# Discrete Stochastic Microfacet Models

Wenzel Jakob[1] Miloš Hašan[2] Ling-Qi Yan[3] Jason Lawrence[4] Ravi Ramamoorthi[3] Steve Marschner[5]
[1]ETH Zürich [2]Autodesk [3]University of California, Berkeley [4]University of Virginia [5]Cornell University

**Figure 1:** ***Left****: fancy pair of women's dress shoes with a glittery finish modeled using our discrete microfacet BRDF.* ***Right****: Christmas ornaments illustrating a range of model parameters including different particle counts, surface roughness, and anisotropy.*

## Abstract

This paper investigates rendering glittery surfaces, ones which exhibit shifting random patterns of glints as the surface or viewer moves. It applies both to dramatically glittery surfaces that contain mirror-like flakes and also to rough surfaces that exhibit more subtle small scale glitter, without which most glossy surfaces appear too smooth in close-up. These phenomena can in principle be simulated by high-resolution normal maps, but maps with tiny features create severe aliasing problems under narrow-angle illumination. In this paper we present a stochastic model for the effects of random subpixel structures that generates glitter and spatial noise that behave correctly under different illumination conditions and viewing distances, while also being temporally coherent so that they look right in motion. The model is based on microfacet theory, but it replaces the usual continuous microfacet distribution with a discrete distribution of scattering particles on the surface. A novel stochastic hierarchy allows efficient evaluation in the presence of large numbers of random particles, without ever having to consider the particles individually. This leads to a multiscale procedural BRDF that is readily implemented in standard rendering systems, and which converges back to the smooth case in the limit.

**CR Categories:** I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

**Keywords:** Stochastic reflectance model, glitter, sparkle, glint, spherical conic section, microfacet BRDF

**Links:** DL PDF WEB VIDEO CODE

## 1 Introduction

Many surfaces have a glittery appearance, characterized by bright glints that suddenly appear and disappear with changes in the lighting and view directions. Obvious examples include materials designed to glitter, containing small mirror-like flakes distributed on the surface (craft glitter used in decorations, glitter in eye makeup) or below the surface (metallic car paint). Many natural materials also look glittery due to surface or internal reflections from crystals (mica flakes in rock, ice crystals in snow or frost).

But literal mirror flakes are not actually necessary to create a glittery appearance. The random surface features on roughened shiny surfaces (bead blasted aluminum, fine-textured plastic, matte finish photo paper) also produce glints that come and go with changes in reflection geometry. Indeed, a smooth BRDF is only a convenient idealization of a statistical process that is assumed to happen at a scale much smaller than pixels. The model proposed in this paper is based on the idea of discrete mirror flakes, but it can also model the material appearance due to features on smoother surfaces.

To understand the behavior of glitter patterns, consider a surface sprinkled with flakes and illuminated by an area source. Each flake has a position and a normal, and some reflect the light to produce glints. This happens when a flake's position is inside the pixel's footprint on the surface and its normal vector is halfway between the view direction and the direction to some point on the light. Thus the contribution of glitter to a pixel is determined by the number of

flakes whose position lies in a particular area of surface and whose normal also lies in a particular solid angle—we are counting points in a volume of the 4D space of surface positions and flake normals.

The appearance of glitter in the image depends on the average number of points found. For example, on an ornament covered in craft glitter, there are relatively few flakes with normals broadly distributed, so if the light source is small the average number of glints per pixel is much less than 1, leading to occasional, widely separated glints. In a metallic car paint under the same lighting, flakes are more numerous, so there are many glints in every pixel, resulting in a spatial noise pattern rather than individually distinguishable points. With the same car under an overcast sky there will be orders of magnitude more glints per pixel, so the relative variation in their number will be low and the surface will appear smooth.

The appearance of glitter also changes with viewing distance (pixels in closer views see fewer particles, resulting in more dramatic glitter), with the area density of particle positions (fewer particles per unit area looks more glittery, more particles looks smoother), and with the density of flake normals (more aligned flakes look smoother). Temporal appearance is another very important aspect of glitter: with smooth motion of camera or object, each glint will appear, persist for a time (dependent on the lighting as well as the speed of motion), and then disappear.

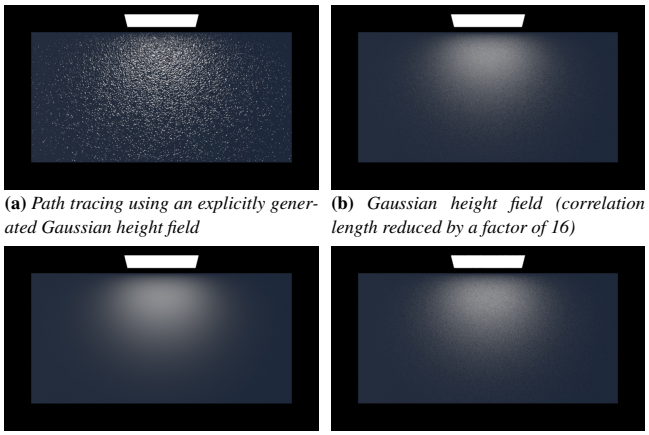
One approach to rendering glittery surfaces is to use high resolution normal maps. However, when the details are much smaller than pixels, this can create difficulties with pixels containing very tiny bright highlights that standard antialiasing methods can only discover by chance. On the other hand, when all structure is subpixel, the expressive power of a normal map is unnecessary (Figure 2).

In this paper we introduce a new kind of reflectance model that creates a glittery appearance without the need to model surface details explicitly. Like the microfacet model, we treat a surface as a collection of microscopic facets. But instead of assuming an infinite collection of facets, leading to a smooth and invariant BRDF, we use a fixed, finite collection of flakes, leading to a non-smooth and spatially varying BRDF. A different specific set of flakes is seen at each pixel, giving rise to the characteristic glints.

A simple but impractical way to implement our model would be to generate such a collection, each flake having a random position and normal, and store them in a list. Then, to decide how many glints lie in a pixel, simply check each flake to see if it lies in the pixel and reflects the light. This will produce the correct kind of spatial patterns, and when the camera or object moves over time, glints will appear and disappear at the right rate as individual flakes start and stop reflecting the source. This search for contributing flakes could be accelerated with a 4D position-normal hierarchy, but it would still take too much memory to store the particles and the hierarchy.

A more practical approach for still frames is to analytically work out the statistical distribution of per-pixel flake counts under given viewing and illumination conditions. Since disjoint regions on the surface are uncorrelated, the flake counts for each pixel could be generated independently, without generating the flakes and counting them. This is far more efficient, but it cannot be used for animation because frames will be independent, rather than showing the distinctive temporal effects of glitter.

Our method combines these two ideas: 4D search using a position-normal hierarchy and randomly generating particle counts rather than counting particles. The key algorithmic novelty is to do both at once, to define a pseudorandom hierarchy that can be generated on the fly while it is searched. It uses very little memory and can efficiently decide how many flakes fall in any given area, in a way that is consistent from one query to the next.



**(a)** *Path tracing using an explicitly generated Gaussian height field*

**(b)** *Gaussian height field (correlation length reduced by a factor of 16)*

**(c)** *Standard smooth microfacet model*

**(d)** *Discrete stochastic microfacet model*

**Figure 2:** *Comparison between a smooth microfacet model, Gaussian random heightfields, and our discrete stochastic microfacet BRDF. (**b**) and (**d**): our model works best when the material's correlation length is much smaller than the distance between pixels. (**a**): when this is not the case, the correlations can lead to visible structure in reflections that our method does not simulate. In this case, the approach of Yan et al. [2014] may be preferable.*

Our model is easy to use because it generalizes the widely used microfacet reflectance model by adding an intuitive parameter—the number of particles per unit area—that gives access to a range of appearances, from subtly glittery to dramatically sparkly, while maintaining the same average BRDF.

In the following sections we develop this *discrete stochastic reflectance model*, which involves the following contributions:

- We introduce the notion of a multiscale BRDF that is queried not at single points and directions but over finite areas and solid angles. Within this setting, we define a new type of microfacet model based on a *discrete* microfacet distribution that describes the reflectance of a surface with finitely many facets.
- We define a specific model based on a finite collection of mirrorlike scattering particles, or flakes, that is implicitly defined by an efficient pseudorandom hierarchy.

