# Evaluating and Sampling Glinty NDFs in Constant Time

PAULI KEMPPINEN, Adobe Research & Aalto University, Finland
LOIS PAULIN, Adobe Research, France
THÉO THONAT, Adobe Research, France
JEAN-MARC THIERY, Adobe Research, France
JAAKKO LEHTINEN, NVIDIA & Aalto University, Finland
TAMY BOUBEKEUR, Adobe Research, France

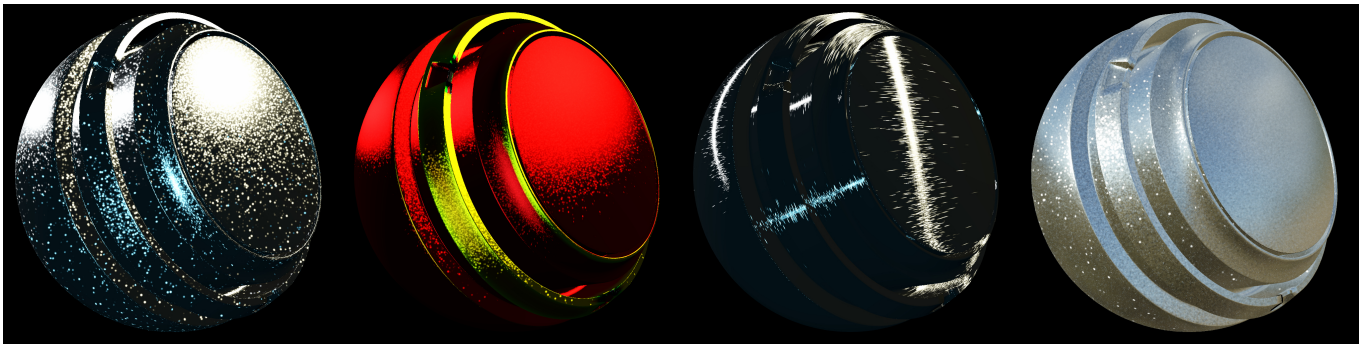


Fig. 1. From left to right: our glittery version of the a microfacet BRDF with a Trowbridge-Reitz (GGX) normal distribution, a Beckmann variant showing per-facet color (red/yellow) control, an anisotropic variant and an image-based lighting setup illustrating the support of importance sampling.

Geometric features between the micro and macro scales produce an expressive family of visual effects grouped under the term "glints". Efficiently rendering these effects amounts to finding the highlights caused by the geometry under each pixel. To allow for fast rendering, we represent our faceted geometry as a 4D point process on an implicit multiscale grid, designed to efficiently find the facets most likely to cause a highlight. The facets' normals are generated to match a given micro-facet normal distribution such as Trowbridge-Reitz (GGX) or Beckmann, to which our model converges under increasing surface area. Our method is simple to implement, memory-and-precomputation-free, allows for importance sampling and covers a wide range of different appearances such as anisotropic as well as individually colored particles. We provide a base implementation as a standalone fragment shader.

CCS Concepts: • **Computing methodologies** → **Reflectance modeling**.

Additional Key Words and Phrases: Rendering, Glints, BRDF, High Frequency Reflectance

**ACM Reference Format:**
Pauli Kemppinen, Lois Paulin, Théo Thonat, Jean-Marc Thiery, Jaakko Lehtinen, and Tamy Boubekeur. 2025. Evaluating and Sampling Glinty NDFs in Constant Time. *ACM Trans. Graph.* 44, 6, Article 255 (December 2025), 11 pages. https://doi.org/10.1145/3763282

Authors' addresses: Pauli Kemppinen, Adobe Research & Aalto University, Finland, pauli.kemppinen@aalto.fi; Lois Paulin, Adobe Research, France, paulin@adobe.com; Théo Thonat, Adobe Research, France, thonat@adobe.com; Jean-Marc Thiery, Adobe Research, France, jthiery@adobe.com; Jaakko Lehtinen, NVIDIA & Aalto University, Finland, jaakko.lehtinen@aalto.fi; Tamy Boubekeur, Adobe Research, France, boubek@adobe.com.



## 1 INTRODUCTION

Product design industries, such as automotive, electronics or fashion, extensively utilize sophisticated materials characterized by "glints" – specular reflections resulting from microflakes embedded in the material structure, or fine geometric detail. Accurately modeling and rendering these materials in real time, such as car paints, brushed metals or complex plastics, was previously addressed using various trade-offs between computational performance and visual fidelity. Glint rendering amounts to a special-purpose anti aliasing solution; the goal is to efficiently evaluate the integral between the complicated surface reflection and the pixel filter. In offline rendering scenarios, glints can be represented explicitly either as points or highly detailed normal maps using acceleration structures to search for the sparse reflections [Jakob et al. 2014; Wang et al. 2020]. These methods provide the full range of visual effects of glints and offer high quality, but are computationally expensive.

In real-time scenarios [Deliot and Belcour 2023; Zirr and Kaplanyan 2016], the common strategy is to generate a stochastic but spatially-and-angularly-coherent approximation of the material response that *could* have been produced by the exact surface. This strategy leads to inconsistencies between different levels of detail that have a low visual impact due to the flickering nature of glints. However, by discarding the representation of micro-scale structures they cannot offer tools requiring access to the BRDF such as importance sampling, which is crucial to handle complex lighting situations such as image-based lighting. This deficiency makes them much lower quality than the best offline methods. We reach real-time performance while obtaining high-quality results by making

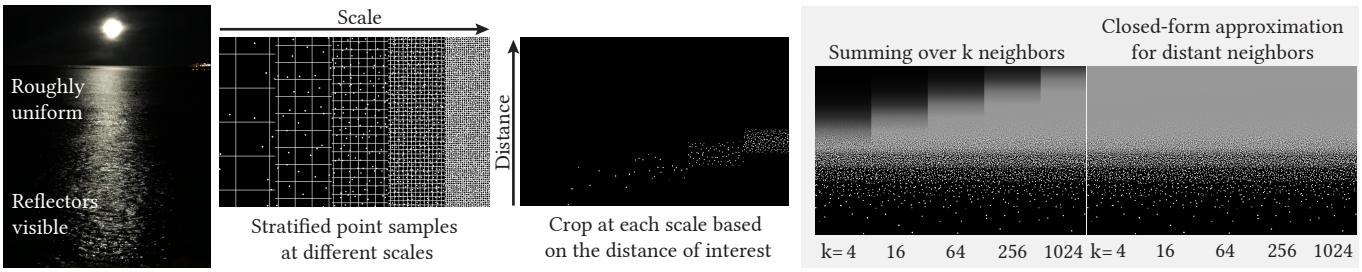



Fig. 2. An intuitive view of our point-based strategy. Left: a photograph depicting glints at different distances. Middle left: stratified point samples at multiple scales. Center: restriction of each scale to its relevant distance range. Middle right: sum over the scales using a growing number $k$ of neighboring cells. Right: as the distribution tends to become uniform when denser, we substitute the sum with a closed-form approximation for further neighbors, inspecting only a fixed number of cell at each scale. While visual discontinuities remain at transitions between levels, those are mitigated by our roulette-based mip-mapping scheme.

our glinty model compatible with the necessary features, such as importance sampling.

To do so, we represent glint rendering as a 4D point process, but avoid complex geometric structures or potentially unbounded computations even for highly anisotropic levels of detail, and model glints implicitly as a multiscale generation process and choose the right process per viewing condition. This results in a small, constant per-pixel computational cost coping with real-time frame rates at high screen resolution on commodity hardware, yet preserving a physically-grounded behavior.

*Contributions.* We present an easy-to-implement real-time glint rendering method, relying on an implicit multiscale grid to leverage spatial and angular resolution, that integrates seamlessly into standard microfacet-based shading models and supports **(i)** an explicit pixel filter, improving rendering quality and temporal stability compared to grid interpolation, **(ii)** importance sampling, **(iii)** anisotropic glints, **(iv)** individual glint colors, and **(v)** UV-free geometry. We also provide a new Russian Roulette based interpolation to reduce the visual impact of interpolation on glints intensity.

