Fast Global Illumination with Discrete Stochastic Microfacets Using a Filterable Model
Beibei Wang, Lu Wang, Nicolas Holzschuch

To cite this version:

HAL Id: hal-01856980
https://inria.hal.science/hal-01856980
Submitted on 13 Aug 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Fast Global Illumination with Discrete Stochastic Microfacets Using a Filterable Model

Beibei Wang\textsuperscript{1,2}, Lu Wang\textsuperscript{3}, Nicolas Holzschuch\textsuperscript{4}

\textsuperscript{1}School of Computer Science and Engineering, Nanjing University of Science and Technology
\textsuperscript{2}Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education
\textsuperscript{3}School of Software, Shandong University
\textsuperscript{4}Inria, Univ. Grenoble-Alpes, CNRS, LJK.

Abstract

Many real-life materials have a sparkling appearance, whether by design or by nature. Examples include metallic paints, sparkling varnish, but also snow. These sparkles correspond to small, isolated, shiny particles reflecting light in a specific direction, on the surface or embedded inside the material. The particles responsible for these sparkles are usually small and discontinuous. These characteristics make it difficult to integrate them efficiently in a standard rendering pipeline, especially for indirect illumination. Existing approaches use a 4-dimensional hierarchy, searching for light-reflecting particles simultaneously in space and direction. The approach is accurate, but still expensive. In this paper, we show that this 4-dimensional search can be approximated using separate 2-dimensional steps. This approximation allows fast integration of glint contributions for large footprints, reducing the extra cost associated with glints be an order of magnitude.

1. Introduction

Many materials have a sparkling or glittering appearance. Examples include some car paints, glittery lipsticks but also natural materials such as snow or sand. These sparkles are an essential part of the material appearance. Simulating them is essential for photorealistic rendering, but also difficult. These sparkles are caused by small patches, with random distribution. These patches are not necessarily part of a continuous surface. Each individual patch provides a bright reflection for a small set of directions, and is not visible otherwise.

As the patches are very small, sampling them accurately is time consuming, and they risk causing noise or aliasing by being under-sampled. We need the glints to have a random distribution, but we also require temporal coherence in animations.

Jakob et al. [JHY\textsuperscript{14}] introduced a discrete stochastic microfacet model for glints. They represent glints as small specular surfaces, as in the microfacet model [CT82] [WMLT07], but with a discrete
distribution instead of a continuous surface. These specular particles are organized in a hierarchy. They extend the microfacet model from a point-direction model to an area-solid angle model and a 4-dimensional search in that hierarchy, to find the number of particles that are located in the query domain, both in space and angle. These particles are used to compute the distribution function and the reflectance.

Their algorithm is purely procedural, without the need to store or compute any textures. It is also physically-based, and temporally coherent. Due to the 4-dimension traversal, computing glints is more expensive than computing standard reflectance model. The extra cost is acceptable for direct illumination, but becomes prohibitive for global illumination.

Our contributions in this paper are two fold: first, we show that the 4-dimension hierarchy traversal can be approximated with two separate 2-dimension traversals, one in the spatial domain and one in the angular domain. This decoupling allows several optimizations: we locate the spatial position directly, eliminating the need for a top-down traversal and we precompute the angular probability of glints depending on direction. Second, we use this simplified model for automatic filtering in global illumination: we compute the footprint of secondary rays and integrate the glints contribution directly. The global model results in very fast global illumination, 30 times faster than the original algorithm on complex scenes.

We review previous work on rendering glints and sparkles in the next section. We then present the original discrete stochastic microfacet model in Section 3. We describe our own algorithm in Section 4. In Section 5, we compare our method with previous works and reference solutions. We conclude in Section 6.

2. Previous Work

2.1. Glint Rendering

Jakob et al. [JHY’14] proposed a stochastic model based on microfacet theory to simulate glittery surfaces without any textures. Their algorithm is purely procedural, without the need to store or compute any textures. It is also physically-based, and temporally coherent. Due to the 4-dimension traversal, computing glints is more expensive than computing standard reflectance model. The extra cost is acceptable for direct illumination, but becomes prohibitive for global illumination.