## 2 Prior Work

The phenomenon of glitter is of great interest to manufacturers of cars and car paints, and the perception of glitter has been studied in that context in color science [McCamy 1996; McCamy 1998; Kirchner et al. 2007; Dekker et al. 2010]. This work has identified many salient perceptual effects, including the texture characteristics of diffuse coarseness (contrast of the noise seen under diffuse illumination) and glints (bright points seen under unidirectional illumination). Cars are also an important rendering application, and several models for realistic rendering of metallic car paints have been proposed that handle glitter using procedural normal maps [Günther et al. 2005], measured textures [Rump et al. 2008], or noise patterns [Ershov et al. 1999; Ershov et al. 2001].

The probabilistic approach of Ershov et al. is closest to the present paper: it calculates the parameters of a Poisson distribution describing the number of flakes contributing to a pixel, then chooses independent Poisson deviates per pixel to simulate the reflection from flakes. However, this method cannot be used for animation, because there is no way to preserve consistency when the light or viewer moves. By carefully computing the statistics of a particular, though


**Figure 3:** *The scenes from Figure 1 rendered using classical smooth microfacet models (all other parameters are unchanged). Note the overly smooth and synthetic appearance.*

implicit, set of flakes, our method is able to overcome this problem.

The idea of using procedural noise to achieve a random but consistent surface appearance is a common way to model spatial variations in materials [Perlin 2002; Cook and DeRose 2005; Lagae et al. 2009]. Antialiasing for procedural textures that use these noise functions is usually achieved by filtering high spatial frequencies from the noise—an approach that does not work for variation in surface normals.

As we have observed, our work is closely related to microfacet models for reflection from rough surfaces [Cook and Torrance ; Walter et al. 2007]; we use the same framework but with a discrete set of normals rather than a continuous distribution.

Our work is also related to techniques for antialiased rendering of normal maps. Most of these techniques assume very smooth normal distributions, such as Gaussians [Toksvig 2005; Olano and Baker 2010] or Gaussian mixture models [Han et al. 2007]. The only normal-map rendering method capable of reproducing glittery appearance is our concurrent work [Yan et al. 2014]. That method calculates accurate BRDF values for a normal map over arbitrary regions of the surface by using a hierarchical search to locate normals that are close to the half vector. Because it makes minimal assumptions about a pixel's NDF, this approach can resolve the glints that create glittery appearance.

Rendering with explicit normal maps works for any surface, even when there is important spatial structure larger than the pixel scale, whereas the stochastic model presented here assumes there is no visible structure. When the normal map does not show any correlations above the pixel scale, both methods can be applied and both will produce an unstructured pattern of glints in the image. However, the normal-map rendering approach accurately renders a particular normal map and maintains the angular structure of the glints produced by a smooth surface, whereas using our stochastic model for this type of surface amounts to replacing the surface with a random point set that has the same large-scale BRDF but will show different angular structures at the pixel scale. On the other hand, the stochastic method does not require an explicit normal map, instead calculating the required statistics on the fly, and it is able to answer large queries quickly without deciding where the particular glints contributing to the total are, whereas the normal-map method has to consider all the contributing parts of the surface.

# 3 Discrete Stochastic Reflection Models

The standard definition of the BRDF relates the scattered radiance to incident irradiance for a pair of directions and a position on a reflecting surface. Section 3.1 introduces the concept of a *multiscale* BRDF, which is defined analogously but involves reasoning about the reflection within *finite* spatial areas and solid angles. Within this framework, we propose a concrete model based on microfacet theory. A key step in this process, and an important technical contribution of this paper, is a mathematical characterization of the reflection of a cone of exitant directions, discussed in Section 3.2.

Our BRDF is also *stochastic* in the sense that its evaluation is driven by a specially chosen stochastic process. In Section 3.3, we propose an efficient way of evaluating the reflectance function along with the stochastic process using a lazy traversal of a hierarchical data structure. Finally, Section 3.4 explains the stochastic process in more detail, including how to sample it deterministically to produce coherent results in animations.

## 3.1 Multiscale BRDFs

To motivate our approach, we begin with the standard definition of the BRDF: suppose a differential surface area $\mathrm{d}A$ containing the position $\mathbf{x}$ is illuminated from direction $\boldsymbol{\omega}_i$. The BRDF is defined as the ratio of $\mathrm{d}^2\Phi$, the differential power scattered into a solid angle $\mathrm{d}\boldsymbol{\omega}_o$ containing $\boldsymbol{\omega}_o$, to $E$, the incident irradiance. More formally,

$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \coloneqq \frac{\mathrm{d}^2\Phi(\mathbf{x}, \boldsymbol{\omega}_o)}{E(\boldsymbol{\omega}_i)\, a(\mathrm{d}A)\, \sigma(\mathrm{d}\boldsymbol{\omega}_o)\, (\mathbf{n}_\mathbf{x} \cdot \boldsymbol{\omega}_o)}. \quad (1)$$

Here $a(\mathrm{d}A)$ is the surface area of the differential $\mathrm{d}A$, $\sigma(\mathrm{d}\boldsymbol{\omega}_o)$ denotes the area of $\mathrm{d}\boldsymbol{\omega}_o$ on the unit sphere, and $(\mathbf{n}_\mathbf{x} \cdot \boldsymbol{\omega}_o)$ is a foreshortening term involving the surface normal $\mathbf{n}_\mathbf{x}$. All areas and solid angles in the above definition are infinitesimal.

A property of $f_r$ that is characteristic of glinty scattering behavior is that the function contains many very narrow spikes dispersed throughout the $(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ space. It is in principle possible to model such effects using a standard BRDF model, but undesirable in practice due to the difficulty in resolving the spikes using point queries when computing integrals involving $f_r$.

A multiscale BRDF then expresses the power that the finite area $A$ around $\mathbf{x}$ scatters into a finite solid angle $\Omega_o$ around $\boldsymbol{\omega}_o$:

$$\widehat{f}_r(A, \boldsymbol{\omega}_i, \Omega_o) \coloneqq \frac{1}{a(A)\, \sigma(\Omega_o)} \int_A \int_{\Omega_o} f_r(\mathbf{x}, \omega_i, \omega_o)\, \mathrm{d}\omega_o\, \mathrm{d}\mathbf{x}. \quad (2)$$

Due to the way that multiscale BRDFs are integrated into our system, we still use a single incident direction $\omega_i$ rather than a set $\Omega_i$ around $\omega_i$. However, this latter variant could be an interesting avenue for future research.

The ability to compute averages over regions of the domain is effectively a form of antialiasing to approximate relevant integrals with a lower number of function evaluations. Like other antialiasing methods (e.g. filtered texture lookups), this introduces a small amount of bias into the computation. We can also draw an analogy to weak formulations of mathematical problems, which result from integrating the underlying system against a family of test functions. In our case, these are the indicator functions on the sets $A$ and $\Omega_o$, whose precise shape will be clarified shortly when we show how a multiscale BRDF can be integrated into an actual rendering system.

To derive a multiscale BRDF based on Equation (2), we must first specify the nature of the underlying classical BRDF model $f_r$. We use microfacet theory for this purpose.

Microfacet models calculate the light reflected from a surface made of randomly oriented microscopic facets, a subset of which reflect from a source to a receiver under a particular illumination and viewing geometry. The amount of light reflected over a large area is determined by calculating the expected total area of reflecting facets, producing a smooth function. Microfacet BRDFs are defined as a product of terms that account for different physical effects. In the

context of computer graphics, the expression for the reflective case is usually written as [Cook and Torrance ; Walter et al. 2007]

$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{F(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_h)\, D(\mathbf{x}, \boldsymbol{\omega}_h)\, G(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \boldsymbol{\omega}_h)}{4\,(\boldsymbol{\omega}_i \cdot \mathbf{n}_\mathbf{x})\,(\boldsymbol{\omega}_o \cdot \mathbf{n}_\mathbf{x})} \tag{3}$$

where $\boldsymbol{\omega}_h \coloneqq (\boldsymbol{\omega}_i + \boldsymbol{\omega}_o)/\|\boldsymbol{\omega}_i + \boldsymbol{\omega}_o\|$ is the half angle, $F$ denotes the Fresnel reflection coefficient, $D$ is the microfacet distribution, and $G$ models shadowing and masking.

