Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering

XAVIER CHERMAIN, SIMON LUCAS, BASILE SAUVAGE, and JEAN-MICHEL DISCHLER, Université de Strasbourg, CNRS, ICube UMR 7357, France
CARSTEN DACHSBACHER, Institute for Visualization and Data Analysis, Karlsruhe Institute of Technology, Germany

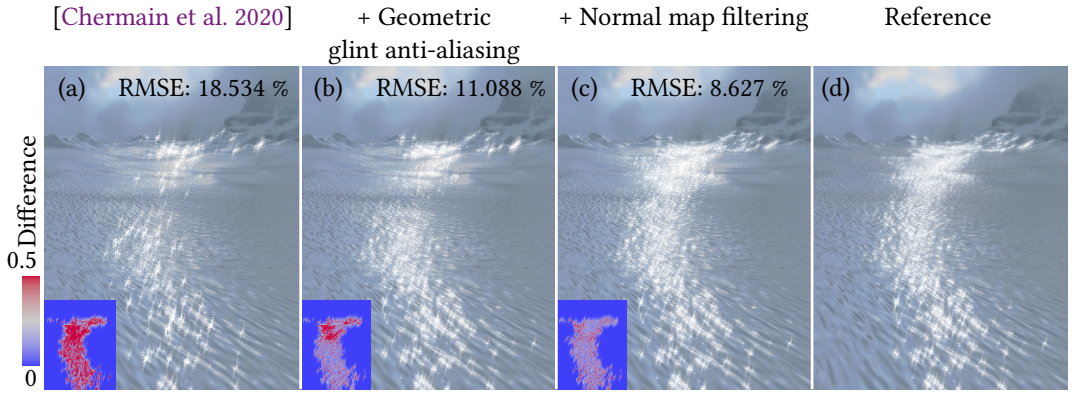

Fig. 1. An arctic landscape with a normal mapped surface. (a) Rendering using the glinty BRDF of Chermain et al. [2020], which is prone to geometric glint aliasing. Our geometric glint anti-aliasing (GGAA) method is shown in (b) without normal map filtering and in (c) with normal map filtering. (d) Reference using 1,024 samples per pixel. Difference maps with false colors highlight pixel-wise errors on the glinty part.

Real-time geometric specular anti-aliasing is required when using a low number of pixel samples and high-frequency specular lobes. Several methods have been proposed for mono-lobe bidirectional reflection distribution functions (BRDFs), but none for multi-lobe BRDFs, e.g., a glinty BRDF. We present the first method for real-time geometric glint anti-aliasing (GGAA). It eliminates most of the inconsistent appearing and disappearing of glints on surfaces with significant curvatures during animations. The technique uses the glinty BRDF of Chermain et al. [2020] and leverages hardware GPU-filtering of textures to filter slope distributions on the fly. We also improve this glinty BRDF by adding a correlation factor of slope. This BRDF parameter allows convergence to normal distribution functions that are not aligned on the surface's axes. Above all, this parameter makes glint rendering compatible with normal map filtering using LEAN mapping. Using GGAA increases the rendering time from 0.6 % to 4.2 % and it requires 1/3 more memory due to MIP mapping of tabulated slope distributions. The results are compared with references using a thousand samples per pixel.

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

Authors' addresses: Xavier Chermain, chermain@unistra.fr; Simon Lucas, simon.lucas@etu.unistra.fr; Basile Sauvage, sauvage@unistra.fr; Jean-Michel Dischler, dischler@unistra.fr, Université de Strasbourg, CNRS, ICube UMR 7357, Illkirch, France, F-67400; Carsten Dachsbacher, dachsbacher@kit.edu, Institute for Visualization and Data Analysis, Karlsruhe Institute of Technology , Karlsruhe, Germany, 76131.

Additional Key Words and Phrases: geometric glint anti-aliasing, normal map filtering, microfacet, real-time rendering

ACM Reference Format:
Xavier Chermain, Simon Lucas, Basile Sauvage, Jean-Michel Dischler, and Carsten Dachsbacher. 2021. Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 1 (May 2021), 16 pages. https://doi.org/10.1145/3451257

1 INTRODUCTION

Reflectance modeling, anti-aliasing, and normal mapping are essential rendering techniques in computer graphics. Reflectance modeling allows the simulation of light scattering on the 3D scene's objects' surfaces and permits representing materials compactly. Normal maps add details to surfaces, thus making visual appearance more complex and realistic at little cost. Anti-aliasing is crucial in rendering in different areas: albedo or normal map anti-aliasing, known as texture filtering, removes moiré patterns and "salt and pepper" artifacts. Geometric anti-aliasing allows smooth edges at object borders and geometric *specular* anti-aliasing removes jarring visual shimmering on high curvature regions due to bright highlights at the sub-pixel level. In this paper, we focus on reflectance modeling of sparkling surfaces combined with normal map filtering and geometric specular anti-aliasing.

Sparkling materials are very common in our environment: sand, snow, ice, ocean, glittery materials, etc., but they are challenging to render because their reflectance distributions contain high frequencies and have hundreds of sharp lobes. Methods exist to render these kinds of materials [Jakob et al. 2014; Yan et al. 2016], and some of them provide means to do so in real-time [Chermain et al. 2020; Zirr and Kaplanyan 2016]. Although these methods handle high-frequency lobes (glints), some of these lobes are missed when the surface is strongly curved (especially normal mapped surfaces) and when the number of pixel samples is low. We call this phenomenon *geometric glint aliasing* (Figure 5a). During animations, this geometric glint aliasing produces disturbing and non-realistic specular temporal incoherence between frames. Although geometric specular anti-aliasing methods exist for rough surfaces modeled using a single specular lobe [Kaplanyan et al. 2016; Tokuyoshi and Kaplanyan 2019], to the best of our knowledge, no such method exists for glinty surfaces. We close this gap and propose a method targeting real-time geometric glint anti-aliasing (GGAA). Since geometric glint aliasing primarily occurs when using normal maps, we also present a method to make glints rendering compatible with efficient normal map filtering using Linear Efficient Anti-aliased Normal (LEAN) mapping [Olano and Baker 2010].

Geometric specular anti-aliasing is generally achieved by convolving the slope distribution function (SDF) of the microfacet BRDF with a normalized kernel, i.e., by filtering the SDF. If the SDF is tabulated, anti-aliasing using efficient texture filtering based on a MIP pyramid can be used. We propose to apply this principle to the glinty BRDF of Chermain et al. [2020], which uses 1D tabulated SDFs (stored in textures). In a second step, we improve this BRDF by adding a new parameter: a slope correlation coefficient. In the former method, only the slope standard deviation in x and y directions can be controlled by linearly transforming a target SDF. Our new linear transformation allows a user to edit the slope correlation coefficient. This is of core importance, as it makes this BRDF compatible with normal map filtering. Compared to the original model, the memory footprint of our GGAA glinty BRDF increases by one third due to storing MIP maps for filtering the tabulated SDFs efficiently (which amounts to 512 KiB total in our implementation). The rendering performance remains almost unchanged (between 0.6 % to 4.2 % of rendering time overhead) because texture filtering on the GPU is extremely optimized. The integration of the correlation coefficient does not impact performance either. Lastly, GGAA preserves the normalization of the distributions and so the BRDF is still energy preserving.

To summarize, our contributions are as follows:

- We propose a real-time geometric glint anti-aliasing (GGAA) method (Section 4.2).
- We introduce a slope correlation factor to the BRDF of Chermain et al. [2020] (Section 5.2).
- We combine real-time glint rendering and normal map filtering.

