

# Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces

Ling-Qi Yan<sup>1\*</sup> Miloš Hašan<sup>2\*</sup> Wenzel Jakob<sup>3</sup> Jason Lawrence<sup>4</sup> Steve Marschner<sup>5</sup> Ravi Ramamoorthi<sup>1</sup>  
\*dual first authors <sup>1</sup>Univ. of California, Berkeley <sup>2</sup>Autodesk <sup>3</sup>ETH Zürich <sup>4</sup>Univ. of Virginia <sup>5</sup>Cornell Univ.



**Figure 1:** A rendering of highly specular objects under point lighting. A high-resolution normal map ( $2048^2$ ) makes rendering impractical with standard techniques: the highlights are missed by naive pixel sampling. **Left inset:** Our solution is based on the concept of a pixel normal distribution function ( $\mathcal{P}$ -NDF), which can be highly complex. **Right inset:** Our method is accurate even in a moving-light sequence.

## Abstract

Complex specular surfaces under sharp point lighting show a fascinating glint appearance, but rendering it is an unsolved problem. Using Monte Carlo pixel sampling for this purpose is impractical: the energy is concentrated in tiny highlights that take up a minuscule fraction of the pixel. We instead compute an accurate solution using a completely different deterministic approach. Our method considers the true distribution of normals on a surface patch seen through a single pixel, which can be highly complex. We show how to evaluate this distribution efficiently, assuming a Gaussian pixel footprint and Gaussian intrinsic roughness. We also take advantage of hierarchical pruning of position-normal space to rapidly find texels that might contribute to a given normal distribution evaluation. Our results show complex, temporally varying glints from materials such as bumpy plastics, brushed and scratched metals, metallic paint and ocean waves.

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

**Keywords:** glints, specular highlights, high-resolution normal maps, normal distribution functions

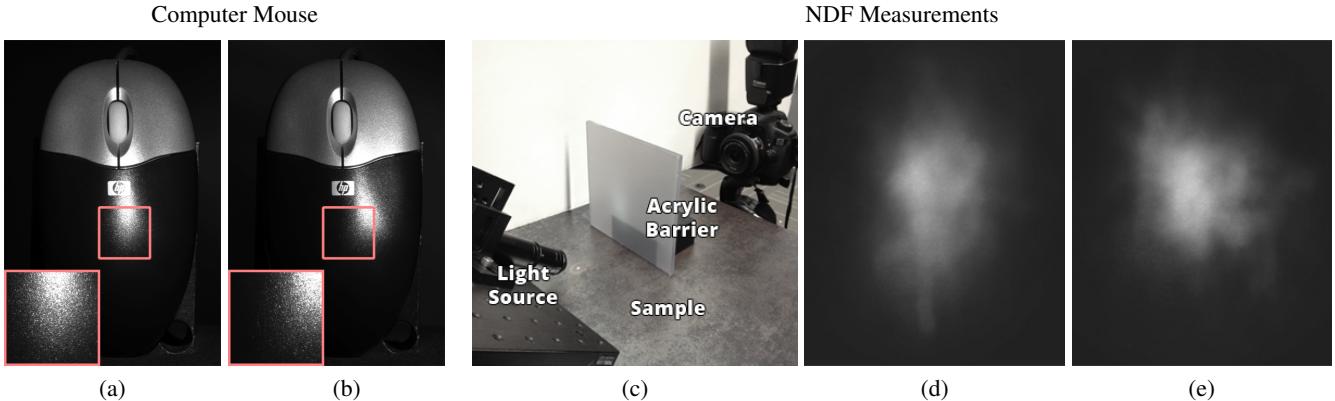
**Links:** DL PDF

## 1 Introduction

Conventional BRDFs model complex microgeometry using a smooth normal distribution function (NDF) of infinitely small microfacets. However, real surface features are certainly not infinitely small. Bumps and flakes from anywhere between a few microns (brushed metal) to about 0.1 mm (flakes in metallic paints) to centimeters (ocean waves) can produce interesting glint behavior that is visible with the naked eye. These glints are very pronounced with a light source that subtends a small solid angle, such as the sun and small light fixtures. This is true for surfaces specifically designed to glint, such as metallic paints with embedded flakes or decorative brushed metals, but also for everyday objects such as plastics or ceramics (see Figure 2, left). In fact, smooth surfaces that meet the microfacet assumption are the exception rather than the norm. Most shiny surfaces that one encounters in reality have this type of glint behavior, readily observed under sharp lighting.

Our goal is to simulate glint appearance in still images and animations (see Figure 1 and the supplemental video). Representing geometry at a resolution sufficient to reveal the features that cause glints is not difficult: we use high-resolution normal maps. A much harder challenge is rendering a complex specular surface under sharp lighting. Standard uniform pixel sampling techniques for direct illumination have extremely large variance, and using them for this purpose is impractical. The reason is that most of the energy is concentrated in tiny highlights that take up a minuscule fraction of a pixel, and uniform pixel sampling is ineffective at hitting the highlights (Figure 3). An alternative explanation is that the space of valid camera-surface-light paths is complicated and cannot be easily sampled from the camera or from the light. In some sense, we need to search the surface for normals aligned with the half vector, and this cannot be done by brute-force sampling.

Normal map filtering techniques [Toksvig 2005; Han et al. 2007; Olano and Baker 2010; Dupuy et al. 2013] also do not fully solve the problem. These methods attempt to approximate the NDF at a



**Figure 2:** (a-b) Two photographs of an injection molded plastic computer mouse illuminated by a small light source ( $\sim 3.5 \times 10^{-5}$  sr solid angle) reveal its glinty appearance. These effects are impractical to simulate using uniform pixel sampling. (c-e) Real-world normal distribution functions of a dark bumpy ceramic tile were measured by illuminating the surface with a small focused incoherent source ( $\sim 6.2 \times 10^{-5}$  sr solid angle covering a surface patch of  $\sim 0.5^2$  mm). The images in (d) and (e) were captured by a camera located opposite a diffuse acrylic barrier from the source. They reveal a distinct non-Gaussian distribution of scattered light, corresponding to the  $\mathcal{P}$ -NDF of the surface patch, only slightly warped and blurred because of the optical limits of our setup.