## 2 RELATED WORK

*Offline methods.* Offline glint rendering has been greatly researched, either using point models [Jakob et al. 2014], normal maps [Wu et al. 2025; Yan et al. 2014] or high-dimensional distributions such as the P-NDF [Wu et al. 2025; Yan et al. 2016]. These methods build upon earlier foundations [Ďurikovič and Martens 2003; Ershov et al. 1999, 2001] addressing specific use cases such as car paint and pearlescent appearances.

Recently, Chermain et al. [2021b] proposed an importance sampling scheme coping with the multi-lobe visible normal distribution functions (NDFs) of the glittering NDF proposed in Chermain et al. [2020], their core idea being to tabulate and store the CDF of the 1D marginal distribution of the corresponding slope distribution. Deng et al. [2022] proposed to precompute and compress range queries over the NDF to reach a constant space representation NDF.

Neural models [Kuznetsov et al. 2019] have also been used as a constant space and compute time solution for glint rendering. Shah et al. [2024] used neural histograms generating per-pixel NDFs with arbitrary positions and sizes, offering efficient memory usage, yet remaining far outside the real-time realm. Similarly, methods based on normal maps [Wu et al. 2025; Yan et al. 2014] incur significant memory cost.

*Real-time methods.* Zirr & Kaplanyan [2016] pioneered the use of a binomial law to statistically count the number of reflective facets for a given pixel footprint. Although fast, their approach presents blending artifacts for distant rendering and pixel footprints which map to anisotropic surface regions.

Chermain et al. [2020] modeled glinty BRDFs using a dictionary of 1D marginal distributions to represent the core NDF. Their procedural method is physically-grounded and converges to the smooth BRDF but is restricted to the Beckmann case.

Deliot & Belcour [2023] proposed a stable statistical counting of glints over the pixel footprint, based on an implicit grid structure of on-surface random numbers which handles anisotropic filtering at constant complexity. Their framework is compatible with arbitrary NDFs and guarantees stable performance for real-time applications but is not compatible with importance sampling for e.g., image-based lighting and has a linear cost in the number of light sources, which is a common pitfall for methods that do not handle area lights or image based lighting.

Wang et al. [2020] precompute directional probability functions stored in GPU buffers together with a spatial tree structure used to count glints at runtime. These precomputed structures enable real-time glint rendering but induce a severe memory footprint and are not suited for fully dynamic scenarios.

The challenging case of area lighting has been addressed by Kneiphof & Klein [2024] who extend previous counting methods by combining linearly transformed cosines [Heitz et al. 2016] and locally constant approximations to model the probability of a microfacet to reflect light from an emitter toward an observer.

We build our method upon the key idea of using a random point distribution on the surface to produce a stable glinty appearance. By handling explicit NDFs instead of only synthesizing their visual response as Zirr & Kaplanyan [2016] and Deliot & Belcour [2023] do, we offer crucial features, such as importance sampling, for high-quality production that require complex lighting environments. Contrary to Zirr & Kaplanyan [2016] and Chermain et al. [2020], we offer a model that takes the NDF as an input to ensure the generality and expressivity of our method.

# 3 METHOD

## 3.1 Overview

We propose a transformation of the NDF of a microfacet model into a semi-discrete one matching a glinty material, with glints seen as 4D points – 2 dimensions representing their position on the surface $\mathcal{X}$ and 2 dimensions representing their normal orientation on the hemisphere $\Omega$ (see Fig. 2). We provide an implicit representation of the set of points on $\mathcal{X}$ as the result of a sampling process, which allows direct access and the ability to iterate over the points close to a given position and orientation. We use this process to provide an efficient estimation of the visual response by iterating over highly-contributing facets and using an analytical estimation of low-contribution ones. By using our semi-discrete NDF and iteration method, we offer an importance sampling strategy, which makes our method a viable drop-in replacement of standard NDF models in rendering pipelines.

## 3.2 Deriving the Glinty NDF

We derive our glinty material from the microfacet model of a BRDF. Starting with the Cook-Torrance microfacet model

$$\rho(\omega_o, \omega_i) = \frac{F(\omega_o, \omega_h)G(\omega_o, \omega_h, \omega_i)D(\omega_h)}{4\left|\omega_g \cdot \omega_o\right|\left|\omega_g \cdot \omega_i\right|} \tag{1}$$

and we modify the NDF $D$ to make it glinty; the other terms preserve their usual forms, and we use the Smith masking-shadowing function for $G$ in our work as it is widely used. A V-cavity masking-shadowing function would be a good alternative as its physical assumption of spatially decorrelated normals matches the derivation of our density. $D$ represents a continuous density of microfacet normals; to produce a glinty appearance, we replace it with a spatially-varying distribution $D_\star(x, \omega_h)$ that models a discrete set of reflective particles, queried at location $x \in \mathcal{X}$. We preserve the property that $D_\star$ converges to $D$ when viewed from afar, while letting us see the individual reflectors building up the smooth aggregate BRDF when looking closer. This is equivalent to having the distributions match on average. For $\mathcal{X}$ a surface and $|\mathcal{X}|$ its area:

$$\lim_{|\mathcal{X}|\to\infty} \frac{1}{|\mathcal{X}|} \int_{\mathcal{X}} D_\star(x, \omega_h)\mathrm{d}x = D(\omega_h). \tag{2}$$

One can view the point-based NDF as one possible discrete realization of the continuous one for a finite number of facets. We use a jittered grid sampler to generate uniform samples and a NDF-specific mapping $T$ that maps the NDF $D$ defined on the hemisphere $\Omega$ onto the uniform distribution on some domain $\mathcal{U}_\Omega$ (usually a disk or a square, see Figure 3 for the Trowbridge-Reitz case), which reads mathematically

$$|J_T(\omega)| = D(\omega)/|\mathcal{U}_\Omega|. \tag{3}$$

with $J_T$ the Jacobian of the transformation $T$ and $|J_T|$ denoting the determinant of said Jacobian. In this paper we will use a mapping such that $\mathcal{U}_\Omega$ is a disk inscribed in the unit square.

Once discrete facet orientations are realized we need a per-facet reflection distribution. Previous methods widen the light source [Jakob et al. 2014], use a convolution [Chermain et al. 2021a; Zirr and Kaplanyan 2016] or introduce an ad-hoc multiplier [Deliot and Belcour 2023]). For perfectly specular facets, the Dirac distribution would be the natural choice. However, the macroscopic scale of glints makes their angular response non-singular after aggregation over their covered surface, and we opt for a Gaussian reflection distribution in practice, thus leading to a BRDF that can be easily convolved and integrated within arbitrary intervals, making our model practical. Therefore, we write the NDF as a 2D mixture of Gaussians $\mathcal{N}(x, \mu, \sigma) := \exp(-|x-\mu|^2/(2\sigma^2))/(\sigma\sqrt{2\pi})$ at $x$:

$$D_\star(x, \omega_h) := \frac{|J_T(\omega_h)|}{K_P} \sum_i^{K_P} \mathcal{N}\left(T(\omega_h), \mu_i, \alpha_\star\sqrt{|J_T(\omega_h)|}\right)\delta(x_i - x), \tag{4}$$

$i$ ranging over $K_P$ points, $\mu_i$ and $x_i$ being the orientation and position of point $i$ and $\delta$ the Dirac delta function. The factor of $\sqrt{|J_T(\omega_h)|}$ compensates for the scaling introduced by the mapping T. The average of the lobes is used to preserve the integral of the smooth distribution. The base standard deviation $\alpha_\star$ (Eq. (4)) is referred to as the *microroughness*: that is the roughness of the reflectance distribution of a single facet. Smaller values give more animated and glittery behavior, while larger ones give a more stable look. Figure 3, bottom right, shows $D_\star$ on the hemisphere. Table 1 presents the principled parameters of our glinty NDF.

