Interactive Rendering of Acquired Materials on Dynamic Geometry Using Frequency Analysis

Mahdi Mohammad 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 precomputation 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 unit 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. Photorealistic 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 postdesign offline 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 precomputation of radiance transfer (including visibility) and prevents geometry editing [1]. 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 variations across nearby pixels.

We leverage theories in frequency domain light transport [2] to systematically exploit the relation between sampling in image-space (reconstruction) and sampling for shading (integration). For reconstruction, we propose a new multiresolution 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 upsampling.

● *M.M. Bagher is with the University of Montréal, Dept. informatique et de recherche operationelle, Faculte des arts et des sciences, C.P. 6128, succ. Centre-Ville, Montréal, Québec H3C 3J7, Canada.*
  E-mail: m.m.bagher@gmail.com.
● *C. Soler and N. Holzschuch are with the INRIA Rhone-Alpes, 655 Avenue de l'Europe, 38334 Montbonnot ZIRST, France.*
  E-mail: {Cyril.Soler, Nicolas.Holzschuch}@inria.fr.
● *K. Subr is with University College London, Malet Place, London WC1E 6BT. E-mail: kartic@gmail.com.*
● *L. Belcour is with Inria Bordeaux - Sud-Ouest 200, avenue de la Vieille Tour 33405 Talence Cedex - France. E-mail: laurent.belcour@gmail.com.*

Our contributions are the following:

1. *Rapid bandwidth and variance computation*. We quickly (about 8 ms) predict local variation in the image due to reflected illumination, and expected variance for each pixel.
2. *Multiresolution shading*. Our multiresolution denferred-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 multisample antialiasing*. We only compute subpixel 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 prefiltered shading (ASFPCS)*. Reflectance usually filters out high frequencies in the illumination. We take advantage of this fact by taking less samples for integration using prefiltered illumination.
6. *Local light sources*. We show applications of our technique with local light sources such as an area-light source.

The first four contributions were also published in [3], while the last two 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., [4]). It is more efficient for computationally expensive shaders, but incompatible with multisampling antialiasing methods [5]. Our algorithm allows adaptive sampling with complex materials and incoming light, and sublinear multisample antialiasing.

GPU rendering of complex materials. Heidrich and Seidel [6], [7] interactively rendered simple BRDFs with environment maps by prefiltering the environment map. Interactively rendering arbitrary BRDFs has also been done using separable approximations [8], homeomorphic factorization [9], spherical harmonics compactness properties [10], and spectral properties [11]. Claustres et al. [12] provide real-time rendering of acquired data using the sparsity of a wavelet representation and low rank of the reflection operator. Wang et al. [13] 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. [2] 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 [14] and depth of field [15]. Ramamoorthi extended the Fourier analysis to gradients [16]. These works provide key insights and understanding of the variations in light transport. They do not provide practical mechanisms to propagate the frequency content to image space in real

time. Propagating sampled spectra [15] is costly. Egan et al. [14] derive formulas 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.

Multiresolution screen-space algorithms. These techniques render by heuristically shading pixels at varying levels of coarseness, then upsampling. Multiresolution splatting for indirect illumination [17] and hierarchical image space radiosity [18] use a fast virtual point light source approach for indirect illumination. They hierarchically combine rasterized images using bilateral upsampling [2], [19]. Similar techniques include computing interactive lighting from dynamic area light sources [20], and indirect illumination in glossy scenes [21], [22]. Light gathering methods can be improved using GPU-friendly interleaved sampling [23]. Ritschel et al. [24] achieve global illumination on GPU. They do not fully take advantage of the bandwidth of the reflectance or illumination.

Precomputed transport. Precomputed approaches compress or represent the transport operator in an alternative basis [1], [25]. Although they can depict rich materials [26], their primary goal is to precompute 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 multiresolution shading algorithm that implements this twofold strategy.
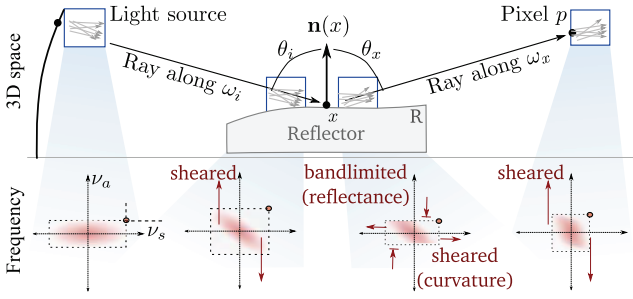
Fig. 1. Flatland illustration of local bandwidth propagation.

2 REAL-TIME FREQUENCY ANALYSIS

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

Two-dimensional bandwidth. We analyze the local lightfield using the parametrization of Durand et al. [2]. Their parametrization is in 4D and we define the bandwidth of the local light field as a 2D vector with the maximum nonzero Fourier frequencies in space and angle. For robustness, we use a quantile (the 95th percentile) of the power spectra rather than the absolute maximum. For nonband-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 antialiasing 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 (Fig. 2).

From [2], we derive simple linear transformations undergone by ν for each step of the transport process (see Fig. 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 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 [2], the transported bandwidths can be written as $T_d\nu$ for transport by a distance of d (see Fig. 1), where T_d is defined in Fig. 4.
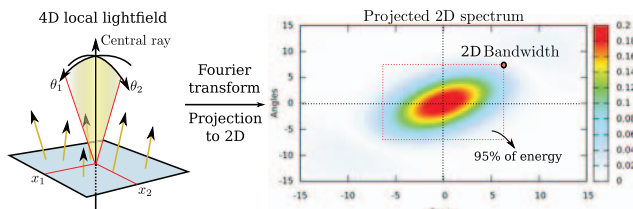
	4D ray space	Fourier domain
Transport (free space)	spatial shear	angular shear
Occlusion	product	convolution (spatial)
Curvature	angular shear	spatial shear
BRDF	convolution (angular)	product
Texture	product	convolution (spatial)

Fig. 3. Review of spectral operations from [2].