given scale by broad lobes, but the true NDF is highly complex; it cannot be approximated well using a single Gaussian lobe, or even a small number of lobes (Figure 4). Although these approaches avoid aliasing artifacts, they are not able to reproduce glinty appearance under high-frequency lighting. We instead desire to compute the true solution that Monte Carlo would eventually converge to, using a completely different deterministic approach with minimal approximations.<sup>1</sup>

We consider the actual, unsimplified NDF of a surface patch  $\mathcal{P}$  seen through a single pixel (an example is shown in Figure 1, left inset). This  $\mathcal{P}$ -NDF can be easily estimated by binning: repeatedly choose a point on the patch, take its normal, perturbed by the intrinsic surface roughness, and add it into a bin. The key problem is that for direct illumination, we need to evaluate the  $\mathcal{P}$ -NDF for a *single* half-vector. Clearly, it would be extremely inefficient to use the binning approach here, wasting all but a single bin. Indeed, this is equivalent to what a standard renderer would do, trying to hit a tiny light source by chance. Instead, we require evaluation of the density of a single normal coming from anywhere on the patch. Moreover, the  $\mathcal{P}$ -NDF is different for every pixel, so computations cannot be reused. In our method, the  $\mathcal{P}$ -NDF is just a mathematical tool to derive what the correct pixel brightness should be; it is never fully constructed, and only evaluated for a single vector.

We introduce an algorithm for  $\mathcal{P}$ -NDF evaluation in Section 4. The key assumptions that make the evaluation possible are a Gaussian pixel filter and a tiny amount of Gaussian roughness on the specular surface. These combine into a single 4-dimensional Gaussian “query” that is *analytically integrated* across the normal map, avoiding random sampling. A basic computational block of our solution is an integral of a 2-dimensional Gaussian over a triangular domain, described in Section 5. We hierarchically prune position-normal space to quickly find texels that might contribute to a given  $\mathcal{P}$ -NDF evaluation (Section 6). Our results show complex, temporally varying glints from bumpy plastics, brushed and scratched metals, metallic paint and ocean waves; see Section 7 and Figure 10.

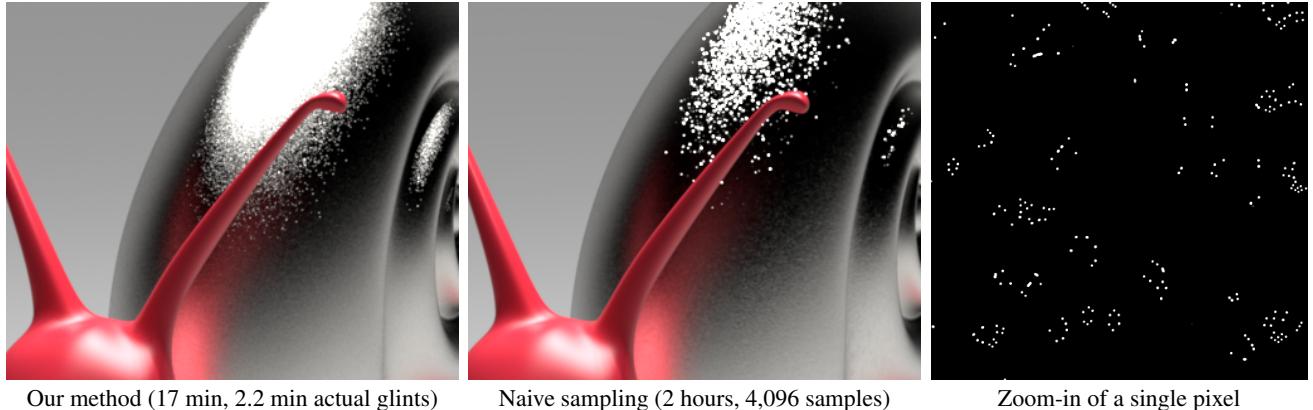
<sup>1</sup>Specifically, constant view and light direction over  $\mathcal{P}$ , and the approximations made when solving the integral in Section 5.

## 2 Related work

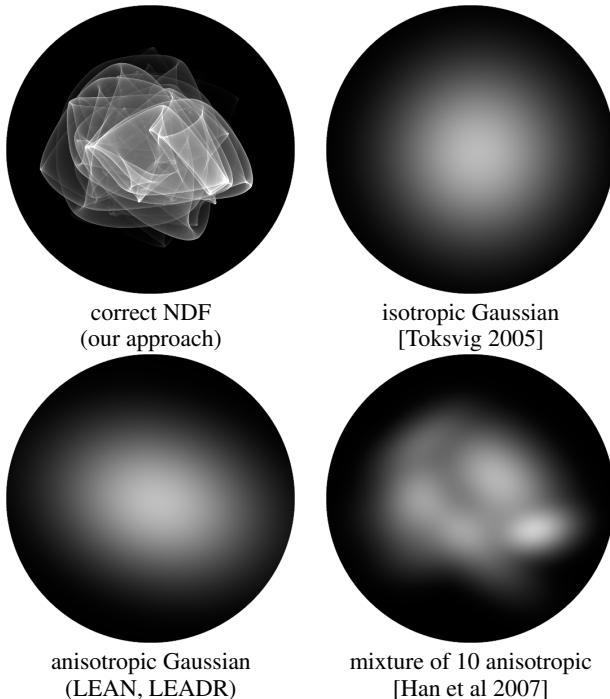
**Naive pixel sampling.** The standard approach to compute direct illumination on a bumpy specular surface is to trace a ray through the pixel, evaluate the normal of the hit point, and shade the point from a light source using the point’s finite roughness BRDF; this fails at rendering glints (Figure 3). Multiple importance sampling [Veach 1997] does not help, because it is the pixel integral that is inefficiently sampled, rather than the BRDF/light combination. The REYES approach of surface subdivision into micropolygons [Cook et al. 1987] is equally inefficient, since it would require micropolygons as small as the highlights. Though we use fine triangulations of the normal map for smoothness, our method can handle highlights that are arbitrarily smaller than the triangles.

**Normal map filtering techniques** can deliver artifact-free renderings by approximating a pixel’s NDF by a single lobe [Toksvig 2005; Olano and Baker 2010; Dupuy et al. 2013] or a small number of lobes [Han et al. 2007]. However, none of these methods can correctly capture glinty appearance. The core of the problem is that the true NDFs can be highly complex, and their sharp features directly translate into spatial and temporal glinting. Approximating them by broad lobes is only applicable under low-frequency illumination that would filter the complex features anyway. Figure 4 shows the effect of replacing the true NDF by a single Gaussian or a mixture of Gaussians, thus losing the sharp features.