Our algorithm extends over Jakob et al. [JHY’14]. We begin by a short description of this work in Section 3. Our goal is to produce glittery effects quickly, with a lesser focus on physical accuracy.

2.2. Real-time Glint Rendering

Shopf [Sho12] used a 3D jittered grid of sparkle shapes to render sparkles in the snow surface in real-time. Bowles and Wang [BW15,BW16] introduced a procedural sparkle approach for snow; this technique has been used in production, Zirr et al. [ZK16] derived a stochastic biscale microfacet model to fit for real-time application. All these algorithms are faster than ours, but are not physically based.

Figure 2: Our algorithm: In a preprocessing step, we compute and store the probability for a particle to be reflected around the direction $\omega_r$ with a tolerance of 1 degree. This probability is indexed by the incoming and outgoing directions $\omega_i, \omega_o$. For each scene, given the scene-specific search range $\gamma$, we accumulate table values in the search range. We then prefilter the new table using Gaussian blurring to build a hierarchical representation. During rendering (right), we trace the path footprint for each surface interaction, and use it to compute the contribution. Depending on the footprint size, we either compute individual glint contributions or the average contribution from all glints.
3. Background: Discrete Stochastic Microfacet Model

3.1. Description

Jakob et al. [JHY+14] introduced the discrete stochastic microfacet model for glittery surface. Their algorithm extends the microfacet BRDF model to take into account a finite extent in space and angle, instead of diracs:

$$
\hat{f}(A, \omega_i, \Omega_o) := \frac{1}{a(A)\sigma(\Omega_o)} \int \int \int_{\Omega_i} f_i(x, \omega_i, \omega_o) \, d\omega_o \, dx.
$$

Where $A$ is a finite area around point $x$, $\Omega_o$ a finite solid angle around outgoing direction $\omega_o$, $f_i$ is the usual microfacet BRDF [CT82] [WMLT07]:

$$
f(x, \omega_i, \omega_o) = \frac{F(\omega_i \cdot \omega_o)D(x, \omega_i,G(\omega_i,\omega_o,\omega_o))}{4(\omega_i \cdot \omega_i)(\omega_o \cdot \omega_o)},
$$

Where $F$ represents the Fresnel reflection coefficient, $D$ is the microfacet normal distribution, and $G$ is the shadowing and masking term.

If we assume the surface to be made of discrete small mirrors or particles instead of continuous microfacets, we get the new flakes BRDF model:

$$
\hat{f}(A, \omega_i, \Omega_o) = \frac{(\omega_i \cdot \omega_o)F(\omega_i \cdot \omega_o)\hat{D}(A, \Omega_o,G(\omega_i,\omega_o,\omega_o))}{a(A)\sigma(\Omega_o)(\omega_i \cdot \hat{n}_i)(\omega_o \cdot \hat{n}_o)},
$$

where $\hat{D}$ is a sum over the finite set of particles:

$$
\hat{D}(A, \Omega_o) := \frac{1}{N} \sum_{k=1}^{N} I_{\Omega_o}(\omega_i^{k})L_1(x^k).
$$

For each particle $k$, the element in the sum is 1, if its position $x^k$ and half-vector $\omega_i^{k}$ fall within the search domain in space and angle $A \times \Omega_o$. $\hat{D}$ is the number of particles that fall within this search domain. $N$ is the particle count.

A usually corresponds to a pixel footprint, and $\Omega_o$ to a cone of directions around which particles can reflect light. The cone half-angle $\gamma$ is a parameter of the model. In practical applications, it ranges from $1^\circ$ to $6^\circ$.