To ensure fast anti-aliased computation of pixel values, we analytically convolve our NDF with the pixel filter. As commonly done [Heckbert 1989], we find an affine mapping of the screen-space pixel into the texture space, as it is convenient to define our

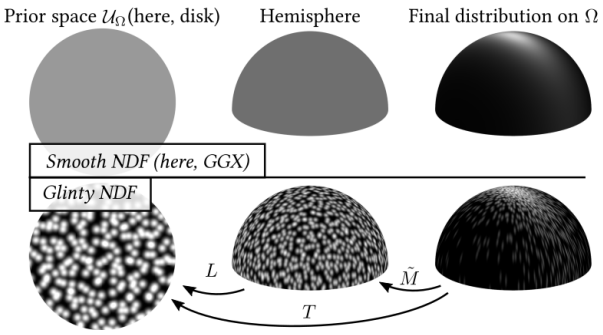



Fig. 3. A geometric view of the map $T : \Omega \mapsto \mathcal{U}_\Omega$ from the Trowbridge-Reitz (GGX) distribution on $\Omega$ (right) to the prior space $\mathcal{U}_\Omega$ (left) via the uniform-energy-hemisphere (middle): $T := L \circ \tilde{M}$. Compared to the smooth (non glittery) NDF (top), our approach (bottom) acts on the prior and *concentrates energy around the facets*.

Table 1. Practical parameterization of our model

| | |
|---|---|
| **Microroughness** | roughness of a single facet $\alpha_\star$ |
| **Roughness** | roughness of the microfacet distribution $\alpha$ |
| **Density** | number of facets per area unit $K_P$ |
| **Anisotropy** | anisotropic extend of the facets $A_\star$ |

process there. The filtered NDF is then

$$D^{\mathrm{f}}_{\star}(x,\omega_h) := \int_{\mathcal{X}} D_{\star}(y,\omega_h) f(y-x,\Sigma_s)\mathrm{d}y \tag{5}$$
$$= \frac{|J_T(\omega_h)|}{K_P} \sum_i^{K_P} \mathcal{N}\left(T(\omega_h), \mu_i, \alpha_{\star}\sqrt{|J_T(\omega_h)|}\right) f(x_i - x, \Sigma_s)$$

$f$ being the filter and $\Sigma_s$ its 2D covariance matrix ($\cdot_s$ indicating action in the spatial domain). Any pixel filter $f$ can be employed [Greene and Heckbert 1986; Heckbert 1989]: in practice, we use a Gaussian.

We have presented our setup for NDF models featuring mappings $T : \Omega \mapsto \mathcal{U}_\Omega$. Importance sampling is commonly done using such mappings, making our work compatible with most NDFs studied in the literature. Next we detail $T$'s construction for the most widely used models: the Trowbridge-Reitz (GGX) and Beckmann models.

*3.2.1 The Trowbridge-Reitz (GGX) NDF.* We use the microsurface transformation framework of Atanasov et al. [2022] to represent different roughness, including anisotropic ones, as a transformation $\tilde{M}$ of the NDF to a uniform distribution over $\Omega$:

$$\tilde{M}(\omega) := M^T\omega/\|M^T\omega\|, \text{ leading to } \left|J_{\tilde{M}}(\omega)\right| = |M|/\left\|M^T\omega_h\right\|^4, \tag{6}$$

$M$ depending on roughness and anisotropy. We compose it with Lambert's area-preserving azimuthal projection [Lambert 1772]

$$L(x,y,z) := (x,y)/\sqrt{1+z}, \tag{7}$$

and obtain $T$ as $T := L \circ \tilde{M}$. Figure 3 details this construction.

*3.2.2 The Beckmann NDF.* It may be tempting to use, for the Beckmann case, the usual mapping $T$ given by the importance sampler of the NDF [Walter et al. 2007]. Unfortunately, the original derivation is given in spherical coordinates, leading to a discontinuity in the azimuth angle and strong distortions near the pole, which distorts the shape of the resulting glints. As the Beckmann NDF is a Gaussian on the tangent plane at $\Omega$'s apex, we can use the sampling strategy of Marsaglia [1962] which produces samples from a Gaussian distribution by scaling a random point on a disk, leading to lower distortion. We start by mapping $\mathcal{U}_\Omega$ to the Gaussian:

$$B_1(x,y) = (x,y)\cdot\sqrt{-\alpha_b^2\log(1-x^2-y^2)/(x^2+y^2)},$$

that we then project onto the hemisphere via normalization:

$$B_2(x,y) = (x,y,1)/\sqrt{1+x^2+y^2},$$

and obtain $T := B_1^{-1} \circ B_2^{-1}$ that we can use in our method.

## 3.3 Procedural glint generation and rendering

We now turn to a practical implementation of the model. Exact evaluation of the glinty BRDF requires summing over all visible facets, which quickly becomes intractable as their number grows. We observe that the contribution of a facet $(x_i, \mu_i)$ highly depends on its proximity in space and angle to the shading location $x$ and the half-vector $\omega_h$. With this in mind, we present a point process built for quick enumeration of the most relevant ones.

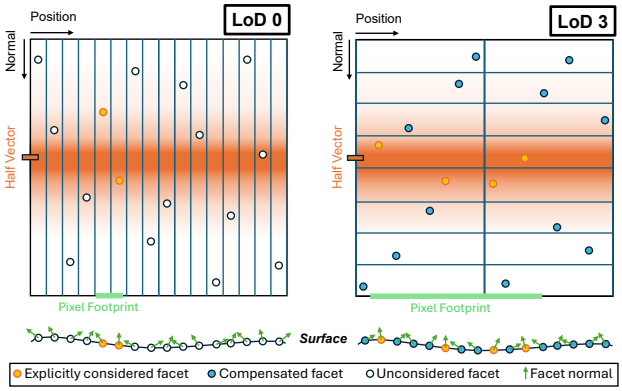



Fig. 4. Our 4D implicit grid structure (here abstracted as a 2D grid), trading off spatial for angular resolution, queried for a given filter in the spatial/angular domains in close-up (left) and far away from the surface (right). Unconsidered points fall outside the pixel footprint and do not contribute to the pixel value. Background color gradient represents individual glint angular response centered on the half-vector between light and view directions.

*3.3.1 Implicit grid-based procedural generation.* Glints are points in the 4D space $[0,1)^2\times\mathcal{U}_\Omega$. We generate them by sampling $[0,1)^4$ and rejecting points outside $\mathcal{U}_\Omega$. We partition $[0,1)^4$ in a 4D implicit grid structure with edge length $\left(\frac{1}{nx}, \frac{1}{ny}, \frac{1}{n\omega_x}, \frac{1}{n\omega_y}\right)$, ending in a total of $K_P := nx \times ny \times n\omega_x \times n\omega_y$ grid cells. We generate all points representing facets by randomly placing a single point in each of these cells, giving in cell $(i,j,\omega_x,\omega_y)$ a point positioned at $\mathbf{x}_{ij\omega_x\omega_y} = \frac{(i,j,\omega_x,\omega_y)+r_{ij\omega_x\omega_y}}{(nx,ny,n\omega_x,n\omega_y)}$ with $r_{ij\omega_x\omega_y} \in [0,1)^4$ given by a random number generator (RNG) seeded by $(i,j,\omega_x,\omega_y)$. We use a congruential RNG as they are fast to seed. Given a cell index, we can thus directly access the position of the single point in it with no memory allocation and very little computation.

