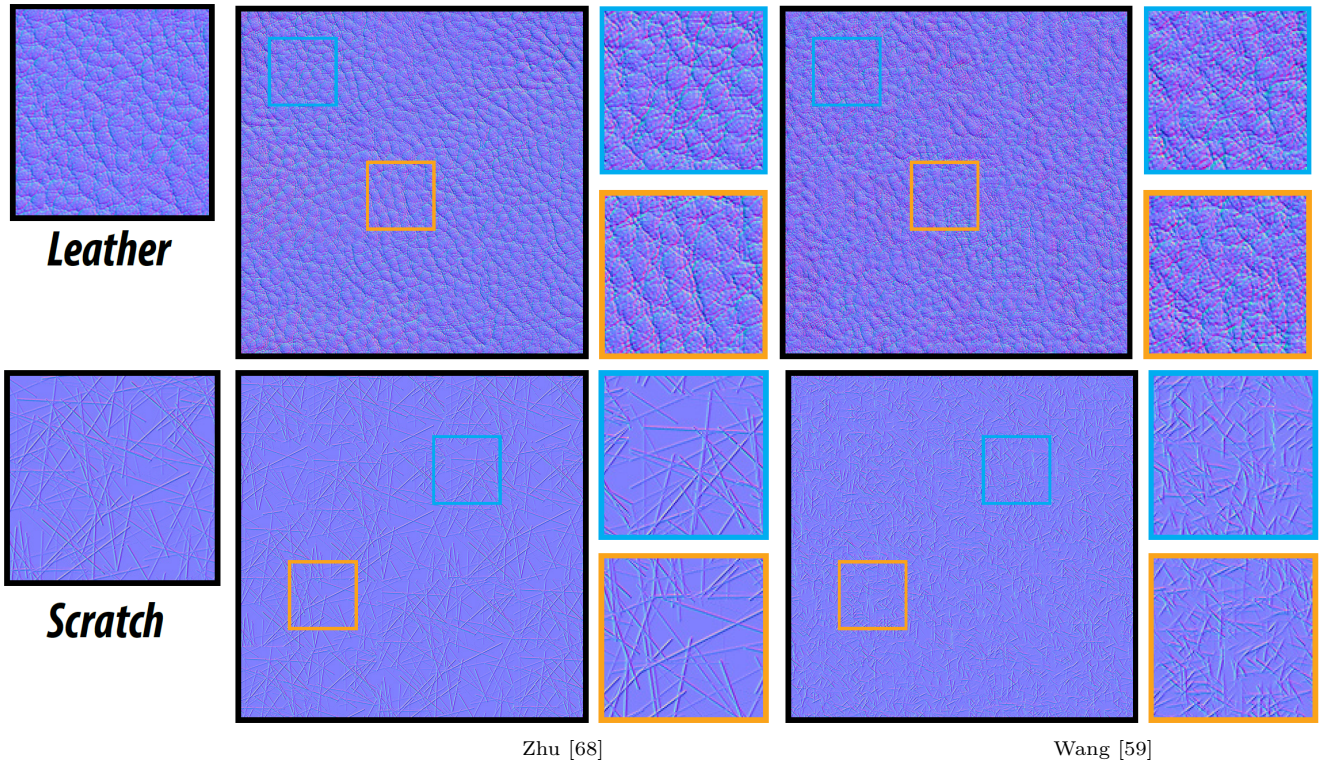


Fig. 8 Texture synthesis methods of Zhu et al. [68] and Wang et al. [59]. For leather, images synthesized by both methods are seamless and maintain leather features. However, for scratches, Wang's method produces blurry results.

the fly to reduce the memory cost. The inverse Fourier transform method [53] can generate tileable noise-like bumps. Texture synthesis methods can also turn normal maps into tileable patches, stitch patches and obtain high-resolution representations. Texture synthesis methods can be categorized into three different kinds: expansion [14, 15], blending [26] and tiling [9, 61]. The first two have been used for explicit representations. To be clear, even if synthesis methods generate representations on the fly, we still consider them to be explicit representations, as the input normal map is in an explicit form, and furthermore, the generation rules are general and do not rely on specific kinds of materials.

The expansion method [14, 15] extends a small texture into a new larger texture dynamically. Zhu et al. [68] apply the expansion idea to high-frequency rendering. They take a small-sized microstructure as a sample, repeatedly select blocks from the sample using some generation rules, and stitch the selected blocks together into a high-resolution microstructure. For continuity, they additionally generate some new Gaussian elements to fill seams between the selected blocks. This method requires only about 1% of the storage needed by the method of Yan et al. (see Fig. 8).

Blending methods [26] assume that any pixel in the resulting texture is a blend of several blocks sampled from the input texture. Wang et al. [59] apply blending-based texture synthesis in their work. They take small microstructure samples as input and generate Gaussian elements on the fly, using constant storage. However, the glinty effects can be blurred in some cases, especially for materials with scratches (see Fig. 8).

Existing synthesis-based methods are effective in solving memory problems. However, a common limitation is that they usually fail to keep global features. For example, when scratches are long, synthesis-based methods do not keep the global distribution of scratches.

3.2 Implicit Representation

Some studies try to generate glinty appearance through parameterized implicitly procedures. By using an implicit representation, one can generate infinitely large, non-repeating microstructures on the fly with little additional storage.

Procedural noise methods [18, 36, 46] use a few parameters to control the appearance of a noise function over an infinitely large space. These methods can generate random patterns controlled by

Tab. 1 Characteristics of representations. Element type: representation form in discrete elements. Storage: run-time memory for the material taking a 2K×2K normal map as input (or 512×512 for [68] and [59]). Appearance: representable appearance types. Multi-scale: whether the method supports multi-scale zoomed rendering. Tileable: whether the microstructure repeated when it is extremely large. Efficiency: element calculation speed, longer being more efficient. One dot: method is unlikely to finish in a reasonable time. Two dots: method is significantly slower than traditional BRDF methods. Three dots: method is slightly slower than traditional methods. Four dots: method is almost as fast as the traditional methods. Five dots: method runs in real-time.