Our new model treats surfaces as being made up of a specific set of *finitely many* randomly oriented facets, and we therefore define a *discrete* microfacet distribution as

$$D(\mathbf{x}, \boldsymbol{\omega}_h) \coloneqq \frac{1}{N} \sum_{k=1}^{N} \delta_A(\mathbf{x},\, \mathbf{x}^k)\, \delta_{\Omega_h}(\boldsymbol{\omega}_h,\, \boldsymbol{\omega}_h^k),$$

where $\mathbf{x}^k$ and $\boldsymbol{\omega}_h^k$ are the positions and normals of a list of $N$ facets. The Dirac delta functions above are defined with respect to the normal integration measures on $A$ and the half vector space $\Omega_h$. Next, we insert this definition into Equation (3) and substitute the resulting expression into Equation (2), at which point the integrals over positions and outgoing directions simplify to a sum over facets. We must remember to apply a change of variables factor of $4\,(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_h^k)$ to account for the different integration measures: Equation (3) integrates over outgoing angles $\Omega_o$, whereas the delta function in $D$ is defined with respect to half-direction vectors on the set $\Omega_h$ [Torrance and Sparrow 1967]. The resulting expression reads

$$\widehat{f}_r(A, \boldsymbol{\omega}_i, \Omega_o) = \frac{1}{N\, a(A)\, \sigma(\Omega_o)} \left[ \sum_{k=1}^{N} \mathbf{1}_{\Omega_h}(\boldsymbol{\omega}_h^k)\, \mathbf{1}_A(\mathbf{x}^k) \right. \\ \left. \frac{(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_h^k)\, F(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_h^k)\, G(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o^k, \boldsymbol{\omega}_h^k)}{(\boldsymbol{\omega}_i \cdot \mathbf{n}_{\mathbf{x}^k})\,(\boldsymbol{\omega}_o^k \cdot \mathbf{n}_{\mathbf{x}^k})} \right] \tag{4}$$

where $\Omega_h \coloneqq \{(\boldsymbol{\omega}_i + \boldsymbol{\omega}_o)/\|\boldsymbol{\omega}_i + \boldsymbol{\omega}_o\| \mid \boldsymbol{\omega}_o \in \Omega_o\}$ is the set of microfacet normals that reflect from $\boldsymbol{\omega}_i$ into the solid angle $\Omega_o$ and $\mathbf{1}_X$ denotes the indicator function on the set $X$. The directions $\omega_o^k$ are defined as the outgoing directions after specular reflection by microfacet $k$ i.e. $\boldsymbol{\omega}_o^k = 2(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_h^k)\boldsymbol{\omega}_h^k - \boldsymbol{\omega}_i$.

The indicator functions in Equation (4) determine which facets in a given spatial region reflect light into a solid angle of outgoing directions. When just a few facets participate, there is considerable variation from pixel to pixel, producing a strongly glittery appearance; when many facets participate, there is less dramatic variation and the surface reflectance appears smoother and more like a traditional BRDF.

The sets $A$ and $\Omega_o$ are generally small in size: $A$ is related to the footprint of a pixel, and we use sets $\Omega_o$ no larger than 0.034 steradians. Consequently, various terms in the sum (Fresnel, geometric, and foreshortening terms) only vary minimally between different facets, and the dominant effect is caused by the indicator functions.

For this reason, we introduce an approximation that will lead to an efficient implementation later on: we remove all superscripts involving $k$ from the fraction term on the second line of Equation (4). The corresponding quantities without superscripts are defined as the centerpoints of their respective sets. Following this, the term can be moved outside the sum, which yields the final form of our BRDF:

$$\widehat{f}_r(A, \boldsymbol{\omega}_i, \Omega_o) = \\ \frac{(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_h)\, F(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_h)\, \widehat{D}(A, \Omega_h)\, G(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \boldsymbol{\omega}_h)}{a(A)\, \sigma(\Omega_o)\, (\boldsymbol{\omega}_i \cdot \mathbf{n}_\mathbf{x})(\boldsymbol{\omega}_o \cdot \mathbf{n}_\mathbf{x})}, \tag{5}$$

where $\widehat{D}$ is a multiscale version of the microfacet distribution:

$$\widehat{D}(A, \Omega_h) \coloneqq \frac{1}{N} \sum_{k=1}^{N} \mathbf{1}_{\Omega_h}(\boldsymbol{\omega}_h^k)\, \mathbf{1}_A(\mathbf{x}^k). \tag{6}$$

This function resolves the fraction of particles located in the query area $A \times \Omega_h$. Most of the remainder of the paper focuses on how to evaluate it efficiently.

**Usage in a rendering system:** In our implementation, $A$ is defined as a parallelogram in texture space, which approximates the region of the surface visible through a pixel and is readily computed using ray differentials [Igehy 1999]. We define the finite solid angle into which surface microfacets may reflect light as a cone of radius $\gamma$ centered around a central outgoing direction $\boldsymbol{\omega}_o$. The smaller the angle $\gamma$, the fewer particles will be found, though the $\sigma(\Omega_o) = \pi(1 - \cos\gamma)$ term in the denominator of $\widehat{f}_r$ compensates for this by weighting each particle higher. In this way, the $\gamma$ parameter offers some control over the variance of the integrand that is exposed to the rendering system. Another interpretation of its effect is as a smoothing kernel that convolves the lighting with a circular unit step function. We use a constant $\gamma$ between 0.5 and 6 degrees in our scenes (Table 1). More involved approaches that decompose the incident lighting into different-sized cones that drive the $\gamma$ parameter are conceivable, but we leave this as a future work.

The parameter $N$ specifies the total number of facets, which are in essence points in a 4D domain: two dimensions of surface position and two of normal direction, with a distribution that's uniform in space and defined by a microfacet distribution in direction. We postpone a full discussion of how these points are generated until Section 3.4. Because of the additional parameter dependences, and because the set of microfacet normals $\Omega_h$ is a function of $\boldsymbol{\omega}_i, \boldsymbol{\omega}_o$, and $\gamma$, we will from now on write the multiscale microfacet distribution $\widehat{D}$ using the arguments $\widehat{D}(D, N, A, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \gamma)$.

The key to achieving a practical model is having the ability to quickly count the facets that fall within a query region (spatial region + angular cone), which implies that we cannot afford to ever enumerate them individually. In the remainder of this section, we will develop the ingredients necessary to achieve this goal, culminating in an efficient hierarchical data structure that supports lazy evaluation of Equation (6).

### 3.2 Reflection Geometry

Consider the problem of identifying which microfacet normals $\mathbf{m}$ reflect light from a specified incident direction $\boldsymbol{\omega}_i$ into a cone of out-

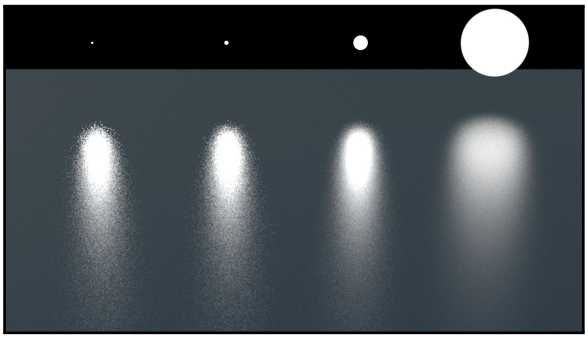
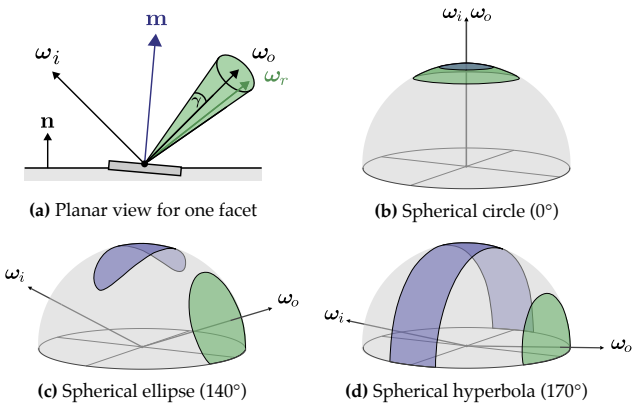


**Figure 4:** *When illuminating a surface by sphere sources, changing the query radius parameter $\gamma$ of our model is equivalent to scaling the source radius while keeping its power constant. A higher query radius will select more particles, producing less glinty appearance.*




**(a)** Planar view for one facet **(b)** Spherical circle (0°)

**(c)** Spherical ellipse (140°) **(d)** Spherical hyperbola (170°)

**Figure 5:** ***(a)*** *The microfacet normals* $\mathbf{m}$ *that reflect light from a fixed incident direction* $\boldsymbol{\omega}_i$ *into a cone of radius* $\gamma$ *around* $\boldsymbol{\omega}_o$ *(green set) form a spherical conic section (blue set).* ***(b)-(d)*** *show three cases for different angles between* $\boldsymbol{\omega}_i$ *and* $\boldsymbol{\omega}_o$ *(in parentheses).*

going angles of radius $\gamma$ around a central direction $\boldsymbol{\omega}_o$ (Figure 5a).

