Interactive rendering of acquired materials on dynamic geometry using frequency analysis

Mahdi M. Bagher, Cyril Soler, Kartic Subr, Laurent Belcour, and Nicolas Holzschuch

Abstract—Shading acquired materials with high frequency illumination is computationally expensive. Estimating the shading integral requires multiple samples of the incident illumination. The number of samples required may vary across the image, and the image itself may have high and low frequency variations, depending on a combination of several factors. Adaptively distributing computational budget across the pixels for shading is a challenging problem.

In this paper we depict complex materials such as acquired reflectances, interactively, without any pre-computation based on geometry. In each frame, we first estimate the frequencies in the local light field arriving at each pixel, as well as the variance of the shading integrand. Our frequency analysis accounts for combinations of a variety of factors: the reflectance of the object projecting to the pixel, the nature of the illumination, the local geometry and the camera position relative to the geometry and lighting. We then exploit this frequency information (bandwidth and variance) to adaptively sample for reconstruction and integration. For example, fewer pixels per area are shaded for pixels projecting onto diffuse objects, and fewer samples are used for integrating illumination incident on specular objects.

Index Terms—Computer Graphics, Rendering, Illumination Simulation, Measured Reflectance, Fourier Analysis

1 Introduction

Real materials can exhibit subtle and rich effects, such as colors changing depending on the viewing direction. While acquired real-world reflectance data are publicly available, their realistic depiction under environmental lighting conditions is slow. Photo-realistic shading involves costly numerical integration per pixel over multiple incident directions.

For some applications, such as shape design, interactive feedback of material appearance is desirable. The need to allow dynamically changing geometry poses the first challenge to interactive shading. In addition, it is important that the shading is realistic and consistent with its final appearance after post-design off-line rendering.

A gamut of approaches partially address this problem. At one end, fast algorithms focus on editable geometry with simple material models. Other algorithms strive to depict a variety of effects such as global illumination. The latter approach typically requires the pre-computation of radiance transfer (including visibility) and prevents geometry editing [**?**]. The combination of editable geometry and realistically portraying complex materials such as acquired BRDFs is still an open research problem and is the focus of this paper. We achieve this combination at the cost of global illumination and visibility.

Simulating material appearance under all frequency illumination requires the estimation of an integral of the incident illumination at each pixel modulated by the material's reflectance function. The integrands are typically sampled, and the sampling rate depends on the material: diffuse materials require many samples over incident directions, but exhibit low variation between neighboring pixels; specular materials require fewer samples over incident directions but cause large variation across nearby pixels.

We leverage theories in frequency domain light transport [**?**] to systematically exploit the relation between sampling in image-space (reconstruction) and sampling for shading (integration). For reconstruction, we propose a new multi-resolution algorithm. For integration, we predict the required *number of samples*. Our prediction may be used in conjunction with any sampling strategy for numerical integration.

In this paper, we introduce the concept of computing and storing the maximum local frequencies of the radiance field. We propose a practical representation of —*local bandwidth*—, as well as an approximation of the —*variance*— of the shading integrand, along with a fast algorithm to compute it. We use this information to adapt sampling rates for reconstruction and integration during rendering. Our rendering algorithm consists of two major steps. First, for each pixel, we estimate the screen-space bandwidth and variance. This information is stored, hierarchically, in a buffer having the same size as the picture generated. Next, we use this information to sample the image. We shade fewer pixels in smoothly varying areas, and adapt the number of samples according to the predicted variance. We render the final image using the scattered shaded pixels and up-sampling.

Our contributions are the following:

1) *Rapid bandwidth and variance computation*: we

- *Université de Grenoble and CNRS, Laboratoire Jean Kuntzmann, BP 53, 38041 Grenoble Cedex 9, France*
- *Maverick, INRIA Grenoble Rhône-Alpes, Montbonnot, 38334 Saint Ismier Cedex, France*
- *University College London*

quickly (about 8ms) predict local variation in the image due to reflected illumination, and expected variance for each pixel.
2) *Multi-resolution shading*: our multi-resolution deferred-shading algorithm uses the local frequency information for efficient sampling. We adaptively sample for reconstruction (shading only some pixels) and for integration (number of light samples for each shaded pixel)
3) *Adaptive multi-sample anti-aliasing*: we only compute sub-pixel shading for those pixels where the predicted image-space frequency is greater than 1 pixel^{-1}.
4) *Reflectance bandwidth-estimation*: we estimate local bandwidth of arbitrary reflectance functions using wavelets.
5) *Adaptive sampling for pre-convolved shading*: Reflectance usually filters out high frequencies in the illumination. We take advantage of this fact by taking less samples for integration using pre-convolved illumination.
6) *Local light sources*: We show applications of our technique with local light sources such as an area-light source.

The first 4 contributions were also published in [**?**], while the last 2 are new extensions to our previous work.

1.1 Related Work

Deferred shading: unlike common rendering methods on the GPU, *Deferred shading* postpones shading until occlusions are resolved in image space (see, e.g. [**?**]). It is more efficient for computationally expensive shaders, but incompatible with multi-sampling anti-aliasing methods [**?**]. Our algorithm allows adaptive sampling with complex materials and incoming light, and sub-linear multi-sample anti-aliasing.

GPU rendering of complex materials: Heidrich and Seidel [**?**], [**?**] interactively rendered simple BRDFs with environment maps by pre-filtering the environment map. Interactively rendering arbitrary BRDFs has also been done using separable approximations [**?**], homeomorphic factorization [**?**], spherical harmonics compactness properties [**?**] and spectral properties [**?**]. [**?**] provides real time rendering of acquired data using the sparsity of a wavelet representation and low rank of the reflection operator. Wang *et al.* [**?**] proposed a real-time rendering technique based on a spherical Gaussian approximation of the BRDF. Our main contribution is accurate shading of acquired materials while only shading a subset of the pixels.

Analysis of light transport: Durand *et al.* [**?**] study the properties of the Fourier spectrum of the local light field around a central ray. They derive transformations that propagate these spectra. Subsequent work has provided interesting applications of this theory, including simulation of motion blur [**?**] and depth of field [**?**]. Ramamoorthi extended the Fourier analysis to gradients [**?**]. These works provide key insights and understanding of the variation in light transport. They do not provide practical mechanisms to propagate the frequency content to image space in real time. Propagating sampled spectra [**?**] is costly. Egan et al. [**?**] derive formulae for transformations to the 3D lightfield (space and time) assuming diffuse objects. Our approach builds upon these works, but we propose a rapid method to estimate maximum variations along space and angle, rather than the entire spectra.

Multi-resolution screen-space algorithms: These techniques render by heuristically shading pixels at varying levels of coarseness, then up-sampling. Multi-resolution splatting for indirect illumination [**?**] and hierarchical image space radiosity [**?**] use a fast virtual point light source approach for indirect illumination. They hierarchically combine rasterized images using bilateral up-sampling [**?**], [**?**]. Similar techniques include computing interactive lighting from dynamic area light sources [**?**], and indirect illumination in glossy scenes [**?**], [**?**]. Light gathering methods can be improved using GPU-friendly interleaved sampling [**?**]. Ritschel *et al.* [**?**] achieve global illumination on GPU. They do not fully take advantage of the bandwidth of the reflectance or illumination.

Pre-computed transport: Pre-computed approaches compress or represent the transport operator in an alternative basis [**?**], [**?**]. Although they can depict rich materials [**?**] their primary goal is to pre-compute effects such as soft-shadows and global illumination as a function of the illumination. They require that the geometry remains static. In comparison to these algorithms, we are restricted to first bounce radiance without global illumination or visibility for shadows, but allow interactive editing of the geometry.

None of the approaches described above achieves realistic material depiction on dynamically editable geometry.

1.2 Overview

The radiance arriving at each pixel p after one-bounce direct reflection at a point x (ignoring visibility) is

$$L_p = \int_{\Omega_x} L_i(\omega)\ \rho(x, \omega, \omega_{x\to p})\ \omega.\mathbf{n}(x)\ \mathrm{d}\omega. \qquad (1)$$

Here $\omega_{x\to p}$ denotes the direction from x to the eye through pixel p, $\mathbf{n}(x)$ is the normal at x, L_i is radiance from illumination, Ω_x is the set of incident directions on the hemisphere above the local tangent plane, and ρ is the reflectance function. This integral is typically computed using Monte Carlo estimators as an average of N_p illumination samples:

$$L_p \approx \frac{G}{N_p} \sum_{i=1}^{N_p} \frac{L_i(\omega_i)}{g(\omega_i)}\ \rho(x, \omega_i, \omega_o)\ \omega_i.\mathbf{n}(x) \qquad (2)$$

where the $\omega_i \in \mathcal{S}^2$ are random incidence directions distributed according to the importance function g which integral over Ω_x is G.