*3.3.2 View-dependent Implicit Grid Size.* As shown in Figure 4, a single 2D cell $(i,j)$ of size $\frac{1}{nx}\times\frac{1}{ny}$ in position space contains $n\omega_x \times n\omega_y$ points with normal orientations covering the whole range. We use our implicit grid to quickly iterate over the highest contributing facets. To do so, we adapt our implicit grid parameters to the viewing conditions so the pixel footprint covers a low number of positional cells. We do this by choosing $nx$ and $ny$ so that the pixel footprint size fits between once and twice the cell size. As our number of points $K_P = nx\times ny\times n\omega_x\times n\omega_y$ is constant, decreasing $nx$ and $ny$ increases $n\omega_x$ and $n\omega_y$: we effectively see the same number of points throughout the various levels, and access them differently across those. When seen from afar this allows us to iterate on points in a wider area while restricting ourselves to closer normal orientations around the half-vector. From even further away the angular filter covers a large number of reflecting points – to handle this, we add a compensation term that accounts for the contribution of the points not considered to be close neighbors (see Sec. 3.3.4). This is close in spirit to the gated Bernoulli approximation of Deliot & Belcour [2023], and is intuitively sensible for glint rendering: individual facets only matter when there are only a few of them

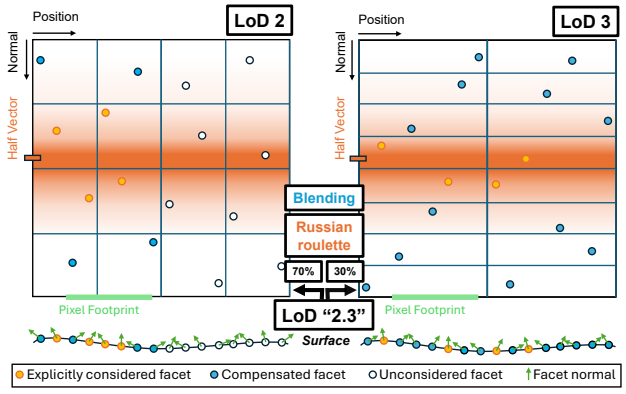



Fig. 5. We use a Russian roulette to randomly select the right portion of evaluated points (in orange) and simply use standard linear blending to interpolate between the compensation terms (in blue) of the two levels. Color gradient represents the angular response of an individual facet centered on the half-vector between light and view directions.

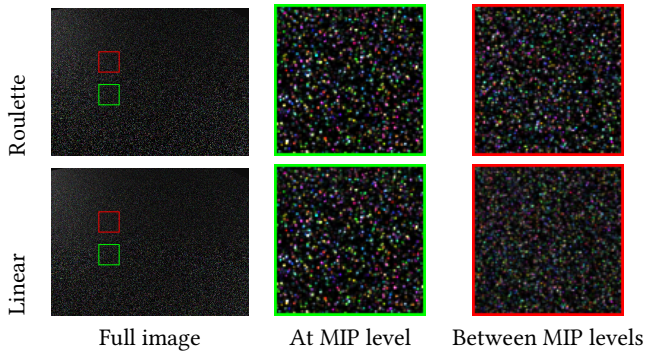



Fig. 6. Blending between MIP levels, using Russian roulette based (top) and linear (bottom) interpolation. The insets show a case with one MIP level contributes most to the image (green), and a case where two levels contribute roughly equally (red). The appearance of linear blending is not consistent when between levels, as averaging two sets of points does not preserve their statistics – at mid level, the density is doubled but the intensity is halved. Our russian roulette based interpolation better preserves the appearance.

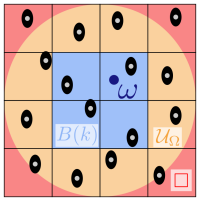
(Figure 2). When the number of facets on the edge of a highlight grows, the appearance quickly converges to that of a smooth BRDF.

*3.3.3 Interpolation via Russian roulette.* As shown by Deliot & Belcour [2023], linear blending of point distributions is not desirable: the appearance at the midpoint of interpolation is that of twice as many glints with half the intensity (Figure 6 bottom right). To this end, inspired by Tokuyoshi & Harada [2017], we use a linearly blended weight for a *per-point* Russian roulette (Figure 5), so that on expectation we get linear blending but individual points get quickly enabled and disabled instead of being smoothly blended (Figure 6 bottom). To keep the appearance smooth under animation, we slightly mollify the roulette by replacing the Heaviside function of the exact roulette by a function that goes from zero to one rapidly but smoothly:

$$R(r, w) = \text{smoothstep}(\max(0, r - \varepsilon), \min(1, r + \varepsilon), w) \tag{8}$$

where $r$ is a uniform random number and $w$ the MIP level weight. We use $\varepsilon = 0.1$ for all of our results.

*3.3.4 Analytic compensation.* When viewed from sufficiently far away, many points contribute to the density. While, ideally, we would like to sum over all points falling inside the pixel footprint, this number grows rapidly with increasing distance to screen, making direct evaluation infeasible. In this case, the exact point positions and orientations become however less significant, and using their expected contribution in place of their exact evaluation works well: $\mathbb{E}_{N\to\infty}\left[\frac{1}{N}\sum_i^N \mathcal{N}(\omega, \mu_i, \sigma)\right] = \int_{\mathcal{U}_\Omega} \mathcal{N}(\omega, \mu, \sigma)\mathrm{d}\mu$.




We adopt a two-level approximation scheme that builds upon this practical observation. Given a half-vector $\omega$ and points generated implicitly for the pixel footprint **at an appropriate LoD** (see Algo. 1), we consider the $k$ points falling into the closest cells $B(k)$ in the angular domain (blue in the inset), and evaluate only those explicitly. In order to estimate the contribution of the other points whose lobe are contained inside $\mathcal{U}_\Omega$ but further away than $B(k)$ from $\omega$ (orange in the inset), we substitute their exact contribution with their expectation, and we define our compensation term as

$$C(\omega, \alpha_\star) := \int_{\mathcal{U}_\Omega \setminus B(k)} \mathcal{N}\left(\omega, \mu, \alpha_\star \sqrt{|J_T(T^{-1}(\omega))|}\right) \mathrm{d}\mu. \tag{9}$$

As our kernels are Gaussian, integrating is more easily done on rectangular domains (using the error function `erf` with a simple rational polynomial approximation) than on non-centered disks. We make a final approximation by considering the unit square $\square$ enclosing $\mathcal{U}_\Omega$ (red in the inset) and estimate $C(\omega, \alpha_\star)$ as

$$C(\omega, \alpha_\star) \approx \int_{\square} \mathcal{N}(\omega, \mu, \alpha_\star \sqrt{|J_T(T^{-1}(\omega))|})\mathrm{d}\mu - \int_{\square \cap B(k)} \mathcal{N}(\omega, \mu, \alpha_\star \sqrt{|J_T(T^{-1}(\omega))|})\mathrm{d}\mu. \tag{10}$$

Note that the correction term is only applied in the angular direction; we choose the spatial-angular trade-off such that the small spatial neighborhood is always large enough to cover the pixel filter sufficiently (the contribution of points outside of the neighborhood is negligible). The number $k$ of angular neighbors considered is a performance-quality trade-off that is easy to change; we find that $k = 4$ is already good enough for most cases (see Figure 2). Note that the compensation terms are computed per level and blended linearly during mip mapping (see Figure 5).

Adding our compensation term, our final NDF reads:

$$D^{\mathrm{f}}_\star(x, \omega_h) = |J_T(\omega_h)| * \left(\left(\sum_{i \in B(k)} \frac{\mathcal{N}\left(T(\omega_h), \mu_i, \alpha_\star \sqrt{|J_T(\omega_h)|}\right)}{K_P} f(x_i - x, \Sigma_s)\right) + C(\omega_h, \alpha_\star)\right). \tag{11}$$