By the law of specular reflection, the reflected direction $\boldsymbol{\omega}_r$ after scattering from a microfacet with normal $\mathbf{m}$ is given by

$$\boldsymbol{\omega}_r = 2(\mathbf{m} \cdot \boldsymbol{\omega}_i)\mathbf{m} - \boldsymbol{\omega}_i. \tag{7}$$

We may generalize this relationship by considering the set of directions that scatter into the cone around $\boldsymbol{\omega}_o$:

$$\begin{aligned} & \boldsymbol{\omega}_r \cdot \boldsymbol{\omega}_o > \cos\gamma \\ \Leftrightarrow \quad & 2(\mathbf{m} \cdot \boldsymbol{\omega}_i)(\mathbf{m} \cdot \boldsymbol{\omega}_o) - \boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_o > \cos\gamma \\ \Leftrightarrow \quad & \mathbf{m}^T\big(\mathbf{I}(\cos\gamma + \boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_o)/2 - \boldsymbol{\omega}_i\boldsymbol{\omega}_o^T\big)\mathbf{m} \le 0 \end{aligned} \tag{8}$$

Equation (8) specifies a second-order cone [Boyd and Vandenberghe 2004], and the set $\Omega_h$ thus is the intersection of such a cone with the unit sphere (Figure 5). In analogy to the planar case, this is referred to as a *spherical conic section*.

Spherical conic sections have been studied before; a good starting point are two articles by Booth [1844; 1852]. However, we could not find any reference of this particular spherical conic section and its relation to the reflection geometry in the literature and believe that it has not been analyzed before. To obtain some intuition about its shape, we use a more natural coordinate frame in terms of the half and difference vectors of $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$:

$$\hat{\mathbf{x}} = \frac{\boldsymbol{\omega}_i \times \boldsymbol{\omega}_o}{\|\boldsymbol{\omega}_i \times \boldsymbol{\omega}_o\|}, \quad \hat{\mathbf{y}} = \frac{\boldsymbol{\omega}_i - \boldsymbol{\omega}_o}{\|\boldsymbol{\omega}_i - \boldsymbol{\omega}_o\|}, \quad \hat{\mathbf{z}} = \frac{\boldsymbol{\omega}_i + \boldsymbol{\omega}_o}{\|\boldsymbol{\omega}_i + \boldsymbol{\omega}_o\|},$$

In this frame, and after scaling Equation (8) to normalize the third eigenvalue, we obtain the standard form of the conic section:

$$\|\mathbf{m}\| = 1 \text{ and } \mathbf{m}^T\mathbf{C}\mathbf{m} \le 0, \text{ where } \mathbf{C} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T \text{ and} \tag{9}$$

$$\mathbf{Q} = \begin{bmatrix} | & | & | \\ \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \\ | & | & | \end{bmatrix}, \quad \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & -1 \end{bmatrix},$$

$$\lambda_1 = \frac{\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_o + \cos\gamma}{1 - \cos\gamma}, \quad \lambda_2 = \cot^2\frac{\gamma}{2}.$$

The first two eigenvalues are the squared cotangents of the principal angles, leading to the following observations: when $\boldsymbol{\omega}_i = \boldsymbol{\omega}_o$, the conic section starts off as a spherical circle with radius $\gamma/2$. As the directions become increasingly separated, the circle becomes a spherical ellipse and, finally, a spherical hyperbola that wraps around the sphere in the shape of a band. The transition from the elliptic to the hyperbolic case occurs when $\lambda_1$ becomes negative, i.e. when $\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_o = -\cos\gamma$, which happens when $-\boldsymbol{\omega}_i$ enters the cone around $\boldsymbol{\omega}_o$. Figure 5 visualizes these three cases for different angles between $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ with $\gamma$ fixed at $30^\circ$.

### 3.3 Hierarchical Traversal

Recall that the search query consists of the Cartesian product of a spherical conic $\Omega_h$ and a parallelogram in texture space. For now, we assume that the locations of the particles in the spatial-directional domain are known ahead of time, and that they are organized in a 4D spatial subdivision acceleration data structure. Briefly disregarding lines 11 and 12, Algorithm 1 implements a standard breadth first search traversal. In each iteration, the algorithm pops a 4D tree node from a queue and checks for overlap with the query. If the node contains no particles or does not overlap the query, it is immediately discarded. If the node is fully contained in the query region, the number of particles within it, $|\,\texttt{node}\,|$, is added to a running counter. Otherwise, the node's children are obtained via the `split()` operation and pushed onto the queue for later processing. The final line returns the number of detected particles divided by their total number on the full domain.

**Adaptive error criterion:** In the form described above, the counting algorithm would traverse the hierarchical data structure until the exact answer is obtained. However, this level of precision is often unnecessary and incurs a heavy cost in terms of total rendering time: when the query covers tens of thousands or even millions of particles, it is excessive to accurately resolve the result down to the last particle. Therefore, we use an approximation in line 12 of Algorithm 1 that stops the recursive expansion of a node when the expected error relative to the current particle count is small. In this case, a *fraction* of the particles contained in a node are added to the running particle count based on how much it overlaps the query, where `.vol()` computes the 4D volume of a set.

Let us denote the overlap as $p \in [0, 1]$, and let $n$ be the number of particles in the node. Under the assumption of a uniform distribution within the cell, the probability of an individual particle falling inside the query is equal to $p$. Since there are $n$ particles, the number of particles contained in the query follows a binomial distribution with parameter $p$ and $n$ draws. Our approximation is to replace the actual number of particles in the intersection with the expected number. This introduces an expected error equal to the distribution's standard deviation and leads to a simple adaptive er-

**Algorithm 1** Breadth-first search evaluation of $\widehat{D}$

```
1  function D̂(D, N, A, ω_i, ω_o, γ)
2      query ← A × ConicSection(ω_i, ω_o, γ)
3      queue ← { root node of tree }
4      count ← 0
5      while queue ≠ ∅ do
6          node ← queue.pop()
7          if node ∩ query = ∅ or |node| = 0 then
8              pass
9          else if node ⊆ query then
10             count ← count + |node|
11         else if error criterion (10) satisfied then
12             count ← count + |node| ·
                 (node ∩ query).vol()/node.vol()
13         else
14             for c in node.split() do
15                 queue.push(c)
16     return count / N
```

