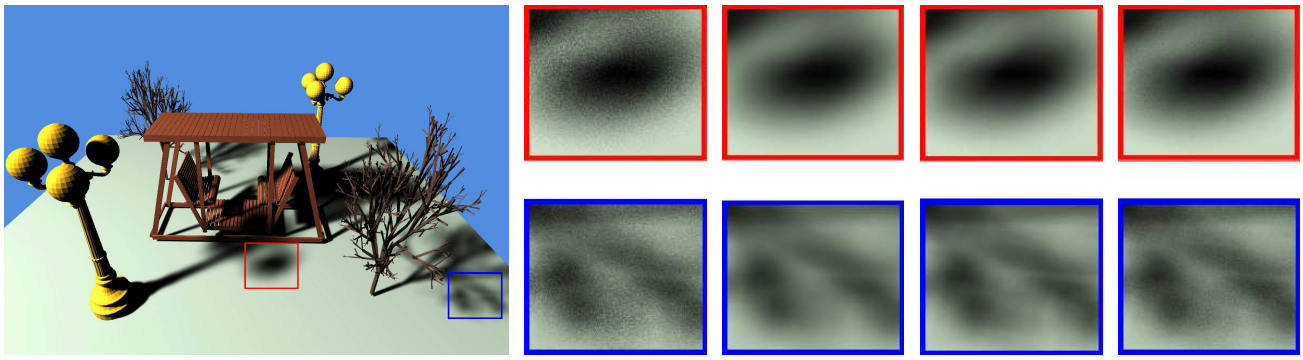
Axis-Aligned Filtering for Interactive Sampled Soft Shadows

Soham Uday Mehta Brandon Wang Ravi Ramamoorthi

University of California, Berkeley *



(a) *our method, 37 spp (samples per pixel)* **(b)** *equal time, unfiltered, 37 spp* **(c)** *our method, 37 spp* **(d)** *ground truth 4000 spp* **(e)** *equal error, unfiltered, 153 spp*

Figure 1: *(a) Soft shadows from a planar area light are computed accurately by raytracing at 2.3 fps, with adaptive sampling and filtering using our method. The scene has 300K vertices and complex shadows. Our method is simple, requires no precomputation and directly plugs into NVIDIA's OptiX or other real-time raytracer. (b) Soft shadows without filtering, equal time; note the considerable noise. (c) Our method compares well to ground truth (d). (e) Equal error without filtering still has some noise making it visually less acceptable. The scene is based on one first used in [Overbeck et al. 2006]. Readers are encouraged to zoom into the PDF in all figures to see the noise and shadow details.*

Abstract

We develop a simple and efficient method for soft shadows from planar area light sources, based on explicit occlusion calculation by raytracing, followed by adaptive image-space filtering. Since the method is based on Monte Carlo sampling, it is accurate. Since the filtering is in image-space, it adds minimal overhead and can be performed at real-time frame rates. We obtain interactive speeds, using the Optix GPU raytracing framework. Our technical approach derives from recent work on frequency analysis and sheared pixel-light filtering for offline soft shadows. While sample counts can be reduced dramatically, the sheared filtering step is slow, adding minutes of overhead. We develop the theoretical analysis to instead consider *axis-aligned* filtering, deriving the sampling rates and filter sizes. We also show how the filter size can be reduced as the number of samples increases, ensuring a consistent result that converges to ground truth as in standard Monte Carlo rendering.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Shadow Algorithms

Keywords: Fourier, filtering, shadows, raytracing

*(sohamum,ravir)@cs.berkeley.edu, brandonwang@berkeley.edu

Links: DL PDF VIDEO CODE

1 Introduction

This paper focuses on accurate and efficient rendering of soft shadows from planar area lights. Soft shadows are a key effect in photo-realistic rendering, but are expensive because every location on the area light must be considered and sampled. While real-time techniques [Hasenfratz et al. 2003; Johnson et al. 2009] have achieved impressive results, they rely on a variety of approximations, making no guarantee of convergence to ground truth while retaining some artifacts. On the other hand, Monte Carlo shadow-ray tracing is the preferred offline rendering method since it is physically-based, accurate and artifacts (noise that goes away with more samples) are well understood. Monte Carlo rendering can now be GPU-accelerated, and ready-to-use raytracers are available; we use NVIDIA's Optix [Parker et al. 2010]. However, the number of samples per pixel for soft shadows remains too large for interactive use.

We build on [Egan et al. 2011b] to significantly reduce the number of Monte Carlo samples needed, while still keeping the benefits of accurate raytraced occlusion. [Egan et al. 2011b] developed a sheared filter in the 4D pixel-light space, that combines samples from many different pixels, at different light locations. However, the filtering step introduces considerable overhead of minutes, and the technique is offline. In this paper, we use simpler *axis-aligned* (rather than sheared) filters. (In this context, axis-aligned or sheared refers to the shadow light field in the pixel-light domain, rather than the 2D image—we also use separable 1D filters along the image axes as a practical optimization, but this is less critical.)