2 RELATED WORK

We are not aware of any previous work that explicitly addresses real-time geometric glint anti-aliasing (GGAA). To this end, we review and discuss specular anti-aliasing, glint rendering, and normal map filtering (with a focus on real-time methods), which are related to our work.

2.1 Geometric Specular Anti-aliasing

The article of Amanatides [1992] introduces the core concept to address geometric specular aliasing and proposes to clamp the Phong-BRDF exponent to avoid it. Hill and Baker [2012] and later Vlachos [2015] use built-in GPU-functions, e.g., `dFdx` and `dFdy` in OpenGL, to compute differences between the half vector of two adjacent pixels. These values are used to clamp the roughness of the material. Kaplanyan et al. [2016] project the pixel footprints into the slope domain and use them to filter the SDFs of microfacet BRDFs. Tokuyoshi and Kaplanyan [2019] reduce the estimation error by evaluating the derivatives in the projected half vector space instead of slope space.

In this paper, we draw from these works and use the geometric specular anti-aliasing method of Kaplanyan et al. [2016]. We convolve the glinty SDF of Chermain et al. [2020] with a slope kernel. We also estimate the derivatives in the projected half vector space [Tokuyoshi and Kaplanyan 2019].

2.2 Glint Rendering

Normal maps can be used to represent the microgeometry of glittery surfaces [Loubet et al. 2020; Yan et al. 2014, 2016], but this implies high memory cost because a specific data structure for acceleration must be built from the normal map. Some works use procedural normal distribution functions (NDFs) [Atanasov and Koylazov 2016; Jakob et al. 2014; Wang et al. 2018] to reduce memory footprints and rendering time. Still, these techniques are not efficient enough to be used in real-time.

The procedural BRDF of Zirr and Kaplanyan [2016] is the first real-time BRDF producing glints controlled by a microfacet density parameter. This BRDF model is not physically based, but it is the fastest to evaluate and does not require memory. Chermain et al. [2020] propose a physically based procedural BRDF, which converges to the BRDF of Cook and Torrance [1982] as microfacet density increases to infinity. They use a dictionary of 1D distributions whose memory footprint is less than one MiB. Wang et al. [2020] use a stochastic three-scale BRDF, which can handle environment maps, contrary to other real-time methods; however, at the expense of more than 256 MiB of pre-computed data.

When the surface is relatively flat, these three methods [Chermain et al. 2020; Wang et al. 2020; Zirr and Kaplanyan 2016] do not produce glint aliasing because the NDF depends on the pixel footprint, i.e., spatial filtering is performed. When the surface is curved, the observation and incident directions vary significantly over the pixel footprint and then these methods are prone to geometric glint aliasing as they all use high-frequency NDFs.

Geometric specular anti-aliasing can be used to account for the curvature by convolving the NDF with a low-pass filter. Zirr and Kaplanyan [2016] and Wang et al. [2020] compute their NDFs by counting the number of flakes in a pixel's footprint. Consequently, convolving their NDFs is not trivial and would require the filtering of the counting process. The NDF of Chermain et al. [2020]

is a mixture of 1D SDFs stored in 1D textures. In this paper, we propose to perform the convolution by filtering these 1D textures.

2.3 Normal Map Filtering

Normal map anti-aliasing by considering a given underlying BRDF is a longstanding problem in rendering. It was first addressed by Fournier [1992], who applied multiple Phong lobes to fit a base normal distribution, and, a bit later, by Becker and Max [1993], who provided smooth transitions between three different rendering techniques. Toksvig [2005] proposes an implementation that substitutes the power function with a two-dimensional texture look-up. Han et al. [2007] introduce a representation based on spherical harmonics for low frequency reflectance and von Mises-Fisher distributions for high-frequency reflectance, thus allowing more accurate filtering in frequency domain. A much faster, simpler, and linear method for real-time filtering of specular highlights on normal mapped surfaces is LEAN mapping [Olano and Baker 2010]. It uses a representation based on the first and second moments of a Gaussian normal distribution, expressed in the tangent space of the textured surface. LEADR mapping [Dupuy et al. 2013] improves this model by more accurate masking-shadowing terms and projection weights.

In this paper, we use LEAN mapping to extract the roughness and the correlation factor of the normal distribution induced by the normal map. We then use these statistics in our improved version of the glinty BRDF of Chermain et al. [2020]. Unfortunately, LEADR mapping is not compatible because only the V-cavity masking can be used with this BRDF, and, to the best of our knowledge, there is no V-cavity masking function for non centered distributions.

3 REAL-TIME GLINT RENDERING

In this section, we briefly recapitulate the real-time glinty BRDF of Chermain et al. [2020].

Microfacet BRDF. Almost all glint reflection models are based on a microfacet BRDF [Heitz 2014], which assumes that the microsurface is composed of numerous tiny mirrors (microfacets). The orientation of the microfacets follows a normal distribution function (NDF) and is denoted as $D(\omega_m)$, which is the distribution of micronormals $\omega_m = (x_m, y_m, z_m)^T$ at a surface position.

The microfacet BRDF f is based on the NDF and defined as

$$f(\omega_o, \omega_i) = \frac{F(\omega_o, \omega_h) G_2(\omega_o, \omega_i, \omega_h) D(\omega_h)}{4(\omega_o \cdot \omega_g)(\omega_i \cdot \omega_g)}, \tag{1}$$

where F is the Fresnel term and G_2 is the masking-shadowing function. The normalized directions ω_i and ω_o are the incident direction and the outgoing (observation) direction, respectively. Finally, the geometric normal is ω_g, and the half vector $\omega_h = (x_h, y_h, z_h)^T$ is computed as $\omega_h = (\omega_o + \omega_i)/||\omega_o + \omega_i||$.

Most NDFs are derived from a slope distribution function (SDF) defined as

$$P^{22}(\tilde{m}) = D(\omega_m)(\omega_m \cdot \omega_g)^4, \tag{2}$$

where $\tilde{m} = (-x_m/z_m, -y_m/z_m)^T$ is the microslope according to the micronormal ω_m. Analytical SDFs with mono-lobe shapes (Figure 2, left), like Beckmann [Beckmann and Spizzichino 1963] or GGX [Walter et al. 2007] distributions, are commonly used in computer graphics.

Glinty SDF. Glints are produced by sharp specular highlights that introduce high frequencies in the SDF (Figure 2, right). Chermain et al. [2020] define a procedural physically based SDF P^{22}_{glint}

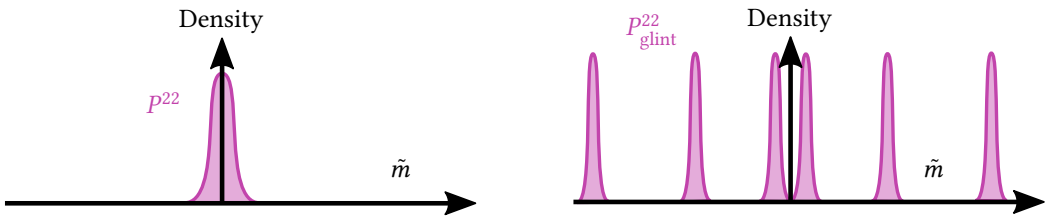

Fig. 2. Left: mono-lobe SDF used for smooth surfaces. Right: multi-lobe SDF used for glinty surfaces.

with sharp lobes targeted for real-time rendering as

$$P^{22}_{\text{glint}}(\tilde{m}, \mathcal{P}) = \sum_{l=\lfloor l_{\mathcal{P}} \rfloor}^{\lfloor l_{\mathcal{P}} \rfloor + 1} w(l) \sum_{s_0 \in \mathcal{P}} W_{\mathcal{P}}(l, s_0) P^{22}_M(\tilde{m}, l, s_0). \tag{3}$$

It is a patch-SDF, i.e., it is defined for any patch $\mathcal{P}$, which is actually the pixel footprint. It is based on a virtual MIP hierarchy with levels of detail l (LOD), and spatial 2D cells s_0. The outer sum in Equation 3 interpolates between two consecutive LODs, where $l_{\mathcal{P}}$ is the continuous LOD associated with $\mathcal{P}$, and $w(l)$ are linear weights. The inner sum approximates an integration over $\mathcal{P}$ by a sum over the cells $s_0 \in \mathcal{P}$, weighted by $W_{\mathcal{P}}(l, s_0)$ so that $\sum_{s_0 \in \mathcal{P}} W_{\mathcal{P}}(s_0, l) = 1$.