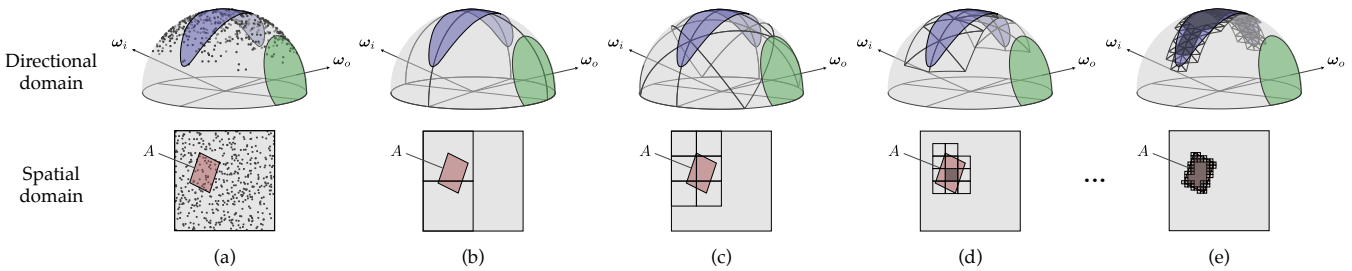



**Figure 6:** ***(a)*** *Our multiscale BRDF models the radiance that region $A$ (marked in red) scatters into a solid angle around $\boldsymbol{\omega}_o$ (marked in green). Scattering is due to small mirror facets, each of which has a position and an orientation (we indicate their projections onto the spatial and directional dimensions as black dots). For a particle to reflect light from $\boldsymbol{\omega}_i$ into the cone around $\boldsymbol{\omega}_o$, its normal must lie in the highlighted blue set. Evaluating the BRDF thus entails counting the number of particles that are contained in both the blue and red sets.* ***(b)-(e)*** *show iterations 1, 2, 3, and 5 of a breadth first search that implements this counting operation (particles hidden for clarity).*

ror criterion: we only use the approximation when the underlying smooth microfacet distribution is close to uniform within a node (more on this later), and when

$$\sqrt{np(1-p)} < \varepsilon \cdot \texttt{count}, \tag{10}$$

where $\varepsilon$ is a relative error threshold. We initially chose $\varepsilon = 1\%$, but found that we could use values as high as 10% without noticeable errors. All of our results thus use $\varepsilon = 10\%$. This error criterion is the reason why breadth first search is the preferred traversal method: we want to increase `count` as quickly as possible so that the criterion can be used in more cases.

**Data structure:** Thus far, Algorithm 1 is fully generic and could be implemented using a range of different tree data structures. The only operations that must be supported by its nodes are intersection "$\cap$" and containment "$\subseteq$" tests, enumerating children via `split()`, and overlap computations against the query.

We experimented with different tree constructions and found the following to work best from an implementation effort and performance perspective: the footprint of a tree node is defined as the Cartesian product of a bounding box in texture coordinate space and a spherical triangle in direction space. Each node has four children that are obtained by splitting the node either in space or direction. In the first case, the bounding box is cut into four equal-sized sub-boxes, and in the second case the spherical triangle is cut into four sub-triangles by inserting vertices at the midpoints of the edges. The root of the tree is the Cartesian product of texture space and the unit hemisphere, i.e. $[0,1]^2 \times \mathcal{H}^2$. Since the hemisphere is not a spherical triangle, the first directional subdivision is a special case: here, we simply cut the hemisphere into four quadrants. A node is split in space if

$$E\left[A(\texttt{query})\right]/A(\texttt{node}) < E\left[\sigma(\texttt{query})\right]/\sigma(\texttt{node}) \tag{11}$$

and in direction otherwise. $A$ and $\sigma$ measure the texture space area and subtended solid angle, respectively. The expected values in Equation 11 are obtained by collecting statistics during a brief prior run of the algorithm; they ensure that the data structure is built to efficiently handle queries with the expected footprint in space and direction. Figure 6 illustrates the first traversal steps, where boxes and triangles highlight the nodes that are visited in each iteration and dark patches mark regions classified as being inside the query (thus needing no further refinement). For simplicity, the figure shows the spatial-directional recursion as occurring in lock-step, but in practice the sequence of spatial and directional splits alternates according to Equation (11).

**Intersection and overlap computations:** Due to the Cartesian product structure, all intersection, containment, and overlap computations can be implemented separately for the spatial and directional dimensions. The part that occurs in texture space involves standard methods (rectangle-parallelogram intersection), hence we will not discuss it here (see e.g. [Schneider and Eberly 2002]). For direction space, we need to determine the intersection of a spherical conic $C$ with a spherical triangle $T$. If $\partial C$ and $\partial T$ are disjoint, then $C$ contains $T$ if it contains any of its vertices; otherwise $T$ contains $C$ if $T$ contains a point in $C$; otherwise $C$ and $T$ are disjoint. If the boundaries intersect, $C$ and $T$ partially overlap.

To test if a point lies inside a conic section, we simply evaluate Equation (8). Determining if a point lies in a spherical triangle involves three plane-point sidedness tests, where each plane contains one of the triangle edges and the center of the sphere. Next, we need the ability to intersect the edge of a spherical triangle, i.e. a spherical arc, against the edge of the conic section. Here, we benefit from our representation of the conic section: the spherical arc from $\mathbf{a}$ to $\mathbf{b}$ intersects the spherical conic section iff the straight line from $\mathbf{a}$ to $\mathbf{b}$ intersects the second order cone. This leads to a simple quadratic equation. We use the following symmetric formulation for its good numerical behavior:

$$\mathbf{p}(t)^T\mathbf{C}\mathbf{p}(t) = 0 \text{ where } \mathbf{p}(t) \coloneqq \mathbf{c} + t\mathbf{d},\ \mathbf{c} \coloneqq \mathbf{a} + \mathbf{b},\ \mathbf{d} \coloneqq \mathbf{a} - \mathbf{b}.$$

The corresponding quadratic equation is given by

$$\mathbf{d}^T\mathbf{C}\mathbf{d}t^2 + 2\mathbf{d}^T\mathbf{C}\mathbf{c}t + \mathbf{c}^T\mathbf{C}\mathbf{c} = 0,$$

and the edge intersects the conic iff there is a root in $[-1, 1]$.

Finally, we must implement a routine that computes the overlap between a spherical triangle and a spherical conic section. Even fairly basic area computations on spherical conic sections involve complex reductions to elliptical integrals of the third kind [Booth 1852], but the overlap is only needed when error criterion (10) is satisfied, at which point the triangle's edges have small radii of a few degrees (and thus low spherical excess). In this setting, the spherical triangles are flat enough to ignore the spherical excess and simply do the overlap computation in the plane containing the triangle vertices, involving the corresponding *planar* conic section. After a suitable nonuniform scaling transformation, this problem reduces to the intersection of a 2D triangle against the unit circle or unit hyperbola, which may be solved via Stoke's theorem. This entails finding all linear and curved segments of the intersection and integrating the 1-form $x\mathrm{d}y - y\mathrm{d}x$ over them.

### 3.4 Stochastic Process

So far, we have described the model as if the input was an explicitly enumerated set of facets organized in a hierarchical data structure.

While this is certainly possible, such an approach would be cumbersome and could easily exceed the available memory for large numbers of microfacets (e.g. $N = 10^9$ in Figure 4 or even more).

To avoid these large space and time demands, we never explicitly store individual microfacets and instead generate them *on the fly* using a stochastic process that is evaluated while traversing the hierarchy. To ensure temporal coherence, we seed the process in a deterministic way so that repeated queries will give consistent results. This is important even when rendering still frames: a Monte Carlo rendering algorithm that takes multiple samples per pixel will average out any "true" randomness in the reflection model, converging to a smooth BRDF without glitter. Making model evaluations deterministic avoids this issue.

The main idea of our new stochastic process is to generate particle counts from the top down, always respecting choices that were made higher in the tree. The process is easiest to visualize in 2D space; the 4D case is completely analogous. Suppose we start with a rectangle known to contain $n$ uniformly distributed points. Dividing the rectangle into four equal subrectangles causes the points to be partitioned into four subsets, each with expected size $n/4$. Given no further information, the set of possible partitions $\mathbf{X} = (X_1, \ldots, X_4)$ can be seen to follow a multinomial distribution with probability vector $\mathbf{p} = (1/4, 1/4, 1/4, 1/4)$ and $n$ trials. So when traversing the hierarchy, we draw a sample from this multinomial distribution, at which point the numbers of points, or counts, for the four child nodes become known.

This process applies recursively to child nodes, thereby procedurally defining a count for *every* node in the (infinite) tree, starting with $N$ at the root. Using the multinomial distribution ensures that the counts of the children always sum to the count of their parent, so any cut through the tree will always find the same total number of points. With further subdivision, counts will eventually be 1 or 0, and every point can be localized as precisely as needed, since recursion can be continued arbitrarily far, trapping the points in ever-smaller regions. Thus, this algorithm can be seen to *define* a set of $N$ uniformly distributed points while allowing the number contained in any node to be found without generating the points. The traversal is entirely procedural; the nodes of our hierarchy never need to be stored.