While the number of samples per pixel is somewhat increased in our axis-aligned method as opposed to sheared filtering, post-processing reduces to a simple adaptive 2D image-space filter, rather than needing to search over the irregular sheared filter for samples in the full 4D shadow light field. Our method is easily integrated with existing Monte Carlo renderers, reducing the samples

required by 4×-10×. The main benefit is that overhead is minimal (about 5ms), even in GPU raytracers, and interactive performance can therefore be achieved. Our specific contributions are:

1. Derivation of adaptive sampling rates and adaptive filter sizes for axis-aligned pixel-light filtering of soft shadows.
2. Consistent sampling, wherein the filter size is adjusted for the sampling rate, ensuring convergence as in standard Monte Carlo. (Previous sheared filtering approaches still contain some artifacts and typically do not guarantee convergence).
3. A simple Optix implementation where the filtering has minimal overhead. We achieve interactive frame rates (Fig. 1).

2 Previous Work

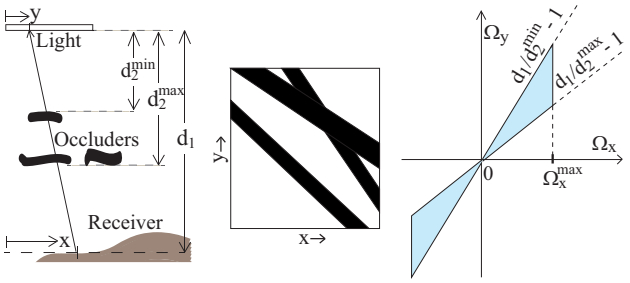
Real-Time and Accelerated Soft Shadows: The shadow mapping method [Williams 1978] can be extended to soft shadow maps that consider occlusion from the entire area source [Guennebaud et al. 2006; Annen et al. 2008]. As noted in [Johnson et al. 2009], these methods make various tradeoffs of speed and accuracy. Soler and Sillion [1998] provide an analytic solution, but only for geometry in parallel planes. Shadow volumes [Crow 1977] can also be extended to soft shadows using geometric ideas like penumbra wedges [Assarsson and Möller 2003; Laine et al. 2005] An analytic approach based on beam tracing is proposed by [Overbeck et al. 2007], but is not yet fast enough for real-time use, especially on complex scenes. Another body of work is precomputed relighting [Sloan et al. 2002], but these are usually limited to static scenes lit by environment maps. We require no precomputation and raytrace each frame independently, allowing for dynamic scenes.

Monte Carlo and Ray-Traced Shadows: A number of accelerations that exploit coherence have been proposed [Hart et al. 1999; Agrawala et al. 2000], as well as methods to separate near and far-field effects [Arikan et al. 2005], prefilter visibility [Lacewell et al. 2008] and accelerate ray packets [Benthin and Wald 2009]. In contrast, we directly leverage a GPU raytracing framework in Optix [Parker et al. 2010] but add an adaptive image-space filter in post-processing.

A few recent works have explored GPU acceleration of occlusion queries [Eisemann and Decoret 2007; Forest et al. 2008] and sampled visibility [Sintorn and Assarsson 2008]. It should be possible in future to combine our image-space filtering approach with these GPU methods, but we currently only use the basic Optix raytracer.

Adaptive Sampling and Sheared Reconstruction: Our method adaptively samples the image plane, inspired by the offline rendering methods of [Guo 1998], and more recently [Hachisuka et al. 2008] and [Overbeck et al. 2009]. We build most closely on recent approaches for frequency analysis of shadows and light transport [Chai et al. 2000; Durand et al. 2005; Lanman et al. 2008]. In particular, Egan et al. [2011b] develop a method for caching the 4D shadow light field with very low sampling density, followed by a sheared pixel-light filter for reconstruction. Several other recent papers have explored similar ideas for motion blur, depth of field, ambient occlusion and more general effects [Egan et al. 2009; Soler et al. 2009; Egan et al. 2011a]. However, the filtering phase is slow, often taking longer than actual shadow casting. Sheared filtering must store the full 4D light field, and perform an irregular search over it at each pixel. [Lehtinen et al. 2011] use a GPU-accelerated reconstruction, but are still too slow for interactive use. We use simpler axis-aligned filtering to develop a very efficient post-processing algorithm that reduces to simple 2D adaptive screen-space filtering.

De-Noising: Our post-process is essentially an image de-noising operator, building on [Rushmeier and Ward 1994; McCool 1999; Xu and Pattanaik 2005]. Recently, [Sen and Darabi 2012] describe an iterative approach to filter Monte Carlo noise, but this is still an offline procedure for global illumination. We are also inspired by many recent image-processing algorithms, such as [Dabov et al. 2007]. However, these previous approaches are not designed for real-time use, and also assume limited a-priori information about the scene. We use our theoretical analysis to estimate the precise extent of the filter needed adaptively at each pixel.