3.2. Use in Path Tracing

To use the model in path-tracing, for each bounce, we integrate over outgoing directions:

$$
L_o(A, \omega_i, \omega_o) = \int_{\Omega_o} L_i(\omega_i, \omega_o') \hat{f}_o(\lambda, \omega_i, \omega_o') \, d\omega_o'.
$$

4. Filterable Discrete Stochastic Microfacet Models

4.1. Our Separable Model

The main element in the Discrete Stochastic Microfacet Models is the 4-dimensional multiscale microfacet normal distribution $\hat{D}$ (see Equation 4). It corresponds to the count of particles located in the 4-dimensional finite domain $A \times \Omega_o$. To compute this number of particles, the original algorithm uses a 4-dimensional hierarchy traversal. Our first idea is to replace the particle count with a particle probability function, which can be carried out of the integral, allowing the decoupling between space and angle.

Under the assumption of separability, $\hat{D}$ can be considered as a joint distribution of two independent variables of $x$ and $(\omega_i, \omega_o)$. We introduce the directional probability function (DPF), $P(\omega_i, \omega_o, \gamma)$, expressing the probability that a particle exists that reflects light incoming from direction $\omega_o$ into a cone centered around direction $\omega_i$ with half-angle $\gamma$.

We rewrite Equation 4 using this directional probability function:

$$
\hat{D}(A, \omega_i, \omega_o) = \frac{1}{N} \sum_{k=1}^{N} I_{\Omega_o}(\omega_i^{k}L_1(x^k))P(\omega_i, \omega_o, \gamma)
$$

The directional probability function $P(\omega_i, \omega_o, \gamma)$ is expressed as

$$
P(\omega_i, \omega_o, \gamma) \approx \frac{1}{N} \sum_{k=1}^{N} I_{\Omega_o}(\omega_o^{k}),
$$

The directional probability function $P$ represents a continuous approximation of the flakes reflection. To reproduce the flakes discrete behaviour, we threshold it against a random variable of position (See Figure 3), $x(x)$:

$$
\hat{D}(A, \omega_i, \omega_o) = \frac{1}{N} \sum_{k=1}^{N} I_{\Omega_o}(x^k)H(\lambda(x) - P(\omega_i, \omega_o, \gamma))
$$

To compute this integral in practice, the flakes BRDF model $\hat{f}_i$ is sampled to obtain the next outgoing direction $\omega_o^*$. The spread of outgoing directions depends on material roughness: for rough materials, the discrete stochastic model has a large variance. A large number of samples are then required for convergence, even only considering direct illumination.
The traversal in angle corresponds to a look-up in our precomputed tables: we first select the appropriate hierarchical level, depending on material roughness and incoming light’s frequency. We then extract the precomputed value for $P$ at this hierarchical level.

For the traversal in space, we conduct the traversal in texture space, as in [JHY14] and assume flakes are distributed uniformly in this space. Having separated the spatial and angular traversals allows us to speed up the traversal (see Figure 5): we begin with the axis-aligned bounding box of the pixel footprint in texture space, and use it to find the starting point of the traversal. For each node encountered in the traversal, we compute its intersection with the pixel footprint, and use it to estimate the number of flakes generated. For each of these flakes we generate a random value, seeded with node ID. If this value is larger than one, we continue the traversal. Otherwise, we stop and record the result.

Algorithm 1 Spatial Traversal

\begin{verbatim}
function $\tilde{D}$($A$, $\omega_i$, $\omega_o$)
  query ← $A$
  count ← 0
  $p$ ← $P(\omega_i, \omega_o)$
  while query ≠ $\emptyset$ do
    node ← query.pop()
    if node $\cap$ query $=$ $\emptyset$ or $\lvert$node$\rvert$ = 0 then pass
    else if node $\subseteq$ query then
      $n_i$ ← $\lvert$node$\rvert$
      for $i < n_i$ do $\psi$ ← $\lambda$($k$)
        if $\psi > p$ then count ← count + 1
      end if
    end if
    else if error criterion satisfied then
      overlap ← (node $\cap$ query).vol() / node.vol()
      $n_k$ ← $\lvert$node$\rvert$
      for $i < n_k$ do $\psi$ ← $\lambda$($k$)
        if $\psi > p$ then count ← count + overlap
      end if
    end if
    else for $c$ in node.split() do queue.push(c)
  end for
  end if
end while
return count
end function
\end{verbatim}

References:

than the probability evaluated during the search in the directional domain, then the particle will be reflected (see Algorithm 1).