Another way to think about this is that the initial $N$ particles within the root node are "fuzzy" in the sense that their positions and orientations have not been individually resolved; the only knowledge we have about them is their number in aggregate. Each splitting operation reduces the uncertainty by one bit, and our traversal algorithm continues splitting nodes until the number of particles that fall within the query region is known to sufficient certainty.

Generating a multinomial sample is slightly more complicated in the case of a directional split, since the particles follow a nonuniform distribution over the hemisphere. Our goal is that the discrete reflectance model reduces to the traditional smooth case in the limit of small $\gamma$ and $N \to \infty$, which implies that the probability vector $\mathbf{p}$ for a directional subdivision is found by integrating the smooth microfacet distribution $D$ over the four child spherical triangles and renormalizing so that the four probabilities sum to one.

Doing this analytically is difficult and restrictive in the sense that it would limit our method to certain distributions—hence we take a numerical approach. Before the rendering process, the function `integrate` (Algorithm 2) is invoked on the four spherical triangles corresponding to quadrants of the hemisphere. This function integrates the microfacet distribution using both a midpoint and corner-based quadrature rule. When these two results differ by more than a specified error threshold, our algorithm recurses and returns the sum of the integrals of the triangle's four sub-triangles.

**Algorithm 2** Numerical integration of $D$ over a spherical triangle

```
1  function integrate(T)
2      rule1 ← T.area() · D(T.center())
3      rule2 ← 1/3 · T.area() · (D(T.v1) +
                                 D(T.v2) + D(T.v3))
4      if |rule1 − rule2| < δ_abs or
         |rule1 − rule2| / rule2 < δ_rel then
5          return rule2
6      else
7          rule3 ← 0
8          for c in T.split() do
9              rule3 ← rule3 + integrate(c)
10         hashtable[T.id] ← rule3
11         return rule3
```

Importantly, for each spherical triangle that required such a composite integration step, the function creates an entry in a globally accessible hash table on line 10.

At render time, whenever we need to integrate the microfacet distribution over a spherical triangle, we simply check if the triangle has an entry in the table. If so, the stored entry is returned. Otherwise, we are "allowed" to treat the distribution as close to uniform and perform the integration using a simple 1-point rule (if this was not sufficient, an entry would have been created). This memoization approach is quite efficient, since only nonuniform triangles must be stored in the hash table. Depending on roughness, our example scenes used between 500-1000 entries when $\delta_{abs} = \delta_{rel} = 10^{-5}$.

**Optimizations:** The breadth first search traversal (Algorithm 1) is key to the efficiency of our algorithm, and we experimented with different ways of accelerating it. Sampling the multinomial distribution exactly, even using an elaborate high-performance implementation [Galassi 2009; Kachitvichyanukul and Schmeiser 1988], proved to be a serious performance bottleneck. Instead, we use a simple but quite accurate approximation: whenever the number of particles in a node, $n$, exceeds 8, we draw a sample from the continuous analog of the multinomial, a multidimensional normal distribution with mean $n\mathbf{p}$ and covariance $n(\mathrm{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T)$. This produces a realization $\mathbf{X}$ that sums to $n$ but does not generally have integer entries. We round $\mathbf{X}$ to the nearest positive integer 4-vector and keep randomly incrementing (or decrementing) component $i$ with probability $p_i$ until $\sum X_i = n$. When $n < 8$, we sample exactly by simply choosing $n$ integers using the probability vector p and counting the frequencies. We implemented the sampling routine, including 8 rounds of the Tiny Encryption Algorithm [Wheeler and Needham 1995] as the original source of pseudorandomness, in branchless 4-lane SIMD arithmetic using Intel SSE instructions.

The Tiny Encryption Algorithm depends on an index that acts as the random seed when using it as a pseudorandom number generator; we use a unique number that is assigned to each tree node (the root is labeled 1, and child $j$ of node $i$ is labeled $4i + j - 2$).

**Importance sampling:** To effectively use our new BRDF model in a modern rendering system, we require a way of sampling directions from a distribution that is close to the BRDF. We experimented with approaches that directly sample the discrete integrand by means of another tree traversal but ultimately found that a simple technique based on a different smooth density function had better efficiency in the scenes we considered. Such an approach is permissible in a Monte Carlo sampling context, as long as we account for the discrepancy between the target BRDF and the used density function in the underlying statistical estimator.

To sample $\boldsymbol{\omega}_o$ for a given incident direction $\boldsymbol{\omega}_i$ we first use the sampling method of the corresponding smooth microfacet model (refer

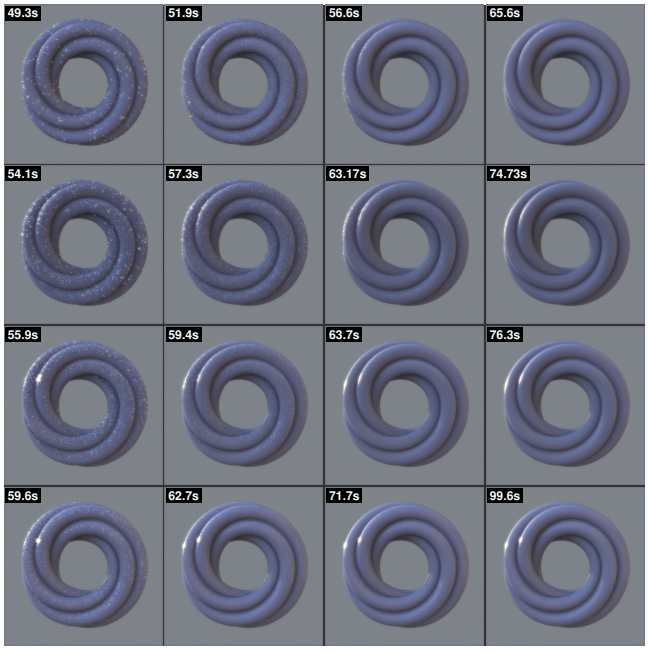



**Figure 7:** *A diffuse object sprinkled with dielectric flakes of different density (**left to right**: $N = 10^5, 10^6, 10^7$, and $10^8$ flakes) and spread (**top to bottom**: $\alpha = 0.81, 0.29, 0.09$, and $0.03$), postprocessed with a bloom filter. The bottom left of this matrix approaches traditional smooth microfacet models, whereas the top right reproduces the sporadic bright reflections observed in craft glitter.*

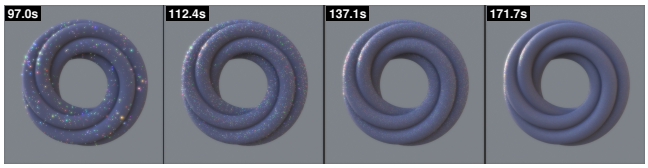



**Figure 8:** *It is possible to extend our BRDF model by associating additional state with each particle. Here, they have spectrally varying interaction cross-sections that modulate their contribution. The other parameters are set to match the second row of Figure 7.*

to Walter et al. [2007] for details), producing a direction $\boldsymbol{\omega}_o'$. Like Walter et al., we also use a scaled roughness parameter to reduce the variance of the resulting sampling method. In our case, a relatively large scale factor ($\alpha_{\text{sample}} = 2\alpha$) is necessary due to the discrete nature of our model: with low probability, some particles can end up in spherical triangles that have very low integrated $D$, causing occasional but very noticeable high variance pixels at the fringe of highlights. The scaling widens the lobe so that these directions are sampled with sufficiently high probability.

Afterwards, $\boldsymbol{\omega}_o$ is found from $\boldsymbol{\omega}_o'$ by uniformly choosing a direction in a cone of radius $\gamma$ around $\boldsymbol{\omega}_o'$. The density function of this sampling method is the density function of the smooth model with scaled roughness, convolved with a circular kernel. We evaluate the density function by performing the convolution using Quasi Monte Carlo integration; for the range of angles $\gamma$ used in this paper, 64 samples of a $(0, 2)$ rule [Keller et al. 2012] proved adequate.

**Additional state:** Our approach can be extended to allow for additional per-particle state, such as surface area or varying color due to albedo or interference coatings. The added state of each particle is modeled as a sample from a distribution over states, and a node splitting operation then draws a sample from a derived distribution.