Reflection. In the frequency analysis framework, reflection is realized in four steps [2]:

1. Reparametrization 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. Reparametrization 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 Fig. 4). The reparametrization (first and last steps) is simply scaling (P_i and P_x). The curvature causes a shear of ν by matrices C_c and C_{-c}. The mirror reparametrization 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.

We quickly precompute 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:

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

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

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

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

Fig. 4. Matrix operators on 2D bandwidth.

Fig. 2. *Left:* 4D local light field parametrization adopted by Durand et al. [2]. *Right:* 2D parameterization introduced by Soler et al. [15]. We define local bandwidth $\nu \equiv [\nu_s \ \nu_a]^T$ (black dot) so that 95 percent of the spectral energy lies in the dotted rectangle.
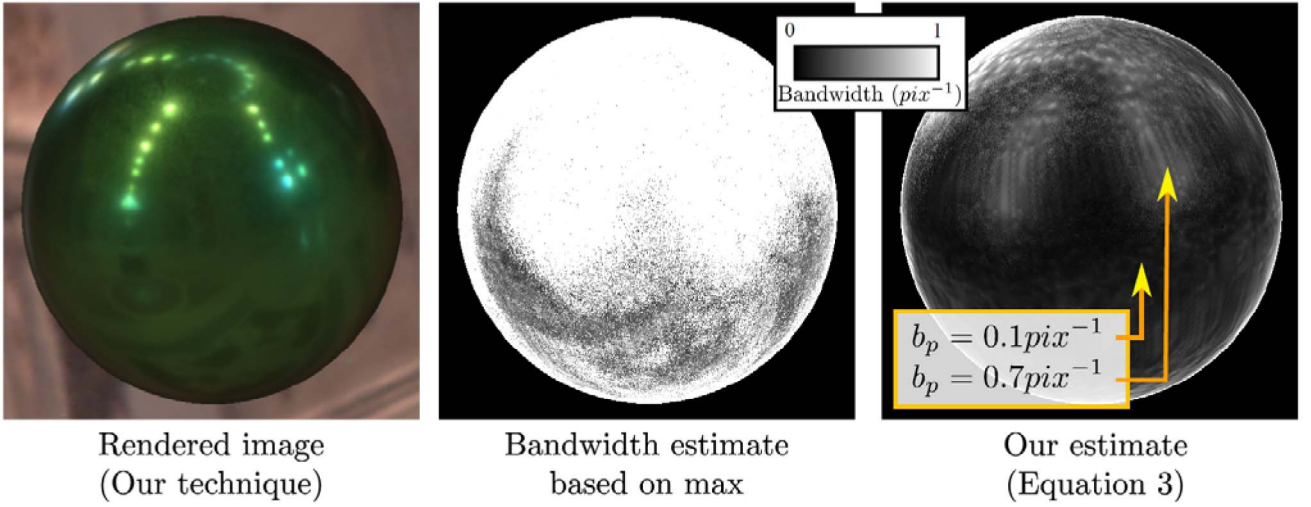
Fig. 5. Combining bandwidth estimates from sampled incident directions. *Middle:* Applying a *max* overestimates sampling rates (1 pixel^{-1} almost everywhere on the sphere). *Right:* Our approach. Equation (5) predicts view-dependent sampling rates. *Left:* Final result.

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

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

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^i_l 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 Fig. 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 Fig. 5). Equation (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 with the same energy, 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})

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

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 over-estimation of 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

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

(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

$$I_w = \underset{i}{\arg\min} \sum_{k=i}^{n} \lambda_k^{max} < \epsilon \max_k \lambda_k^{max}. \tag{7}$$

(a) Input texture (environment map)

(b) 2D wavelet transform

(c) Result image-space bandwidth texture

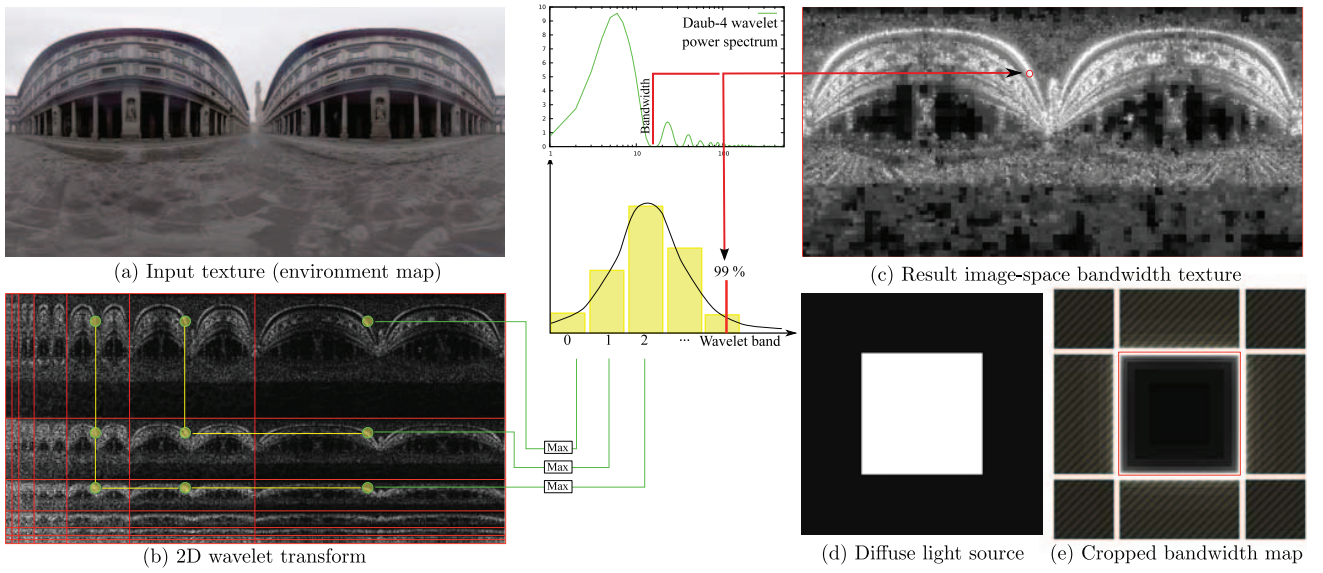(d) Diffuse light source   (e) Cropped bandwidth map

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 95th 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 textured 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.