**Single-point evaluation of caustics.** Caustics are related to our work, since glints can be interpreted as “directional caustics”. Most methods sample paths (particles, photons) and accumulate them in a data structure (kd-tree, hash grid, or bins). However, this is not sufficient for our purposes; we require point evaluation, which is much harder. Walter et al. [2009] compute volumetric caustics due to the refraction of a point light into a scattering volume through a bumpy interface. This is related to our approach: linear normal interpolation over triangles is used, a discrete set of specular connections is enumerated, the Jacobian determinant term determines highlight intensity, and a hierarchy is used to speed up the enumeration. However, no intrinsic roughness is considered (resulting in singularities), and the phenomenon rendered is quite different. Mitchell and Hanrahan [1992] compute reflected caustics from an implicit surface by enumerating the discrete set of valid light paths through interval arithmetic. They used waveform tracing as a way to compute the contribution of a valid specular path; this is again



**Figure 3:** *Naive pixel sampling fails at rendering complex specular surfaces under small lights. The reason is that the highlights are too small to be efficiently hit by uniform pixel sampling, which is obvious from the zoomed pixel on the right. Multiple importance sampling would not help, since it is the pixel integral that is inefficiently sampled, not the light/BRDF combination.*



**Figure 4:** *Approximating the true NDF by a single Gaussian (Toksvig, LEAN, LEADR) or a small number of Gaussians (Han et al.) loses the sharp features that cause glinting.*

equivalent to the Jacobian determinant term for a single reflection, with the associated singularities.

**Other work on specular paths.** Jakob and Marschner [2012] is an extension of Metropolis light transport, which allows mutation of a specular path at a single diffuse vertex; however, in our case, no diffuse vertex is available for mutations. In the perfectly specular case, there is a discrete set (rather than a manifold) of valid paths, as already noted above. Moon et al. published several approaches to approximate higher-order specular bounces, e.g. [2007], but low-order specular paths are still computed brute-force with a relatively large light source.

symbol	domain	definition
$\mathcal{D}$		unit disk (proj. hemisphere)
$\perp$		invalid normal
$\mathbf{s} = (s, t)$	$\mathcal{D}$	unit disk parameters, defining vectors $(s, t, \sqrt{1 - s^2 - t^2})$
$\mathbf{u} = (u, v)$	$\mathbb{R}^2$	texture space parameters
$n(\mathbf{u})$	$\mathbb{R}^2 \rightarrow \mathcal{D}$	normal map function
$J(\mathbf{u})$	$\mathbb{R}^2 \rightarrow \mathbb{R}^{2 \times 2}$	Jacobian of $n(\mathbf{u})$
$\mathcal{P}$		pixel footprint
$G_p(\mathbf{u})$	$\mathbb{R}^2 \rightarrow \mathbb{R}$	pixel Gaussian
$G_r(\mathbf{s})$	$\mathbb{R}^2 \rightarrow \mathbb{R}$	intrinsic roughness Gaussian
$G_c[\mathcal{P}, \mathbf{s}](\mathbf{x}, \mathbf{y})$	$\mathbb{R}^4 \rightarrow \mathbb{R}$	combined Gaussian query for footprint $\mathcal{P}$ and normal $\mathbf{s}$
$D(\mathbf{s})$	$\mathcal{D} \cup \perp \rightarrow \mathbb{R}$	normal distribution function

**Table 1:** Notation used in the paper.

**Stochastic reflectance.** [Jakob et al. 2014] is concurrent work that also addresses the problem of glinty surfaces, using a stochastic approach. Rather than work from a normal map, that method models the surface as a procedural random collection of specular flakes that occur according to a particular normal distribution. The key to their method is counting up the particles contributing to a particular illumination calculation without actually generating them, providing efficiency for large query areas where many particles contribute. When used as a model for a bumpy smooth surface, the stochastic approach is phenomenological: the random-flake approximation replaces the  $\mathcal{P}$ -NDF. In contrast, our algorithm exactly determines how a given specular surface, defined by a particular normal map, really looks under given sharp illumination. Moreover, normal maps can express surface features large enough to be visible in the image, e.g. the scratched and brushed examples in this paper.

### 3 Preliminaries

Solving our problem requires thinking about a surface patch  $\mathcal{P}$  seen through a pixel all at once, rather than one point at a time. Just as every surface point has a local BRDF, we can think of areas of the surface having  $\mathcal{P}$ -BRDFs that describe how the total contribution to the pixel depends on the illumination. Rendering detailed normal maps requires an efficient way to evaluate the area-integrated  $\mathcal{P}$ -BRDF, rather than letting the pixel filter do it implicitly by point sampling.

For a specular normal-mapped surface, this area-integrated BRDF is primarily determined by the distribution of surface normals over the relevant patch of surface: we need to be able to ask “how often” a given normal vector occurs in the patch. We call this distribution the  $\mathcal{P}$ -NDF; it is just like the microfacet distribution in a standard BRDF model, but it gives the normal distribution for a particular area rather than a global average over the whole surface. A crucial observation is that the  $\mathcal{P}$ -NDF is not a simple, broad function. It contains a surprising amount of structure (Figure 5) even when the surface patch is far larger than the features in the normal map. It also varies dramatically across the surface. Evaluating the  $\mathcal{P}$ -NDF efficiently while preserving this detailed spatio-angular structure is the key to accurately capturing glinty appearance.

Let us define these terms more precisely. Table 1 lists the symbols used throughout the paper.