	Method	Element Type	Storage	Appearance	Multi-scale	Tileable	Efficiency
Explicit	naïve	normal map	46.52 MB	all types	✗	✓	●○○○○
	[65]	triangle	1.5 GB	all types	✗	✓	●●○○○
	[66]	Gaussian lobe	1.76 GB	all types	✗	✓	●●●○○
	[5]	Gaussian lobe	2.1 GB	all types	✗	✓	●●●○○
	[68]	Gaussian lobe	102.0 MB	all types	✓	✗	●●●○○
	[59]	Gaussian lobe	35.0 MB	all types	✗	✗	●●●○○
	[19]	histogram	2.37 GB	all types	✓	✓	●●●●○
	[2]	histogram	36.0 MB	all types	✓	✓	●●●○○
Implicit	[30]	flake	N/A	glittery	✗	✗	●●●●○
	[69]	flake	N/A	glittery	✓	✗	●●●●●
	[48]	scratch	N/A	scratched surface	✓	✗	●●●●○

noise, such as bumps. Guo et al. [22] propose an implicit representation method for procedural material parameter estimation. They introduce a Bayesian inference approach using Hamiltonian Monte Carlo methods to sample the space of plausible material parameters, and fit procedural models to a range of materials such as wall plaster, leather, wood, anisotropic brushed metals and metallic paints. However, for glinty appearance rendering, we need not only the microstructure, but also a corresponding acceleration method to prune non-contributing regions. Unfortunately, none of these methods currently support queries in an arbitrary range.

So far, only two kinds of glinty appearances can be represented implicitly: glittery materials and scratched materials. Mirror-like flakes cause glittery effects. Unlike a general microfacet model, a glittery surface contains a collection of tiny and discrete flakes which are supposed to be defined using a non-smooth, spatially varying BRDF. Some methods [13, 16] regard the discrete flakes as random normals with positions. Gunther et al. [21] store the normals and positions of flakes to avoid flickering between frames but requires much memory to do so. Jakob et al. [30] represent the glittery material by stochastically generating flakes on the surface. They assume surfaces to be a collection of a specific set of randomly oriented facets and use a random index to store the count of flakes in a specific area and solid angle. This method supports range queries on the microstructure. Zirr and Kaplanyan [69] also model the glittery appearance as a set of implicitly represented flake elements. They derive a stochastic bi-scale model based on flake elements and implement this model in real-time.

For scratches, we need to consider two levels of microstructure distribution. The first level describes global scratch trajectories visible to the naked eye, defined as curves. The second level characterizes the microstructure profile of a single scratch considered as an element. Raymond et al. [48] model the microstructure profile for a single scratch as a multi-scale spatially varying BRDF. They use noise functions to generate the scratch distributions, with statistics determining the orientation and position of the scratch. For the range query, they adopt a simple idea, calculating the area occupied by a scratch element in a pixel. They further use the area and the BRDF of a single element to evaluate the contribution of a scratch element. However, their method does not take the cross section of two elements into consideration.

The above implicit representation methods can generate infinitely large parameterized microstructures of non-repeating patterns. However, a common challenge for such methods is how to efficiently integrate and evaluate the $\mathcal{P}$-NDF. Another limitation is that they can only represent a few kinds of glinty appearances, using specifically designed generation methods.

We summarize explicit and implicit representation methods in Tab. 1. Explicit methods can represent all kinds of glinty appearances but require large storage and are generally slower than implicit methods. Implicit methods can represent non-repeating large microstructures, require less computation and incur little storage overhead, but so far can only represent glittery and scratched materials. Some methods [59, 68] apply texture synthesis methods, which significantly reduces the storage. These methods can generate non-

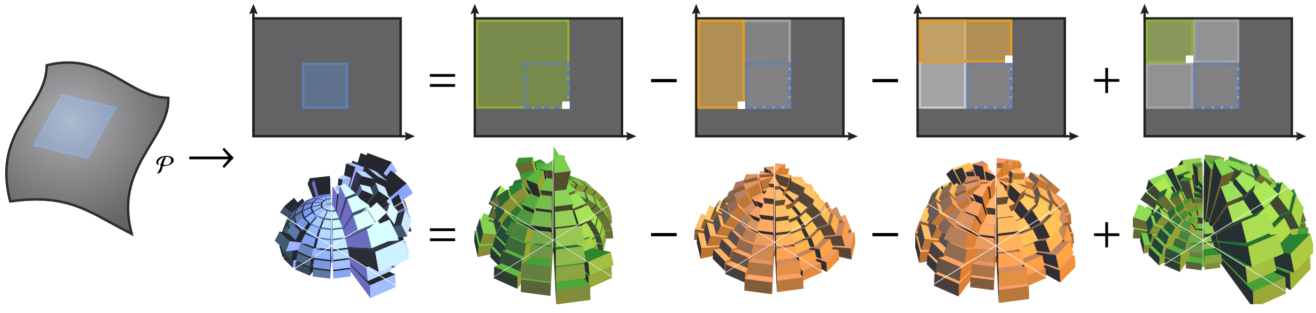


Fig. 9 Querying a multi-scale NDF histogram for an arbitrary filtering patch $\mathcal{P}$. The method accumulates the multi-scale NDF histogram and organizes it with a summed area table (below) for efficient querying. Image from Gamboa et al. [19].

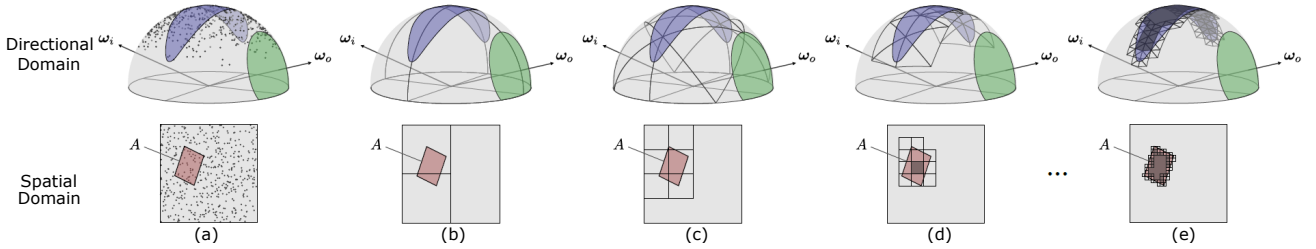


Fig. 10 Searching for discrete 4D flakes. (a): Initial state. The number of facets that lie in region A (red), scatter into a solid angle around ω_o (green) and hence point to the blue area are counted. (b–e): Breadth first searching process. Image from Jakob et al. [30]

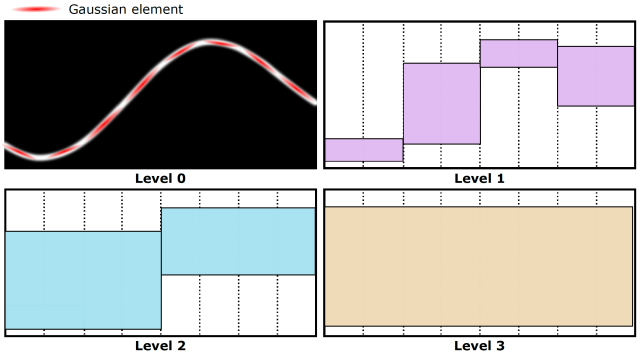


Fig. 11 Min-max hierarchy. Top left: Flatland visualization of a position-normal distribution, fitted with Gaussian elements. Others: Different levels of min-max hierarchy over the normal map that bounds the sets of normals within spatial ranges. Image from Wang et al. [60].

repeating large microstructures, but rendering is still slow.