Recall that thus far, each particle within the query region made a contribution of $1/N$ to the value of $\widehat{D}$. To demonstrate how additional state can be integrated, we consider an example where each particle instead makes a contribution of $a_i$—we can think of $a_i$ as the fractional "area" of particle $i$ (in quotes because we continue to resolve particles as points in query operations). Suppose furthermore that the $a_i$ are independent and uniformly distributed on the interval $[0, 1]$, and that the particle area is merely redistributed but not changed, i.e. $\sum a_i = 1$. As a consequence of the sum constraint, the particle areas take on a special conditional probability density, which is described by a $N$-dimensional Dirichlet distribution with parameter $(1, \ldots, 1)$. One way to integrate these areas into our method would thus involve generating and storing a sample from such a distribution during a preprocess step—but as before, this is generally infeasible due to the sheer size of $N$.

Instead, we prefer to determine surface areas on the fly while traversing the acceleration data structure. Suppose that the set of $N$ particles into is partitioned into four subsets during a recursive traversal operation, where the particle counts of the subsets satisfy $N_1 + N_2 + N_3 + N_4 = N$. The Dirichlet distribution has a convenient addition property which, in this case, states that the total surface areas of the child nodes are distributed according to another four-dimensional Dirichlet distribution with parameters $(N_1, N_2, N_3, N_4)$. We therefore generate a sample from this distribution (details in Devroye's [1986] book) in every traversal operation using the previously discussed approach to deterministically seed the underlying pseudorandom number generator. Figure 8 shows a rendering where we track separate particle areas for each color channel (i.e. spectral interaction cross-sections). This adds chromatic variation to the glints, which slowly vanishes as the their number within a pixel grows.

## 4 Results

We implemented our BRDF as a shading model plugin for the Mitsuba renderer [Jakob 2010] using C++ and SSE intrinsics for the performance-critical sampling code.

Our example images were rendered using Mitsuba's backwards path tracing integrator with multiple importance sampling [Veach 1997]. This integrator supplies texture space differentials to BRDF implementations, but it only does so for the first scattering event along a path. Our plugin detects this and reverts back to the smooth case when no differentials are specified. In practice this means that the first bounce of a rendering involving interreflecting objects relies on the discrete stochastic model, whereas later bounces use the smooth model.

Table 1 reports rendering time and other statistics collected on a dual 2.2 Ghz Intel Xeon E5-2660 machine with 16 hyper-threaded cores. Unless otherwise stated, we use the Beckmann microfacet model as the underlying smooth distribution $D$, which has a single roughness parameter $\alpha$ with lower values corresponding to

**(a)** *Ground truth* **(b)** *Multinomial approx.* **(c)** *Equation* (4) $\rightarrow$ (5)

**Figure 9:** *Effects of approximations in our model: switching to a simpler sample generator in (**b**) results in a different realization of the stochastic process and large image differences. Image (**c**) shows the errors that result from the approximation to Equation* (4)*. The noise characteristics remain unchanged in both cases.*

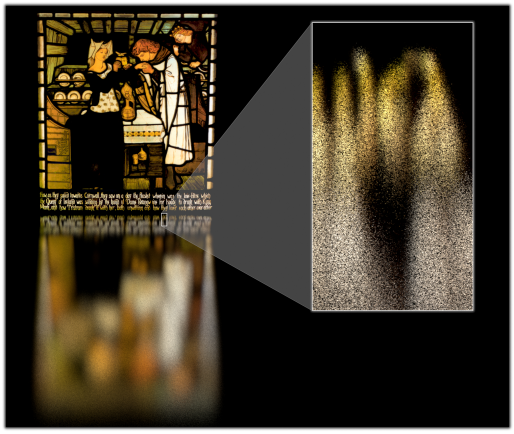


**Figure 10:** *Reflection of a stained glass window on a shiny floor ($N = 10^8$, Beckmann roughness $\alpha = 0.01$). **Left**: as the number of visible facets within an individual particle increases, our model converges to a conventional BRDF. **Right**: under great magnification, the reflection resolves into individual particles.*

smoother surfaces. Except for Figure 7, we simulate metallic flake particles without Fresnel effects (i.e. $F = 1$). Color-coated flakes are obtained by simply multiplying the BRDF value $\hat{f}$ with the transmittance of the coating. Altogether, we rendered the following scenes:

RED SLIPPERS: The left side of Figure 1 shows a fancy pair of women's dress shoes modeled as a linear combination of a red diffuse BRDF and a monochromatic discrete stochastic microfacet BRDF. The scene is lit by a mostly smooth interior environment map with one strong directional source from the left. The supplementary video shows how glints appear and disappear as the viewing position changes. The statistics are for an uncropped version of this image rendered at full HD resolution (1920x1080 pixels) using 1024 samples per pixel to resolve single and multiple scattering. Rendering interreflecting glossy objects under sharp illumination remains a difficult problem in rendering, and our model is no different in this respect. To improve convergence, we use smoother lighting when sampling the illumination on an object that is only seen through a reflection

CHRISTMAS ORNAMENTS: The right side of Figure 1 shows a range of glittery Christmas ornaments inside a box made of glossy plastic. We modeled the plastic box as a diffuse layer coated with a smooth microfacet BSDF ($\alpha = 0.1$), and the ornaments use the discrete stochastic model. Our model can work with any smooth microfacet distribution, and in this rendering the top right ornament uses the anisotropic $D$ proposed by Ashikhmin and Shirley [2000]. The detailed parameters are as follows (from left to right, and top to bottom): $N = (8{\cdot}10^5, 9{\cdot}10^6, 8{\cdot}10^5, 4{\cdot}10^6, 4{\cdot}10^5, 4{\cdot}10^5, 2{\cdot}10^6)$ and $\alpha = (0.2, [0.1, 0.4], 0.1, 0.1, 0.2, 0.2, 0.2)$, where the two numbers in brackets correspond to the tangential roughness values of the anisotropic ornament. The parameter $\gamma$ is set to a constant value of $5^\circ$. As before, statistics in Table 1 are for an uncropped full HD version of this image. To resolve interreflections between the plastic box and the ornaments, a relatively high number of 2048 samples per pixel was necessary for this scene.

TWISTED TORUS: Figure 7 shows a matrix of direct illumination renderings of a diffuse material test object that has been sprinkled with dielectric flakes with an index of refraction of $\eta = 1.5$. This figure illustrates the effect of the $N$ and $\alpha$ parameters. For high $N$ and low $\alpha$, evaluations of $\hat{D}$ find more particles, leading to a visually smooth appearance that approximates traditional microfacet models. For high $\alpha$ and low $N$, the appearance reproduces the sporadic and very bright highlights also observed in craft glitter. These images were rendered at resolution 768x768 pixels with 256 samples per pixel and post-processed using a bloom filter to better indicate the intensity of the highlights.

SPHERE LIGHTS: Figure 4 shows a direct illumination rendering of four different-sized sphere lights, each having the same power, over a reflective floor, rendered with 256 samples per pixel. This image illustrates how illumination from smaller sources produces more glints, whereas illumination from large sources leads to reflections that approximate the smooth case.

STAINED GLASS: The scene in Figure 10 shows the reflection of one of the Tristram and Isoude stained glass panels by Morris, Marshall, Faulker & Co. in a reflective floor having low roughness and a high particle count ($\alpha = 0.01$, $N = 10^8$). Statistics are reported for the first HD frame of the animation, rendered with 512 samples per pixel. From a distance, the floor appears visually uniform, but its discrete nature becomes visible in magnifications and, as shown in the video, when moving the camera close to the floor.

**Comparison against smooth models:** Figure 3 shows the RED SLIPPERS and CHRISTMAS ORNAMENTS scenes once more, this time rendered using classical smooth microfacet models. These images were faster to generate due to the considerably simpler query operations. The corresponding renderings with the discrete stochastic model took $1.62\times$ and $1.44\times$ longer, respectively.

## 5 Conclusion

This paper has introduced a new kind of multiscale BRDF model that extends the widely used microfacet models to produce controllable, random spatial variation that exhibits all the characteristics of glittery appearance, including temporal coherence in animation, without explicitly storing any surface detail information.