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

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 Fig. 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 afterward. Fig. 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: a 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 $1,024^2$ 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

$$\sigma_p^2 = \mathrm{E}\left(\chi_i^2\right) - \mathrm{E}\left(\chi_i\right)^2$$
$$\leq \mathrm{E}\left(\chi_i^2\right),$$

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,

$$\mathrm{E}\left(\chi_i^2\right) = \mathrm{E}\left(\hat{L}_i^2 \otimes \hat{\rho}^2\right),$$
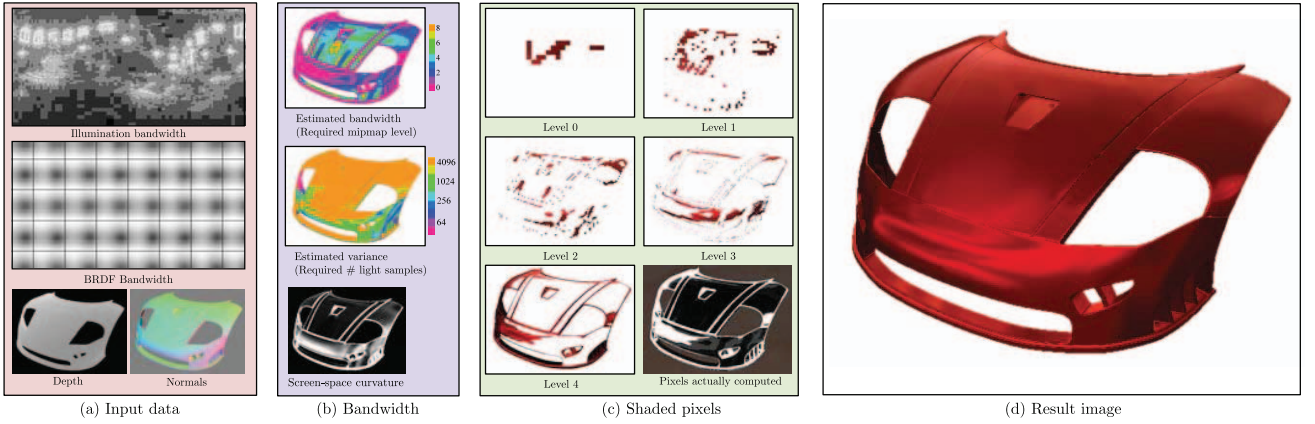
Fig. 7. Our rendering pipeline: (a) at each frame, we first render G-Buffers. From these, we compute an additional *bandwidth buffer* (b) that stores image space bandwidth and shader integrand variance maps as well as screen-space curvature. The former is stored in a multiresolution pyramid. During rendering of the final image from coarse to fine scale, depending on our bandwidth and variance predictions, pixels are either explicitly shaded (numerical integration) or upsampled from parent pixels (c). Once we reach the finest level, the image is fully rendered (d).

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., [27]). The convolution is in the angular domain. We have

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

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

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

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 Fig. 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 Prefiltered 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

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

$$\approx \int_{\Omega_x} L_i^l(\omega)\rho(x,\omega,\omega_{x\to p})\mathbf{n}.\omega d\omega.$$

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

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

To compute the shading integral, we pick samples in the prefiltered illumination L^l (e.g., a prefiltered 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 prefiltered environment map shading [7]. 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 Fig. 4. These matrices require the curvature c, normal n, the incident and outgoing angles (θ_i and θ_o), the precomputed spatial and angular bandwidth of the material (resp. t and ρ) for the current pixel and direction ω_i, and the precomputed bandwidth of the light source ν_l^i in direction ω_i.

3 HIERARCHICAL SHADING ALGORITHM

Our rendering algorithm consists of four steps (see Fig. 7):

1. load the precomputed bandwidth for BRDF(s) and compute illumination bandwidth on the fly.
2. render G-buffers such as depths, positions, 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 multiresolution shading step, interleaved with upsampling.

Rendering G-Buffers is a classical geometry pass where we store normals, positions, 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 multiresolution 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 (Fig. 13). Rather than storing b_p (see (6)) in the bandwidth buffer, we store

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

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 subpixels 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 Upsampling

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 upsample 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 (Fig. 8). The last term g_a is an anisotropic 2D Gaussian defined as

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

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.

```
for all points p at level L do
    if b_w(p) < L then
        compute w_0, ..., w_3
        c(p) ← ∑_k w_k c(p_k)
    else if b_w(p) == L then
        shade(p)
    end if
end for
```
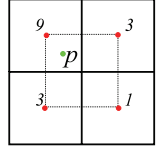

Fig. 8. Upsampling interpolation scheme. *Left:* Pseudocode for the computation of one level. *Right:* relative weights α_i for parent pixels of pixel p at the next level.

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

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 precomputed 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 subpixel 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 1 GB of memory. Fig. 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.

4.1 Behavior of Our Algorithm

Computation time. The computation time for our algorithm scales linearly with the total number of shading samples (Fig. 10, left). The total number of shading samples required depends on the desired image quality, the material, and the environment map. Fig. 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 percent). Shading estimation consumes most of the total time (up to 90 percent).