4 Rendering Solutions

In this section, we consider how to integrate glinty appearance representations into the classic Monte Carlo path tracing pipeline. We discuss the evaluation step in Sec. 4.1 and importance sampling in Sec. 4.2. Multiple scattering is also an important part of path tracing, discussed in Sec. 4.3.

4.1 Evaluation

The $\mathcal{P}$-NDF integral can be defined in a general form as:

$$D_{\mathcal{P}}(\mathbf{s}) = \int_{-\infty}^{\infty} G_p(\mathbf{u}) G_r(n(\mathbf{u}) - \mathbf{s}) \mathrm{d}\mathbf{u}, \tag{2}$$

where $\mathbf{s}$ is a given direction to be queried. Within a pixel coverage G_p, we visit each microfacet at position $\mathbf{u}$ and use G_r to decide whether its normal $n(\mathbf{u})$ is close enough to the direction $\mathbf{s}$ of the query. r is the intrinsic surface roughness, used to define the value of closeness. In this way, we can evaluate the density for any direction $\mathbf{s}$ on $\mathcal{P}$.

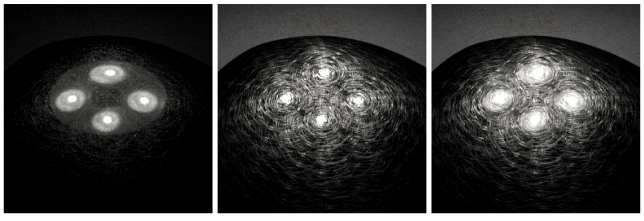
In order to compute the integral and evaluate the $\mathcal{P}$-NDF analytically, researchers use discrete piece-wise elements to describe the microstructure. There may be millions of elements in total in query patch $G_p(\mathbf{u})$ but only a few of them make a non-negligible contribution to the given query vector $\mathbf{s}$.

The $\mathcal{P}$-NDF can be rewritten in discrete form as:

$$D_{\mathcal{P}}(\mathbf{s}) \approx \sum_{i=1}^{m} G_p(\mathbf{u}_i) E_i(\mathbf{u}, \mathbf{s}), \tag{3}$$

where m is the number of elements. E_i is the contribution of the i^{th} element at position u and direction s. The weighted sum of these elements gives the desired value.

Evaluation in all methods with an explicit representation can be described by this equation.



(a) BRDF Sampling (b) Light Sampling (c) Combined

Fig. 12 Multiple importance sampling. (a): BRDF sampling captures the reflection of the light in flat areas, but is sub optimal for rendering scratches. (b): Light sampling captures illumination from high-intensity parts of HDR light texture onto scratches. (c) Combining them has the benefits of both. From Yan et al. [66].

Evaluation in methods with an implicitly represented NDF [30, 48] cannot be simplified by Alg. 3 as the NDF does not store the correspondence between position and normal. Raymond et al. [48] directly calculate the ratio of the scratched area to the patch area $\mathcal{P}$, and evaluate the NDF according to the ratio, scratch orientation, and measured BRDF.

Flake high-frequency material methods [30] generate a sequence of random numbers that represent the distribution of flakes. They then count particles that contribute to illumination without actually generating them. In their stochastic approach, random-flake approximation replaces the $\mathcal{P}$-NDF evaluation.

In general, an acceleration hierarchies is used. Figs. 10 and 11 illustrate two typical acceleration hierarchies for the query.

Yan et al. [1, 65] create a 4D bounding box for each element and build a min-max structure to organize these bounding boxes in a top-down manner (see Fig. 11). Given a rectangle bounding the patch Gaussian $G_p(\mathbf{u})$ and a cone bounding direction s as input, the method queries the hierarchy tree by top-down traversal to find contributing Gaussians.

The evaluation process in [30] is performed on a 4D search tree as shown in Fig. 10. Each tree node also contains a 4D bounding box defined as the Cartesian product of a bounding box in texture space and a spherical triangle in direction space. Each branch node is further split both in texture and in direction space. In texture space, the bounding box is cut into four equal-sized sub-boxes, and in direction space, the spherical triangle is cut into four sub-triangles by inserting vertices at the midpoints of the edges. The search proceeds alternately in spatial and in directional domains.

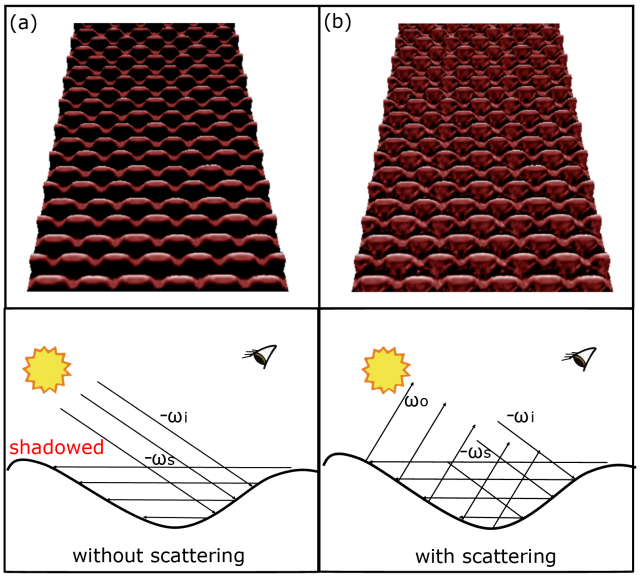
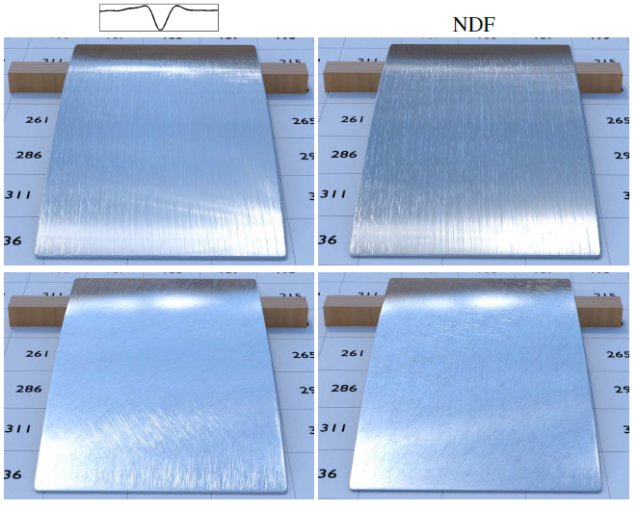


Fig. 13 Multiple scattering in microstructures can have a strong impact on the overall appearance. Above: rendered image, (a) without self-scattering, and (b) including self-scattering effects. Below: schematic diagram showing how light interacts with the geometry.