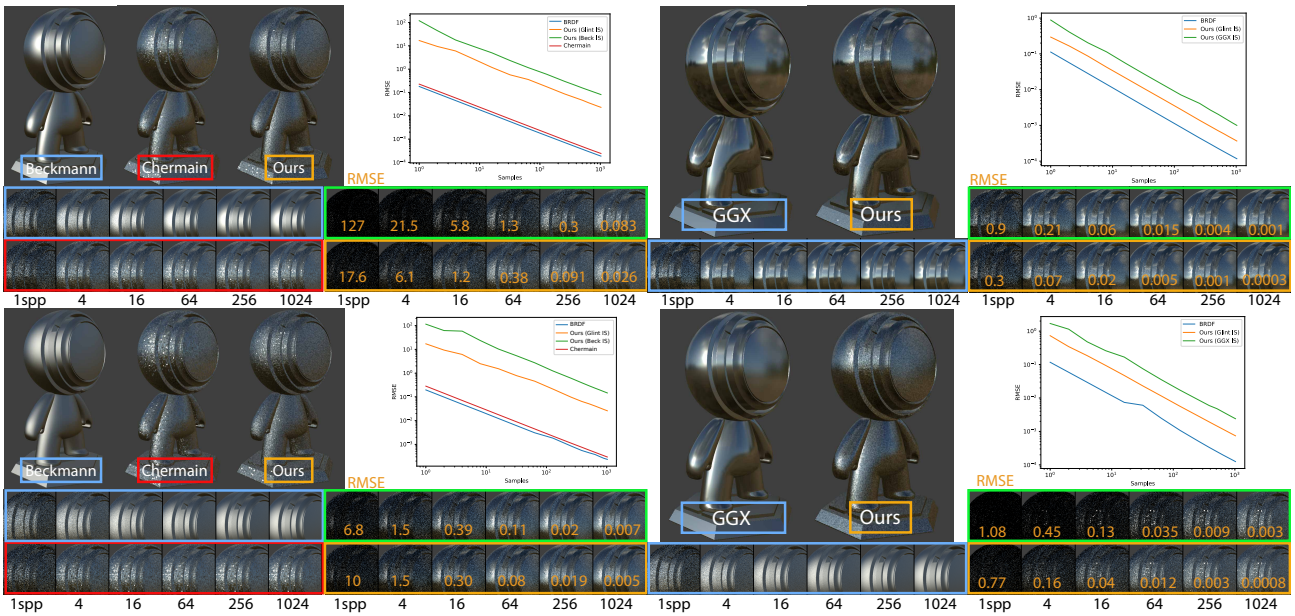



Fig. 7. We render glints with image-based lighting without angular prefiltering for different base NDFs (left: Beckmann, right: Trowbridge-Reitz) in order to validate our importance sampling scheme. We provide on the left a comparison to the method of Chermain et al. [2021b], which is the only other real-time glint rendering method that allows for importance sampling. Partial convergence is shown at the bottom with a growing number of samples per pixels. We use references computed with 8192 samples per pixel to compute the root mean square error (RMSE.) We use RMSE log-log plots which show power convergence as straight lines. As dictated by theory, importance sampling offers a constant offset in RMSE log-log plots. Note that the smooth case, Chermain, and ours all converge towards different target images, and thus the error plots aren't directly comparable. As expected, importance sampling effectiveness increases with the specularity of the material. In addition to this figure, we provide as supplemental material a live recording of our application, comparing convergence speeds obtained using our importance sampling scheme against using the standard importance sampling scheme of the target NDF.

Note that $K_P$ is still the total number of points per unit texture area, distributed across $[0,1)^4$, and differs from $|B(k)|$. The ratio $|B(k)|/K_P$ corresponds to the area of $B(k)$ and allows weighting the term correctly before summation with the compensation term $C$ (itself integrating over the complement of $B(k)$: $\mathcal{U}_\Omega \setminus B(k)$).

*3.3.5 Importance sampling.* Importance sampling our model is conceptually straightforward: we sample in $\mathcal{U}_\Omega$ from the sum of Gaussians of Eq.(11) generated by our point process (Figure 3 bottom left) and use $T^{-1}$ to map the samples to $\Omega$. This process samples a normal at the given point. To sample outgoing direction from a view direction we take the reflection of view direction according to the sampled normal. Algorithm 2 details our method. We first sample a direction $x_a$ according to the NDF to get a neighborhood $B(k)$ for Eq. (11). We then perform reservoir sampling [Vitter 1985] on all terms of the sum. Different areas are sampled per MIP level, thus separate reservoirs are required. We choose to not sample the compensation term, as it is only significant if the Gaussians are wide enough to sample the whole area roughly uniformly on their own. As shown in Figure 7, importance sampling the glint distribution leads to considerably less rendering noise compared to the simple alternative of importance sampling the original smooth distribution. The error plots also empirically confirm that our sampling scheme is unbiased. We further show in Figure 8 results of our approach for a complex environment lighting for different combinations of roughness/microroughness/density parameters.

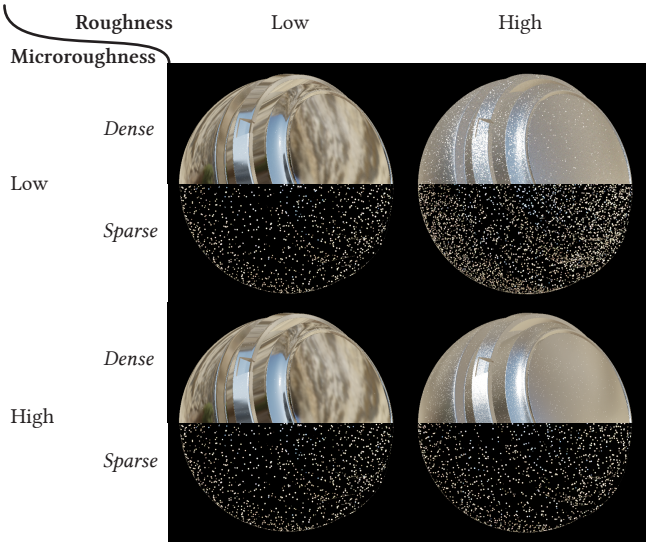



Fig. 8. The appearance of our model under environment lighting for a combination of roughness/microroughness/density values, rendered using our importance sampling scheme.

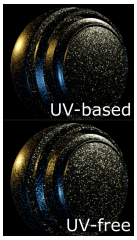
Fig. 9. **Rendering glints from NDFs.** Left: Trowbridge-Reitz (smooth and ours). Right: Beckmann (smooth and ours).

#### 3.3.6 *Extensions of our model.*

*Anisotropy.* Since we choose our virtual MIP level in a simple isotropic fashion, there is no limitation in scaling in the UV space. Hence, like Chermain et al. and Zirr and Kaplanyan, we can render materials like brushed metals by stretching the UV space and inversely stretching the angular space; see Algorithm 1 for details[1].

*Colors.* As we treat individual points explicitly, we can set their colors based on the cell index. The compensation term becomes colored, so the average color must be possible to compute efficiently.

*UV-free.* Our method supports uv-free rendering via triplanar mapping (see the inset) by emulating 3 different point processes $\{\mathcal{P}_i\}$ (aligned in 3 orthogonal planes). The same roulette term is used to blend between the 3 processes, using as weight the absolute value of the dot product between the triangle normal and the plane normal. However, this requires more evaluations and is slower than our base method, and prevents a simple definition of anisotropy as we use the uv-gradients to compute preferred anisotropy directions.




## 4 IMPLEMENTATION & RESULTS

We implement our method as a WebGPU shader, and provide an interactive implementation at https://www.shadertoy.com/view/tcdGDl[2]. The required inputs are the lighting and viewing directions, the surface normal, and the uv coordinates including their partial derivatives with respect to screen space. We can relax the requirement for uv coordinates via triplanar mapping, but the partial derivatives are still required. Pseudo code of our base method is provided in Algorithm 1[3], and Algorithm 2 details our importance sampling scheme. Only a few lines of code are required to implement our method in a standard PBR engine, as neither preprocessing nor storage are required, and our approach is compatible with various smooth NDF models (Figure 9).

We recall that our model is controlled by parameters listed in Table 1, which can all be spatially-varying, i.e. controlled with procedural or raster maps. Figure 10 illustrates the impact of those parameters independently, Figure 11 shows results of our approach across different geometries and shading styles, and Figure 12 shows results of our approach guided by spatially-varying maps. Note that too high-frequency spatially-varying parameters can result in noise under animation because neighboring pixels can get contributions from points from unrelated MIP levels, so a step interpolation instead of a smooth interpolation is sometimes more desirable.