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 (four RGBA buffers for position, normal, tangent, material ID, and the

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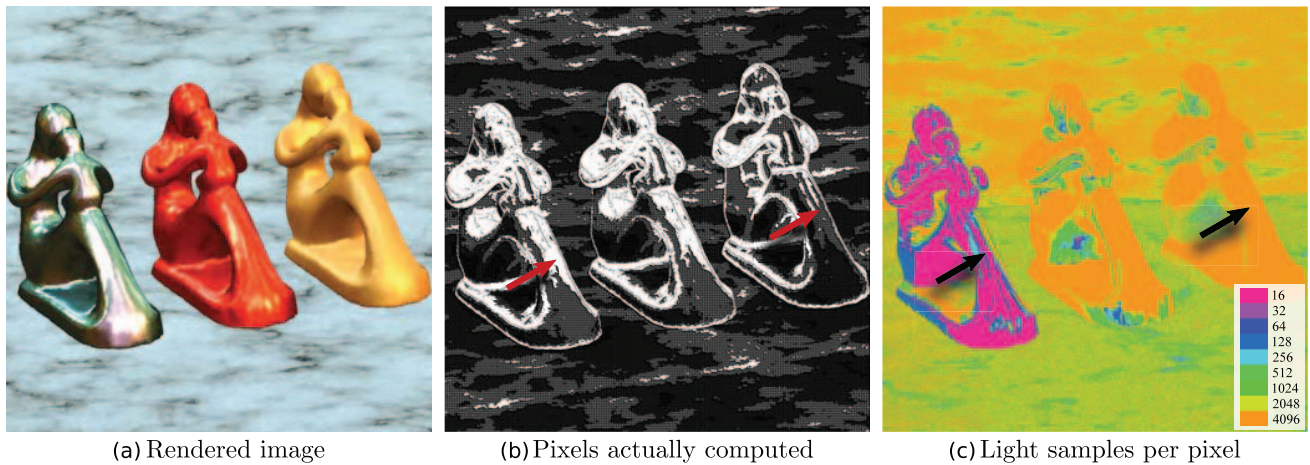
Fig. 9. Three identical objects with different materials (`color-changing-paint3`, `red-metallic-paint`, and `gold-paint`), and a marble textured ground floor with `white-marble` material. We shade more pixels per area for specular materials (red arrows) but we use a larger number of samples to compute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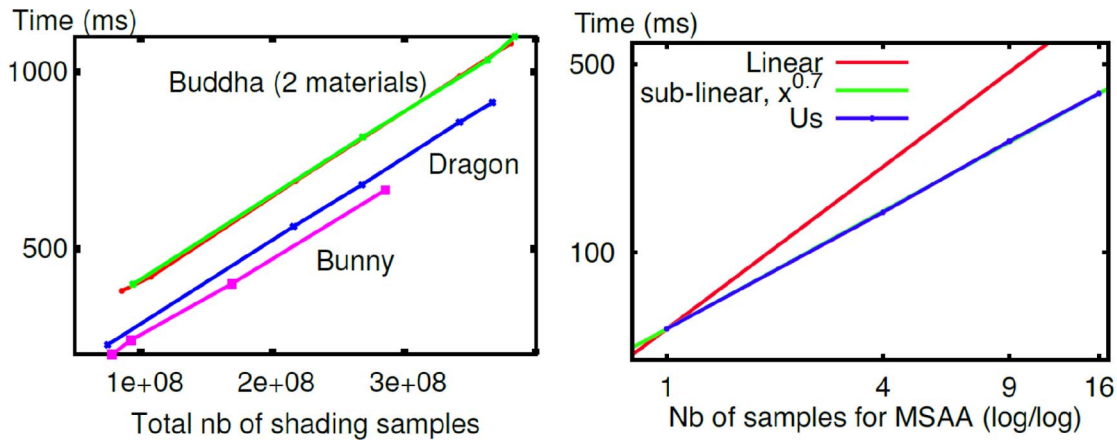

Fig. 10. *Left*: Rendering times are linear in the number of shading samples for various models. *Right*: Rendering time against number of samples for antialiasing (blue) in log scale. Our algorithm scales sublinearly (with an exponent of 0.7) for antialiasing.

depth buffer); two mip-mapped buffers (5.5 MB each) for the bandwidth and variance and for the shading computations with upsampling; two RGB buffers of 6 MB for the environment map and its bandwidth; and two 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 $1{,}024 \times 1{,}024$ and 448 MB at $2{,}048 \times 2{,}048$, which is currently the maximum for our algorithm.

Validation. Fig. 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)). Fig. 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. Fig. 14 compares our result with a forward shading reference computed using importance sampling and a fixed number of shading samples per pixel.

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.

---

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.

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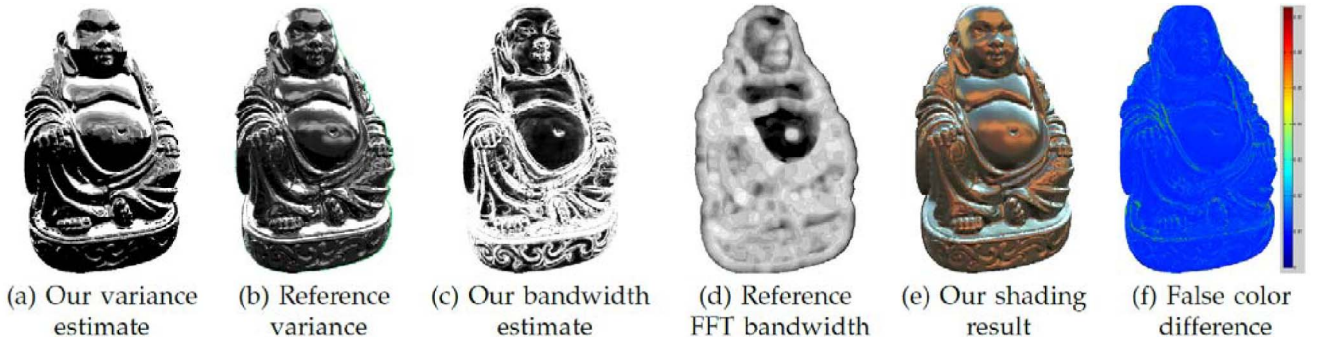
Fig. 12. Validation of predicted variance and bandwidth. (a) our estimate for the variance of the shading integrand (9), (b) reference variance computed using brute-force sampling, (c) our estimate of the image bandwidth, (d) reference image bandwidth (windowed FFT of reference image), (e) our result,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