4.2 Importance Sampling

Fig. 12 shows a multiple importance sampling experiment in which a scratched surface is illuminated with a textured area light. The reflection from flat parts of the surface can easily be rendered by BRDF sampling (a), but rendering scratches relies on sampling and evaluating the $\mathcal{P}$-NDF in the light direction (b). A combined image (c) has the benefits of both.

We can sample the underlying normal map directly, or the discrete elements of $\mathcal{P}$-NDF. Yan et al. [65] take the normal of a random surface point and perturb the normal by the intrinsic roughness value. This only works for normal map based representations. Yan et al. [66] also sample from the discrete Gaussian elements. They select a Gaussian element proportional to its contribution to the patch and then pick a normal from that element. Their sampling method works well without the input normal maps. Raymond et al. [48] sample directions around $\omega_\mathbf{i}$ which are randomly generated within the reflection cone following a 1D probability distribution function based on the mirror scratch BRDF. Instead of sampling the actual $\mathcal{P}$-NDF, Jakob et al. [30] define a smooth density function to describe the distribution of flakes and use for sampling. The advantage is that they do not generate all elements reducing time and space costs. However, the function is more capable of describing surfaces with discrete flakes.



(a) With multiple scattering (b) Without multiple scattering

Fig. 14 Scratched metal rendered with and without multiple scattering. Left: metal modeled with real geometry yields stronger contrasts and reflections due to multiple scattering. Right: the same metal modeled with NDF appears less bright, since scratches reflect little or none of the incoming radiance when higher-order bounces are neglected. From Raymond et al.[48]].

4.3 Multiple Scattering

By multiple scattering, in this survey, we mean self-scattering between microstructures. Modeling multiple scattering in glinty appearance rendering is considered to be an important open problem, since a non-negligible fraction of the energy leaving the surface occurs due to paths with multiple reflections. Fig. 13 illustrates the principle of multiple scattering. Fig. 14 shows how, for materials with scratches for example, the results will be less bright if multiple scattering is neglected.

While considering multiple scattering produces better results (see Figs. 13 and 14), it also requires much more computation and storage. Only a few investigations of glinty appearance rendering integrate multiple scattering into their solutions.

Raymond et al. [48] randomly distribute scratches on a surface and users can modify the profiles, orientations and density of the scratches. They simulate multiple-scattering by pre-computing distributions of scratch profiles. Their method gives realistic results but only works for scratched surfaces. Chermain et al. [5] derive an energy-compensation BRDF to compensate for the energy lost by single scattering. Their method leverages a local energy preserving BRDF by faking normal perturbations. Turquin [56] deduce a compensated $\mathcal{P}$-BRDF for glinty appearance, and produce noticeably improved results.

Such methods are results-based, and deriving a physically-based analytical multiple scattering model is an open problem in glinty appearance rendering.

5 Experiments and Analysis

In this section, we first define quality criteria for glinty appearance rendering methods. Then, we introduce our unified experimental platform that integrates dozens of recent glinty appearance rendering methods. We show some methods' results produced by the platform and compare them according to our criteria.

5.1 Quality Criteria

Verifying complex surface rendering quality can be difficult due to the lack of mathematical criteria and the subject nature of human perception.

A direct approach to verifying the 'correctness' of rendering results is to render texture-based glinty appearances with an extremely high number of samples per pixel (spp) and to consider the image result at convergence as 'ground truth' or reference. Comparing the rendered results to the reference, we may estimate the degree of degradation. We render the reference images by brute force path tracing. How can we determine whether an image has converged or not? We can assume that the image converges when there is no significant change (when the RMSE of two images is smaller than 0.03) as we increase the number of samples. The time needed to compute such a converged image is about 14–20 hours.

Another effective term to model glinty appearance is NDF. Thus, we adopt the correctness of the NDF into our quality criteria. Designing a metric suitable for comparing a model's NDF to the original normal distribution is necessary. In practice, we find comparing a visual comparison of the $\mathcal{P}$-NDF to the reference is adequate to represent correctness.

In addition to rendering quality, rendering efficiency, memory consumption, and versatility are also useful criteria to assess the strengths and weaknesses of different methods. Energy conservation can be verified by a white furnace test, but since most existing techniques are not energy-conserving, we do not discuss this much further.

A further issue is that researchers and artists should not simply judge a method as good or bad based on the quality criteria alone. Instead, they should remember that different methods are applicable to different rendering requirements. Implicit rendering methods [30, 48] are suitable when rendering scenes with glittery or scratched appearances, due to their good