We accelerate rendering by first, *avoiding shading all pixels*: we compute the integral for an adaptively sparse set of pixels depending on local variations, and upsample from neighboring pixels for the others (Section 3.2). Second, for each pixel p where we estimate the integral, we *adaptively choose* N_p according to the predicted variance of the shading integrand (Section 2.4). In Section 3, we present a multi-resolution shading algorithm that implements this two-fold strategy.

2 Real-time frequency analysis

We only propagate maximum local frequencies (bandwidth) about light paths — See Figure 1.

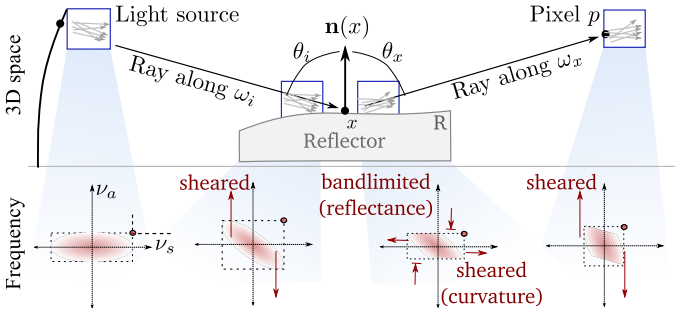


Fig. 1. Flatland illustration of local bandwidth propagation.

2D bandwidth: We analyze the local lightfield using the parametrization of Durand et al. [**?**]. Their parametrization is in 4D and we define the bandwidth of the local light field as a 2D vector with the maximum non-zero Fourier frequencies in space and angle. For robustness we use a quantile (the 95^{th} percentile) of the power spectra rather than the absolute maximum. For non-band-limited signals, we store an arbitrarily large value until the final calculation in image-space, where we clamp to the maximum representable frequency, which depends on the extent of anti-aliasing chosen. We denote the bandwidth using $\nu \equiv [\nu_s\ \nu_a]^T$ so that the rectangle with opposite corners $(-\nu_s, -\nu_a)$ and (ν_s, ν_a) contains the 2D spatio-angular spectrum of the local light field around a central ray (Figure 2).

From [**?**], we derive simple linear transformations undergone by ν for each step of the transport process (see Figure 3). We describe how to derive sampling rates using the bandwidth information. Finally, we explain how to estimate the variance of the shading integrand for adaptive sampling.

2.1 Computing one-bounce 2D bandwidth

At the light: the bandwidth of the local light field leaving light sources depends on the geometry and emission of the light sources. For distant illumination, ν_s is zero and

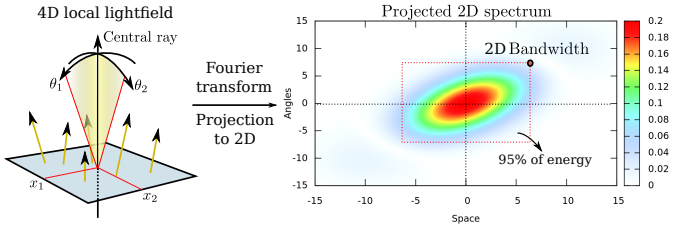


Fig. 2. *Left:* 4D local light field parametrization adopted by Durand et al. [**?**]. *Right:* 2D parametrization introduced by Soler et al. [**?**]. We define local bandwidth $\nu \equiv [\nu_s\ \nu_a]^T$ (black dot) so that 95% of the spectral energy lies in the dotted rectangle.

	4D ray space	*Fourier domain*
Transport (free space)	spatial shear	angular shear
Occlusion	product	convolution (spatial)
Curvature	angular shear	spatial shear
BRDF	convolution (angular)	product
Texture	product	convolution (spatial)

Fig. 3. Review of spectral operations from [**?**].

ν_a is directly computed from the environment map. For local light sources at finite distance, such as an area-light source, ν_a is zero and ν_s is computed directly from the light source.

Transport through free space: since transport through free space results in an angular shear of the local light field's spectrum [**?**], the transported bandwidths can be written as $T_d\nu$ for transport by a distance of d (see Figure 1), where T_d is defined in Figure 4.

Reflection: in the frequency analysis framework, reflection is realized in four steps [**?**]:

1) Re-parametrization of the incident light field into the frame of the reflecting surface. This is a spatial scale by $\cos\theta_i$ of the spectrum, followed by a spatial curvature shear of length c in the frequency domain (c is the Gaussian curvature of the surface expressed in m^{-1});
2) The product with the incident cosine, which is an angular convolution in Fourier space with a Bessel function;
3) The angular convolution of the light field with the BRDF is a band-limiting product in the frequency domain, while the spatial product by the texture is a convolution by the spectrum of the texture in the Fourier domain.
4) Re-parametrization along the outgoing direction. This is a mirror reflection in the spatial domain, followed by a spatial curvature shear of length $-c$ and a spatial scale of $1/\cos\theta_x$.

We translate these into matrix operations applicable to the bandwidth vector ν of the incident local light field (see Figure 4). The re-parametrization (first and last steps) are simply scaling (P_i and P_x). The curvature causes a shear of ν by matrices C_c and C_{-c}. The mirror re-parametrization is a multiplication by matrix S. The reflectance function band-limits angular frequencies

based on its own angular bandwidth ρ while the convolution with local texture augments the spatial bandwidth by the bandwidth t of the texture. We denote this using the operator $\mathcal{B}_{t,\rho}$. We neglect the product by the incident cosine, which only adds a small constant to the angular frequency.

$$T_d = \begin{bmatrix} 1 & 0 \\ -d & 1 \end{bmatrix} \qquad P_x = \begin{bmatrix} \frac{1}{\cos\theta_x} & 0 \\ 0 & 1 \end{bmatrix}$$

$$P_i = \begin{bmatrix} \cos\theta_i & 0 \\ 0 & 1 \end{bmatrix} \qquad S = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$C_c = \begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix} \qquad \mathcal{B}_{t,\rho}\ \nu \equiv \begin{bmatrix} \nu_s + t \\ \min(\rho, \nu_a) \end{bmatrix}$$

Fig. 4. Matrix operators on 2D bandwidth.

We quickly pre-compute angular and spatial bandwidths of the reflectance distribution (and texture). This computation is applicable to any type of reflectance function (analytical BRDFs, acquired BRDFs or artistic shaders).

The overall transformation undergone by incident bandwidths during reflection can thus be represented by a reflection operator $\mathcal{R}$ over the bandwidth vector:

$$\mathcal{R} = P_x\ C_{-c}\ S\ \mathcal{B}_{t,\rho}\ C_c\ P_i \tag{3}$$

The bandwidth around a light path arriving at pixel p after one-bounce of a single ray from the light is

$$\nu^i = T_{d'}\ \mathcal{R}\ T_d\ \nu_l^i \tag{4}$$

In this expression, d is the distance from the light source to the bouncing point on the surface, d' is the distance from the surface to the image plane and ν_l^i is the bandwidth while originating at the light source along direction ω_i.

2.2 Image-space bandwidth and sampling rate

The bandwidth at pixel p depends on the choice of ω_i sampled at x. That is the 2D bandwidth ν at pixel p is a combination of the individual bandwidths ν^i along the sampled directions.

We compute the 2D bandwidth at each pixel ν as a weighted average of the sampled incident illumination $L_i(\omega_i)$ at x, reflectance and the 2D bandwidths of the associated one-bounce paths ν^i:

$$\nu = \begin{bmatrix} \nu_s \\ \nu_a \end{bmatrix} = \frac{1}{\sum_{j=1}^{n_b} \lambda_i} \sum_{i=1}^{n_b} \lambda_i \nu^i \tag{5}$$

with

$$\lambda_i = L_i(\omega_i)\ \rho(\omega_i, \omega_{x\to p})\ \omega_i.\mathbf{n_x}$$

Although the bandwidth at each pixel is estimated using multiple samples, a small value of n_b is sufficient (see Figure 13). Using a max operation in place of the sum in (5) would be the only conservative choice, but it does not capture view-dependent effects. For instance bandwidth after reflection from a specular sphere would be equally high regardless of viewing or light direction (see Figure 5). (5) is a heuristic approximation to the actual variance of the shading integrand that is bounded to the range of bandwidths from contributing light paths while giving more credit to light paths with larger energy (If all light paths contribute the same bandwidth, the approximation is exact). This accounts for the material reflectance, relative orientation of illumination and view, and local geometry.

The required sampling rates at the image plane are twice the local image-space bandwidth (Nyquist criterion) b_p (in pixel^{-1}):

$$b_p \quad = \quad \nu_a\ \max\left[\frac{f_x}{W}, \frac{f_y}{H}\right] \tag{6}$$

where f_x and f_y are the horizontal and vertical fields of view, and the rendered image is $W \times H$ pixels.