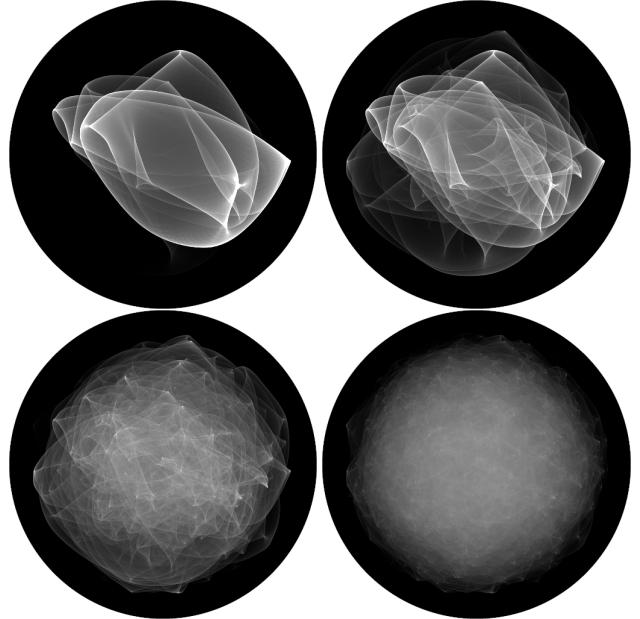
**Pixel footprint.** We assume a Gaussian pixel reconstruction filter. This projects to an approximately Gaussian footprint  $\mathcal{P}$  in the uv-parameterization of the normal map, whose covariance matrix is easily computed by propagating ray differentials to the surface [Igehy 1999]. In practice, we actually subdivide pixels into  $4 \times 4$  subpixels, and make the footprints smaller accordingly. This handles edges better, but for simplicity we will talk about pixel rather than subpixel footprints.

**Projected hemisphere.** We will use the unit disk  $\mathcal{D}$  to express hemispherical unit vectors. The point  $\mathbf{s} = (s, t) \in \mathcal{D}$  represents the unit vector  $(s, t, \sqrt{1 - s^2 - t^2})$  on the hemisphere. Let us also define the *extended unit disk* as the union of the unit disk and a special symbol  $\perp$ , which allows for normal distributions that sometimes return invalid normals. This is less common than working with hemispheres, but it will be useful shortly.

**Normal maps** can be given directly or as the derivative of a heightfield. We use the direct option, though all but one normal map in our examples do come from a heightfield (the exception is the metallic paint flakes). The normal map is then defined as a function  $n : \mathbb{R}^2 \rightarrow \mathcal{D}$  from points  $\mathbf{u} = (u, v)$  in texture space to normals  $\mathbf{s} = (s, t)$ . The Jacobian of  $n(\mathbf{u})$ , denoted  $J(\mathbf{u})$ , plays an important role in determining highlight brightness, and points where  $\det J(\mathbf{u}) = 0$  cause problems unless we are careful.

**Intrinsic roughness.** We could treat the surface as perfectly specular; however, we found that it is useful to consider a small amount of unresolved fine roughness. This matches the real world in that perfect smoothness is unachievable and the limits of geometric optics are reached at very high resolutions. It also prevents singularities (infinitely bright highlights), which arise with perfectly specular surfaces when  $\det J(\mathbf{u}) = 0$ , and cleanly deals with normal maps that contain piece-wise constant regions.

**NDFs.** We can now define a normal distribution function (NDF) as a probability distribution on the extended unit disk, with the obvious measure. (The associated random event is simply a “choice of normal”.) This definition slightly deviates from standard references such as [Walter et al. 2007] and [Burley 2012], but it is fully compatible with them, and is actually more convenient. In hemispherical terms, NDFs like Beckmann and GGX require an additional cosine term to integrate to 1, and their associated sampling routines also bake in a cosine (see eq. (4) and (28) in Walter et al. [2007]); in our formulation, no cosines need to be worried about. Furthermore, we now have more freedom in what passes as an NDF: any suitable plane function can be restricted to the unit disk and properly normalized. In particular, Gaussians are perfectly good NDFs, and this includes anisotropic and non-centered ones. Finally, statements such as “blur an NDF by a Gaussian” now have a very precise meaning. Even though this is different from spherical convolutions with vMF or Kent distributions, the difference is not critical to us:



**Figure 5:** The  $\mathcal{P}$ -NDFs of a smooth specular heightfield with a Gaussian power spectrum, with a pixel footprint covering about  $15 \times 15$ ,  $30 \times 30$ ,  $90 \times 90$  and  $300 \times 300$  texels respectively.

we simply use the convolutions to avoid singularities coming from unrealistically perfect surfaces.

**The  $\mathcal{P}$ -NDF** can now be defined as the probability distribution of the random variable defined by sampling the footprint  $\mathcal{P}$ , evaluating the normal at the sampled location, and perturbing by the intrinsic roughness kernel. The last step can sometimes result in a normal outside of the unit disk; these events are collected by the probability of  $\perp$ , and are often near zero in practice. Figure 5 shows different  $\mathcal{P}$ -NDFs as the size of the pixel footprint increases. Note that quite large footprint sizes are required for these NDFs to start to mimic analytic normal distributions like Beckmann.

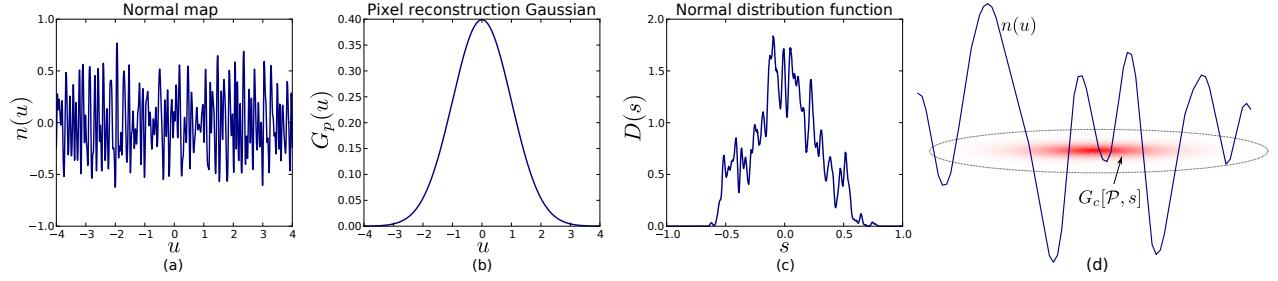
## 4 $\mathcal{P}$ -NDF evaluation in flatland and 3D

Our core challenge is to find an evaluation algorithm for the  $\mathcal{P}$ -NDF  $D(\mathbf{s})$  for a half-vector  $\mathbf{s}$ , corresponding to a given footprint on a given normal map and with a given intrinsic roughness; indeed, with such an algorithm at hand, it is straightforward to plug the  $\mathcal{P}$ -NDF into a standard microfacet BRDF, which can be used for direct illumination calculations:

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

where  $\mathbf{h} = (\mathbf{i} + \mathbf{o})/\|\mathbf{i} + \mathbf{o}\|$  is the half vector,  $\mathbf{n}$  is the unmapped surface normal,  $F$  is the Fresnel term, and  $G$  is a shadowing-masking term (only needed to avoid infinities at grazing). In the following sections, we will first make the  $\mathcal{P}$ -NDF evaluation problem more approachable by analyzing the situation in flatland, and then present the full 3D solution, which naturally follows from the flatland case.