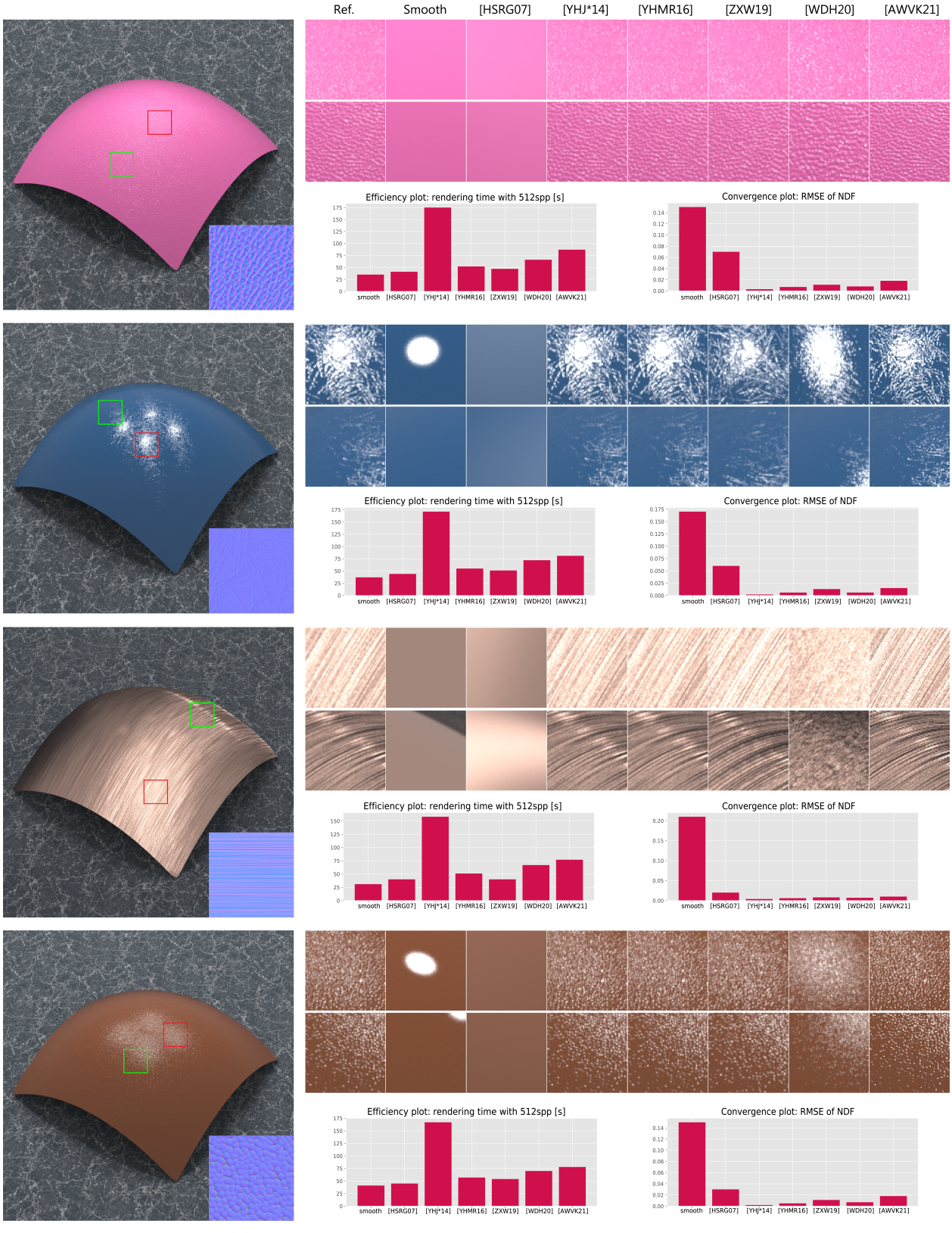


Fig. 15 Four glinty appearances rendered on the Bent Quad scene: leather, a scratched coating, brushed metal, and bumpy plastic, using seven different methods on our unified experimental platform; the reference is also shown. Tables to the right of each scene compare rendering time and correctness of the $\mathcal{P}$-NDF for different methods. The sizes of microstructures are as given in Tab. 1.

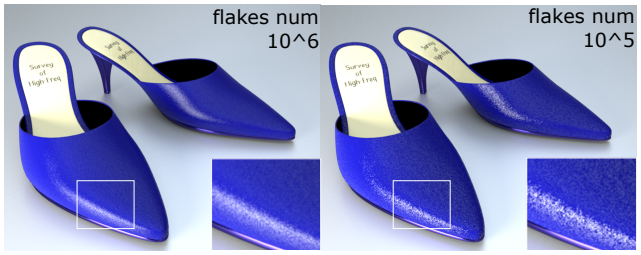


Fig. 16 Fancy shoes with glints rendered using the method in [30] and 1024 spp. Left: 10^6 flakes, rendering time = 3.2 minutes. Right: 10^5 flakes, rendering time = 1.7 minutes.

performance, but they have limited expressiveness. We need explicit rendering methods for other more complex scenes. Even so, there are big differences between different explicit rendering methods. For example, the method of [66] can accurately render scenes with complex glinty appearances. If the light sources are not sharp, it is more efficient to use methods in [12, 19]. When there are many glinty appearances in a scene, the LOD method of [68] is efficient and storage-saving.

5.2 Unified Experimental Platform

To evaluate methods according to the quality criteria for glinty appearance rendering technologies defined above, we built a unified experimental platform based on the Mitsuba framework [29]. It:

- implements more than a dozen advanced offline glinty appearance rendering methods,
- provides RMSE and HDR visual difference maps to compare different rendering results,
- provides NDF visualization and quantitative NDF statistics,
- provides example scenes (leather shoes, a wooden ball and a scratched kettle), including geometric models, scene files, glint features and a converged reference for comparison,
- collects statistics of memory and rendering time.

Using the platform, researchers can easily compare their methods with previous work, and artists can quickly select methods that meet their needs. Our platform only supports comparison of offline methods.

5.3 Results

In Fig. 15, we compare typical explicit glinty appearance rendering methods on our unified experimental platform in terms of rendering effects, accuracy of $\mathcal{P}$-NDF and rendering time. All timings in this section were measured on a 2.20 GHz Intel Xeon with 22 cores and 128 GB of memory. We consider four glinty appearances: leather, a scratched coating, brushed metal, and bumpy plastic, on the simple bent quad geometry, illuminated by an environment map and four tiny area lights (which can be considered to be point lights). All timings are for pictures with 1024×1024 pixels.

First, let us observe the rendering results. Compared to the reference, the traditional microfacet model and the pre-filtering method [23] fail to render the glinty effect, while results from methods in [65, 66] are very close to the reference. Atanasov et al. [2] capture glint effects, but compared to the reference, their results differ somewhat in close up. For the texture synthesis methods, except for slight discontinuities on the brushed metal, the results of Zhu et al. [68] are visually identical to the reference; the results of Wang et al. [59] look blurry for the scratched coating, brushed metal and the bump plastic scenes.

In terms of rendering speed, the method of [65] is significantly slower than other methods. Yan et al. [66] take only about 1.4× as long as standard microfacet BRDF rendering. [59, 68] represent the microstructure as Gaussian elements like [66]. Zhu et al. [68] use a clustering method, so their method is slightly faster than that of [66]. Wang et al. [59] generate the Gaussian elements on the fly, so their method is slower than that of [66]. [2] takes a longer time than [59] because their methods have higher computational complexity.

We also compare the correctness of the $\mathcal{P}$-NDF. In practice, when the RMSE is smaller than 0.01, the $\mathcal{P}$-NDF can be considered correct. We can observe that all glinty appearance rendering methods compared correctly evaluate the $\mathcal{P}$-NDF. Correctness of the $\mathcal{P}$-NDF is also an essential requirement of a glinty appearance rendering method.

Fig.16 illustrates rendering results of the method developed by Jakob et al. [30]. These methods are efficient and require no additional storage space; rendering speed is correlated with the number of flakes, and they only need to store a few parameters.

6 Extensions

So far, we have presented practical methods for rendering surfaces with glint features accurately and efficiently, and showed they are able to render credible results. In theory, using these methods, the rendering results should exactly match the actual appearance, but we still require better speed and realism.