2.3 Illumination, textures, and BRDF bandwidth

We present a general technique to estimate local bandwidth on 2D signals. We used it to feed our algorithm with bandwidth from textures, distant illumination, BRDF and local light sources. All of them can be easily converted to 2D images — or sets of 2D images.

A naïve way to estimate 2D bandwidth would be to compute a windowed Fourier transform of the input signal (2D image) and measure the bandwidth of the spectrum. Windowed FFT is a costly computation, and implies a compromise between locality and the range of possibly measured frequencies.

For reasons of efficiency as well as improved localization in space and frequency, we use wavelets. The set of wavelet basis functions that contribute to the image value at a given pixel provides an estimate of the local spectrum in the image. However, summing power spectra of these different wavelets neglects their phase, and results in an overestimate the image bandwidth. In our experiments, the benefits from efficiency and simplicity outweighed potential error from overestimation.

More formally, for a given point x in the image signal s, we have:

$$s(x) = \sum_i \beta_i\ \phi_i(x) \quad + \quad \sum_i \sum_j \lambda_{i,j}\ \frac{1}{2^i}\ \psi\left(\frac{x - 2^i j}{2^i}\right).$$

(where ϕ is the scale function and ψ is the mother wavelet). Wavelets of the same scale have identical bandwidths, so we compute the maximum wavelet coefficient $\lambda_i^{max} = \max_j |\lambda_{ij}|$ per frequency band, and estimate the wavelet level as:

$$I_w = \text{argmin}_i \sum_{k=i}^{n} \lambda_k^{max} < \epsilon \max_k \lambda_k^{max} \tag{7}$$

The result is independent of ϵ as long as it is small enough. We used $\epsilon = 0.01$ in all our experiments. Figure 6 illustrates this process.

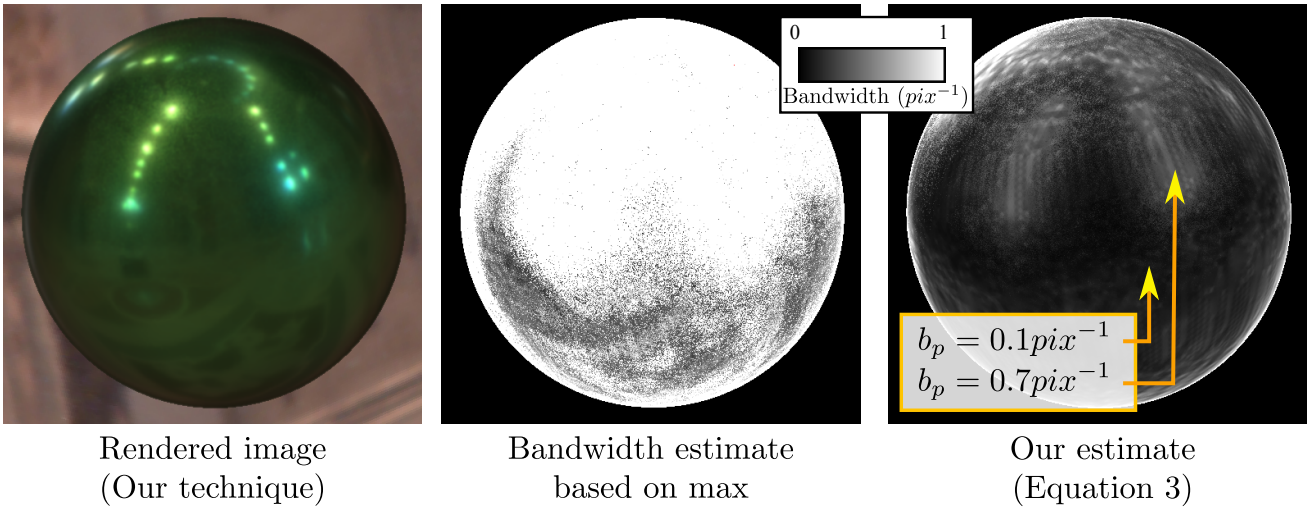


Fig. 5. Combining bandwidth estimates from sampled incident directions. *Middle:* Applying a *max* overestimates sampling rates (1 pixel^{-1} almost everywhere on the sphere). *Right:* Our approach. (5) predicts view-dependent sampling rates. *Left:* Final result.

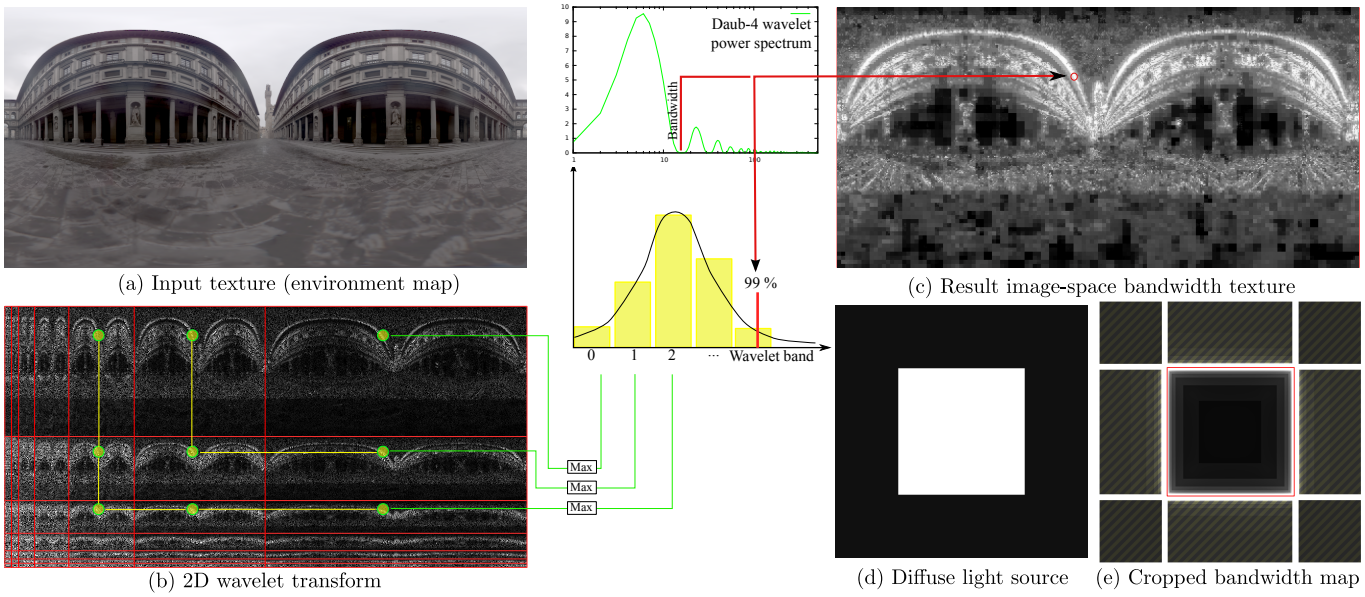


Fig. 6. Instant 2D bandwidth estimation using wavelets. The source image ((a) Environment map, or (d) diffuse local light source) is first converted into 2D wavelets, alternating horizontal and vertical pyramid steps (*left*). For each pixel, we build a histogram of wavelet energy per frequency band (*middle*). We extract the 95^{th} percentile of this energy as the maximum order of significant wavelets bands per pixel, and then converted into bandwidth using the (precomputed) bandwidth of the mother wavelet. For textures local light sources, we compute bandwidth similarly. We first embed the source in a larger space, compute the bandwidth and only consider the computed bandwidths within the region occupied by the source. The process of bandwidth computation is the same for spatial (local light sources) as well as angular (distant illumination) domains.

Distant illumination: The local 2D bandwidth of distant illumination along ω is purely angular:

$$\nu(\omega) = [0, \nu_a(\omega)]^T$$

The distant illumination is first converted into an angular map. Then, for each pixel, the image wavelet hierarchy level I_w from (7) is converted back into angular bandwidth using

$$\nu_a(\omega) = \frac{2\pi}{2^{I_w}\lambda_{max}}$$

where λ_{max} is the maximum eigenvalue of the Jacobian for the 2D mapping of spherical coordinates onto the image plane. This approach allows to compute instant angular bandwidth in real time on GPU for environment maps. See Figure 6 for an example.

Local light sources (spatial): The bandwidth of an area-light source is purely spatial, and is computed using the above method. The area-light source can have any planar shape and be textured.

To correctly account for the boundary effects of a finite-size area-light source, we pad the area-light source image with zeros, so that its size is doubled. We compute the bandwidth of the zero-padded image and crop it back to the original size afterwards. Figure 6 shows an example of a solid white square-shaped area-light source. We treat textured area-light sources the exact same way.

Texture (spatial): We extract the spatial bandwidth using the same approach, this time accounting for the Jacobian of the mapping onto the surface so that the bandwidth is correctly expressed in inverse meters.