**The flatland situation** is simpler: there is only one texture parameter  $u$ . The normal map can be written as a function  $n(u)$  returning normals in  $(-1, 1)$ , which is analogous to the unit disk from the 3D case. The full normal vector is  $(n(u), \sqrt{1 - n(u)^2})$ . The pixel footprint  $\mathcal{P}$  will turn into a Gaussian reconstruction kernel  $G_p(u)$  that integrates to 1. Let  $X$  be a random variable that is distributed



**Figure 6:** Flatland illustration of  $\mathcal{P}$ -NDF sampling and evaluation. **(a)** A normal map is a 1D curve  $n(u)$  of the texture coordinate  $u$ . (The other component of the normal vector is  $\sqrt{1 - n(u)^2}$ ). **(b)** The pixel of interest projects to a Gaussian footprint given by  $G_p(u)$ . **(c)** The  $\mathcal{P}$ -NDF  $D(s)$  giving the probability density of a given normal  $(s, \sqrt{1 - s^2})$ , assuming an intrinsic roughness kernel  $G_r(s)$  with  $\sigma = 0.01$ . **(d)**  $\mathcal{P}$ -NDF evaluation in flatland can be visualized as integration of the combined Gaussian query  $G_c[\mathcal{P}, s]$  over the segmented graph of the normal map. In areas where the Gaussian is effectively zero (outside of the ellipse) we can prune the segments using a hierarchy.

according to  $G_p(u)$ . The key question is, what is the distribution of the random variable  $n(X)$  on  $(-1, 1)$ ? This is not a simple multiplication or convolution of the normal map with  $G_p$ , but instead a pdf of a dependent random variable. The situation is illustrated in Figure 6.

We can write down the  $\mathcal{P}$ -NDF as:

$$D(s) = \int_{-\infty}^{\infty} G_p(u) \delta(n(u) - s) du = \sum_i \frac{G_p(u_i)}{|n'(u_i)|}, \quad (2)$$

where  $u_i$  are the roots of the equation  $n(u) = s$ . The delta function restricts the integral to points where  $n(u) = s$ , and the second equation intuitively accounts for the “speed” of crossing the root; it only works if a finite set of roots exists. As we can see, the  $\mathcal{P}$ -NDF will have singularities at points where  $n'(u) = 0$ . These correspond to inflection points of the original heightfield. This analysis shows that the  $\mathcal{P}$ -NDF can have infinite values. If we use a pin-hole camera and a point light, this can cause infinitely bright pixels. (Our distant light/camera approximation is not the culprit; infinities could occur even if we did not make this approximation.) Furthermore, there could be constant regions in the normal map, so we get  $n'(u) = 0$  for whole intervals, and corresponding delta functions in the  $\mathcal{P}$ -NDF.

To avoid singularities and other problems inherent in perfect specular surfaces, we introduce a tiny amount of finite roughness to the normal-mapped surface. Since the  $\mathcal{P}$ -NDF is just a function on the interval  $(-1, 1)$ , we can convolve it with a Gaussian  $G_r(s)$  easily:

$$\begin{aligned} D(s) &= \int_{-1}^1 G_r(s - s') \int_{-\infty}^{\infty} G_p(u) \delta(n(u) - s') du ds' \\ &= \int_{-\infty}^{\infty} G_p(u) \int_{-1}^1 G_r(s - s') \delta(n(u) - s') ds' du \\ &= \int_{-\infty}^{\infty} G_p(u) G_r(n(u) - s) du \\ &= \int_{-\infty}^{\infty} G_c[\mathcal{P}, s](u, n(u)) du. \end{aligned} \quad (3)$$

In the last step, we combined the two 1D Gaussians into a single 2D one:

$$G_c[\mathcal{P}, s](x, y) = G_p(x) G_r(y - s). \quad (4)$$

By changing the integration order and eliminating delta functions, we have removed any notion of root finding or singularities from the problem, leaving a single well-defined integral of a one-dimensional real function. An elegant way to intuitively visualize

the result is that we would like to integrate the combined reconstruction kernel  $G_c[\mathcal{P}, s]$  along the *graph* of the normal function, the plane curve  $(u, n(u))$ . Note, though, that the measure is the standard line measure on the  $u$  axis, not arc length along the graph. Figure 6 (d) illustrates this intuition, and immediately leads to an accelerated query idea: we can use a hierarchy to prune all normal map segments in areas where  $G_c[\mathcal{P}, s]$  is effectively zero.

In flatland,  $G_c$  is a 2D Gaussian, so we can subdivide the graph into many line segments, and integrate the combined kernel along the line segments. This leads to integrals of 1-dimensional Gaussians over the segments, which can be computed easily in terms of  $\text{erf}(\cdot)$ . This shows the benefit of choosing Gaussian filters; other choices such as splines would lead to integration problems without closed-form solutions.

Also note that we made the *graph* piecewise-linear, instead of the full integrand  $G_c(u, n(u))$ : the latter would be a bad choice, since *the Gaussian can be much narrower than the discretization step*. We would like to handle specular highlights arbitrarily smaller than the finest discretization level, and this choice is key to achieving that goal.

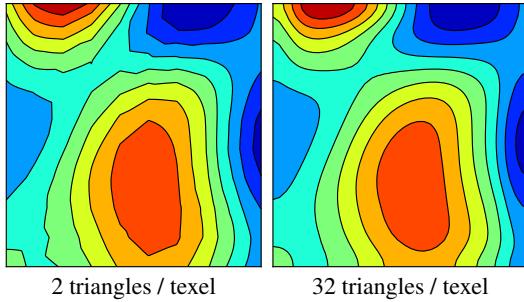
**3D analysis.** We can extend the above line of thinking to three dimensions, with two-dimensional texture space parameterized by  $\mathbf{u} = (u, v)$ , and a normal function  $n : \mathbb{R}^2 \rightarrow \mathcal{D}$ .