Several techniques and applications leverage or extend the idea of glinty appearance rendering for broader applications and more realistic rendering. This

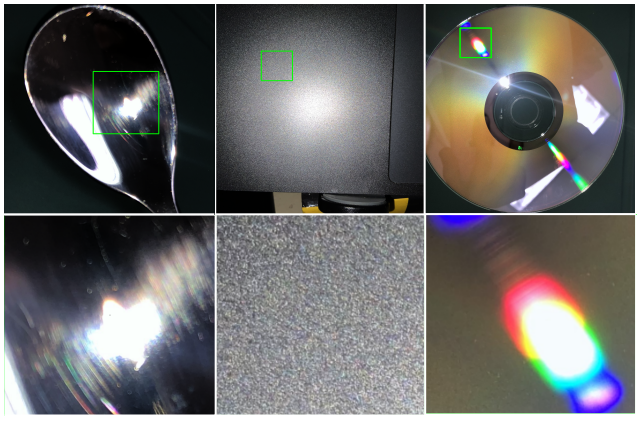


Fig. 17 Left to right: a spoon with iridescent scratches, metallic paint coating a laptop, and a CD with colorful anisotropic highlights. Above: photographs taken with a smartphone, with flash. Below: close-ups.

section provides a brief overview of these areas.

6.1 Wave Optics

When we look at actual photographs (see Fig. 17), we can observe that there may be colored glint features even when a white light source illuminates the object. This is an interesting phenomenon, and is not possible in traditional geometric optics, which only produces white highlights from a white light source. This phenomenon is explained by wave optics.

In wave optics, light is described by complex-valued fields. Scalar diffraction models, such as those proposed by Harvey-Shack [24, 34] or Kirchhoff [40, 44], can be used to estimate the reflected field from a rough surface. For scratches, Werner et al. [62] derive a wave-optical and analytical shading model based on Harvey-Shack theory [24], where the surface is represented as a collection of randomly oriented scratches over a smooth BRDF. This work is further extended to real-time by Velinov et al. [58]. For mirror flakes, Guo et al. [32] extend the stochastic model [30] to take wave-optical effects due to thin-film interference into account, reproducing iridescent reflection. Yan et al. [67] present a solution to derive a wave effect-aware BRDF model on surfaces described as a discretized height field. The BRDF model is estimated by simulating diffraction effects of coherent light over the corresponding area on the height field, allowing their method to support arbitrary glint features. While capturing wave effects accurately, wave optics glinty appearance rendering methods are about one order of magnitude slower than those using geometric optics; further acceleration should be explored.

6.2 Machine Learning Methods

There are some glinty appearance-related studies in the field of machine learning. They are more concerned with the representation of details than improving the actual rendering process. We introduce them in this section as a complement to previous representation approaches.

Again, we classify machine learning-based methods in two categories: one uses an inverse model to provide an explicit representation, the other uses a procedural model for an implicit one.

The inverse model basically estimates spatial varying material parameters (diffuse albedo, roughness and normal for a microfacet model) from measured data like images and derives per-pixel BRDFs inversely, as an SVBRDF. These parameters are stored explicitly in a texture. Chandraker et al. [4] utilize motion cues to jointly recover shapes and BRDFs of objects from images. Hui et al. [27] recover SVBRDFs and shape from multiple images taken with a fixed viewpoint and varying illumination. Riviere et al. [49], Hui et al. [28], and Li et al. [37] propose SVBRDF recovery algorithms under a simpler setup consisting of a camera and a flashlight, as commonly found in mobile devices. Li et al. [38] improve the approach to handle larger inputs. Gao et al. [20] present a unified framework for estimating SVBRDFs.

Generally, the inverse model heavily relies on a specific reflection model, which, in most cases, is the microfacet model, and which is not suitable for glinty appearance rendering as explained in Sec. 2.1. In order to obtain an SVBRDF at microscale resolution, Nam et al. [42] propose a microscopic material acquisition system. They use machine learning to compress the vast amounts of data. While different reflection models could be used, this approach cannot handle advanced complex models like the wave optical one introduced in Sec. 6.1.

The procedural model learns glint features by fitting an implicit pattern distribution instead of explicitly estimating parameters everywhere on the surface. Guo et al. [22] focus on procedural material parameter estimation by employing a Bayesian framework and using an image classification neural network (usually VGG) as a descriptor. Their procedural material models generally consist of a microfacet SVBRDF and an explicitly constructed procedural height field (usually a noise texture). Kuznetsov et al. [35] propose a procedural model and use a conditional generative adversarial network to learn to generate $\mathcal{P}$-

NDF for every location with a fixed patch $\mathcal{P}$. Using a relatively lightweight GAN network, their procedural model can reproduce a $\mathcal{P}$-NDF close to the ground truth with lower storage due to its implicit nature, while supporting traditional geometric optics and wave optics. Moreover, this serves as a precomputation for evaluating the $\mathcal{P}$-NDF integral (Alg. 2) and speeds up evaluation when rendering with wave optics compared to the method proposed by Yan et al. [67]. However, their work is limited to a fixed patch size and may introduce artifacts in the near-field and grazing angles where the size of $\mathcal{P}$ varies dramatically.

6.3 Real-time Rendering

We have mainly discussed offline glinty appearance rendering methods above. It is not easy to apply offline glinty appearance models to real-time rendering. The first reason is that GPU and CPU architectures differ, so offline acceleration structures cannot be applied to real-time rendering. The second reason is the small video memory of the GPU: the storage required for explicitly represented materials is large.

Current real-time research work mainly deals with implicitly represented materials. Zirr et al. [69] propose a stochastic bi-scale microfacet model for real-time rendering of multi-scale glint features including discrete flakes and brushed marks. Wang et al. [59] propose a pre-filtering method for the stochastic discrete microfacet model to simulate glints under both environment maps and point light sources in real-time. Velinov et al. [58] treat scratches under wave optics. Chermain et al. [7] propose a method of rendering flakes in real-time. They use mip-map structures to speed up rendering. They further propose an anti-aliasing method [6] in real-time. However, no existing real-time solutions can consistently process glinty appearances explicitly defined by high-resolution normal maps, limiting the variety of glinty appearances in real-time rendering. The limited video memory available in real-time rendering makes it difficult to store complete mapping-based glinty appearance information. Also, texture synthesis methods are challenging to implement and accelerate reasonably well in real-time rendering.

7 Conclusions and Future Work

Glinty appearance methods have made promising steps toward modeling and rendering visual appearances with real-world complexity. In this survey, we have discussed recent advances in glinty appearance rendering by broadly categorizing approaches based on *representation* and *practical rendering*.