One limitation of our current implementation is that it requires texture parameterizations to map surface area to texture-space area with reasonable uniformity, for the observed particle density to be constant across the surface (though preservation of angles is not required). Our method could be extended to allow spatially varying particle density, which would remove this requirement and also add additional expressiveness.

Currently, we set $\gamma$ to a constant value per scene. As mentioned earlier, it is potentially superior to cluster the lighting into a superposition of different-sized cones that drive the $\gamma$ parameter.

Our model can be used to create the intense glints of purposely glittery surfaces in sunlight, but it can also be used in almost any scene to add subtle visual richness to surfaces that just look a bit too smooth and perfect without any noise. When using a Monte Carlo renderer one is often drawn to less-than-converged renderings because surfaces look more realistic with some noise, and our model provides a principled, controllable, and animation-compatible way to introduce that noise.

A back-of-the-envelope analysis begins to explain why so few surfaces look correct in closeups with smooth BRDFs. A remarkable number of glints per pixel is required to reach the threshold of noise visibility—about 400 for around 5% RMS noise [Fairchild and Johnson 2005]. In high-resolution closeups, pixel footprints can easily be 100 microns or smaller, and fitting that many glints begins to run up against the limits of geometric optics. Even features just 5 to 10 microns in size (for example, the dents on a bead-blasted aluminum notebook computer) will produce visible glitter under sharp illumination. In a sense, smooth BRDFs simply don't exist in close-up views of glossy materials.

| Scene | N | $\gamma$ | $\alpha$ | $\overline{\text{nodes}}$ | $\overline{\text{count}}$ | Time |
|---|---|---|---|---|---|---|
| RED SLIPPERS | $10^6$ | $6^\circ$ | 0.1 | 17.19 | 0.11 | 9m 34s |
| CHR. ORNAMENTS | $4{\cdot}10^5$-$9{\cdot}10^6$ | $5^\circ$ | 0.1-0.4 | 50.32 | 1.51 | 36m 17s |
| SPHERE LIGHTS | $10^9$ | $0.5^\circ$ | 0.1 | 43 | 0.66 | 3m 25s |
| STAINED GLASS | $10^8$ | $1^\circ$ | 0.01 | 104 | 10.71 | 6m 0s |
| TWISTED TORUS | $10^5$-$10^8$ | $1^\circ$ | 0.03-0.81 | 9.6-36.6 | $3{\cdot}10^{-6}$-0.53 | 54s-99s |

**Table 1:** *Scene parameters and statistics collected during rendering: $N$ is the total number of particles, $\gamma$ the query radius, and $\alpha$ the Beckmann roughness. The overlined quantities denote the average number of tree nodes visited and particles found per query.*

## 6 Acknowledgments

The authors are indebted to Olesya Isaenko, who kindly designed the scenes shown in Figure 1. Emily Whiting provided voice narration for the video. Funding for this work was provided by NSF grants 1011832, 1011919, and the Intel Science and Technology Center for Visual Computing. Wenzel Jakob was supported by an ETH/Marie Curie fellowship. We acknowledge equipment and support from NVIDIA, Nokia and Samsung.

## References

ASHIKHMIN, M., AND SHIRLEY, P. 2000. An anisotropic Phong BRDF model. *J. Graph. Tools 5*, 2 (Feb.), 25–32.

BOOTH, J. 1844. IV. on the rectification and quadrature of the spherical ellipse. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 25*, 163, 18–38.

BOOTH, J. 1852. Researches on the geometrical properties of elliptic integrals. *Philosophical Transactions of the Royal Society of London 142*, 311–416.

BOYD, S., AND VANDENBERGHE, L. 2004. *Convex Optimization*. Cambridge University Press, New York, NY, USA.

COOK, R. L., AND DEROSE, T. 2005. Wavelet noise. *ACM Trans. Graph. (SIGGRAPH 2005 Proceedings) 24*, 3, 803–811.

COOK, R. L., AND TORRANCE, K. E. A reflectance model for computer graphics. In *Proceedings of SIGGRAPH '81*, ACM.

DEKKER, N., KIRCHNER, E. J. J., SUPÈR, R., VAN DEN KIEBOOM, G. J., AND GOTTENBOS, R. 2010. Total appearance differences for metallic and pearlescent materials: Contributions from color and texture. *Color Research & Application 36*, 1 (Dec.), 4–14.

DEVROYE, L. 1986. *Non-Uniform Random Variate Generation*. Springer-Verlag.

ERSHOV, S., KHODULEV, A., AND KOLCHIN, K. 1999. Simulation of sparkles in metallic paints. In *Proceedings of Graphicon 1999*, 121–128.

ERSHOV, S., KOLCHIN, K., AND MYSZKOUWSKI, K. 2001. Rendering pearlescent appearance based on paint-composition modeling. *Comp. Graphics Forum (Proc. EUROGRAPHICS) 20*, 3.

FAIRCHILD, M. D., AND JOHNSON, G. M. 2005. On the salience of novel stimuli: Adaptation and image noise. In *Proceedings of the IS&T/SID Color Imaging Conference*, 333–338.

GALASSI, M. E. A. 2009. *GNU Scientific Library Reference Manual*, 3rd ed ed. Network Theory Ltd.

GÜNTHER, J., CHEN, T., GOESELE, M., WALD, I., AND SEIDEL, H.-P. 2005. Efficient acquistion and realistic rendering of car paint. In *VMV 2005 Proceedings*.

HAN, C., SUN, B., RAMAMOORTHI, R., AND GRINSPUN, E. 2007. Frequency domain normal map filtering. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2007) 26*, 3, 28:1–28:12.

IGEHY, H. 1999. Tracing ray differentials. In *Proceedings of SIGGRAPH '99*, ACM Press/Addison-Wesley Publishing Co.

JAKOB, W., 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

KACHITVICHYANUKUL, V., AND SCHMEISER, B. W. 1988. Binomial random variate generation. *Commun. ACM 31*, 2 (Feb.), 216–222.

KELLER, A., PREMOZE, S., AND RAAB, M. 2012. Advanced (quasi) monte carlo methods for image synthesis. In *ACM SIGGRAPH 2012 Courses*, ACM, SIGGRAPH '12, 21:1–21:46.

KIRCHNER, E., VAN DEN KIEBOOM, G.-J., NJO, L., SUPÈR, R., AND GOTTENBOS, R. 2007. Observation of visual texture of metallic and pearlescent materials. *Color Research & Application 32*, 4, 256–266.

LAGAE, A., LEFEBVRE, S., DRETTAKIS, G., AND DUTRÉ, P. 2009. Procedural noise using sparse Gabor convolution. *ACM Trans. Graph. 28*, 3 (July), 1.

MCCAMY, C. S. 1996. Observation and measurement of the appearance of metallic materials. Part I. Macro appearance. *Color Research & Application 21*, 4, 293.

MCCAMY, C. S. 1998. Observation and measurement of the appearance of metallic materials. Part II. Micro appearance. *Color Research & Application 23*, 6, 362–373.

OLANO, M., AND BAKER, D. 2010. Lean mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, I3D '10, 181–188.

PERLIN, K. 2002. Improving noise. *ACM Trans. Graph. (SIGGRAPH 2002 Proceedings) 21*, 3, 681–682.

RUMP, M., MÜLLER, G., SARLETTE, R., KOCH, D., AND KLEIN, R. 2008. Photo-realistic rendering of metallic car paint from image-based measurements. *Computer Graphics Forum (EUROGRAPHICS Proceedings) 27*, 2.

SCHNEIDER, P., AND EBERLY, D. H. 2002. *Geometric tools for computer graphics*. Morgan Kaufmann.

TOKSVIG, M. 2005. Mipmapping normal maps. *Journal of Graphics Tools 10*, 3, 65–71.

TORRANCE, K. E., AND SPARROW, E. M. 1967. Theory for off-specular reflection from roughened surfaces. *JOSA 57*, 9, 1105–1112.

VEACH, E. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University.

WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, 195–206.

WHEELER, D. J., AND NEEDHAM, R. M. 1995. TEA, a tiny encryption algorithm. In *Fast Software Encryption*, Springer, 363–366.

YAN, L.-Q., HAŠAN, M., JAKOB, W., LAWRENCE, J., MARSCHNER, S., AND RAMAMOORTHI, R. 2014. Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2014) 33*, 4.