Reflectance (angular):. In this paper, we only demonstrate separable reflectance: an spatially homogeneous angular reflectance distribution along with a texture. However, all the derivations for bandwidth hold for spatially-varying BRDFs.

For each incident direction in the local tangent frame, we compute the angular bandwidth map of the outgoing BRDF lobe. We use the same technique as for distant illumination and apply 2D wavelet transforms on the slices. We store the result for each lobe of the BRDF in a large texture. For the general case of 4D reflectance data we use 16×16 input directions and a 16×16 image for each reflectance lobe, packed into a 1024^2 texture. Since the maximum expressible bandwidth depends on resolution, we compute the bandwidth for higher-resolution angular slices and reduce it to 16×16.

2.4 Adaptive sampling for shading

In this section, we estimate the variance of the Monte-Carlo estimator L_p of (1) in order to determine a suitable number N_p of integration directions for each pixel p. Because there is no analytical formula giving the variance $V(L_p)$ of L_p using importance sampling, we conservatively overestimate it assuming uniform sampling:

$$V(L_p) \leq \frac{4\pi^2}{N_p}\sigma_p^2$$

The variance σ_p^2 of the shading integrand about a single illumination direction ω_i, at a point x that projects to pixel p, is

$$\begin{aligned}\sigma_p^2 &= \mathrm{E}(\lambda_i^2) - \mathrm{E}(\lambda_i)^2 \\ &\leq \mathrm{E}(\chi_i^2)\end{aligned}$$

where $\chi_i = L_i(\omega_i)\rho(\omega_i, \omega_{x\to p})$, and $\mathrm{E}(X)$ denotes the expected value of X. Finally,

$$\mathrm{E}(\chi_i^2) = \mathrm{E}(\hat{L}_i^2 \otimes \hat{\rho}^2)$$

where $\hat{f}$ denotes the Fourier transform of f and $\otimes$ denotes convolution. This equality is a consequence of Parseval's theorem (see, *e.g.* **[?]**). The convolution is in the angular domain. We have:

$$\mathrm{E}(\hat{L}_i^2 \otimes \hat{\rho}^2) \leq (\nu_a^i + \nu_{\rho a}^i)\ \chi_i^2 \tag{8}$$

where ν_a^i is the angular component of ν^i, and $\nu_{\rho a}^i$ is the local angular bandwidth of the reflectance function. Therefore, we have:

$$V(L_p) \leq \frac{4\pi^2}{N_p}\sum_{i=1}^{n_b}(\nu_a^i + \nu_{\rho a}^i)\ \chi_i^2 \tag{9}$$

To keep the variance of the shading globally constant, we need to keep N_p proportional to the sum of the bandwidths, weighted by the illumination and reflectance values, along sampled directions ω_i. The sum is a conservative approximation of the variance of the integrand. $n_b = 16$ provides acceptable quality (see Figure 13).

The summations over incident directions ((5) and (9)) indicate that we implicitly account for the relative alignment (phase) of the illumination and reflectance. Previous approaches that neglect phase cannot predict variation due to view-dependent effects.

2.5 Adaptive sampling for pre-convolved shading

As an approximation, we propose a faster integration scheme, that would need far less directional samples.

Starting again from (1), we can write the illumination to be the sum of a low frequency component L_i^l and a high frequency component L_i^h. The high frequency component is going to cancel out due to the low-pass filter characteristics of the reflectance:

$$\begin{aligned}L_p &= \int_{\Omega_x}(L_i^l(\omega) + L_i^h(\omega))\rho(x,\omega,\omega_{x\to p})\mathbf{n}.\omega d\omega \\ &\approx \int_{\Omega_x} L_i^l(\omega)\rho(x,\omega,\omega_{x\to p})\mathbf{n}.\omega d\omega\end{aligned}$$

Doing this, we can rewrite (9), replacing the bandwidth of the illumination by the bandwidth of the filtered

illumination, which in turn is equal to the bandwidth of the BRDF:

$$V(L_p) \leq \frac{4\pi^2}{N_p} \sum_{i=1}^{n_b} 2\nu_{\rho a}^i \, \chi_i^2. \quad (10)$$

To compute the shading integral, we pick samples in the pre-filtered illumination L^l (e.g a pre-filtered environment map), at the level given by the maximum frequency of the BRDF. The number of samples needed to compute the shading integral is therefore greatly reduced.

The calculation we performed is based on the same principle used by pre-convolved environment map shading [**?**]. The main difference is that we do not project the BRDF nor the illumination onto a basis function. We only filter out parts of the illumination that are out of the support of the spectrum of the BRDF. We still perform a Monte-Carlo integration to compute L_p, but using less samples. Section 4.3 gives examples of using this technique.

2.6 Implementation roadmap

The practical implementation is as follows: for each pixel, we compute the image bandwidth using (5) and (6), and the number of samples for the shading integrand using (9). Both are summations over $n_b = 16$ incoming directions ω_i. For each ω_i, the local 2D bandwidth ν^i is given by (3) and (4), using the matrices listed in Figure 4. These matrices require the curvature c, normal n, the incident and outgoing angles (θ_i and θ_o), the pre-computed spatial and angular bandwidth of the material (resp. t and ρ) for the current pixel and direction ω_i, and the pre-computed bandwidth of the light source ν_l^i in direction ω_i.

3 Hierarchical shading algorithm

Our rendering algorithm consists of four steps (see Figure 7): (1) load the pre-computed bandwidth for BRDF(s) and compute illumination bandwidth on the fly. (2) render G-buffers such as depth, position, normals and material IDs; (3) fill the bandwidth buffer with image-space bandwidth, the number of integration samples to use per pixel and the screen-space bandwidth; (4) run a single-pass multi-resolution shading step, interleaved with up-sampling.

Rendering G-Buffers is a classical geometry pass where we store normals, position, depths and material IDs into a set of screen-space buffers. Note that G-Buffers do not need to be hierarchically built (mip-mapped) in our method; we build a multi-resolution pyramid only for the bandwidth buffer.

3.1 Bandwidth buffer initialization

The bandwidth buffer contains two important values: the local image-space bandwidth and the number of samples to be used for shading each pixel. These are computed using the G-Buffers ((6) and (9)). Although these estimations involve numerical integration, they are several orders of magnitude faster than the actual shading, since a coarse sampling is sufficient (Figure 13). Rather than storing b_p (see (6)) in the bandwidth buffer, we store

$$\min(\ \lfloor \log_2 \frac{1}{b_p} \rfloor,\ \min(\ \log_2(W),\ \log_2(H)\ )\ ) \quad (11)$$

which is the pyramid resolution at which pixel p needs to be shaded, accounting for the local variation at p. The *floor* operation ensures that the Nyquist sampling rate is respected. Storing b_p directly in the bandwidth buffer leads to identical results; our optimization simplifies tests for deciding the pyramid resolution while shading each pixel.

The bandwidth is mip-mapped using a min filter, so that at a given level in the hierarchy, the value for a pixel conservatively tells us whether sub-pixels should be computed at this level. We do the same for the variance estimate using a max filter.

In addition, we estimate screen-space curvature and mip-map it using a max filter.

3.2 Shading and up-sampling

We render the image hierarchically, progressively from coarse to fine. At a given resolution (say $2^k \times 2^k$), we examine the bandwidth buffer and shade the pixels for which the bandwidth buffer pyramid contains the current coarseness resolution k. For pixels whose bandwidth buffer entries are less than the current resolution (i.e.$< k$), we bilaterally up-sample from neighbors at the preceding level of coarseness ($2^{k-1} \times 2^{k-1}$), only accounting for pixels that are already computed. The parents' values are averaged with coefficients