For this scene, we achieve a $2.5\times$ speedup due to our adaptive sampling. The extent of our speedup depends on the material, the environment map and the area occupied by the object on screen. Fig. 11 tabulates rendering times for our algorithm and brute-force rendering for several scenes. We observe a speedup of $4\times$ to $10\times$.

Spherical Gaussian approximation. Fig. 15 compares the results of Wang et al. [13] 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 [13] is visibly different from the ground truth.

4.3 Rendering Results

Adaptive sampling for prefiltered shading. To use adaptive sampling for preconvolved shading, we preconvolve 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 (11) and look up prefiltered illumination for each sample according to the maximum frequency of the BRDF. Fig. 16 depicts examples of using adaptive sampling for prefiltered shading. We achieved a significant speedup (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 toward 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. Fig. 17 depicts two scenes rendered with the area-light sources.

Adaptive multisample antialiasing. Our bandwidth prediction reduces the cost of multisample antialiasing by adaptive sampling. Standard deferred shading evaluates shaders at every sample: 16 samples per pixel costs 16 times more. Our algorithm scales sublinearly in the number of samples (see Fig. 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. Antialiasing only requires a few extra shader evaluations at the finest levels (blue pixels in Fig. 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. Fig. 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
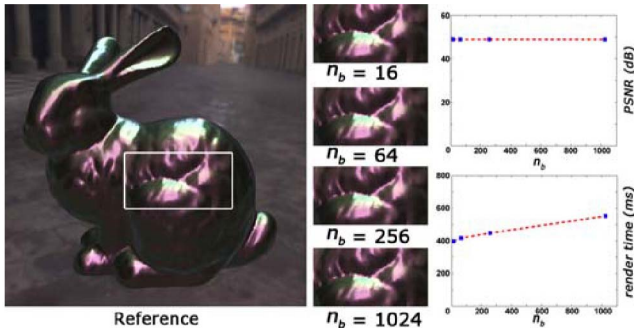

Fig. 13. Effect of the number of sampled directions n_b for bandwidth computation on time (ms) and quality (PSNR). The data points in the plots are at $n_b = 16, 64, 256, 1,024$. The enlarged insets are virtually indistinguishable for the different n_b. The plots depict that fast bandwidth computation ($n_b = 16$) is sufficient. Resolution: 512×512. Material: `color-changing-paint3`.

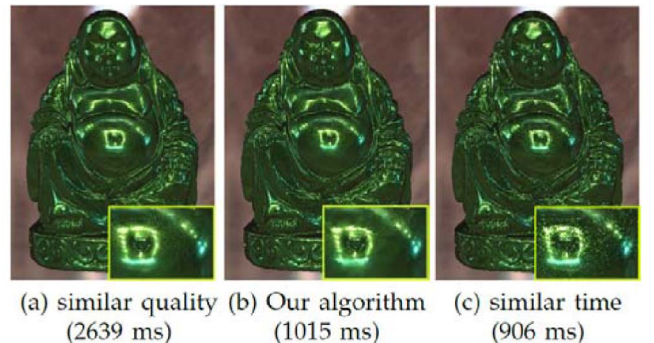(a) similar quality (2639 ms) (b) Our algorithm (1015 ms) (c) similar time (906 ms)