The patch-SDF of Equation 3 is a mixture of linearly transformed SDFs $P^{22}_M(\tilde{m}, l, s_0)$ defined for a microslope $\tilde{m}$, a LOD l and a cell s_0. A 2D linear transformation represented by a 2×2 matrix M is applied on an original SDF P^{22}_o in order to remove glint alignment (rotation) and to control the roughness of the material (scaling):

$$P^{22}_M(\tilde{m}, l, s_0) = \frac{1}{\det(M)} P^{22}_o(M^{-1}\tilde{m}, l, s_0). \tag{4}$$

In other words, the distribution P^{22}_M is computed from P^{22}_o by recovering the original slope:

$$\tilde{m}_o = (x_{\tilde{m}_o}, y_{\tilde{m}_o})^T = M^{-1}\tilde{m}. \tag{5}$$

Procedural generation. The technique of Chermain et al. [2020] is "procedural" as the distributions in Equations 3 and 4 are computed on-the-fly, when the pixel footprint $\mathcal{P}$ is known. This is achieved by defining the original SDF P^{22}_o as the product of two 1D marginal distributions:

$$P^{22}_o(\tilde{m}_o, l, s_0) = P^{2-}_o(x_{\tilde{m}_o}, l, s_0) P^{-2}_o(y_{\tilde{m}_o}, l, s_0). \tag{6}$$

The marginal distributions P^{2-}_o and P^{-2}_o are randomly picked in a pre-computed dictionary composed of N multi-scale 1D distributions P_o (see Figure 3). The highest LOD of all distributions P_o is the marginal distribution of a Beckmann SDF, with isotropic roughness σ_o. To enforce spatial coherence when evaluating Equation 6, the pseudo-random generator uses the cell position s_0 as a seed.

4 REAL-TIME GEOMETRIC GLINT ANTI-ALIASING

When a surface is flat, a glinty BRDF does not produce glint aliasing because the SDF is obtained from filtering using the footprint $\mathcal{P}$. In contrast, on curved surfaces *geometric* glint aliasing occurs in form of missing glints (see Figures 1b, 8 and 9). The assumption that the observation and incident directions vary only slightly over $\mathcal{P}$ does not hold any more on curved geometry. This causes quick variations for the half slopes $\tilde{h}$, as shown in Figures 4a and 4b. The half slope varies strongly across the footprint $\mathcal{P}$ and causes large half slope derivatives, which would require the integration of the BRDF over a solid angle. Integration in the half slope domain is simpler, as demonstrated by recent

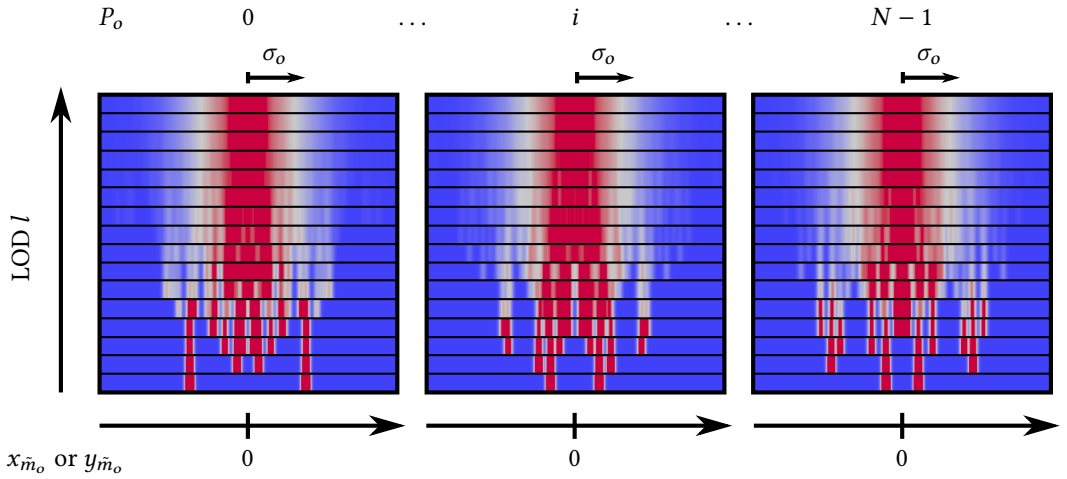

Fig. 3. Pre-computed dictionary of 1D multi-scale distributions P_o [Chermain et al. 2020]. They serve as marginal distributions for P_o^{22}. As the LOD grows, they converge towards a Gaussian distribution with the same standard deviation σ_o.

geometric specular anti-aliasing methods (Section 4.1). We leverage this technique to improve the BRDF of Chermain et al. [2020] (Section 4.2).

4.1 Geometric Specular Anti-aliasing

In order to efficiently pixel filter the microfacet BRDF (Equation 1), Kaplanyan et al. [2016] assume that all terms of the BRDF are constant over the pixel footprint $\mathcal{P}$, except for the SDF (the main source of high frequency). Then, instead of integrating over $\mathcal{P}$ (Figure 4a), they change the integration domain to the microslope domain (Figure 4c), assuming that the half slope $\tilde{h} = (x_{\tilde{h}}, y_{\tilde{h}})^T = (-x_h/z_h, -y_h/z_h)$ varies linearly with the surface position (first-order Taylor approximation). The result is a filtered SDF

$$\overline{P^{22}}(\tilde{m}) = K * P^{22}(\tilde{m}) \tag{7}$$

which can be interpreted as a convolution of the SDF with a normalized kernel K (Figure 4c). Similar to Equation 2, the filtered SDF is used to define the filtered NDF

$$\overline{D}(\omega_m) = \frac{\overline{P^{22}}(\tilde{m})}{(\omega_m \cdot \omega_g)^4}. \tag{8}$$

This last term is used in the microfacet model (Equation 1). Estimating Equation 7 is done by computing the half slope kernel K associated with the pixel footprint $\mathcal{P}$.

Computing the Half Slope Kernel. The kernel K is either a box, a parallelogram or a Gaussian, depending on the expression of P^{22}. For example, if the SDF is a Gaussian, choosing K also as a Gaussian leads to a closed form for $\overline{P^{22}}$. The extent of the kernel K depends on the Jacobian $\frac{\mathrm{d}\tilde{h}}{\mathrm{d}p}$ of the half slope $\tilde{h}$ with respect to the pixel coordinate p. On GPUs, an approximation $\frac{\Delta\tilde{h}}{\Delta p}$ can be computed by using finite differences of $\tilde{h}$ between two adjacent pixels (Figure 4c).