A 2D Gaussian reconstruction kernel  $G_p : \mathbb{R}^2 \rightarrow \mathbb{R}$  now models the pixel footprint  $\mathcal{P}$ . The random process of choosing a position  $\mathbf{u}$  by sampling  $G_p$  and taking its normal will have the following probability distribution:

$$D(\mathbf{s}) = \int_{\mathbb{R}^2} G_p(\mathbf{u}) \delta(n(\mathbf{u}) - \mathbf{s}) d\mathbf{u} = \sum_i \frac{G_p(\mathbf{u}_i)}{|\det J(\mathbf{u}_i)|}. \quad (5)$$

This is in direct analogy to the flatland derivation. While the flatland case has singularities at the inflection points of the original one-dimensional heightfield, here we have singularities at  $\det J(\mathbf{u}) = 0$ , which is a set of curves in uv-space where the curvature of the original heightfield flips between elliptic and hyperbolic. These curves directly correspond to the “folds” we often see in  $\mathcal{P}$ -NDF visualizations. Again, piecewise constant normal maps (or affine regions of the heightfield) make  $\det J(\mathbf{u}) = 0$  over whole regions, causing delta functions in  $D(\mathbf{s})$ . In fact, we have tried to implement eq. (5) using analytic root finding and found it impractical due to the singularities.

Therefore, as in flatland, we introduce intrinsic roughness. This is accomplished by a 2-dimensional Gaussian kernel  $G_r(\mathbf{s})$ , which convolves the  $\mathcal{P}$ -NDF. The derivation is identical to flatland except



**Figure 7:** A patch of the normal map with  $9 \times 9$  texels. The  $z$ -component of the normal is visualized using iso-lines, to clearly depict curvature discontinuities. Using 32 triangles per texel shows better smoothness than 2, at no extra storage.

with bold letters:

$$\begin{aligned} D(\mathbf{s}) &= \int_{\mathcal{D}} G_r(\mathbf{s} - \mathbf{s}') \int_{\mathbb{R}^2} G_p(\mathbf{u}) \delta(n(\mathbf{u}) - \mathbf{s}') d\mathbf{u} d\mathbf{s}' \\ &= \int_{\mathbb{R}^2} G_c[\mathcal{P}, \mathbf{s}](\mathbf{u}, n(\mathbf{u})) d\mathbf{u}. \end{aligned} \quad (6)$$

where

$$G_c[\mathcal{P}, \mathbf{s}](\mathbf{x}, \mathbf{y}) = G_p(\mathbf{x}) G_r(\mathbf{y} - \mathbf{s}) \quad (7)$$

We can again visualize this intuitively as integration of the combined 4D reconstruction kernel  $G_c[\mathcal{P}, \mathbf{s}]$  along the graph of the normal function,  $(\mathbf{u}, n(\mathbf{u}))$ , which is a 2D surface in 4D space. This is hard to plot; however, the intuition that the graph can be triangulated and  $G_c$  reduces to 2D Gaussians over the triangles is correct. The hierarchical pruning idea also carries over from flatland.

In summary, we have observed that the  $\mathcal{P}$ -NDF  $D(\mathbf{s})$  is not trivially evaluated at a single point (direction)  $\mathbf{s}$ . However, under Gaussian pixel and roughness kernels, we have cast this evaluation as an integration problem, which can be solved by discretizing the normal map into small affine patches. (Note, though, that the specular highlights we handle can still be much smaller than the patches.) The next section discusses the details of solving this integration problem.

## 5 Analytic integration

To numerically evaluate equation (6), we choose to discretize the normal map  $n(u, v)$  into triangles, and linearly interpolate the normal across them. More precisely, we linearly interpolate the  $s$  and  $t$  values; the third coordinate is implied.

The simplest solution is to split each normal map texel into two triangles. This is sometimes sufficient, but we found that this discretization can produce triangular artifacts in the  $\mathcal{P}$ -NDF, if the resolution of the normal map is too low compared to the features it depicts. If this is an issue, we can up-sample the normal map, or subdivide texels into  $4 \times 4$  sub-texels using bicubic Catmull-Rom interpolation. Any other subdivision could be used, but  $4 \times 4$  naturally matches the control polygon of the bicubic patch. Figure 7 shows the difference between the two options.

**Integrating a 2D Gaussian over a triangle  $\Delta$ .** Our goal is to compute integrals of the form

$$I = \int_{\Delta} G_c(\mathbf{u}, n(\mathbf{u})) d\mathbf{u} = \int_{\Delta} G(\mathbf{u}) d\mathbf{u}. \quad (8)$$

Since we linearly interpolate the normals,  $n$  is an affine function on  $\Delta$ , which allows us to collapse the 4-dimensional combined Gaussian  $G_c$  into some other 2D Gaussian  $G$ .

This problem has been studied, and an R package implements one possible solution [PolyCub 2004]. There exist numerical algorithms for evaluating the cumulative distribution function  $\Phi(x, y, \rho)$  of a bivariate Gaussian with  $\sigma_x = \sigma_y = 1$  and covariance  $\rho$  [Genz 2004], which can be adapted to evaluate the desired integral. The PolyCub package also takes a similar approach. We have implemented this method and it works correctly, but appears slower than our method. A related problem for spherical Gaussians has been studied by Xu et al. [2014].

Below we describe the implementation that we found to perform well in our case.  $\Delta$  is a triangle from our triangulation; due to its construction, we only have right triangles, with two sides aligned to the axes. If  $\Delta$  is the triangle given by  $(u_0, v_0), (u_1, v_0)$  and  $(u_0, v_1)$ , we obtain an integral

$$I = \int_{u_0}^{u_1} \left( \int_{v_0}^{f(u)} G(u, v) dv \right) du, \quad (9)$$

where  $f(u)$  achieves a triangular integration domain:

$$f(u) = \frac{(u_1 - u)v_1 + (u - u_0)v_0}{u_1 - u_0}. \quad (10)$$

So far, we have just explicitly stated the problem. Eliminating  $v$  by carrying out the inner integration, and substituting  $x$  for the argument of the resulting erf function, this leads to integrals of the form

$$\text{experf}(a, b, x_0, x_1) = \int_{x_0}^{x_1} \exp(-a(x - b)^2) \text{erf}(x) dx \quad (11)$$

for some constants  $a$  and  $b$ , and shifted bounds  $x_0$  and  $x_1$ . In fact, the same form will result if we center the triangle instead of the Gaussian, or if we transform the problem so the Gaussian is unit, or with any other similar approach. This integral does not have an elementary solution, but we can approximate it as follows.