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 antialiasing). Forward-shaded references use BRDF-importance sampling and a fixed number of shading samples per pixel. Material: `green-metallic-paint`.
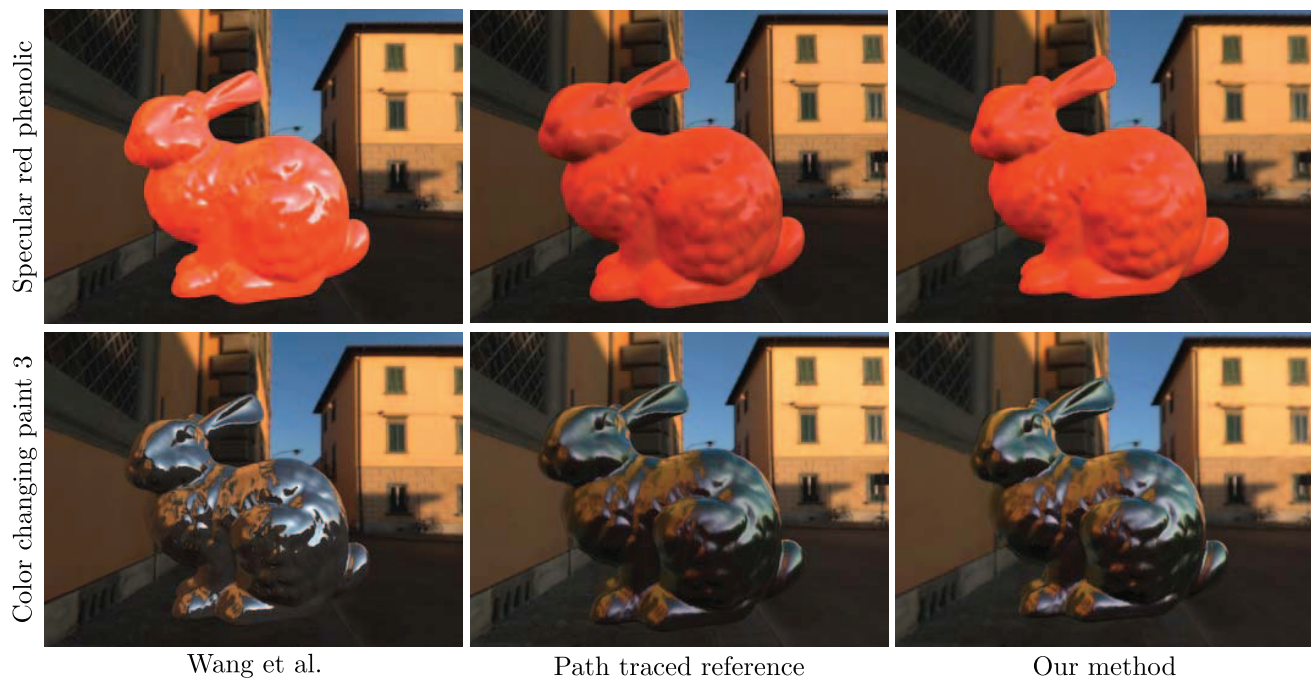
Fig. 15. Comparison with related work [13]. Our algorithm results in pictures that are identical to ground truth, while [13] results in clear differences.
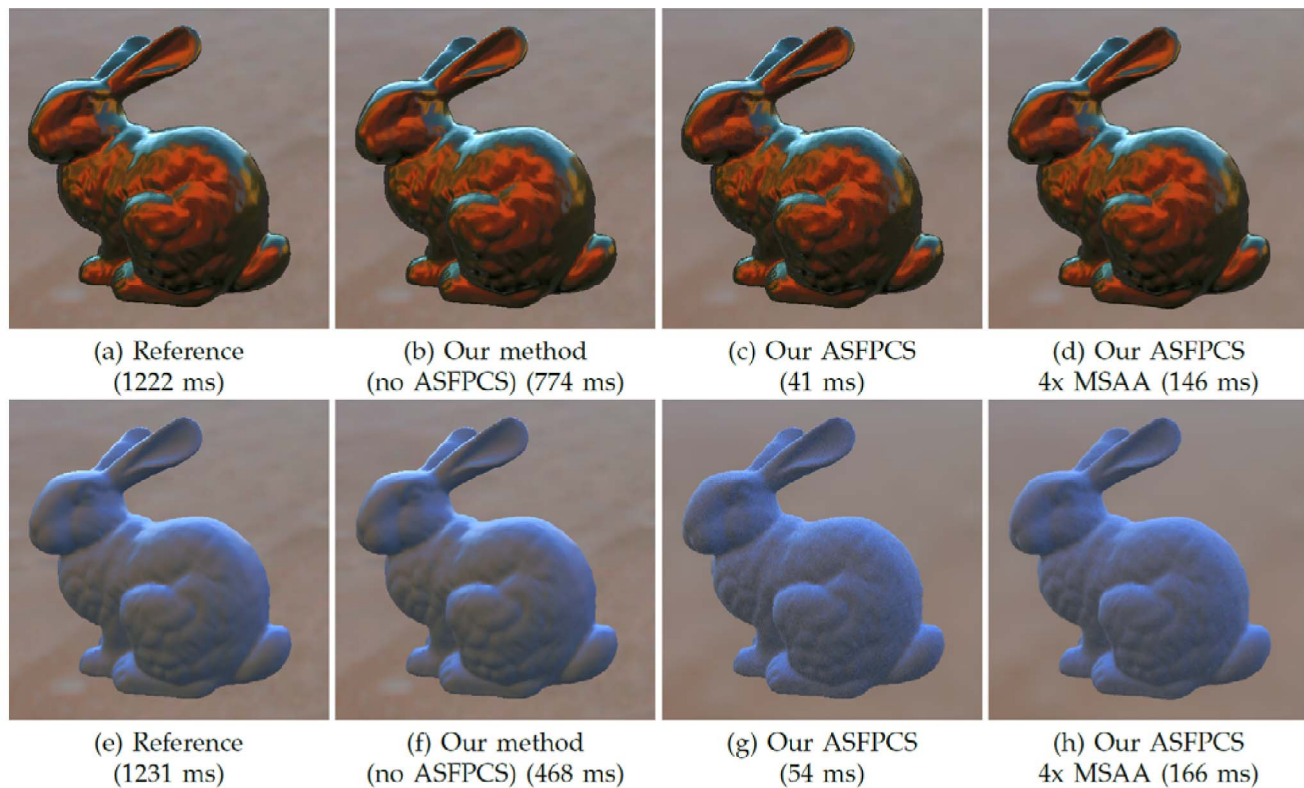

Fig. 16. Adaptive sampling for prefiltered shading examples (c, g) for `color-changing-paint3` and `blue-rubber`, compared to the reference (a,e) and our adaptive sampling method without prefiltered shading (b,f). Our ASFPCS technique with $4\times$ multisample antialiasing is still faster than our method without prefiltered shading. The speedup is 10 to 20 times.

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 predict bandwidth, choosing suboptimal (excessive) sampling over artifacts from insufficient sampling. The

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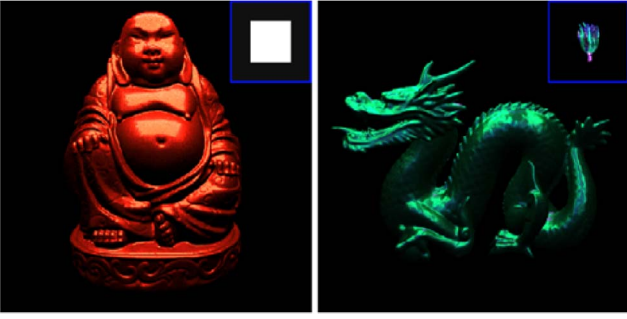
Fig. 17. *Left:* `Buddha` rendered with a solid white square area-light source and `red-metallic-paint` material (513 ms). The area-light source is positioned at the top of the object and is shown as inset. *Right:* `Dragon` rendered with a textured area-light source and `color-changing-paint3` material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

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.