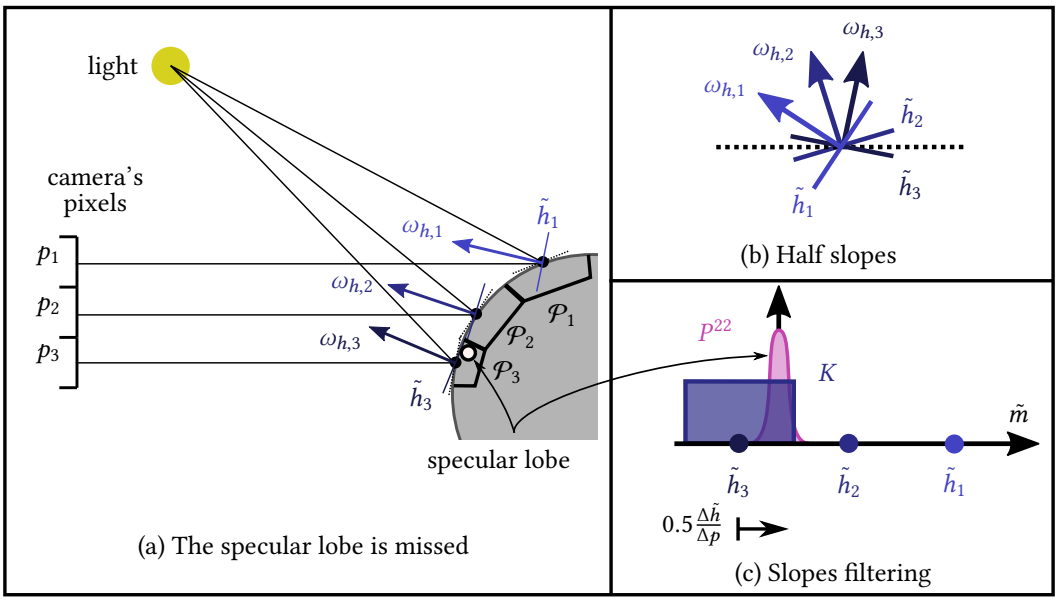

Fig. 4. Geometric specular anti-aliasing: with one sample per pixel, the specular lobe can be missed, *because the surface has a high curvature*. (a) A *curved* surface is seen through three pixels p_1, p_2 and p_3. (b) The curvature produces quick variations of the half-slopes (from $\tilde{h}_1$ to $\tilde{h}_2$ and to $\tilde{h}_3$). (c) The highlight is missed when sampling at $\tilde{h}_1$, $\tilde{h}_2$, and $\tilde{h}_3$. Geometric specular anti-aliasing performs a convolution with a kernel K, whose width is efficiently computed on GPU using finite differences $\Delta\tilde{h}/\Delta\tilde{p}$.

4.2 Geometric Glint Anti-aliasing

As a glinty SDF P^{22}_{glint} is high frequency, some of its lobes can easily be missed during rendering (Figure 5a). We now derive our real-time glint anti-aliasing model. To do so, we filter the procedural patch-SDF P^{22}_{glint} with a kernel K (Figure 5b).

Similar to Equation 7, the filtered procedural patch-SDF is

$$\overline{P^{22}_{\text{glint}}}(\tilde{m}, \mathcal{P}) = \left[K(\cdot) * P^{22}_{\text{glint}}(\cdot, \mathcal{P})\right](\tilde{m}). \tag{9}$$

By expanding P^{22}_{glint} using Equation 3, and by inverting sums and convolutions, we obtain

$$\overline{P^{22}_{\text{glint}}}(\tilde{m}, \mathcal{P}) = \sum_{l=\lfloor l_{\mathcal{P}} \rfloor}^{\lfloor l_{\mathcal{P}} \rfloor + 1} w(l) \sum_{s_0 \in \mathcal{P}} W_{\mathcal{P}}(l, s_0) \overline{P^{22}_{M}}(\tilde{m}, l, s_0), \tag{10}$$

where

$$\overline{P^{22}_{M}}(\tilde{m}, l, s_0) = \left[K(\cdot) * P^{22}_{M}(\cdot, l, s_0)\right](\tilde{m}). \tag{11}$$

To compute $\overline{P^{22}_{M}}$ efficiently, the filtering must be performed in the original slope space, i.e., the domain of the dictionary (Figure 3). Similar to Equation 4 we have

$$K(\tilde{m}) = \frac{1}{\det(M)} K_o(\tilde{m}_o). \tag{12}$$

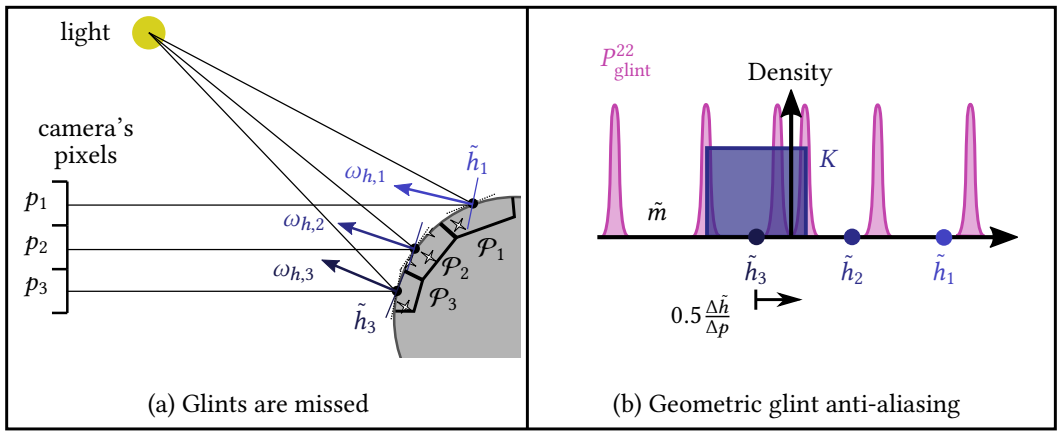

Fig. 5. Geometric *glint* anti-aliasing is similar to geometric *specular* anti-aliasing (Figure 4) but the SDF exhibits several sharp lobes. (a) Glints can be missed because the surface has a high curvature. (b) Geometric glint aliasing is removed by convolving the SDF P^{22}_{glint} with a kernel K (low-pass filter).

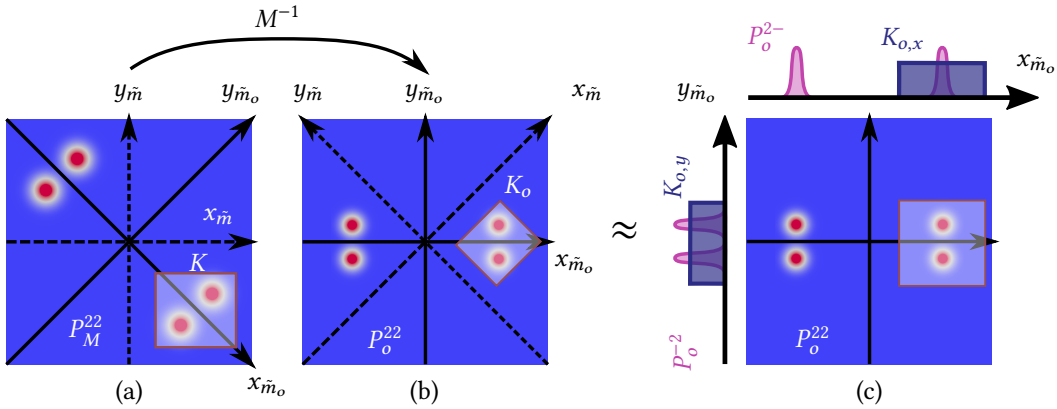

Fig. 6. Approximation of the filtered $\overline{P^{22}_M}$ by a separable convolution in the original half slope space. (a to b) M^{-1} transforms both the distribution P^{22}_M (to P^{22}_o) and the kernel K (to K_o). (b to c) K_o is approximated by a conservative separable form $K_{o,x}(x_{\tilde{m}_o})K_{o,y}(y_{\tilde{m}_o})$.