**Algorithm 1:** Evaluation of $D^{\mathrm{f}}_{\star}$

**Input:** Smooth micro-facet normal distribution $D$ and roughness $\alpha$, number of points per unit area of texture space $N$, glint microroughness $\alpha_\star$ and anisotropy matrix $A_\star$, pixel filter size $f$, texture coordinate $\mathbf{uv}$, and half-vector $\omega_\mathbf{h}$.

**Result:** Glint *(colored)* normal distribution $\mathbf{D}_\star(D, \omega_\mathbf{h}, \mathbf{uv})$

$\mathrm{res} \leftarrow \sqrt{N}\det(A_\star^{-1})$ `// Implicit grid base resolution`
$\mathbf{x_s} \leftarrow \mathbf{A}_\star \cdot \mathbf{uv}$ `// Spatial position`
$\mathbf{x_a} \leftarrow \mathrm{NDF2Disk}(D, \omega_\mathbf{h}, \alpha)$ `// Angular position`
$d \leftarrow D(\omega_\mathbf{h}, \alpha)$ `// Smooth D evaluation`
$\lambda \leftarrow \mathrm{QueryLod}\,(\mathrm{res} \cdot \mathbf{uv})$ `// MIP level`
$D^{\mathrm{f}}_{\star} \leftarrow 0$
**foreach** $l \in \{\lfloor\lambda\rfloor, \lceil\lambda\rceil\}$ **do**
  $w_\lambda \leftarrow 1 - |\lambda - l|$ `// MIP weight`
  $\mathrm{res}_s \leftarrow \mathrm{res} \cdot 2^{-l}$ `// Spatial grid resolution`
  $\mathrm{res}_a \leftarrow 2^l$ `// Angular grid resolution`
  $\Sigma_\mathbf{s} \leftarrow f^2\,\mathbf{J_{uv}} \cdot \mathbf{J_{uv}}^T$ `// Spatial variance`
  $\Sigma_\mathbf{a} \leftarrow d \cdot \left(\alpha_\star\,\mathbf{A}_\star{}^{-1}\right)^2 \cdot \mathbf{I}_2$ `// Angular variance`
  `// Explicit contribution from nearby points`
  **foreach** $\mathbf{i_s} \in k\text{-}NeighbouringCells(\mathbf{x_s}, res_s)$ **do**
    **foreach** $\mathbf{i_a} \in k\text{-}NeighbouringCells(x_a, res_a)$ **do**
      `// Glint position and orientation`
      $\mathbf{g_s} \leftarrow (\mathbf{i_s} \cdot \mathrm{res}_s + \mathrm{Rand2D}(\mathbf{i_s}, \mathbf{i_a}, l))\,/\mathrm{res}_s$
      $\mathbf{g_a} \leftarrow (\mathbf{i_a} \cdot \mathrm{res}_a + \mathrm{Rand2D}(\mathbf{i_s}, \mathbf{i_a}, l))\,/\mathrm{res}_a$
      $\mathbf{c} \leftarrow \mathrm{GlintColor}(\mathbf{i_s}, \mathbf{i_a}, l)$
      $r \leftarrow \mathrm{Rand1D}(\mathbf{i_s}, \mathbf{i_a}, l)$
      $\mathbf{D}_\star \mathrel{+}= R(r, w_\lambda)\,\mathcal{N}(\Sigma_\mathbf{s}, \mathbf{x_s}{-}\mathbf{g_s})\mathcal{N}(\Sigma_\mathbf{a}, \mathbf{x_a}{-}\mathbf{g_a}) \cdot \mathbf{c}/K_P$
  `// Analytic compensation for other points`
  $D^{\mathrm{f}}_{\star} \mathrel{+}= w_\lambda \mathbb{E}[\mathrm{GlintColor}]C(\omega, \Sigma_a)$;
**return** $\pi^{-1} d\, D^{\mathrm{f}}_{\star}$

*Comparison.* Table 2 compares at high-level our technique with previous real-time methods. Our main conceptual difference to the methods of Deliot & Belcour and Zirr & Kaplanyan is that we explicitly model the facets; instead of sampling from a distribution of plausible outcomes, an actual NDF is produced. Like Deliot & Belcour, our method is constant-time to evaluate regardless of viewing anisotropy, but our implicit grid construction is significantly simpler and allows for glint anisotropy. Chermain et al. also produce a NDF and allow for importance sampling it, but their approach is limited to the Beckmann distribution and requires precomputing a set of distributions which limits their generality.

*Pixel filter and Temporal stability.* Explicitly modeling the pixel filter for a set of discrete points produces exactly filtered results and avoids subtle artifacts related to implicit grid interpolation visible in previous works, as shown in Figure 13. On top of allowing us to

[1]We drive the anisotropy through our *diagonal* stretch matrix $A_\star$.
[2]Implementation tip: Eq. (3) states that $\forall\omega \in \Omega, |J_T(\omega)| = D(\omega)/|\mathcal{U}_\Omega|$; this allows us to use $D(\omega)$ to avoid explicit Jacobian computation.
[3]As written in Sec. 3.3.4, we use $k = 4$ for all examples in the paper.

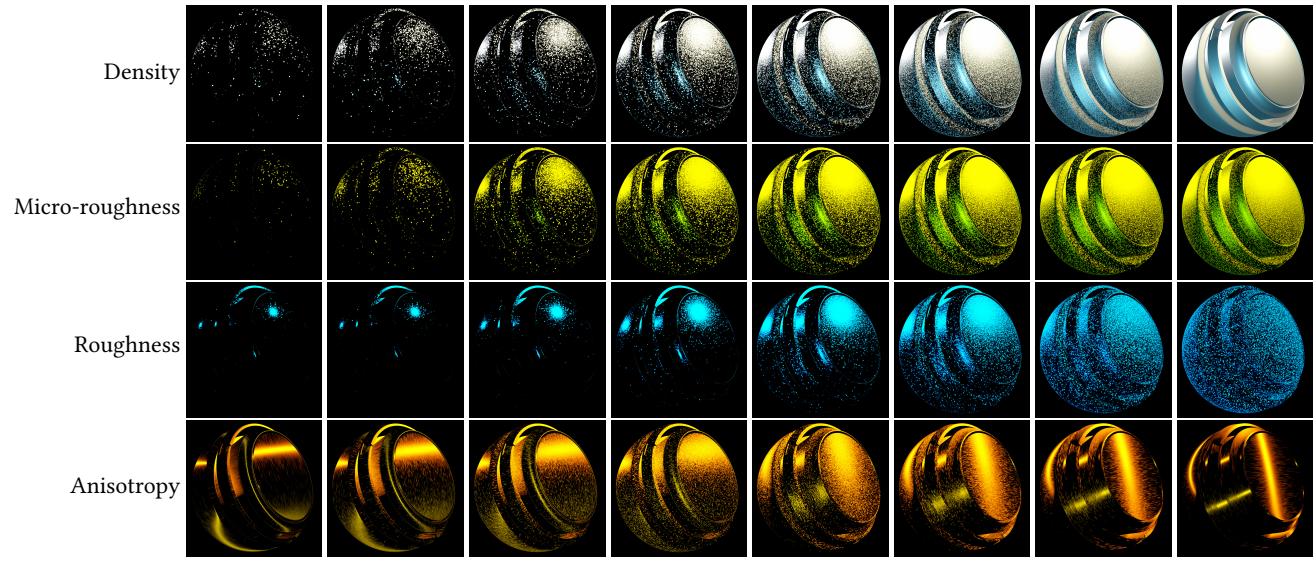



Fig. 10. Our model parameterization and how it translates visually. Each parameter is varied across its row over its range, all other parameters being constant.