Spatially-varying BRDFs. We have only used homogeneous (nonspatially varying) materials, modulated by a texture. We precomputed 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 precomputation.
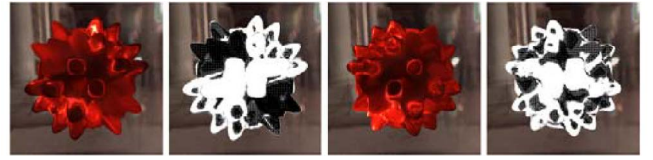

Fig. 18. Application of our algorithm to dynamic geometry, with `red-metallic-paint`. Since our algorithm depends only on data from the G-buffers, it can handle dynamic geometry, displacement mapping, and normal maps. *Top row:* rendered images, *bottom row:* pixels for which we performed shading computations.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

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 sublinearly scale multisample antialiasing 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.
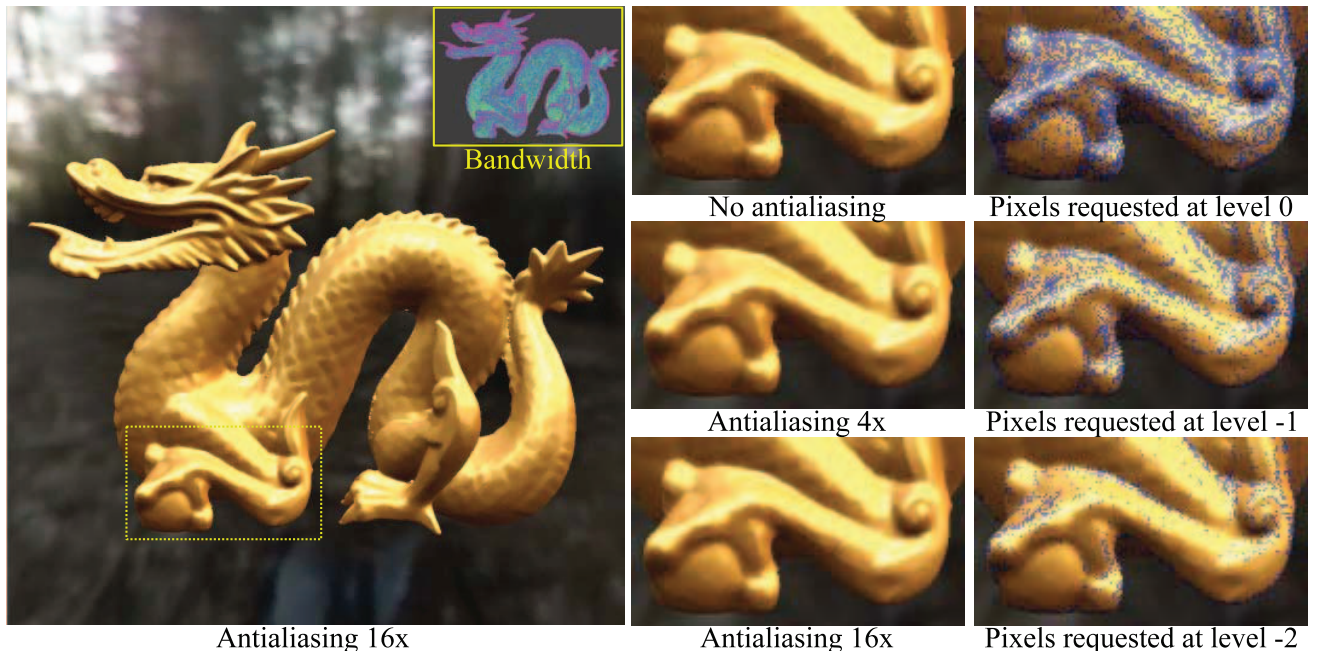

Fig. 19. Adaptive multisample antialiasing using bandwidth information. Aliasing artifacts are visible in the top row of enlarged insets (55 ms). Our multiresolution algorithm handles antialiasing seamlessly by adding extra levels to the pyramid (one level for $4\times$ and two levels for $16\times$). 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 antialiasing is thus reduced: 141 ms for $4\times$, 390 ms for $16\times$. Material: `gold-paint`.

ACKNOWLEDGMENTS

REFERENCES

[1] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 527-536, July 2002.

[2] F. Durand, N. Holzschuch, C. Soler, E. Chan, and F.X. Sillion, "A Frequency Analysis of Light Transport," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 1115-1126, 2005.

[3] M. Bagher, M.C. Soler, K. Subr, L. Belcour, and N. Holzschuch, "Interactive Rendering of Acquired Materials on Dynamic Geometry Using Bandwidth Prediction," *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games (I3D),* M. Garland and R. Wang, eds., pp. 127-134, Mar. 2012.

[4] M. Deering, S. Winner, B. Schediwy, C. Duffy, and N. Hunt, "The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics," *Computer Graphics,* vol. 22, no. 4, pp. 21-30, 1988.

[5] K. Fatahalian, S. Boulos, J. Hegarty, K. Akeley, W.R. Mark, H. Moreton, and P. Hanrahan, "Reducing Shading on GPUs Using Quad-Fragment Merging," *ACM Trans. Graphics,* vol. 29, no. 4, pp. 67:1-67:8, 2010.

[6] W. Heidrich and H.-P. Seidel, "Realistic, Hardware-Accelerated Shading and Lighting," *Proc. ACM SIGGRAPH '99,* pp. 171-178, 1999.