By a change of variables (Equation 5) in the convolution (Equation 11), and by using Equations 4 and 12, we obtain

$$\overline{P^{22}_M}(\tilde{m}, l, s_0) = \frac{1}{\det(M)} \left[K_o(\cdot) * P^{22}_o(\cdot, l, s_0)\right](M^{-1}\tilde{m}). \tag{13}$$

This is illustrated in Figure 6a and 6b. If the kernel K_o is a Dirac distribution, then Equation 13 becomes Equation 4 as expected.

Ultimately, the objective is to separate the convolution in order to independently filter each marginal distribution (Equation 6); recall that P^{2-}_o and P^{-2}_o are stored in 1D textures. To this end, we approximate the original kernel K_o by a conservative separable form (Figure 6c)

$$K_o(\tilde{m}_o) \approx K_{o,x}(x_{\tilde{m}_o})K_{o,y}(y_{\tilde{m}_o}). \tag{14}$$

As a result, Equation 13 is approximated

$$\overline{P_M^{22}}(\tilde{m}, l, s_0) \approx \frac{1}{\det(M)} \left[K_{o,x}(\cdot) * P_o^{2-}(\cdot, l, s_0)\right](x_{\tilde{m}_o}) \left[K_{o,y}(\cdot) * P_o^{-2}(\cdot, l, s_0)\right](y_{\tilde{m}_o}). \tag{15}$$

Since the marginal distributions P_o are normalized, the filtered distribution $\overline{P_M^{22}}$ is normalized and the BRDF is energy preserving if the kernels are normalized.

In summary, we have to implement Equations 15 in order to benefit from GGAA. In Section 6.1 we explain how to implement this Equation, using efficient GPU texture filtering.

5 NORMAL MAP FILTERING

Compared to Figure 1d, the renderings in Figures 1a and 1b are missing many highlights in the distance. This is due to a naive smoothing of the normal map. The problem has been solved by LEAN mapping [Olano and Baker 2010], which is based on the assumption that both the normal map and the microsurface have Gaussian distributions. Since the distribution P_o^{22} converges to a Beckmann (Gaussian) SDF in the distance (larger l in Figure 3), glints can be combined with LEAN mapping (Figure 1c). We first briefly recapitulate LEAN mapping and then propose a non-axis aligned anisotropic extension of Chermain et al. [2020] that makes this combination possible.

5.1 LEAN mapping

LEAN mapping efficiently filters normal maps by approximating not only the first, but also the second moments of normal map distributions within a footprint $\mathcal{P}$. Given a Beckmann microfacet distribution representing the material, and a slope map $\tilde{n} = (x_{\tilde{n}}, y_{\tilde{n}})$, a closed form of the patch-SDF is derived as an anisotropic Gaussian distribution with covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}. \tag{16}$$

All parameters (mean, variances σ_x and σ_y, and correlation factor ρ) can be efficiently computed from precomputed MIP maps of $x_{\tilde{n}}$, $y_{\tilde{n}}$, $x_{\tilde{n}}^2$, $y_{\tilde{n}}^2$, and $x_{\tilde{n}}y_{\tilde{n}}$.

The glinty BRDF of Chermain et al. [2020] allows the control of σ_x and σ_y, but there is no correlation factor ρ. However, as shown in Figure 7, controlling ρ is mandatory for normal map filtering, because it takes the covariance into account. This enables the representation of anisotropic lobes with arbitrary orientation, which appear when normal maps have anisotropic features (Figure 7). In the next section we show how to add a slope correlation factor ρ to the former method.

5.2 Slope Correlation Factor

In order to combine glints and LEAN mapping, we need to control the covariance matrix of P_M^{22} (and hence of P_{glint}^{22}) such that it converges towards Σ (Equation 16) when the LOD l increases. To achieve that, we use the matrix M which controls the variance of the SDF at the highest LOD (Equation 4 and Figure 3).

The original distribution P_o^{22} tends to an isotropic Gaussian with covariance matrix

$$\Sigma_o = \begin{bmatrix} \sigma_o^2 & 0 \\ 0 & \sigma_o^2 \end{bmatrix} \tag{17}$$

where σ_o is the roughness of the dictionary (Figure 3). In the former method [Chermain et al. 2020], the transformation M allows P_M^{22} to converge to *axis aligned* anisotropic Gaussian distribution, that is where the slope correlation factor always equals to zero ($\rho = 0$).

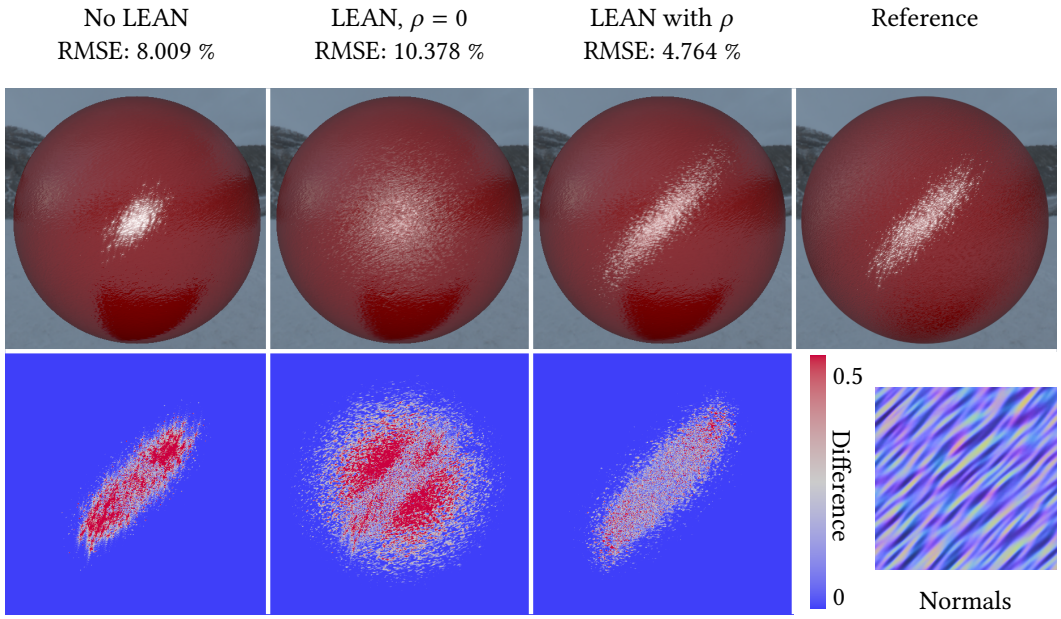

Fig. 7. LEAN mapping is combined with a glinty BRDF. Accounting for the slope correlation factor ρ is necessary to faithfully filter anisotropic normal maps.

To overcome this limitation, we introduce a control of the correlation factor through the new following transformation

$$M = \frac{1}{\sigma_o} \begin{bmatrix} \sigma_x \sqrt{1-\rho^2} & \rho\sigma_x \\ 0 & \sigma_y \end{bmatrix}. \tag{18}$$

By this, the covariance $M\Sigma_o M^T$ of P_M^{22} approaches Σ (Equation 16) as the LOD increases. There exists an infinity of appropriate matrices M, however, this particular one has a simple closed form which is a major advantage in the context of real-time rendering. Since M was already present in the model, the improvement comes almost for free.