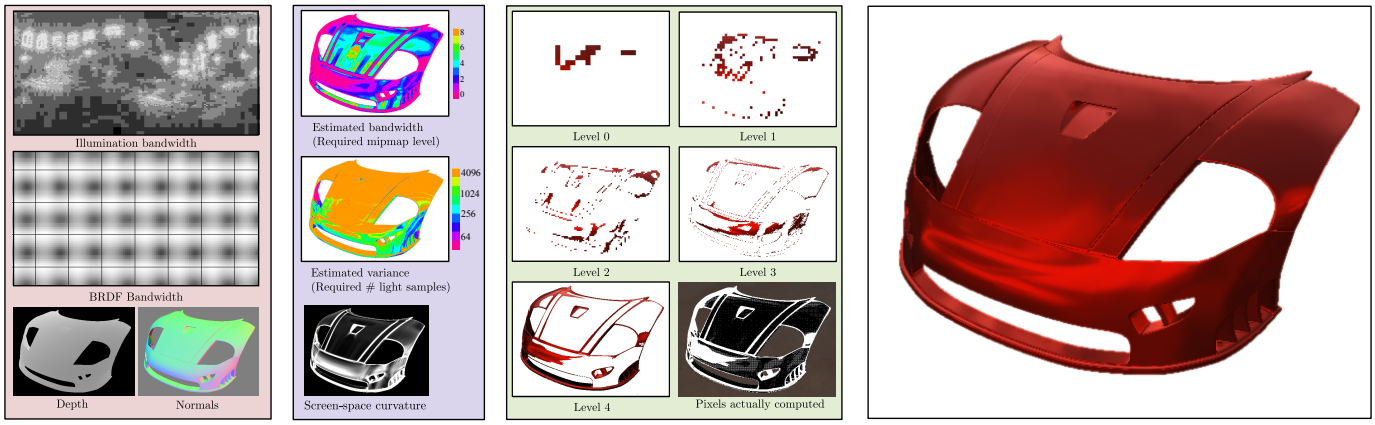
$$w_i = g_z(z - z_i) g_a(p - p_i) \alpha_i$$

where z (resp. z_i) are the depths of the shaded (resp. parent) pixels, and g_z is a Gaussian that cancels out pixels of irrelevant depth, and α_i are bilinear weights (Figure 8). The last term g_a is an anisotropic 2D Gaussian defined as

$$g_a(\mathbf{v}) = e^{-\mathbf{v}^T M \mathbf{v}} \quad \text{with} \quad M = R_\phi^T \begin{bmatrix} 1 & 0 \\ 0 & 1/\cos^2\theta_x \end{bmatrix} R_\phi$$

where ϕ is the angle of the screen-projected normal at the surface, and θ_x the angle between the normal and the view direction. This enables efficient anisotropic filtering aligned with the highest and lowest screen-space frequencies, since $\frac{b_p}{\cos\theta_x}$ and b_p estimate the minimum and maximum directional screen-space bandwidth around current pixel.

We continue this process over successive levels, until we reach the finest resolution where we shade all remaining pixels. Pseudo-code for the algorithm is presented in Figure 8.



(a) Input data (b) Bandwidth (c) Shaded pixels (d) Result image

Fig. 7. Our rendering pipeline: (a) at each frame, we first render G-Buffers. From these, we compute an additional *bandwidth buffer* (b) that stores image space bandwidth and shader integrand variance maps as well as screen-space curvature. The former is stored in a multi-resolution pyramid. During rendering of the final image from coarse to fine scale, depending on our bandwidth and variance predictions, pixels are either explicitly shaded (numerical integration) or up-sampled from parent pixels (c). Once we reach the finest level, the image is fully rendered (d).

3.3 Shading computation

For each shaded pixel we read the number of samples N_p from the bandwidth buffer. We estimate reflected radiance (2) , unless stated otherwise, by importance sampling the BRDF lobe for the current view direction. In our implementation, we read N_p samples randomly, from multiple pre-computed vectors of importance samples that are stored in a texture. Our algorithm is compatible with any importance sampling strategy. In practice, we importance sample the reflectance by numerical inversion of its cumulative distribution.

We always shade with depth and normal values at the finest level, since the G-Buffers are not mip-mapped. This is possible because the bandwidth buffer predicts whether, for any sub-pixel of the current level pixel, the computation will yield similar estimates despite potential variation in the depths, normals and illumination.

4 Results and Discussion

All timings reported in this paper were measured on an NVIDIA GeForce GTX 560 Ti graphics card with 1GB of memory.

Figure 9 shows an example of a scene with three identical objects but different materials and a marble textured ground floor. This is a clear example of how our fast frequency analysis helps us spend the samples where they are most needed.

for all points p at level L **do**
  if $b_w(p) < L$ **then**
    compute $w_0, ..., w_3$
    $c(p) \leftarrow \sum_k w_k c(p_k)$
  else if $b_w(p) == L$ **then**
    shade(p)
  end if
end for

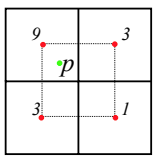
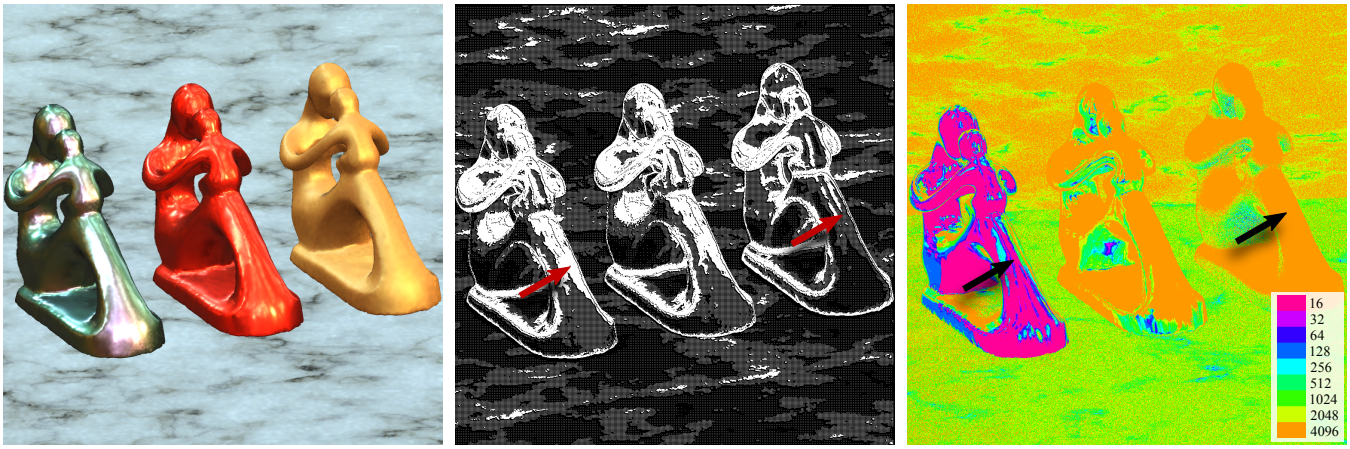


Fig. 8. Up-sampling interpolation scheme. *Left:* Pseudo-code for the computation of one level. *Right:* relative weights α_i for parent pixels of pixel p at the next level.

4.1 Behavior of our algorithm

Computation time: the computation time for our algorithm scales linearly with the total number of shading samples (Figure 10, left). The total number of shading samples required depend on the desired image quality, the material and the environment map. Figure 11 tabulates the computation times for our algorithm to obtain equal quality as ground truth, for several scenes. It details the cost of individual steps: the cost of bandwidth computation is independent of the scene and the material, and negligible compared to the overall cost (8 ms, or $< 0.33\%$). Shading estimation consumes most of the total time (up to 90 %).

Memory cost: The memory footprint of our algorithm on the GPU is approximately 432 MB at a resolution of 512×512 pixels: 17 MB for the G-buffers (4 RGBA



(a) Rendered image (b) Pixels actually computed (c) Light samples per pixel

Fig. 9. Three identical objects with different materials (`color-changing-paint3`, `red-metallic-paint` and `gold-paint`), and a marble textured ground floor with `white-marble` material. We shade more pixels per area for specular materials (red arrows) but we use a larger number of samples to compute shading on diffuse materials (black arrows).

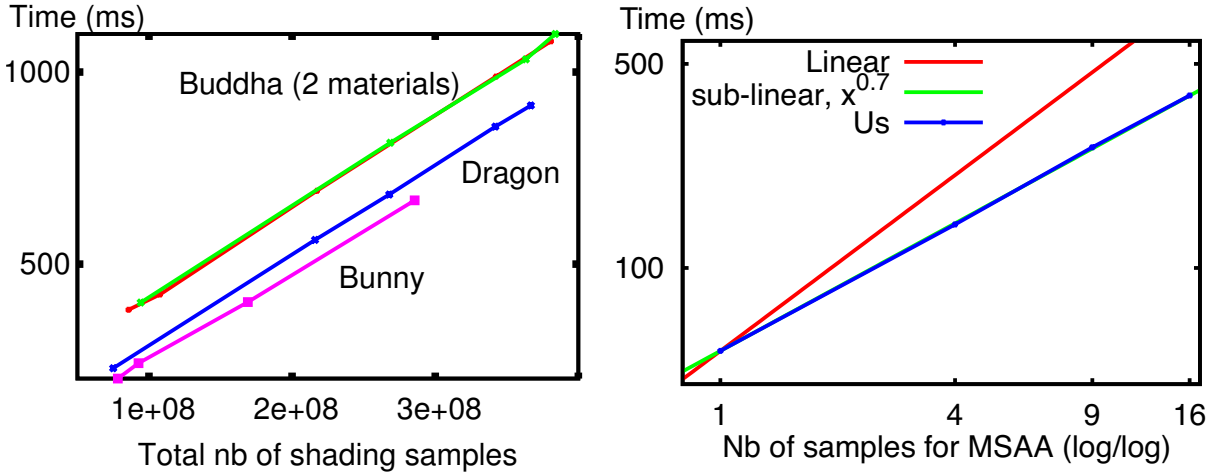


Fig. 10. *Left*: Rendering times are linear in the number of shading samples for various models. *Right*: Rendering time against number of samples for anti-aliasing (blue) in log scale. Our algorithm scales sub-linearly (with an exponent of 0.7) for anti-aliasing.