Glinty appearances can be represented in explicit or implicit form, but both explicit and implicit representation have in common that they take discrete forms. Explicit representation methods have been intensively studied because they can represent all kinds of glinty appearances like bumps, flakes, scratches, leather patterns and dents. However, a common problem for explicit representations is that they require large additional storage to store all microstructure information. Texture synthesis-based methods reduce the storage to a certain extent by generating the large microstructure from small samples. Implicit representations can represent microstructures of unlimited size while requiring little additional storage. As a downside, such methods are not general: existing implicit representations can only represent glittery and scratched materials.

For practical rendering, we have distinguished three sub-problems, *evaluation, importance sampling* and *multiple scattering*. Acceleration hierarchies are used to quickly prune non-contributing parts to efficiently evaluate the glinty appearance. During evaluation, methods with explicit representations are generally slower than those with implicit representations. Because the $\mathcal{P}$-NDF is known, all methods can easily perfectly importance sample the $\mathcal{P}$-NDF. Sometimes, multiple scattering has non-negligible effects on glinty appearance. Existing methods precompute the multiple-scattered BRDF or adopt energy compensation methods. However, these methods are experience-based and usually oversimplify multiple scattering evaluation.

We have also discussed several open problems and potential future research directions in this area. The conflict between representation ability, rendering efficiency and memory is still the core problem in glinty appearance rendering. Explicitly represented methods require more storage and longer rendering time. Implicitly represented methods are not general and can only represent a few appearance types. One solution is to deduce a general implicit model which could represent all types of appearances while incurring no storage, and efficiently evaluating the $\mathcal{P}$-NDF during rendering. Although it is far from easy to settle the range query problem of implicit representation methods, we believe this to be an area that provides potential advantages for further practical applications. Another practical solution is to find a better discrete element form providing easy calculation in explicitly represented methods.

Apart from solving existing problems, new problems are to be found. For example, the glinty appearance rendering of volumes is a more difficult integral problem than glinty surface appearance rendering. Another problem is denoising algorithms. How can we distinguish glint features from noise? All these problems and challenges provide new research opportunities and may stimulate deeper observations and explorations.

Acknowledgements

This work was partially supported by the National Key R&D Program of China under grant No.2020YFB1709200, the National Natural Science Foundation of China under grant No.61872223 and the Shandong Provincial Natural Science Foundation of China under grant No.ZR2020LZH016.

Funding or Conflicts of interests

We declare that we do not have any commercial or associated interest that represents a conflict of interest in connection with the work submitted.

References

[1] A. Atanasov and V. Koylazov. A practical stochastic algorithm for rendering mirror-like flakes. In *ACM SIGGRAPH 2016 Talks*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery.

[2] A. Atanasov, A. Wilkie, K. Vladimir, and J. Krivanek. A Multiscale Microfacet Model Based on Inverse Bin Mapping. *Comput. Graph. Forum (Proc. Pacific Graphics)*, 40(7), 2021.

[3] E. Brusneton and F. Neyret. A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics*, 18(2):242–260, 2011.

[4] M. Chandraker. On shape and material recovery from motion. In *European Conference on Computer Vision*, pages 202–217. Springer, 2014.

[5] X. Chermain, F. Claux, and S. Mérillou. Glint rendering based on a multiple-scattering patch brdf. *Computer Graphics Forum*, 38(4):27–37, 2019.

[6] X. Chermain, S. Lucas, B. Sauvage, J.-M. Dischler, and C. Dachsbacher. Real-time geometric glint anti-aliasing with normal map filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(1):1–16, 2021.

[7] X. Chermain, B. Sauvage, D. Jean-Michel, and C. Dachsbacher. Procedural Physically-based BRDF for Real-Time Rendering of Glints. *Comput. Graph. Forum (Proc. Pacific Graphics)*, 39(7), 2020.

[8] P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno. A general method for preserving attribute values on simplified meshes. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 59–66. IEEE, 1998.

[9] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. *ACM Transactions on Graphics (TOG)*, 22(3):287–294, 2003.

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

[11] K. J. Dana, B. Van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions On Graphics (TOG)*, 18(1):1–34, 1999.

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

[13] R. Ďurikovič and W. L. Martens. Simulation of sparkling and depth effect in paints. In *Proceedings of the 19th spring conference on Computer graphics*, pages 193–198, 2003.

[14] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001.

[15] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[16] S. Ershov, K. Kolchin, and K. Myszkowski. Rendering pearlescent appearance based

on paint-composition modelling. *Computer Graphics Forum*, 20(3):227–238, 2001.

[17] A. Fournier. Normal distribution functions and multiple surfaces. In *Graphics Interface '92 Workshop on Local Illumination*, pages 45–52, Vancouver, BC, Canada, 11 May 1992.

[18] B. Galerne, A. Leclaire, and L. Moisan. Texton noise. *Computer Graphics Forum*, 36(8):205–218, 2017.

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

[20] D. Gao, X. Li, Y. Dong, P. Peers, K. Xu, and X. Tong. Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM Transactions on Graphics (TOG)*, 38(4):134–1, 2019.

[21] J. Günther, T. Chen, M. Goesele, I. Wald, and H.-P. Seidel. Efficient acquisition and realistic rendering of car paint. In *Vision, Modeling, and Visualization*, volume 5, pages 487–494. Akademische Verlagsgesellschaft Aka, 2005.

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

[23] C. Han, B. Sun, R. Ramamoorthi, and E. Grinspun. Frequency domain normal map filtering. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 28–es, New York, NY, USA, 2007. Association for Computing Machinery.

[24] J. E. Harvey. Fourier treatment of near-field scalar diffraction theory. *American Journal of Physics*, 47(11):974–980, 1979.

[25] E. Heitz, J. Dupuy, C. Crassin, and C. Dachsbacher. The sggx microflake distribution. *ACM Transactions on Graphics (TOG)*, 34(4CD):1–11, 2015.

[26] E. Heitz and F. Neyret. High-performance by-example noise using a histogram-preserving blending operator. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2):1–25, 2018.

[27] Z. Hui and A. C. Sankaranarayanan. A dictionary-based approach for estimating shape and spatially-varying reflectance. In *2015 IEEE International Conference on Computational Photography (ICCP)*, pages 1–9. IEEE, 2015.

[28] Z. Hui, K. Sunkavalli, J.-Y. Lee, S. Hadap, J. Wang, and A. C. Sankaranarayanan. Reflectance capture using univariate sampling of brdfs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5362–5370, 2017.

[29] W. Jakob. Mitsuba renderer, 2010.