**Algorithm 2:** Importance sampling of $D^{\mathrm{f}}_{\star}$

**Input:** Similar to Algorithm 1.
**Result:** Sampled direction $\omega_h$ and its associated $pdf$

$\mathrm{res}, \mathbf{x_s}, \lambda$ // Same as Algorithm 1
$\mathbf{x_a} \leftarrow \mathrm{UniformDiskSampling}()$ // Angular position
$pdf \leftarrow 0$
$l \leftarrow \lfloor \lambda \rfloor$
**if** $Rand1D() < \lambda - \lfloor \lambda \rfloor$ **then**
  $l \leftarrow \lceil \lambda \rceil$;
$\sum w_\star \leftarrow 0$ // Cumulative weights
$w_\lambda, \mathrm{res}_s, \mathrm{res}_a, \Sigma_\mathbf{s}, \Sigma_\mathbf{a}$ // Same as Algorithm 1
**foreach** $\mathbf{i_s} \in k\text{-}NeighbouringCells(\mathbf{x_s}, res_s)$ **do**
  **foreach** $\mathbf{i_a} \in k\text{-}NeighbouringCells(x_a, res_a)$ **do**
    $\mathbf{g_s}, \mathbf{g_a}, \mathbf{c}, r$ // Same as Algorithm 1
    $w_\star \leftarrow R(r, w_\lambda)\, \mathcal{N}(\Sigma_\mathbf{s}, \mathbf{x_s} - \mathbf{g_s}) \cdot \int_{B(x_a)} \mathcal{N}(\Sigma_\mathbf{a}, \mathbf{x_a} - \mathbf{x})\mathrm{d}\mathbf{x}\ \mathrm{Luminance}(\mathbf{c})$
    $\sum w_\star \leftarrow w_\star + \sum w_\star$
    $\mathbf{x} \leftarrow \mathrm{Sample}\left(\mathbb{1}_{B(k)} \mathcal{N}_{\Sigma_\mathbf{a}, \mathbf{g_a}}\right)$
    **if** $Rand1D() < w_\star / \sum w_\star$ **then**
      $\omega_\mathbf{h} \leftarrow T^{-1}(\mathbf{x})$
    $pdf \leftarrow pdf + w_\star\ \mathrm{Area}(B(\mathbf{x_a}))\ \mathrm{PDF}(\mathcal{N}_{\Sigma_\mathbf{a}, \mathbf{g_a}})(\mathbf{x})$
$pdf \leftarrow pdf \frac{|J_T(\omega_h)||\omega_h \cdot \omega_g| w_\lambda}{4\pi |\omega_h \cdot \omega_i| \sum w_\star}$

handle a wider range of settings, this also translates to better anti-aliasing and temporal stability as our subpixel details are always smooth. As such phenomena are better perceived in motion, we refer to the supplemental video for additional qualitative results.

*Performance.* Table 3 provides timings for our method and compares it to the approaches of Zirr & Kaplanyan [2016], Chermain et al. [2020] and Deliot & Belcour [2023]. The experimental setup reproduces the one proposed by Deliot & Belcour [2023] with two scenarios: one plane orthogonal to the camera (90° plane), and one plane with a 25° viewing angle, to assess performance when rendering glints at multiple scales/MIP levels. Overall, our method stays in the ballpark established by the state of the art, and doesn't exhibit significantly worse behavior in any specific setting. We measure

Table 2. **Qualitative comparison** with Chermain et al. [2021b] (`Cher`), Zirr & Kaplanyan [2016] (`Zirr`) and Deliot & Belcour [2023] (`Del`).

| Property | **Ours** | `Cher` | `Zirr` | `Del` |
|---|---|---|---|---|
| Importance sampling | y | y | n | n |
| Trowbridge-Reitz / Beckmann | y/y | n/y | n/y | y/y |
| Precomputation free | y | n | y | y |
| Constant time | y | n | n | y |
| Anisotropy | y | y | y | n |
| Individual colors | y | y | y | y |
| Energy conservation | n | y | n | n |

Table 3. **Performance comparison**: in ms per frame on an RTX 3090 (averaged over 1000 frames), at full HD resolution with 3 lights. Timing are given for Beckmann/Trowbridge-Reitz NDFs if applicable. (*): Our implementation of Zirr & Kaplanyan uses the gated Bernoulli approximation which explains the differences in the timings to those in Deliot & Belcour [2023].

| | 90° plane | | 25° plane | |
|---|---|---|---|---|
| Roughness | low | high | low | high |
| **Zirr & Kaplanyan*** | 1.42/NA | 1.48/NA | 2.72/NA | 2.80/NA |
| **Deliot & Belcour** | 2.07/1.99 | 2.06/2.06 | 2.09/2.00 | 2.06/2.07 |
| **Chermain et al.** | 3.70/NA | 3.44/NA | 18.02/NA | 19.44/NA |
| **Ours** | 3.26/1.70 | 3.27/3.33 | 3.31/1.59 | 3.36/3.07 |

Fig. 11. Some results of our glittery BRDF with different set of parameters over different objects and NDFs.

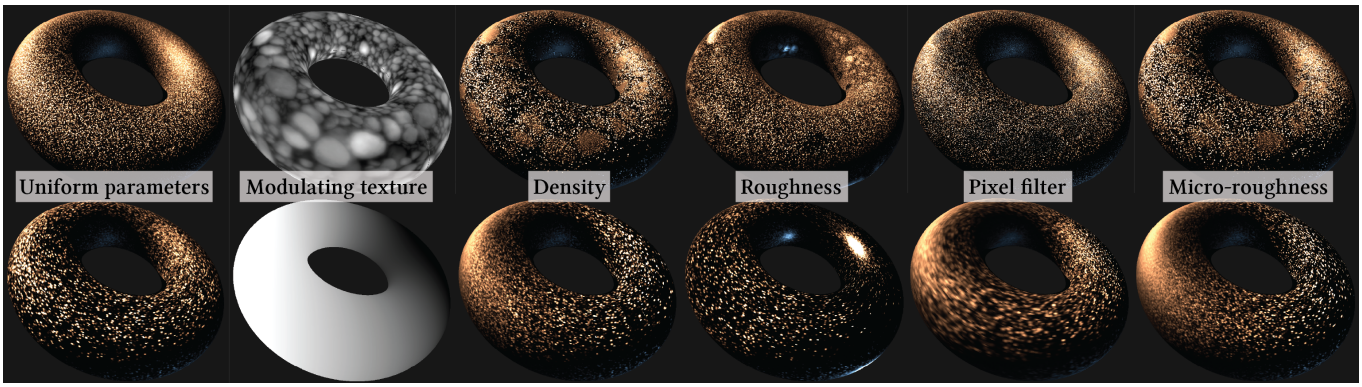



Fig. 12. **Spatially varying parameters**. Examples with parameters modulated using two different textures. From left to right: our method with uniform parameters, the texture, our method with one parameter varied spatially according to the texture while the other parameters stay uniform.

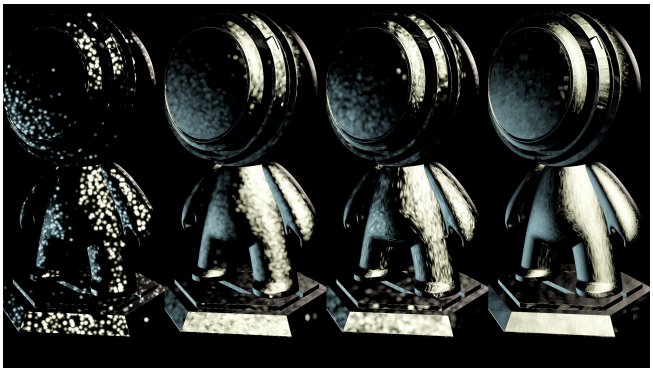


Fig. 13. **Explicit pixel filter (ours) vs grid interpolation methods**. From left to right: our method, Deliot & Belcour [2023], Chermain et al. [2021a] and Zirr & Kaplanyan [2016].