Model	Car body	Bunny	Buddha	Dragon
Reference	2056 ms	1628 ms	2634 ms	1739 ms
Ours: Total	229 ms	419 ms	390 ms	205 ms
Bandwidth calculation	7 ms	8 ms	7 ms	8 ms
Shading integration	216 ms	390 ms	200 ms	182 ms

Fig. 11. Computation times (in ms) at 512×512 screen resolution: Car body (Fig. 9), Bunny (Fig. 13), Buddha (Fig. 12) and Dragon (Fig. 19). Our algorithm provides a 4 to $10\times$ speedup compared to a forward-shaded reference of comparable quality, depending on the material and screen occupancy. Bandwidth computation is fast (< 10 ms) for all scenes.

buffers for position, normal, tangent, material ID and the depth buffer); 2 mip-mapped buffers (5.5 MB each) for the bandwidth and variance and for the shading computations with up-sampling; 2 RGB buffers of 6 MB for the environment map and its bandwidth; and 2 buffers of 196 MB for the raw BRDF data and its importance samples. Increasing the picture resolution only increases the cost of the G-buffers and mip-mapped buffers: 112 MB (instead of 22.5) for 1024×1024 and 448 MB at 2048×2048, which is currently the maximum for our algorithm.

Validation: Figure 12 compares our predictions with reference variance and image-space bandwidth. Our predictions are similar in spatial distribution, and of the same order of magnitude. Our variance estimate is conservative, as explained in Section 2.4. We computed the reference bandwidth[1] using a windowed Fourier transform over the image, with a window size of 32×32 pixels. We computed the reference variance using extensive sampling.

Influence of parameters: the main parameter for our algorithm is the number of samples we use for the bandwidth estimation, n_b (see (5)). Figure 13 shows the influence of varying this parameter. The results are indistinguishable even in the zoomed-in insets and the Peak Signal-Noise Ratio stays almost constant for all values of n_b. The rendering time has a small dependency on n_b. We used a small value, $n_b = 16$, for all results in this paper. This makes sense as n_b is only used to estimate the bandwidth and not for the actual illumination computations.

4.2 Comparison with related work

Brute-force rendering: Figure 14 compares our result with a forward shading reference computed using importance sampling and a fixed number of shading samples per pixel. For this scene, we achieve a $2.5\times$ speedup due to our adaptive sampling. The extent of our speed-up depends on the material, the environment map and the area occupied by the object on screen. Figure 11 tabulates rendering times for our algorithm and brute-force rendering for several scenes. We observe a speed-up of $4\times$ to $10\times$.

Spherical Gaussian approximation: Figure 15 compares the results of Wang *et al.* [**?**] with ours and ground truth computed using path-tracing. The authors were kind enough to provide us with their best images for the materials. Our algorithm accurately shades acquired materials. The fast algorithm [**?**] is visibly different from the ground truth.

1. Reference local frequencies cannot be computed exactly using the windowed Fourier transform because of the uncertainty principle. However, using our wavelet estimation technique here would not constitute a fair test either.

4.3 Rendering results

Adaptive sampling for pre-convolved shading: To use adaptive sampling for pre-convolved shading (ASFPCS), we pre-convolve the environment map with Phong lobes of various exponent and assemble them into an array of textures. At the time of rendering, we estimate the number of adaptive samples using (10) and look-up pre-filtered illumination for each sample according to the maximum frequency of the BRDF. figure 16 depicts examples of using adaptive sampling for pre-convolved shading (ASFPCS). We achieved a significant speed-up (10 to 20 times depending on the material and scene coverage).

Area-light source rendering: To render area-light sources, we propagate bandwidth through the scene from the area-light source towards the camera. Once we know the adaptive number of samples for shading by bandwidth propagation, we sample the area-light source directly (no BRDF importance sampling) and shade in the hierarchical buffer. Figure 17 depicts two scenes rendered with the area-light sources.

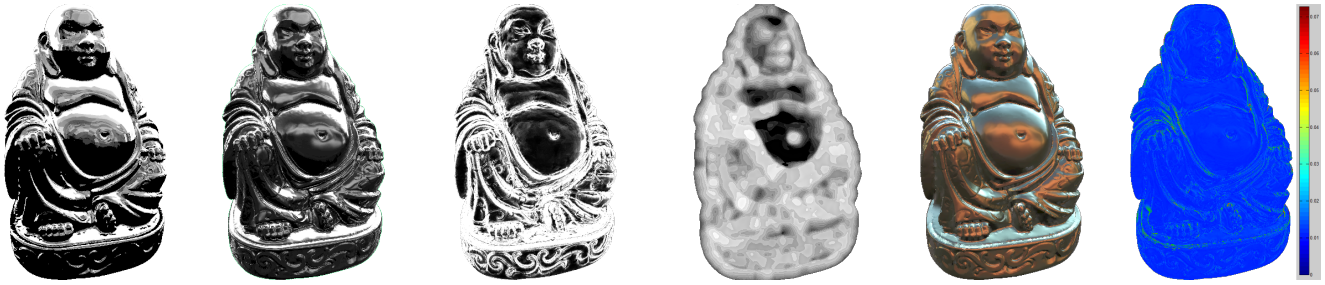
Adaptive multisample anti-aliasing: Our bandwidth prediction reduces the cost of multi-sample anti-aliasing by adaptive sampling. Standard deferred shading evaluates shaders at every sample: 16 samples per pixel costs 16 times more. Our algorithm scales sub-linearly in the number of samples (see Figure 10, right). We render the G-buffers at the higher resolution (4 or 16 times the number of pixels), but compute shading at the appropriate level in the pyramid, depending on the predicted bandwidth. Anti-aliasing only requires a few extra shader evaluations at the finest levels (blue pixels in Figure 19).

Dynamic geometry, normal and displacement mapping: We compute bandwidth and variance estimates using only the information from the G-buffers (normals, geometry and curvature). Our algorithm handles dynamic geometry, displacement mapping and normal maps seamlessly. Figure 18 shows an example of our bandwidth estimation algorithm on a dynamic geometry, running at 15 fps.

4.4 Discussion

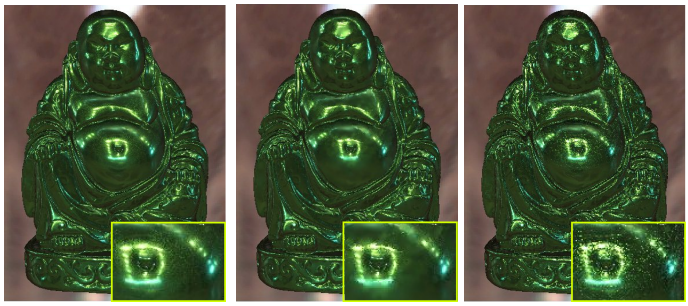
Best and worst case: The rendering cost for our algorithm depends on the total number of shading samples that we predict. Our best cases are when the predicted variance is low (a specular material) or when the spatial variation is low (a smooth diffuse surface). We do not save time when both angular variance and spatial frequencies are high (a bumpy surface with high frequency illumination).

Conservative bandwidth estimate: We conservatively



(a) Our variance estimate (b) Reference variance (c) Our bandwidth estimate (d) Reference FFT bandwidth (e) Our shading result (f) False color difference

Fig. 12. Validation of predicted variance and bandwidth. (a) our estimate for the variance of the shading integrand (9), (b) reference variance computed using brute-force sampling, (c) our estimate of the image bandwidth, (d) reference image bandwidth (windowed FFT of reference image), (e) our result, and (f) relative error of (e) with respect to a path-traced reference. Material: `color-changing-paint3`



(a) similar quality (2639 ms) (b) Our algorithm (1015 ms) (c) similar time (906 ms)

Fig. 14. Comparison of our algorithm (b) with a reference for equal quality (a) and equal time (c). For this scene, we achieve a $2.5\times$ speedup (without anti-aliasing). Forward-shaded references use BRDF-importance sampling and a fixed number of shading samples per pixel. Material: `green-metallic-paint`.