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

[31] H. W. Jensen. *Global illumination using photon maps*, volume 22. Springer, 1996.

[32] G. Jie, C. Yanjun, G. Yanwen, and P. Jingui. A physically-based appearance model for special effect pigments. *Computer Graphics Forum*, 37(4):67–76, 2018.

[33] J. T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[34] A. Krywonos. *Predicting surface scatter using a linear systems formulation of non-paraxial scalar diffraction*. PhD thesis, University of Central Florida, 2006.

[35] A. Kuznetsov, M. Hasan, Z. Xu, L.-Q. Yan, B. Walter, N. K. Kalantari, S. Marschner, and R. Ramamoorthi. Learning generative models for rendering specular microgeometry. *ACM Transactions on Graphics (TOG)*, 38(6):225–1, 2019.

[36] A. Lagae, S. Lefebvre, G. Drettakis, and P. Dutré. Procedural noise using sparse gabor convolution. *ACM Transactions on Graphics (TOG)*, 28(3):1–10, 2009.

[37] X. Li, Y. Dong, P. Peers, and X. Tong. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

[38] Z. Li, K. Sunkavalli, and M. Chandraker. Materials for masses: Svbrdf acquisition with a single mobile phone image. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 72–87, 2018.

[39] W.-C. Ma, S.-H. Chao, Y.-T. Tseng, Y.-

Y. Chuang, C.-F. Chang, B.-Y. Chen, and M. Ouhyoung. Level-of-detail representation of bidirectional texture functions for real-time rendering. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 187–194, 2005.

[40] B. Mityashev. The scattering of electromagnetic waves from rough surfaces *1p. beckman and a. spizzichino, oxford — london — new york — paris. *Ussr Computational Mathematics & Mathematical Physics*, 4(6):247–249, 1964.

[41] K. Murat. A survey of bsdf measurements and representations. *Journal of Science and Engineering*, 20(58):87–102, 2018.

[42] G. Nam, J. H. Lee, H. Wu, D. Gutierrez, and M. H. Kim. Simultaneous acquisition of microscale reflectance and normals. *ACM Transactions on Graphics (TOG)*, 35(6):185, 2016.

[43] F. Neyret. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):55–70, 1998.

[44] Ogilvy and J. A. Theory of wave scattering from random rough surfaces. *Journal of the Acoustical Society of America*, 90(6):2332, 1991.

[45] M. Olano and D. Baker. Lean mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 181–188, 2010.

[46] K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[47] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[48] B. Raymond, G. Guennebaud, and P. Barla. Multi-scale rendering of scratched materials using a structured sv-brdf model. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016.

[49] J. Riviere, P. Peers, and A. Ghosh. Mobile surface reflectometry. *Computer Graphics Forum*, 35(1):191–202, 2016.

[50] F. Suykens, K. Berge vom, A. Lagae, and P. Dutré. Interactive rendering with bidirectional texture functions. *Computer Graphics Forum*, 22(3):463–472, 2003.

[51] P. Tan, S. Lin, L. Quan, B. Guo, and H. Shum. Filtering and rendering of resolution-dependent reflectance models. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):412–425, 2008.

[52] P. Tan, S. Lin, L. Quan, B. Guo, and H.-Y. Shum. Multiresolution reflectance filtering. In *Computer Graphics Forum*, pages 111–116, 2005.

[53] J. Tessendorf. Simulating ocean water. *SIGGRAPH'99 Course Note*, 01 2001.

[54] M. Toksvig. Mipmapping normal maps. *journal of graphics tools*, 10(3):65–71, 2005.

[55] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Josa*, 57(9):1105–1114, 1967.

[56] E. Turquin. Practical multiple scattering compensation for microfacet models. *URL: https://blog. selfshadow. com/publications/turquin/ms_ comp_final. pdf*, 45, 2019.

[57] E. Veach. *Robust Monte Carlo methods for light transport simulation*, volume 1610. Stanford University PhD thesis, 1997.

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

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

[60] B. Wang, M. Haan, N. Holzschuch, and L. Q. Yan. Example-based microstructure rendering with constant storage. *ACM Transactions on Graphics (TOG)*, 2020.

[61] H. Wang. Proving theorems by pattern recognition—ii. *Bell system technical journal*, 40(1):1–41, 1961.

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

[63] H. Wu, J. Dorsey, and H. Rushmeier. Characteristic point maps. *Computer Graphics Forum*, 28(4):1227–1236, 2009.

[64] H. Wu, J. Dorsey, and H. Rushmeier. Physically-based interactive bi-scale material design. *ACM Transactions on Graphics (TOG)*, 30(6):1–10, 2011.

[65] L.-Q. Yan, M. Hašan, W. Jakob, J. Lawrence, S. Marschner, and R. Ramamoorthi. Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Transactions on Graphics (TOG)*, 33(4):1–9, 2014.

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

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

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

[69] T. Zirr and A. S. Kaplanyan. Real-time rendering of procedural multiscale materials. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 139–148, 2016.

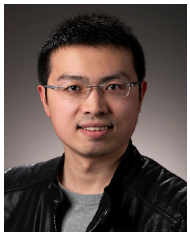
Junqiu Zhu is a Ph.D. candidate at Shandong University, China working under the supervision of Prof. Xiangxu Meng. Her research is in physically-based rendering, realtime ray tracing, and realistic appearance modeling.

Sizhe Zhao is a master's degree student at Shandong University. He received his bachelor's degree from the School of Energy and Power Engineering at Huazhong University of Science and Technology in 2021. His research interest is in computer graphics.

Yanning Xu is an associate professor in the School of Software, Shandong University. He received his Ph.D. degree from Shandong University in 2006. His research interests include photorealistic rendering and high performance rendering.

Xiangxu Meng is a professor in the School of Software, Shandong University. He obtained his Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Science, in 1998. His research covers industrial design, product design, digital media, software services and other applications, human-computer interaction, computer graphics theory and methods, virtual reality and virtual prototyping, grid computing and service computing, manufacturing of information technology, and other areas of theoretical research and system development.

Lu Wang is a professor in the School of Software, Shandong University. She received her Ph.D. degree from Shandong University in 2009. Her research interests include photorealistic rendering and high performance rendering.



Ling-Qi Yan is an assistant professor of computer science at UC Santa Barbara, co-director of the MIRAGE Lab, and affiliated faculty in the Four Eyes Lab. Before that, he received his doctoral degree from the Department of Electrical Engineering and Computer Sciences at UC Berkeley and obtained his bachelor's degree in computer science from Tsinghua University. His research interests include physically-based rendering, realtime ray tracing, and realistic appearance modeling.