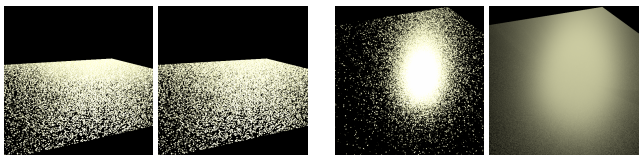


Fig. 14. **Limitations.** Left two images: We assume that only the few closest points are relevant. There are settings where this does not hold, causing the background to brighten before the points uniformly cover the surface (far left, using $k = 4$ points), which can be mitigated by consider a larger neighborhood (middle left, with $k = 16$ points) for a runtime cost. Right two images: Not all parameter combinations are sensible. If we model a smooth surface (middle right) with a microroughness that is too high (far right), we lose the brightness and shape of the highlight. This is physically reasonable, as we are trying to model a sharply reflective surface with diffusely reflective particles, but it does mean that the parameter values are not independent from each other and have to be chosen together with some care.

in the most typical case where most of the surface is covered by points; our method is roughly twice as fast for sparse cases due to early exits for points outside the hemisphere.

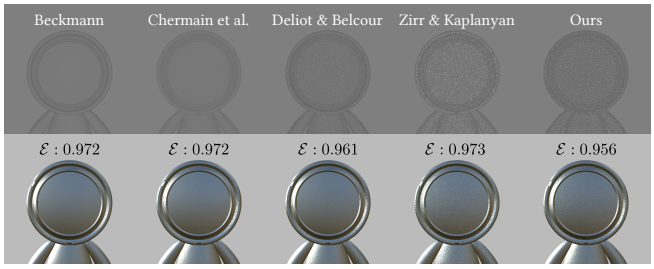



Fig. 15. **Furnace test:** Average energy $\mathcal{E}$ (ideally matches Beckmann or at least be less than 1). Contrary to Chermain et al., our method is not energy preserving nor conserving. Only Zirr & Kaplanyan increases the energy.

## 5 LIMITATIONS & FUTURE WORK

Some parameter settings reveal the structure of our mapping construction. As is visible in the bottom right image of Figure 3, the mapping stretches the shape of the Gaussians. The more extreme this stretch is, the more disturbing the artifact becomes – especially in motion. Our compensation term is designed on the assumption that individual particles are highly specular. When the microroughness is too large, it considers areas outside of $\mathcal{U}_\Omega$ and becomes visible before the glints converge to a uniform mass (Figure 14, left). Additionally, the microroughness and roughness parameters are not independent and some combinations can lead to counter-intuitive results, even if physically reasonable (Figure 14, right). Similarly to Deliot & Belcour [2023], our method is not strictly energy-conserving nor preserving. This is due to our mixture of Gaussians that can gain or lose energy close to the domain boundaries: we do not match the integral of the smooth NDF. To quantify our energy loss we perform a white furnace test (Figure 15). Since the response of our model is black in areas between the particles and arbitrarily high when zoomed in, we perform the test averaged over surface area. Also, our model's parametrization could be optimized to provide coherent parameters with a linear perceptual effect. Last, we could use ray differentials to propagate pixel footprints to potentially support path tracing, including specular indirect lighting e.g., glint caustics.

## ACKNOWLEDGMENTS

We thank Élie Michel for his WebGPU base code and help in using it, Axel Paris for pushing the work forward, and Erik Härkönen for crucial debugging assistance.

## REFERENCES

A. Atanasov, V. Koylazov, R. Dimov, and A. Wilkie. 2022. Microsurface Transformations. *Computer Graphics Forum* 41, 4 (2022), 105–116. https://doi.org/10.1111/cgf.14590 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14590

Xavier Chermain, Simon Lucas, Basile Sauvage, Jean-Michel Dischler, and Carsten Dachsbacher. 2021a. Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering. *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D)* 4, 1 (2021).

Xavier Chermain, Basile Sauvage, Jean-Michel Dischler, and Carsten Dachsbacher. 2020. Procedural Physically-based BRDF for Real-Time Rendering of Glints. *Computer Graphics Forum (Proceedings of Pacific Graphics)* 39, 7 (2020), 243–253. https://doi.org/10.1111/cgf.14141

Xavier Chermain, Basile Sauvage, Jean-Michel Dischler, and Carsten Dachsbacher. 2021b. Importance Sampling of Glittering BSDFs based on Finite Mixture Distributions. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)*. https://doi.org/10.2312/sr.20211289

Thomas Deliot and Laurent Belcour. 2023. Real-Time Rendering of Glinty Appearances using Distributed Binomial Laws on Anisotropic Grids. *Computer Graphics Forum* 42, 8 (2023), e14866. https://doi.org/10.1111/cgf.14866 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14866

Hong Deng, Yang Liu, Beibei Wang, Jian Yang, Lei Ma, Nicolas Holzschuch, and Ling-Qi Yan. 2022. Constant-cost spatio-angular prefiltering of glinty appearance using tensor decomposition. *ACM Transactions on Graphics (TOG)* 41, 2 (2022), 1–17.

Roman Ďurikovič and William L Martens. 2003. Simulation of sparkling and depth effect in paints. In *Proceedings of the 19th spring conference on Computer graphics*. 193–198.

Sergey Ershov, Andrei Khodulev, and Konstantin Kolchin. 1999. Simulation of sparkles in metallic paints. In *Proceeding of Graphicon*. 121–128.

Sergey Ershov, Konstantin Kolchin, and Karol Myszkowski. 2001. Rendering pearlescent appearance based on paint-composition modelling. In *Computer Graphics Forum*, Vol. 20. Wiley Online Library, 227–238.

Ned Greene and Paul S Heckbert. 1986. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications* 6, 6 (1986), 21–27.

Paul S. Heckbert. 1989. *Fundamentals of Texture Mapping and Image Warping*. Technical Report. USA.

Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. 2016. Real-time polygonal-light shading with linearly transformed cosines. *ACM Trans. Graph.* 35, 4, Article 41 (July 2016), 8 pages. https://doi.org/10.1145/2897824.2925895

Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. 2014. Discrete stochastic microfacet models. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Tom Kneiphof and Reinhard Klein. 2024. Real-Time Rendering of Glints in the Presence of Area Lights. *ArXiv* abs/2408.13611 (2024). https://api.semanticscholar.org/CorpusID:271957045

Alexandr Kuznetsov, Milos Hasan, Zexiang Xu, Ling-Qi Yan, Bruce Walter, Nima Khademi Kalantari, Steve Marschner, and Ravi Ramamoorthi. 2019. Learning generative models for rendering specular microgeometry. *ACM Trans. Graph.* 38, 6 (2019), 225–1.

Johann Heinrich Lambert. 1772. *Beiträge zum Gebrauche der Mathematik und deren Anwendung*. Haude und Spener, Berlin.

George Marsaglia. 1962. IMPROVING THE POLAR METHOD FOR GENERATING A PAIR OF NORMAL RANDOM VARIABLES. (1962). https://api.semanticscholar.org/CorpusID:122725153

Ishaan Shah, Luis E. Gamboa, Adrien Gruson, and P.J. Narayanan. 2024. Neural Histogram-Based Glint Rendering of Surfaces With Spatially Varying Roughness. *Computer Graphics Forum (Proceedings of EGSR)* 43, 4 (2024).

Yusuke Tokuyoshi and Takahiro Harada. 2017. Stochastic Light Culling for VPLs on GGX Microsurfaces. *Comput. Graph. Forum* 36, 4 (July 2017), 55–63. https://doi.org/10.1111/cgf.13224

Jeffrey S Vitter. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)* 11, 1 (1985), 37–57.

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*. 195–206.

Beibei Wang, Hong Deng, and Nicolas Holzschuch. 2020. Real-Time Glints Rendering with Prefiltered Discrete Stochastic Microfacets. *Computer Graphics Forum* 39, 6 (Sept. 2020), 144–154. https://doi.org/10.1111/cgf.14007

Liwen Wu, Fujun Luan, Miloš Hašan, and Ravi Ramamoorthi. 2025. Position-Normal Manifold for Efficient Glint Rendering on High-Resolution Normal Map. In *SIGGRAPH*.

Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. 2014. Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–9.

Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. 2016. Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–9.

Tobias Zirr and Anton S. Kaplanyan. 2016. Real-time rendering of procedural multiscale materials *(I3D '16)*. Association for Computing Machinery, New York, NY, USA, 139–148. https://doi.org/10.1145/2856400.2856409