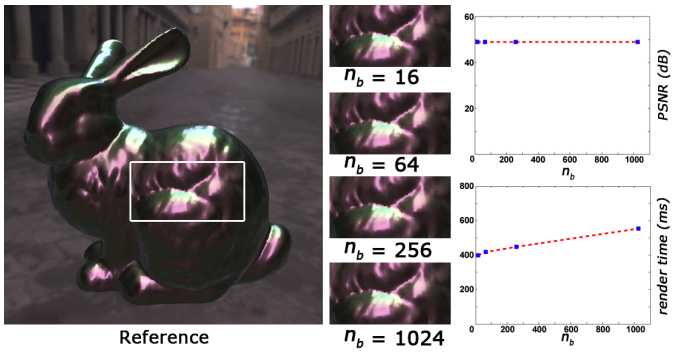


Fig. 13. Effect of the number of sampled directions n_b for bandwidth computation on time (ms) and quality (PSNR). The data points in the plots are at $n_b = 16, 64, 256, 1024$. The enlarged insets are virtually indistinguishable for the different n_b. The plots depict that fast bandwidth computation ($n_b = 16$) is sufficient. Resolution: 512×512. Material: `color-changing-paint3`.

predict bandwidth, choosing suboptimal (excessive) sampling over artifacts from insufficient sampling. The $\min$ operator of Figure 4 is larger than the real bandwidth of the product of the BRDF and illumination; we also over-estimate variance.

Bandwidth estimate and importance sampling: Since we base all our bandwidth predictions on uniform sampling and render the result using BRDF importance sampling, our predictions for variance are sometimes very conservative, and it happens that an acceptable result can still be achieved with less samples. It would be interesting to derive acceptable approximations of the variance for importance sampling, but we keep this as a future avenue.

Visibility and global illumination: we focus on accurate depiction of materials rather than scenes. We ignore effects such as visibility for shadows and global illumination.

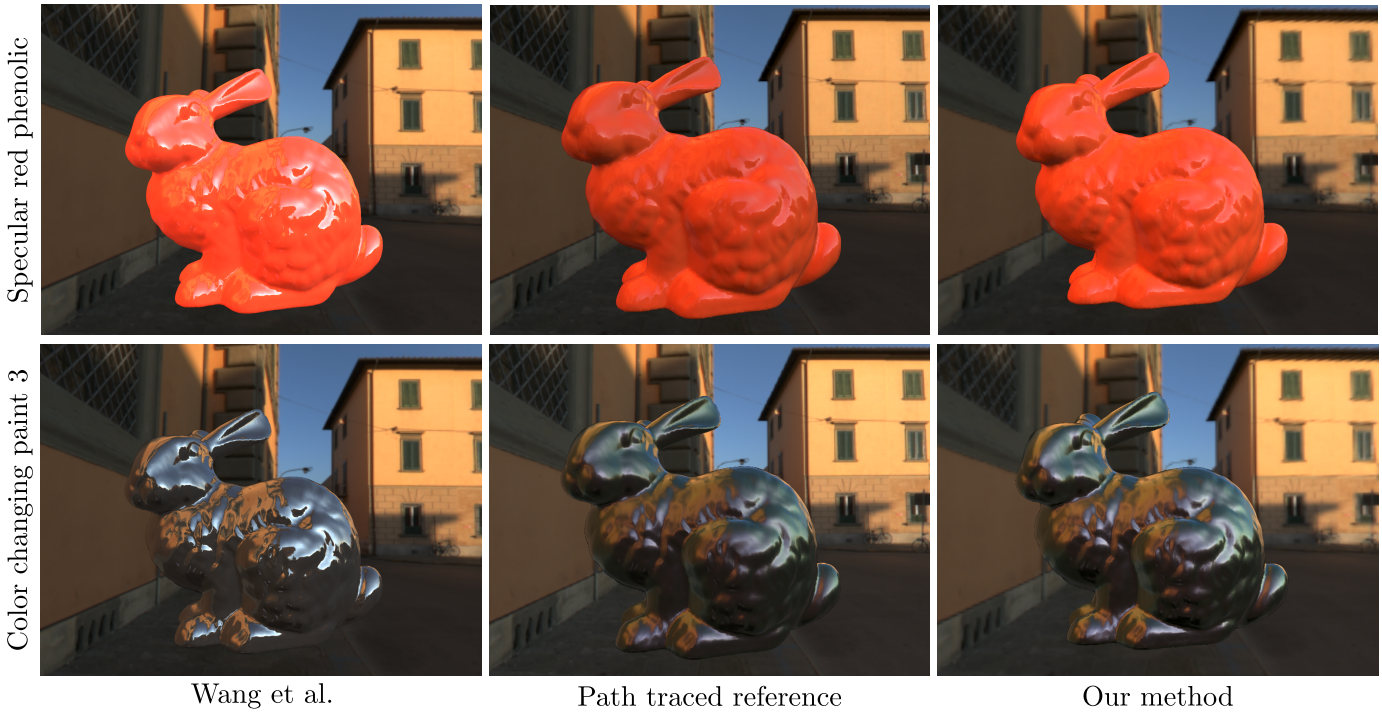


Fig. 15. Comparison with related work [**?**]. Our algorithm results in pictures that are identical to ground truth, while [**?**] result in clear differences.

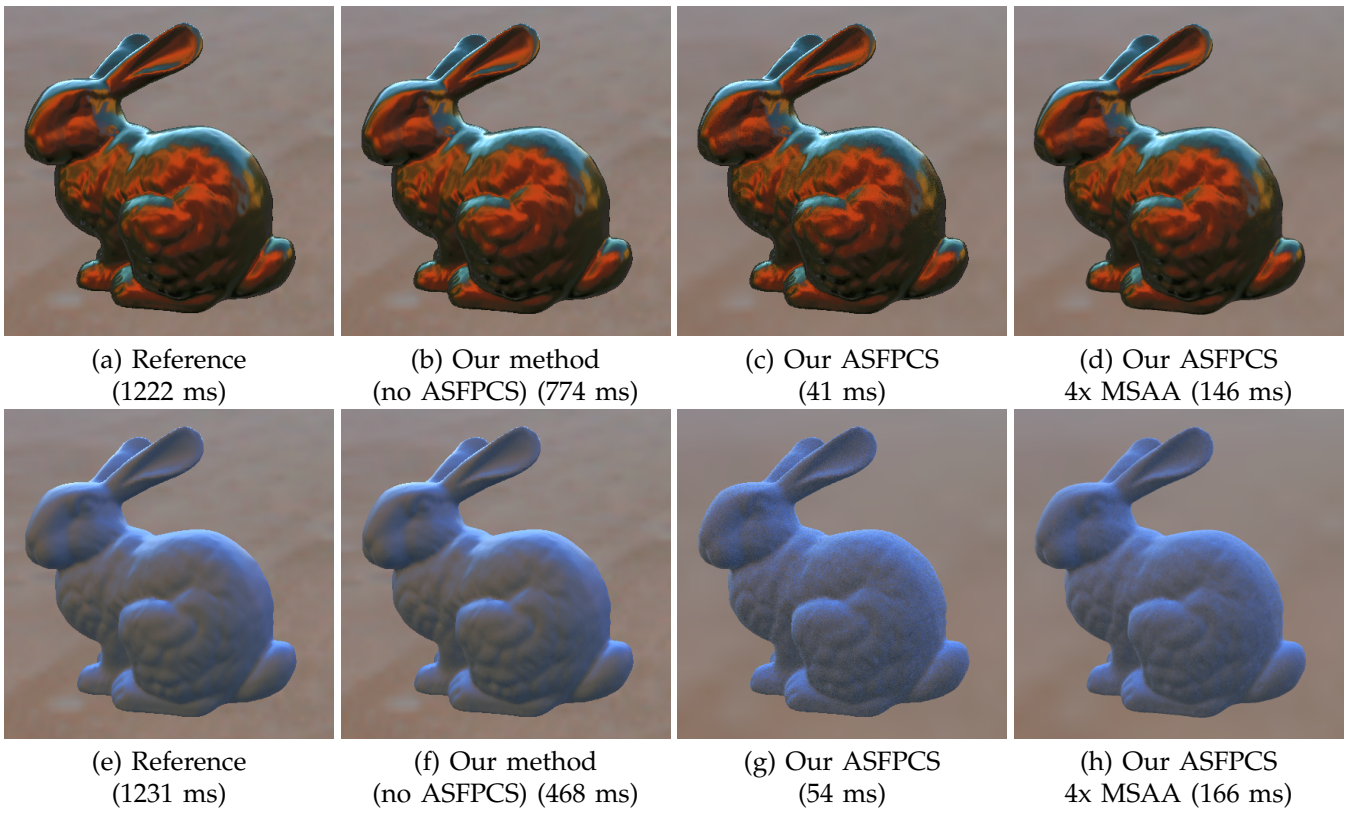


Fig. 16. Adaptive sampling for pre-convolved shading (ASFPCS) examples (c, g) for `color-changing-paint3` and `blue-rubber`, compared to the reference (a,e) and our adaptive sampling method without pre-convolved shading (b,f). Our ASFPCS technique with 4x multi-sample anti-aliasing is still faster than our method without pre-convolved shading. The speed-up is 10 to 20 times.