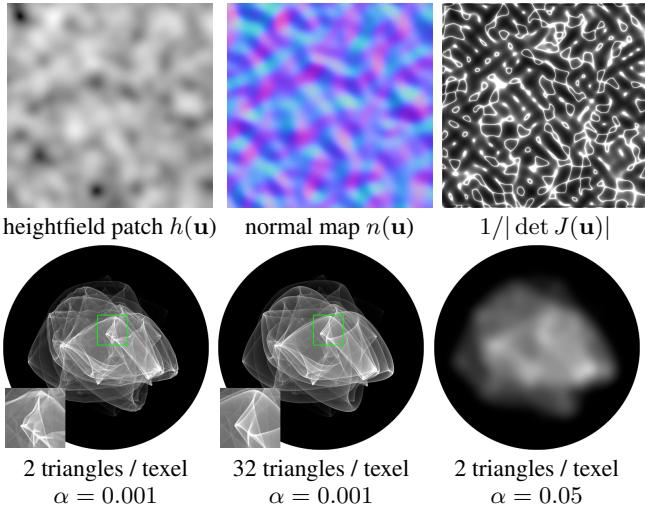
We choose to approximate the function  $\text{erf}(x)$  on the interval  $[-3, 3]$  by a piece-wise quadratic function on six subintervals, and as  $-1$  and  $1$  for  $|x| \geq 3$ . The problem thus separates into integrals of the form

$$\begin{aligned} \text{expquad}(a, b, c_0, c_1, c_2, x_0, x_1) &= \\ \int_{x_0}^{x_1} \exp(-a(x - b)^2) (c_0 + c_1 x + c_2 x^2) dx, \end{aligned} \quad (12)$$

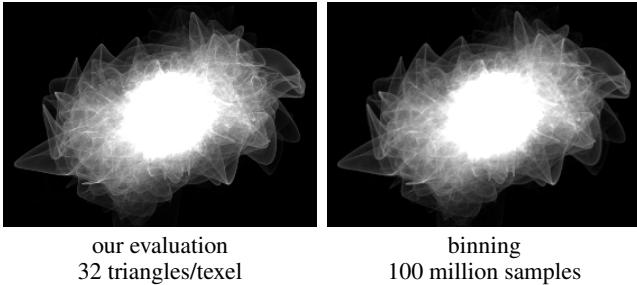
which can be solved analytically using a computer algebra system. The result is long but not fundamentally difficult.

Figure 8 illustrates the result of our integration algorithm on a particular normal map patch.

**Comparison against reference.** The correctness of the derivation can be easily checked against the binning method. That is, we use 100 million samples of  $G_p$ , look-up the normal map, perturb by  $G_r$ , and store the samples in bins. Figure 9 shows the result. The time-sequence comparison in Figure 1 is also computed using this method. Note the excellent match between the two images, computed using completely different methods. A minor difference comes from the fact that the binning inherently computes bin integrals instead of bin center values like our evaluation. The supplementary data contains several different NDFs compared against the reference, in floating point format. Note that we only provide



**Figure 8:** **Top row:** A heightfield  $h(\mathbf{u})$  with a Gaussian power spectrum, its normal map  $n(\mathbf{u})$  and the  $1/|\det J(\mathbf{u})|$  term that specifies the highlight brightness on a perfectly specular surface (with singularities at points where the original heightfield flips curvature). **Bottom row:** the  $\mathcal{P}$ -NDF corresponding to the footprint, computed using our approach. Left to right, with roughness 0.001 and two triangles per texel (showing some artifacts), with 32 triangles per texel, and with roughness 0.05 and 2 triangles per texel (no artifacts).



**Figure 9:** Comparison of the  $\mathcal{P}$ -NDF evaluated by our approach to the reference  $\mathcal{P}$ -NDF computed by binning, demonstrating the correctness of our derivations, for a single pixel of the cutlery model. A minor difference comes from the “anti-aliasing” of the binning method, which naturally computes bin integrals instead of bin center values like our evaluation.

single-pixel rather than full-frame reference comparisons, since the latter would be extremely slow to compute using the 100 million samples (see Figure 3), and would arguably provide less insight than NDF comparisons.

## 6 Implementation

**Hierarchical pruning of texels.** To increase performance, we limit the Gaussians to be non-zero only within  $5\sigma$  (a reasonable approximation). Therefore, many texels can be pruned, because either  $G_p$  or  $G_r$  are zero over the whole texel. We can trivially reject texels that fall outside of  $G_p$ . For  $G_r$  we utilize a min-max hierarchy over the normal map. More precisely, for each texel, we precompute the minimum and maximum value of  $s(u, v)$  and  $t(u, v)$ , and build a quad-tree hierarchy over these bounds. For a given query of  $D(s)$ , we traverse the hierarchy, pruning whole groups of texels where  $G_r$

is guaranteed to be beyond  $5\sigma$ . The recursive traversal is similar to many other bounding volume approaches.

**Importance sampling.** Sampling from a  $\mathcal{P}$ -NDF is easy by definition, using the same technique as was used to create the binning reference: simply take the normal of a random surface point seen through the pixel, and perturb by the intrinsic roughness kernel.

**Adding other light paths.** In our implementation, we separate the glint component of the image (i.e. direct illumination on normal-mapped specular surfaces from point lights) from all other light paths, which are computed using path tracing; any other standard algorithm could be used as well. On the first bounce from the camera, we use the full normal map for importance sampling. On further bounces we use a global  $\mathcal{P}$ -NDF approximation for both sampling and evaluation, since an accurate  $\mathcal{P}$ -NDF no longer makes a difference here. We could also use a normal map mip-mapping method in that case. A simple extension would be to smoothly transition to a normal map mip-mapping method in the distance, once glinting becomes insignificant.

Alternatively, our algorithm can be treated as a new “black-box” BRDF with an additional pixel footprint specification, while keeping all other parts of a renderer unmodified. However, we prefer to get separate timings, and we wanted to make sure the glint component is completely deterministic, to avoid any confusion about how much noise comes from the true glints vs. the algorithm. For this reason, we also do not use area lights, depth of field, or motion blur in our results, though they would be easy to add.