[7] J. Kautz and M.D. McCool, "Approximation of Glossy Reflection with Prefiltered Environment Maps," *Proc. Conf. Graphics Interface,* pp. 119-126, May 2000.

[8] J. Kautz and M.D. McCool, "Interactive Rendering with Arbitrary Brdfs Using Separable Approximations," *Proc. ACM SIGGRAPH '99,* p. 253, 1999.

[9] L. Latta and A. Kolb, "Homomorphic Factorization of BRDF-Based Lighting Computation," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 509-516, 2002.

[10] J. Kautz, J. Snyder, and P.-P.J. Sloan, "Fast Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics," *Proc. Eurographics Symp. Rendering,* pp. 291-296, 2002.

[11] R. Ramamoorthi and P. Hanrahan, "Frequency Space Environment Map Rendering," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 517-526, 2002.

[12] L. Claustres, L. Barthe, and M. Paulin, "Wavelet Encoding of BRDFs for Real-Time Rendering," *Proc. Conf. Graphics Interface (GI),* pp. 169-176, 2007.

[13] J. Wang, P. Ren, M. Gong, J. Snyder, and B. Guo, "All-Frequency Rendering of Dynamic, Spatially-Varying Reflectance," *ACM Trans. Graphics,* vol. 28, no. 5, pp. 133:1-133:10, 2009.

[14] K. Egan, Y.-T. Tseng, N. Holzschuch, F. Durand, and R. Ramamoorthi, "Frequency Analysis and Sheared Reconstruction for Rendering Motion Blur," *ACM Trans. Graphics,* vol. 28, no. 3, pp. 93:1-93:13, 2009.

[15] C. Soler, K. Subr, F. Durand, N. Holzschuch, and F.X. Sillion, "Fourier Depth of Field," *ACM Trans. Graphics,* vol. 28, no. 2, pp. 18:1-18:12, 2009.

[16] R. Ramamoorthi, D. Mahajan, and P. Belhumeur, "A First-Order Analysis of Lighting, Shading, and Shadows," *ACM Trans. Graphics,* vol. 26, no. 1, article 2, Jan. 2007.

[17] G. Nichols and C. Wyman, "Multiresolution Splatting for Indirect Illumination," *Proc. Symp. Interactive 3D Graphics and Games (I3D),* pp. 83-90, 2009.

[18] J. Shopf, G. Nichols, and C. Wyman, "Hierarchical Image-Space Radiosity for Interactive Global Illumination," *Computer Graphics Forum,* vol. 28, no. 4, pp. 1141-1149, 2009.

[19] J. Kopf, M.F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint Bilateral Upsampling," *ACM Trans. Graphics,* vol. 26, no. 3, article 96, 2007.

[20] G. Nichols, R. Penmatsa, and C. Wyman, "Interactive, Multi-resolution Image-Space Rendering for Dynamic Area Lighting," *Computer Graphics Forum,* vol. 29, no. 4, pp. 1279-1288, 2010.

[21] C. Soler, O. Hoel, and F. Rochet, "A Deferred Shading Algorithm for Real-Time Indirect Illumination," *Proc. ACM SIGGRAPH,* p. 18:1, http://hal.inria.fr/inria-00480869/en/, 2010.

[22] G. Nichols and C. Wyman, "Interactive Indirect Illumination Using Adaptive Multiresolution Splatting," *IEEE Trans. Visualization and Computer Graphics,* vol. 16, no. 5, pp. 729-741, Sept./Oct. 2010.

[23] B. Segovia, J.C. Iehl, R. Mitanchey, and B. Péroche, "Non-Interleaved Deferred Shading of Interleaved Sample Patterns," *Proc. Symp. Graphics Hardware (GH),* pp. 53-60, 2006.

[24] T. Ritschel, T. Engelhardt, T. Grosch, H.-P. Seidel, J. Kautz, and C. Dachsbacher, "Micro-Rendering for Scalable, Parallel Final Gathering," *ACM Trans. Graphics,* vol. 28, no. 5, pp. 132:1-132:8, 2009.

[25] R. Ramamoorthi, "Precomputation-Based Rendering," *Foundations and Trends in Computer Graphics and Vision,* vol. 3, no. 4, pp. 281-369, Apr. 2009.

[26] X. Sun, K. Zhou, Y. Chen, S. Lin, J. Shi, and B. Guo, "Interactive Relighting with Dynamic BRDFs," *ACM Trans. Graphics,* vol. 26, no. 3, pp. 27:1-27:10, July 2007.

[27] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing.* Prentice-Hall, 1975.

[28] W. Matusik, H. Pfister, M. Brand, and L. McMillan, "A Data-Driven Reflectance Model," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 759-769, July 2003.

Mahdi Mohammad Bagher received the master's degree in computer graphics, vision, and imaing from UCL, London. He recently finished his PhD degree at Grenoble University in the MAVERICK research team. He is currently a postdoctoral researcher at the University of Montreal in the LIGUM team. He has been mainly conducting research about material appearance representation and rendering.

Cyril Soler has been 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.

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

Laurent Belcour received the PhD degree in 2012, under supervision by Nicolas Holzschuch and Cyril Soler, from the University of Grenoble. He is currently a postdoctoral researcher at Inria Bordeaux in the manao research team. His research interests include Fourier equivalent of local light transport with applications in rendering.
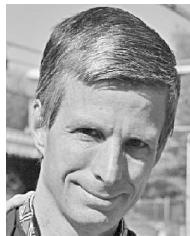
Nicolas Holzschuch received the PhD degree from Grenoble University in 1996, and joined the INRIA in 1997, first in Nancy, then in Grenoble. He is a senior researcher at INRIA Grenoble Rhône-Alpes, and the scientific leader of the MAVERICK research team. His research interests include photorealistic rendering and real-time rendering, with an emphasis on scalability, and multiscale phenomena.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.