The area corresponding to a pixel footprint is much smaller than the texture space of the object. Jumping straight to the hierarchical level corresponding to its bounding box greatly speeds up the spatial hierarchy traversal.

4.5. Filterable Glint Computation

The model described in the previous sections is an approximation of the stochastic microfacet model. By itself, it provides a significant reduction in computation time. But its strongest advantage is that it allows filterable glint computation: to compute the average contribution from glints in a certain area $A$, we only have to multiply the Directional Probability Function $P$ by the surface area:

$$\hat{D}(A, \omega_i, \omega_o) = a(A) \times P(\omega_i, \omega_o, \gamma), \quad (10)$$

where $a(A)$ is the area of $A$. The estimated corresponding glint count is $N \times \hat{D}$.

This allows us to include glints in global illumination simulation efficiently:

- For each path, compute the path footprint at each light bounce (see Section 4.6).
- Compute the average glint contribution at this footprint using Equation 10.
- If the glint count is larger than a given threshold, use the average contribution.
- Otherwise, use the separable model described in Section 4.4.

The idea behind the algorithm is that, if a footprint covers a large surface area, individual glints are not noticeable, and the average contribution is what matters. In our experiments, 16 is a good threshold to switch from the accurate model to the filtered model.

In practice, this makes us switch to the filtered model when the glint material is far from the camera, or after several bounces in global illumination.

4.6. Approximate Footprint Computation

For each pixel, for the first intersection with geometry, we use ray differentials, as described by Igehy [Ige99].

For further interactions along the path, we keep tracing the main ray (the one centered at $p$) as well as the offset rays, with origin $p_s$ (resp. $p_o$) and direction $r_s$ (resp. $r_o$). For specular bounces, the outgoing direction of the offset ray $r'_o$ is the specular bounce of the incoming direction $r_s$. For glossy reflections, we first compute the specular bounce, extend the direction using a glossy cone angle, depending on the BSDF (see Figure 6). We used the glossy cone angle approximation defined by Günther et al. [GRG14]. For the Phong BSDF model, the angle for extension $\beta$ at each bounce is approximated as:

$$\beta \approx 2\sqrt{\frac{2}{g + 2}},$$

where $g$ is the phong exponent. For the Cook-Torrance model with the Beckmann normal distribution function, we use:

$$\beta \approx 2\alpha$$

where $\alpha$ is the roughness of the material.

We then convert the spatial footprint into screen space derivatives.

5. Results and Discussion

We have implemented our algorithm inside the Mitsuba Renderer [Jak10]. We compared our algorithm against [JHY’14] which we consider as the reference for quality validation. We use path tracing as our global illumination algorithm.

All timings in this section are measured on a 2.20GHz Intel i7 (40 cores) with 32 GB of main memory. Unless otherwise stated, we use the Beckmann microfacet model as the underlying smooth distribution $D$, which has a single roughness parameter with lower values corresponding to smoother surfaces.

Figures 1, 8, 9 and 10 show our algorithm (using filtering) with global illumination; Figure 7 shows direct illumination only.

5.1. Qualitative Validation

We first compare our method with the reference solution, Stochastic Microfacet Distribution, by Jakob et al. [JHY’14]. Figure 7 focuses on direct illumination. Qualitatively, our approach produces results that are visually identical to the reference. In terms of computation time, we have reduced the extra cost introduced by glints by a factor of 5.

In Figure 8, we do the same comparison, but with indirect illumination and reflection on a glossy floor. Visually, the results are,
B. Wang & L. Wang & N. Holzschuch / Fast Global Illumination with Discrete Stochastic Microfacets Using a Filterable Model

Our model (with LOD),
Total: 1.9 min, Cost: +12%

Original [JHY*14],
Total: 2.7 min, Cost: +65%

Without Glints,
Total: 1.7 min

Our model (no LOD),
Total: 1.9 min, Cost: +12%

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

Figure 7: Comparison with reference [JHY*14] on the Shoe Scene with direct illumination only.