Combining glints and LEAN mapping boils down to precomputing the MIP maps of $x_{\tilde{n}}$, $y_{\tilde{n}}$, $x_{\tilde{n}}^2$, $y_{\tilde{n}}^2$ and $x_{\tilde{n}} y_{\tilde{n}}$, and then in real-time

- compute σ_x and σ_y, and ρ as in Olano and Baker [2010],
- compute M using Equation 18,
- use M into Equation 4 (without glint anti-aliasing) or into Equation 15 (with glint anti-aliasing).

6 IMPLEMENTATION AND RESULTS

We implemented our GGAA method and the additional correlation factor using the publicly available OpenGL code of Chermain et al. [2020]. Both contributions are independent and can be implemented separately. The `C++` and `GLSL` code is available in the supplemental material. Our implementation uses forward rendering. Note that in all our renderings, we apply the star-shaped bloom post-effect of Zirr and Kaplanyan [2016] and a tonemap operator to better visualize the high dynamic range of the glints. Note that image-based lighting is not supported by the glinty BRDF: consequently, in all our results, glints are a result of point or directional lighting while the environment map is used for the diffuse and smooth specular components. When computing difference maps (in the figures) and

errors (RMSE) we only account for the specular component of the image coming from the glinty BRDF. Our result use only one sample per pixel. The reference renderings use 1,024 samples per pixel.

6.1 Implementing GGAA

The procedural *filtered* patch glinty SDF $\overline{P^{22}_{\text{glint}}}$ is a mixture of *filtered* linearly transformed SDFs $\overline{P^{22}_{M}}$ (Equations 10 and 15). To efficiently approximate the two convolutions involved, we use the OpenGL function `textureGrad` as

$$\begin{aligned}\overline{P^{22}_{M}}(\tilde{h}, l, s_0) \approx \frac{1}{\det(M)} \\ \text{textureGrad}\left(P^{2-}_{o}(l, s_0), x_{\tilde{h}_o}, s_K \frac{\mathrm{d}x_{\tilde{h}_o}}{\mathrm{d}x}, s_K \frac{\mathrm{d}x_{\tilde{h}_o}}{\mathrm{d}y}\right) \\ \text{textureGrad}\left(P^{-2}_{o}(l, s_0), y_{\tilde{h}_o}, s_K \frac{\mathrm{d}y_{\tilde{h}_o}}{\mathrm{d}x}, s_K \frac{\mathrm{d}y_{\tilde{h}_o}}{\mathrm{d}y}\right),\end{aligned} \tag{19}$$

where the derivatives are approximated using the OpenGL functions `dFdx` and `dFdy`. We experimented two variants, namely

$$\frac{\mathrm{d}\tilde{h}_o}{\mathrm{d}x} \approx M^{-1}\mathrm{dFdx}(\tilde{h}) \quad \text{and} \quad \frac{\mathrm{d}\tilde{h}_o}{\mathrm{d}y} \approx M^{-1}\mathrm{dFdy}(\tilde{h}), \tag{20}$$

as in [Kaplanyan et al. 2016]), and

$$\frac{\mathrm{d}\tilde{h}_o}{\mathrm{d}x} \approx M^{-1}\mathrm{dFdx}(h_\perp) \quad \text{and} \quad \frac{\mathrm{d}\tilde{h}_o}{\mathrm{d}y} \approx M^{-1}\mathrm{dFdy}(h_\perp), \tag{21}$$

as in [Tokuyoshi and Kaplanyan 2019]. The second variant, which is our default formula, uses the projected half vector $h_\perp = (x_h, y_h)^T$ to reduce the kernel estimation error at grazing angles, as demonstrated by Tokuyoshi and Kaplanyan [2019].

We decided not to use the derivative functions `dFdx` and `dFdy` directly on $\tilde{h}_o$, because it would imply one evaluation per cell s_0 in Equation 10 due to M which contains a random rotation depending on s_0 [Chermain et al. 2020]. Instead, Equations 20 and 21 evaluate derivative functions `dFdx` and `dFdy` only once per pixel, i.e., once per patch $\mathcal{P}$.

The scaling factor s_K in Equation 19 is set to a default value of 0.5 in order to avoid overlapping filter region between adjacent pixels (Figure 4 and 5). See Section 6.2 for a detailed discussion of this parameter.

To summarize, implementing GGAA boils down to computing the derivative (Equation 20 or 21), and then replacing the `textureLod`-functions in the former method by the `textureGrad`-functions in Equation 19. For each LOD of each tabulated distribution P_o in the dictionary (Figure 3), a MIP map is generated before the rendering. If the distributions are stored in a 1D array texture, using the OpenGL function `glGenerateMipmap` is sufficient. Consequently, the memory footprint increases by one third (from 384 KiB to 512 KiB in our implementation). Lastly, mirrored texture repetition must be activated (see supplemental document for more details).

6.2 GGAA Results

Geometric glint anti-aliasing is applicable to geometry without and with normal maps. In the first case, when the surface has high curvature, our glint anti-aliasing method yields renderings closer to the reference than the method without anti-aliasing, as demonstrated in Figure 8. When using

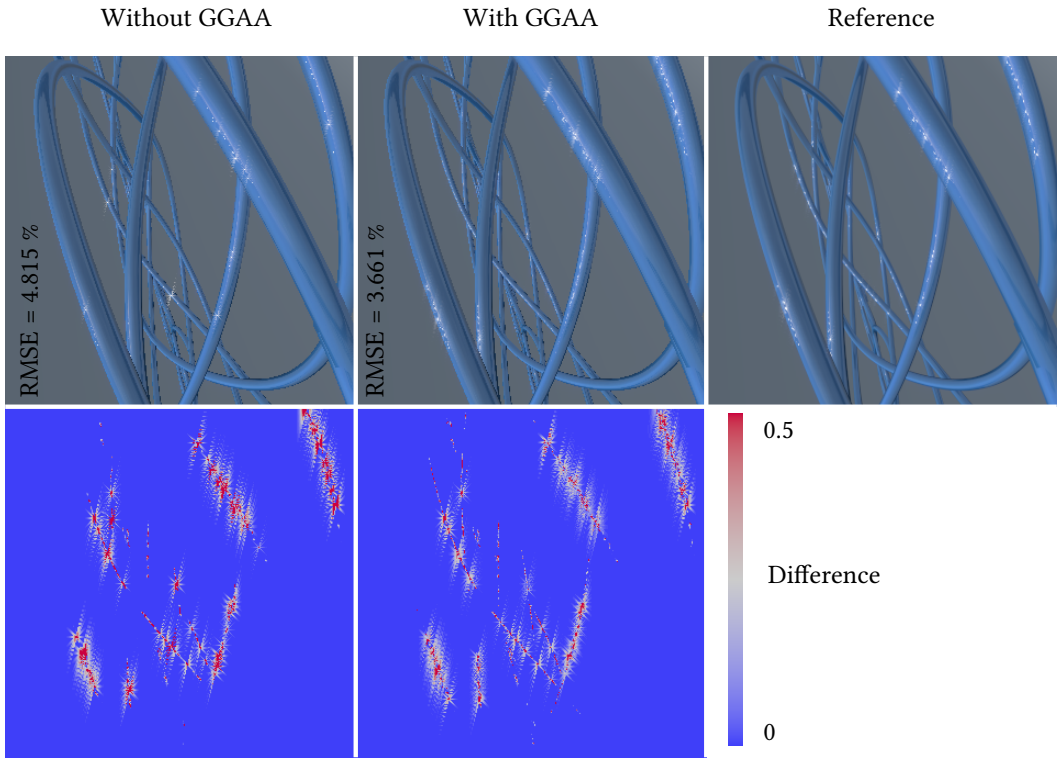

Fig. 8. Geometric glint anti-aliasing on curved metallic tubes with sparkles. GGAA yields smaller differences to the reference.

normal maps (without LEAN filtering), our anti-aliasing technique is also efficient, as demonstrated in Figures 1b and 9. In this case, the half slope is computed in the shading frame defined by the normal of the normal map, then the derivatives are computed (see Kaplanyan et al. [2016] for more details). Above all, GGAA improves the glints coherence between frames (see supplemental video).