Spatially-varying BRDFs: we have only used homogeneous (non spatially-varying) materials, modulated by a texture. We pre-computed local bandwidths separately for reflectance ($4D$) and texture ($2D$). Extending our algorithm to fully spatially varying BRDFs ($6D$) is possible at the extra cost of 6D bandwidth pre-computation.

5 Conclusion

We have introduced an algorithm for interactively and accurately shading dynamic geometry with acquired materials. Our contribution is to: (1) only shade a small fraction of pixels where the local bandwidth is predicted to be large; and (2) adaptively sample shading integrals based on the predicted variances. We achieve these predictions by quickly computing local bandwidth information from standard G-buffers. We have introduced the bandwidth buffer, to store this information, which is exploited to sub-linearly scale multi-sample anti-aliasing with deferred shading.

Our work can be extended in many ways. We would like to apply this technique to indirect lighting and visibility effects as well. This constitutes a real challenge: although extending our theory to these situations seems feasible, the resulting calculations needed by visibility will increase the computation cost significantly. Accounting for anisotropy in the bandwidth estimate is another interesting avenue that will require significantly more calculations.

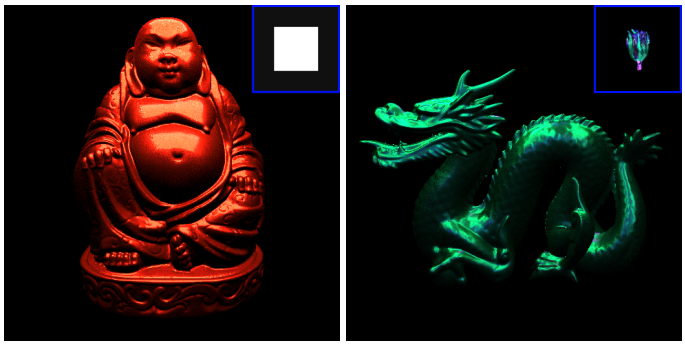


Fig. 17. *Left:* `Buddha` rendered with a solid white square area-light source and `red-metallic-paint` material (513 ms). The area-light source is positioned at the top of the object and is shown as inset. *Right:* `Dragon` rendered with a textured area-light source and `color-changing-paint3` material (376 ms). The area-light source is positioned at the top of the object and is shown as inset.

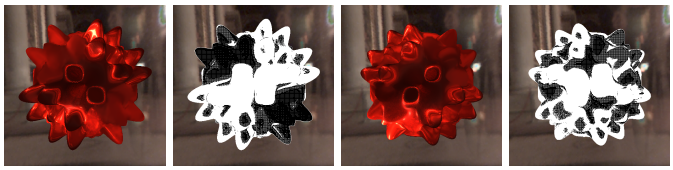


Fig. 18. Application of our algorithm to dynamic geometry, with `red-metallic-paint`. Since our algorithm depends only on data from the G-buffers, it can handle dynamic geometry, displacement mapping and normal maps. *top row:* rendered images, *bottom row:* pixels for which we performed shading computations. The screen-space bandwidth adapts to the curvature (Rendering time: 66 ms).

Acknowledgments

All pictures and the accompanying video were generated using the acquired materials from the MERL BRDF database [**?**]. The Bunny and Dragon model are provided by the Stanford Computer Graphics Laboratory. The top row of Figure 15 was kindly provided by the authors of [**?**].

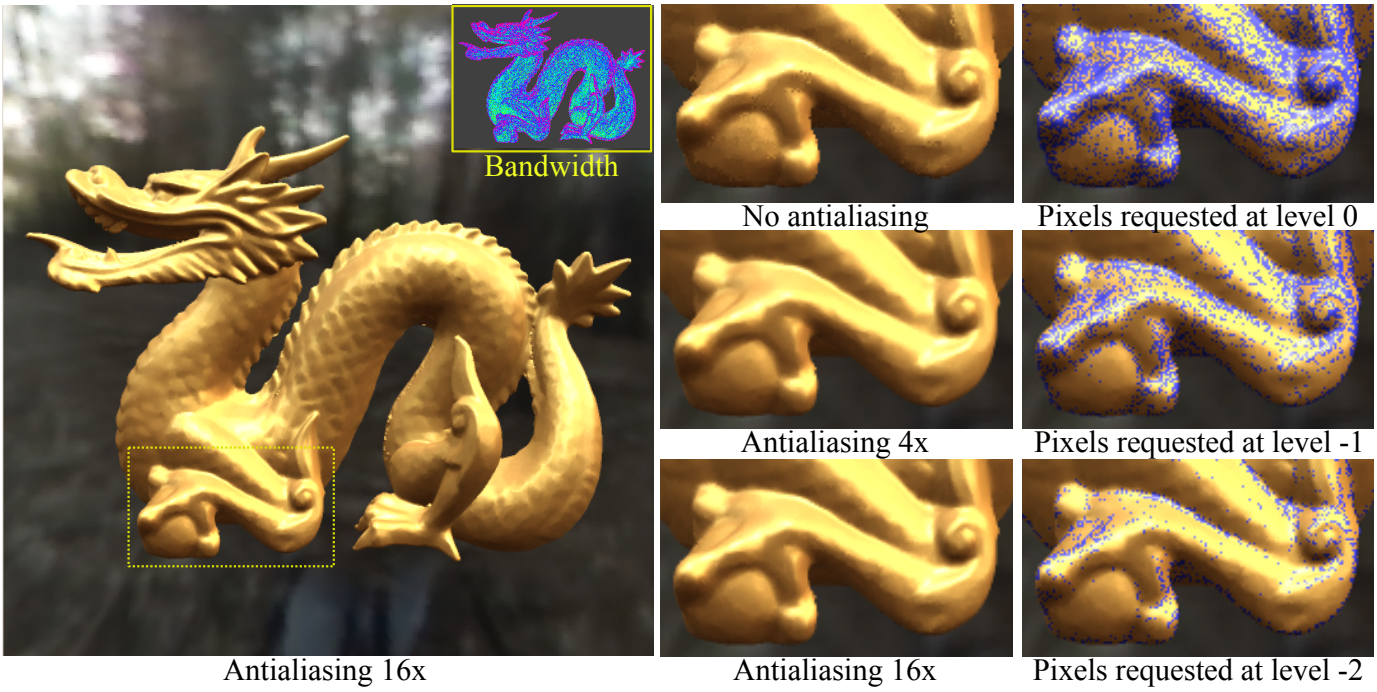


Fig. 19. Adaptive multi-sample anti-aliasing using bandwidth information. Aliasing artifacts are visible in the top row of enlarged insets (55 ms). Our multi-resolution algorithm handles anti-aliasing seamlessly by adding extra levels to the pyramid (one level for $4\times$ and two levels for $16\times$). We only compute shading at the finest level for the blue pixels in the rightmost column, where the predicted bandwidth is high. The cost of anti-aliasing is thus reduced: 141 ms for $4\times$, 390 ms for $16\times$. Material:`gold-paint`.

Mahdi M. Bagher is a PhD candidate at Université de Grenoble in the Maverick research team, INRIA, since 2009. He has been mainly working on materials appearance representation, acquisition and rendering under the supervision of Nicolas Holzschuch and Cyril Soler. He received his Masters degree in computer graphics, vision, and imaging from UCL, London, under suprvision of Jan Kautz.

Kartic Subr is currently a Newton Fellow and Honorary Research Associate at University College London. He was a post-doctoral researcher in the computer graphics group at INRIA-Grenoble from 2008-2010. He received his PhD (2008), under supervision by James Arvo, from the University of California, Irvine. During his PhD, he enjoyed a variety of short stints interning at venues including Rhythm and Hues Studios, NVIDIA Inc. and Columbia University. Kartic's research interests span most aspects of realistic picture generation and manipulation.

Cyril Soler is a researcher at INRIA since 2000. He has worked mainly on global illumination, including various contributions in Fourier analysis of light transport. He also has contributions in real-time rendering, geometry processing and texture analysis and synthesis.

Laurent Belcour is currently a PhD candidate at Université de Grenoble in the Maverick research team. He studies the Fourier equivalent of local light transport with applications in rendering under the supervision of Nicolas Holzschuch and Cyril Soler.

Nicolas Holzschuch is a Senior Researcher at INRIA Grenoble Rhône-Alpes, and the scientific leader of the MAVERICK research team. He got his PhD from Grenoble University in 1996, and joined the INRIA in 1997, first in Nancy, then in Grenoble. His research interests include photorealistic rendering and real-time rendering, with an emphasis on scalability, and multi-scale phenomena.