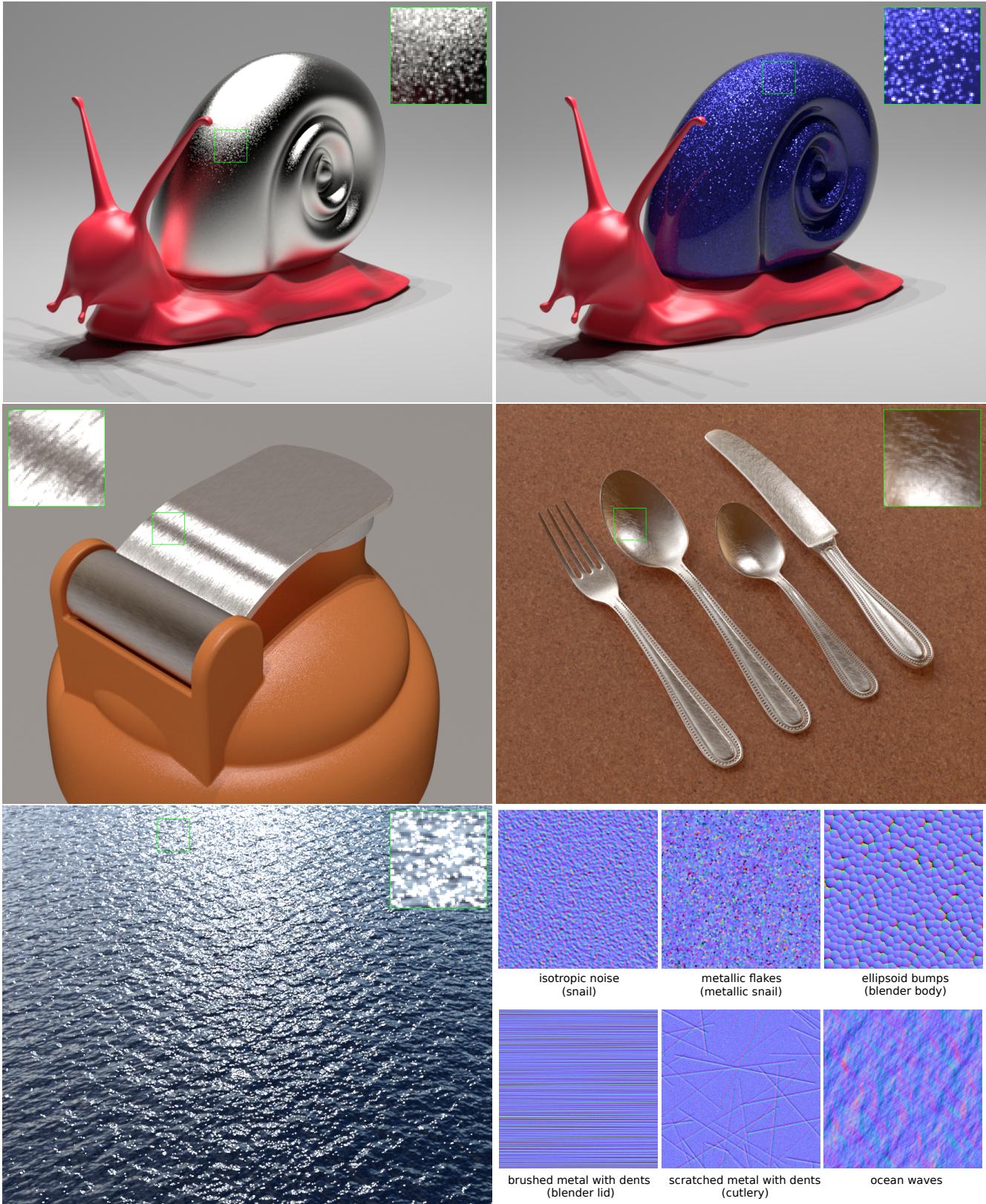
## 7 Results

Our implementation uses the Mitsuba framework [Jakob 2010], and runs on a 6-core Intel i7-4770K desktop at 3.5 GHz, hyperthreaded to 12 threads. Below we describe the scenes shown in Figure 10. Please see their temporal versions in the attached video. Note how the strong glinting is correct, given the normal map and the lighting; our method is entirely deterministic and does not produce any Monte Carlo noise. Our timings (Table 2) refer to one frame ( $1280 \times 720$ ). Note how the overhead of our algorithm is smaller than the standard rendering with other light paths. Also note that our performance depends on the number of pixels with glinty materials, and is independent of scene complexity.

**Snail.** This scene illustrates, on the snail’s shell, a smooth heightfield created by inverse FFT from an isotropic Gaussian spectrum with randomized phase, converted to a normal map. The features of the normal map are smaller than a pixel, and yet the result is far from smooth, producing a fairly dramatic glint effect.

**Metallic paint snail.** Metallic paint, often used on cars, is specifically designed to show glints. Composed of several layers, the most important are the top clear-coat (which provides the smooth specular highlight) and the colored absorptive layer with embedded aluminum flakes [Rump et al. 2008]. We model the flakes using a normal map that is constructed by clustering the pixels into Voronoi cells, whose centers are chosen using Poisson disk sampling, and assigning a fixed normal to each cell, drawn from the Beckmann distribution. No normal interpolation is necessary (or desirable) in this case: each texel has a constant normal. No subdivision beyond 2 triangles is required either. We also added a diffuse lobe to approximate multiple internal reflections between the flakes and the clear-coat. The snail is about 10 cm long, making the flakes more visible than on a car.

**Blender.** This scene shows an energy drink blender with a bumpy plastic body and a brushed metal lid. Brushed metal is notoriously difficult to render under sharp lighting; typical compromises in-



**Figure 10:** Still frames from our five scenes: snail (showing a simple isotropic noise normal map), metallic paint snail (modeling metallic flakes embedded in paint), blender (showing brushed metal with dents and plastic with ellipsoid bumps), cutlery (scratched metal with dents) and ocean (temporally varying waves caused by wind). We used simple sRGB in these images, but any tone-mapping could be applied. The full animations are shown in the supplementary video. Normal map contrast was enhanced for visualization purposes.