In Figure 9, we compare Equations 20 and 21: using finite differences of projected half vectors yields lower errors than half slopes. We also compare different kernel size s by using $s_K = 0.5$ (default value) and $s_K = 1$. Small kernels ($s_K = 0.5$) yield results closer to the reference, but induce more specular shimmering in the animated sequences (see supplemental video). Larger kernels ($s_K = 1$) induce less specular shimmering, however, they lead to overfiltered specular highlights.

Performance. Performance has been measured on an NVIDIA GeForce 2080 RTX GPU with 1920×1080 resolution. The bloom post-effect, the tonemap operator, and LEAN filtering are disabled. As shown in Table 1, the rendering time overhead varies from 0.6 % to 4.2 % in our tests. In the method without GGAA, the MIP map for each distribution P_o is not generated to avoid increased memory traffic, the slopes derivatives are not computed and the call to the `textureGrad`-function is replaced by a call to the `textureLod`-function. These differences are responsible of the performance reduction.

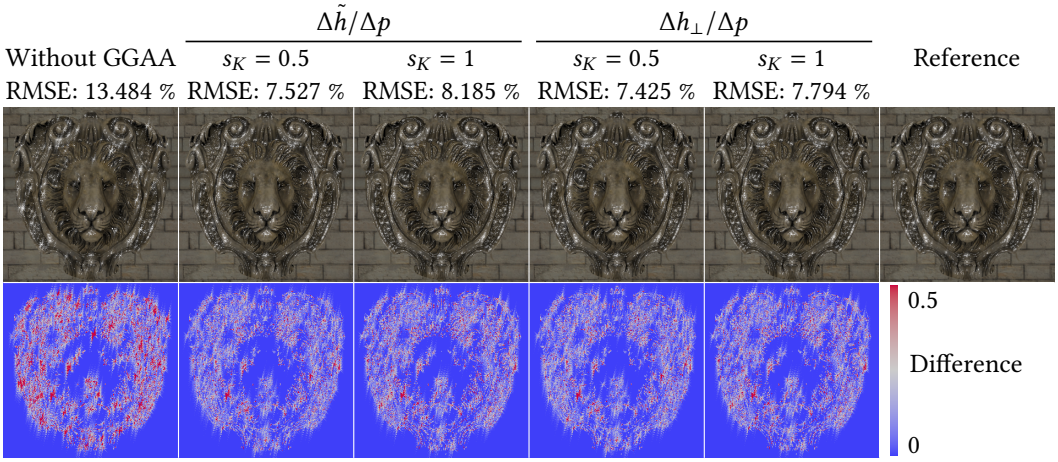

Fig. 9. Sparkling lion head. Comparison between a tight kernel ($s_K = 0.5$), and a large and conservative kernel ($s_K = 1$). RMSEs obtained with $s_K = 1$ are higher than RMSEs obtained with $s_K = 0.5$, but large kernels produce less specular shimmering in animations. Differences to the reference are also more prominent with half slopes derivatives $\Delta\tilde{h}/\Delta p$. For all settings, results with geometric glint anti-aliasing yield lower RMSEs and less specular shimmering.

	Figure 1 459 k triangles	Figure 8 40 k triangles	Figure 9 262 k triangles
No GGAA	3.68 ms	1.36 ms	3.03 ms
GGAA	3.70 ms	1.42 ms	3.15 ms

Table 1. Rendering times in milliseconds per frame (ms). The GGAA overhead is 0.6 % to 4.2 %.

6.3 Slope Correlation Factor Results

The slope correlation factor ρ introduced in Section 5.2 allows us to converge to non-axis aligned Beckmann (Gaussian) distributions as the microfacet density increases, as shown in Figure 10. The former method corresponds to the left column ($\rho = 0$).

The correctness of the matrix M in Equation 18 is tested in this Shadertoy https://www.shadertoy.com/view/3ddfDf. It shows that the transformed isotropic distribution is equivalent to the non-axis aligned Beckmann distribution [Heitz 2014].

6.4 LEAN mapping with Glint Results

In Figure 1c we show that normal map filtering using LEAN mapping combined with glint rendering better matches the reference. The major differences occur in the background, where normals are averaged and the roughness of the surface increases. In Figure 7 a sparkle sphere with an anisotropic normal map is shown with different settings for normal map filtering. The material has little roughness and thus the surface looks smooth without normal map filtering. Using LEAN mapping without the correlation factor ρ leads to an incorrect lobe, whereas the rendering with full LEAN mapping (with ρ) gives an appearance that matches the reference better. In Figure 1c and 7, our GGAA method is activated in order to remove most of geometric glint aliasing. When zooming in, sparkles can be seen individually (see the supplemental video). When zooming out, all the tiny glinty lobes aggregate to form a relatively smooth anisotropic specular lobe, as shown in Figure 7.

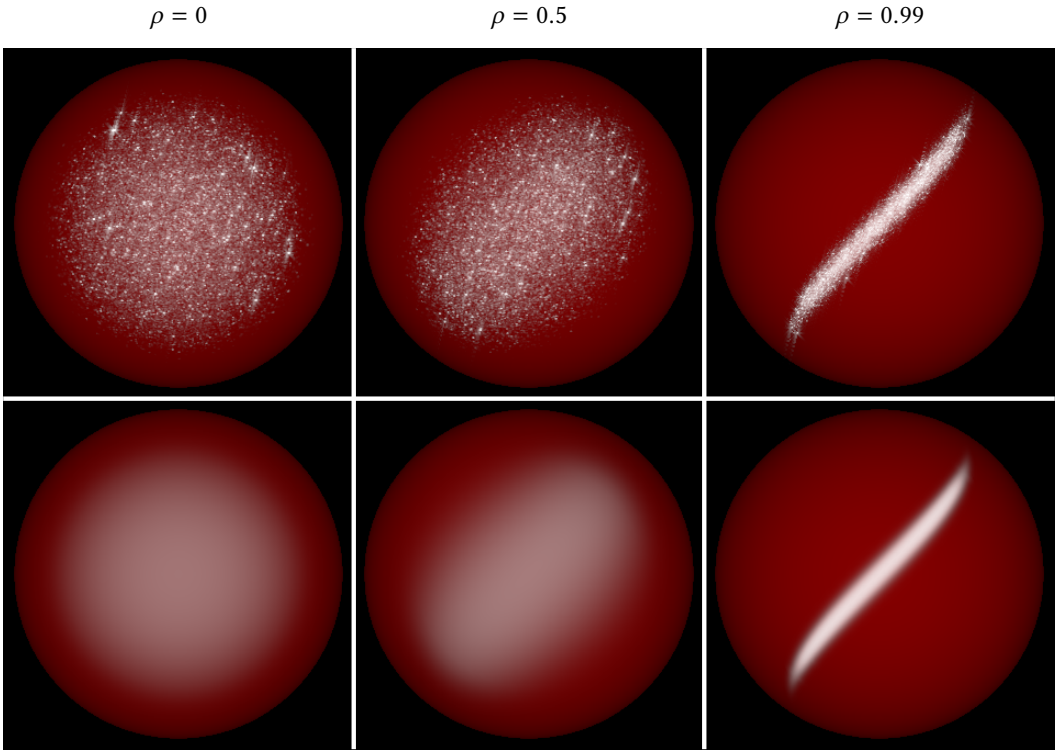

Fig. 10. Controlling the slope correlation factor ρ allows the lobe to be non-axis aligned. Top: glinty lobes (microfacet density equals exp(15)). Bottom: the NDF has converged to a Beckmann distribution (microfacet density equals exp(50))