Figure 8: Comparison with reference [JHY*14] with Shoes on a glossy floor Scene. The first column displays whether we use the accurate model (red) or the filtered version (blue). For the first two rows, the floor is highly specular, so we use the finest version everywhere; the computation cost is the same with and without filtering. For the third row, because of the rough floor, we switch to the filtered version more often, resulting in a significant cost reduction.

For a glossy floor (roughness $\alpha = 0.1$, bottom row), the computation time increases significantly because of the larger footprint caused by the glossy reflection. Our method without filtering is about 8 times faster than the reference, but still quite expensive compared to the scene without glints. The filtered version is triggered for distant reflections, and results in a significant speedup.

One key advantage of our filtered algorithm is its predictability: the extra cost caused by the introduction of glints remains almost constant, between 1.3 and 1.6 mm, independent of the rest of the scene. For comparison, increasing glossiness causes the extra computation time for the original algorithm to raise from 1.7 mm to 36.2 mm.

The same behaviour is apparent in Figures 9 and 10: Our model without filtering provides a significant speedup (8 times faster for the Balls Scene, 6 times faster for the Car Scene). With filtering, the speedup is much larger (more than 25 times faster). With filtering activated, adding glints to the scene amounts to a small extra computational cost, from 15 % to 20 %.

5.2. Performance and Timings

Table 1 displays the settings for the materials and timings for all our test scenes. For all test scenes, we report the computation time...
without glints, and the computation time for the reference method by Jakob et al. [JHY*14] and our method with filtering. To compare methods, we report both the total computation time, and the extra time associated with glint computation.

The overall speedup depends on the complexity of the computations, and the impact of filtering. For direct illumination only, or specular reflections, filtering is not used, and the separable glint model provides a 5 times speedup. For more complex scenes, with indirect illumination and glossy reflections, the speedup is more important, about 30 times faster.

The precomputed directional probability function is extremely fast, and introduces a modest extra memory cost. Table 2 shows computation times and memory costs as a function of material roughness.

The overall speedup depends on the complexity of the computations, and the impact of filtering. For direct illumination only, or specular reflections, filtering is not used, and the separable glint model provides a 5 times speedup. For more complex scenes, with indirect illumination and glossy reflections, the speedup is more important, about 30 times faster.

The precomputed directional probability function is extremely fast, and introduces a modest extra memory cost. Table 2 shows computation times and memory costs as a function of material roughness.

5.3. Parameter Analysis

Figures 11 and 12 show the impact of glint parameters on our algorithm:

- Figure 11 displays computation time as a function of the search angle \( \gamma \) for the Car Scene (Figure 10). Computation time for our algorithm is a constant, independent from this parameter. Computation time for [JHY*14], on the other hand, increases with \( \gamma \), as they are required to search for particles in a larger domain. As a consequence, the larger the search angle, the more important the speedup for our method.

- Figure 12 displays computation time as a function of the particle count \( N \) for the Car Scene (Figure 10). Increasing the number of particles increases the computation time for both [JHY*14] and our method, but the slope for our method is much smaller, resulting in a smaller increase in computation time.

Figures 13 and 14 show the impact of global illumination parameters on our algorithm:

- Figure 13 displays the impact of varying material roughness of the floor material in the Shoe Scene (see results in Figure 8). Increasing roughness increases computation time for all methods, due to the larger footprint for integration. Using our model with filtering keeps this increase manageable, thanks to our fast approximation for large footprints.

- Figure 14 displays computation time as a function of path tracing depth for the Balls Scene (Figure 9). A low value (2) corresponds to only direct illumination. In this setting, the cost of the reference algorithm [JHY*14] is not too high, though substantially more expensive than our method. As the depth increases, we compute more light bounces, increasing the computation time. The slope is much larger with the reference method than with our method using filtering. The computation time levels after 4 bounces, probably due to the scene configuration.