7 DISCUSSION AND LIMITATIONS

Energy preservation. The model remains physically based with the geometric glint anti-aliasing (Section 4), provided that the filtering kernels are normalized. Indeed, the marginal distributions P_o^{2-} and P_o^{-2} are normalized so the filtered distribution $\overline{P_M^{22}}$ in Equation 15 is still normalized. Hence $\overline{P_{\text{glint}}^{22}}$ is also normalized (Equation 10). We noticed no energy creation in our experiments though we have not experimentally proved it. We conjecture that `textureGrad` uses normalized discrete kernels.

Normal map filtering. Our work does not benefit from LEADR mapping and its Smith masking function for non-centered distribution because the BRDF of Chermain et al. [2020] only handles the V-cavity masking function. To the best of our knowledge, there is no V-cavity function for non-centered distributions. Finding it is an interesting future work that would benefit normal map filtering combined with glint rendering.

Our normal map filtering method assumes that the glinty SDF and the distribution induced by the normal map are both Gaussians. If these assumptions are not fulfilled, then the surface can look too rough and/or too glinty.

Remaining Geometric Glint Aliasing. Our GGAA method does not remove glint aliasing entirely, especially when combined with normal maps (see the supplemental video). Indeed, it inherits the limitations of geometric specular anti-aliasing [Kaplanyan et al. 2016; Tokuyoshi and Kaplanyan

2019]. Firstly, geometric aliasing is still there and can also cause specular shimmering. Secondly, these methods and ours rely on several assumptions and approximations: first-order Taylor approximation of the half slopes, minor curvature variation in the local neighborhood of the surface, and finite differences using GPU function. However, our GGAA technique visibly reduces flicker aliasing without significantly impacting memory and performance.

8 CONCLUSION

We have proposed geometric glint anti-aliasing to remove a significant part of the disturbing specular shimmering present during animations when using real-time glinty BRDFs and curved surfaces (normal mapped or not). Our technique requires only one third additional memory while having little impact on performance (the overhead varies from 0.6 % to 4.2 % in our tests). We have also introduced a slope correlation factor to the glinty BRDF of Chermain et al. [2020] and combine normal filtering with real-time glint rendering.

ACKNOWLEDGMENTS

This work has been published under the framework of the IdEx Unistra supported by the Investments for the future program of the french government. This work has been partially funded by the project ReProcTex from the Agence Nationale de la Recherche (ANR-19-CE33-0011-01) and the Deutsche Forschungsgemeinschaft (project 431478017). The Sponza scene is provided by Morgan McGuire (https://casual-effects.com/data) and metallic tubes model is provided by flipkidh (https://blendswap.com/blend/4086).

REFERENCES

John Amanatides. 1992. Algorithms for the Detection and Elimination of Specular Aliasing. In *Proc. of the Conference on Graphics Interface*. 86–93.

Asen Atanasov and Vladimir Koylazov. 2016. A Practical Stochastic Algorithm for Rendering Mirror-like Flakes. In *ACM SIGGRAPH Talks*.

Barry G. Becker and Nelson L. Max. 1993. Smooth Transitions between Bump Rendering Algorithms. In *Proc. ACM SIGGRAPH*. 183–190.

Petr Beckmann and André Spizzichino. 1963. *The scattering of electromagnetic waves from rough surfaces*. Pergamon Press.

Xavier Chermain, Basile Sauvage, Dischler Jean-Michel, and Carsten Dachsbacher. 2020. Procedural Physically-based BRDF for Real-Time Rendering of Glints. *Comput. Graph. Forum (Proc. Pacific Graphics)* 39, 7 (2020), 243–253.

R. L. Cook and K. E. Torrance. 1982. A Reflectance Model for Computer Graphics. *Comput. Graph. (Proc. SIGGRAPH)* 1, 1 (1982), 7–24.

Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. 2013. Linear Efficient Antialiased Displacement and Reflectance Mapping. *ACM Trans. Graph.* 32, 6 (2013), 211:1–211:11.

Alain Fournier. 1992. Normal distribution functions and multiple surfaces. In *Proc. of the Conference on Graphics Interface*. 45–52.

Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. 2007. Frequency Domain Normal Map Filtering. *ACM Trans. Graph. (Proc. SIGGRAPH)* 26, 3 (2007).

Eric Heitz. 2014. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs. *Journal of Computer Graphics Techniques* 3, 2 (2014), 48–107.

Stephen Hill and Dan Baker. 2012. Rock-Solid Shading: Image Stability Without Sacrificing Detail. In *Advances in Real Time Rendering, SIGGRAPH Courses*.

Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. 2014. Discrete Stochastic Microfacet Models. *ACM Trans. Graph. (Proc. SIGGRAPH)* 33, 4 (2014).

A. S. Kaplanyan, S. Hill, A. Patney, and A. Lefohn. 2016. Filtering Distributions of Normals for Shading Antialiasing. In *Proc. of ACM SIGGRAPH / Eurographics conference on High Performance Graphics*. 151–162.

Guillaume Loubet, Tizian Zeltner, Nicolas Holzschuch, and Wenzel Jakob. 2020. Slope-Space Integrals for Specular next Event Estimation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39, 6 (2020).

Marc Olano and Dan Baker. 2010. LEAN Mapping. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 181–188.

Michael Toksvig. 2005. Mipmapping Normal Maps. *Journal of Graphics Tools* 10, 3 (2005), 65–71.

Yusuke Tokuyoshi and Anton S. Kaplanyan. 2019. Improved Geometric Specular Antialiasing. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*.

Alex Vlachos. 2015. Advanced VR rendering. In *Game Developers Conference Talk*.

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet Models for Refraction Through Rough Surfaces. In *Proc. Eurographics Symposium on Rendering*. 195–206.

Beibei Wang, Hong Deng, and Nicolas Holzschuch. 2020. Real-Time Glints Rendering With Pre-Filtered Discrete Stochastic Microfacets. *Comput. Graph. Forum* (2020).

Beibei Wang, Lu Wang, and Nicolas Holzschuch. 2018. Fast Global Illumination with Discrete Stochastic Microfacets Using a Filterable Model. *Comput. Graph. Forum (Proc. Pacific Graphics)* 37, 7 (2018), 55–64.

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 Trans. Graph. (Proc. SIGGRAPH)* 33, 4 (2014).

Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. 2016. Position-Normal Distributions for Efficient Rendering of Specular Microstructure. *ACM Trans. Graph. (Proc. SIGGRAPH)* 35, 4 (2016).

Tobias Zirr and Anton S. Kaplanyan. 2016. Real-time Rendering of Procedural Multiscale Materials. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 139–148.