An important parameter for our method is the threshold for...
Figure 10: Comparison with reference [JHY*14] on the Car Scene with global illumination.

Figure 11: Computation time for reference [JHY*14], our method and rendering without glints, as a function of the searching angle $\gamma$, for the Car Scene, particle count $3 \times 10^7$ and roughness $\alpha = 0.1$ (direct illumination only).

Figure 12: Computation time for reference [JHY*14], our method and rendering without glints, as a function of the number of particles $N$, for the Car Scene with $\gamma = 6$ and roughness $\alpha = 0.1$ (direct illumination only).

Figure 13: Comparison between our algorithm (with and without filtering), reference [JHY*14] and without glint with varying roughness for Shoe Scene (global illumination).

Figure 14: Comparison between our algorithm (with and without filtering), reference [JHY*14] and computation without glint with varying path tracing depth for Balls Scene (global illumination).
switching from individual glint computation to filtered glint representation. We express it as a maximum number of glints in the footprint. Figure 15 displays the algorithm behaviour as we change this parameter, using the Shoe Scene (Figure 8), with a roughness of $\alpha = 0.1$. We compute both the error, evaluated by comparing our algorithm with filtering and without filtering (left scale), and the rendering cost, measured as the extra computation time compared to the same scene without glints (right scale). Increasing the filter threshold decreases the error, at the expense of rendering time. 16 glints as a threshold for switching to the filtered version appears to be a reasonable compromise.

5.4. Discussion and limitation
Our algorithm relies on several approximations and assumes a large number of flakes. The approximations fail for a smaller number of flakes, as in Figure 16. Our method is still much faster, but the glints do not have the accurate distribution provided by the reference method. Our method cannot simulate flakes that have a spatial-angular correlated distribution.

Also, glitter has a distinctive angular behaviour. If individual glints are visible, as the half vector moves through the hemisphere, the pixel value flickers brighter and darker. Our method does not keep this behaviour, as the DPF is a monotonically increasing function when moving towards to the normal direction. The thresholding function cannot provide this brighter and darker appearance. However, the effect is only visible with point lights, and becomes invisible with environment lighting or indirect lighting.

For applications where this flake behaviour is essential, we suggest keeping the original method [JHY*14] for close-range, directly visible glints, and switch to our method for indirect illumination and elements further away from the camera.

6. Conclusion
We have presented a fast model for computing an approximated stochastic microfacet model for glints and flakes in materials. We replace the 4-dimensional search for glints with two 2-dimensional searches, for a significant speedup.

This separation between space and angle in the search allows for further improvements: hierarchical representation for the directional table, jumping straight to the hierarchical level for the spatial search. We also prefilter the directional table, depending on the material roughness for faster convergence in low-frequency environments. Finally, we use this simplified model for automatic filtering in global illumination: we compute the footprint of secondary rays and integrate the glints contribution directly.

Our algorithm is especially interesting for indirect lighting and global illumination, where it is more than 30 times faster than the reference, without any visible quality degradation. Our algorithm is highly consistent: rendering time does not depend on material parameters, making it easy to predict the total rendering time for a given scene. This is an important feature for practical use.

In future work, we want to extend the filtered model to scratches and other local surface details. We also plan to port the model for interactive rendering on the GPU.

Acknowledgements. We thank the reviewer for the valuable comments. We thank the following blendswap.com artists for providing the scenes used in this paper: Angelo Borraccino (Kitchen) and Neil Taylor (Car). This work has been partially supported by the National Key R&D Program of China under grant No. 2017YFB0203000, the National natural science foundation of China under grant No. 61472224, 61472225, U1713208, 61472187, the 973 Program under grant No. 2014CB349303, Program for Changjiang Scholars, the National Science Foundation of Jiangsu under grant No. BK20170857, the fundamental research funds for the central universities No. 30918011320, Open Project Program of the State Key Lab of CAD&CG Zhejiang University under grant No. A1804 and ANR project ANR-15-CE38-0005 "Materials".

References