(a) *Schematic* **(b)** *Occlusion* **(c)** *Occlusion Spectrum*

Figure 2: *(a) The basic setup and coordinate system for analyzing occlusion, (b) The binary occlusion function for flatland, and (c) Fourier spectrum of the occlusion function.*

3 Background

We now briefly introduce the basic spatial and frequency domain analysis of the occlusion function. The ideas are summarized in Fig. 2. In the next section, we define the various frequency bandlimits precisely in physical units, and then proceed to derive filter sizes and sampling rates. As in previous work, we introduce the theory with a 2D occlusion function (1 spatial dimension on the receiver and light source); the extension to 2D images and lights is straightforward and we provide details later of our implementation.

Assumptions: We use y for light coordinates and x for coordinates on a parameterization plane parallel to the light source as shown in Fig. 2(a); Our goal is to compute the result $h(x)$,

$$h(x) = r(x) \int f(x,y) l(y)\, dy, \tag{1}$$

where $f(x,y)$ is the occlusion or shadow function[1], and $l(y)$ is the light source intensity. We assume planar area lights and generally use a gaussian form for $l(y)$. We also focus on the diffuse intensity (assuming the specular term will be added by a separate pass, as is common in many applications), and that the irradiance from the light can be approximated at its center with $r(x)$, so we can focus purely on the occlusion $f(x,y)$. These assumptions are similar to many previous works [Soler and Sillion 1998; Egan et al. 2011b]. Textures can be included directly in $r(x)$. Glossy or other BRDFs can also be combined into the lighting function $l(y)$, as discussed in [Egan et al. 2011a], and simply require us to modify the effective light frequency bandlimit in our formulae. However, we do not include these effects in most of our examples, to focus on shadows. In practice, our method also works fairly well, without modifications, when including only the cosine falloff in $l(y)$, as seen in Fig. 11(a).

Occlusion Function: From the geometry of Fig. 2(a), we can derive the 2D occlusion function $f(x,y)$ in terms of a 1D visibility

[1]Note that [Egan et al. 2011b] first define f in terms of a ray-space parameterization with a plane one unit away and then compute the occlusion; we directly use $f(x,y)$ to denote the occlusion.

($g(\cdot)$ is defined along the occluder plane parallel to the light plane),

$$f(x,y) = g\left(\frac{d_2(x-y)}{d_1} + y\right), \tag{2}$$

where d_2 is the distance from the light to the occluder and d_1 is the distance from the light to the receiver (note that d_1 can depend on x since the receiver may not be planar). The occlusion function $f(x,y)$ has a regular structure with diagonal bands due to equation 2; the slope of the bands will be given by $-d_2/(d_1-d_2)$. If the occluder depths vary between $d_2^{\min} < d_2 < d_2^{\max}$, f typically looks like Fig. 2(b).

Fourier Analysis: The Fourier spectrum for equation 2 lies on a line with slope $s = (d_1/d_2) - 1$ (orthogonal to the spatial domain slope in Fig. 2(b)). In terms of Fourier spectra F and G,

$$F(\Omega_x, \Omega_y) = \frac{d_1}{d_2}\delta(\Omega_y - s\Omega_x)G\left(\frac{d_1}{d_2}\Omega_x\right). \tag{3}$$

As noted in [Chai et al. 2000; Egan et al. 2011b], when we have a range of depths, most of the spectrum will lie in a union of the lines for each depth, and hence be confined to a double wedge as shown in Fig. 2(c).[2] Maximum and minimum slopes respectively are

$$s_1 = \frac{d_1}{d_2^{\min}} - 1 \qquad s_2 = \frac{d_1}{d_2^{\max}} - 1. \tag{4}$$

4 Axis-Aligned Filtering

The frequency spectra define the sampling resolution and hence the filter sizes. In this section, we derive the axis-aligned filters that we use. Note that axis-aligned here refers to the 2D (later 4D) pixel-light space, not the 2D image domain. We begin by introducing the frequency domain bandlimits; we use precise physical units unlike many previous works. We then derive filter sizes and apply them in screen-space (implementation details are later in Sec. 6).

4.1 Preliminaries: Frequency Domain Bandlimits

Frequencies in both dimensions, Ω_x and Ω_y, are measured in m^{-1}, assuming world coordinates are measured in meters. The limited resolution of the output image acts as a low pass filter in the spatial (pixel) dimension. $\Omega_{\text{pix}}^{\max}$ is the maximum displayable frequency in the pixel space, and following earlier approaches, is taken to be $\Omega_{\text{pix}}^{\max} = (1/d)\ m^{-1}$, where d is the projected distance per pixel (i.e., the length in world coordinates that this pixel corresponds to, which also accounts for effects like foreshortening).

The occlusion function $F(\Omega_x, \Omega_y)$ is assumed to have a spatial bandlimit imposed due to the smoothness of geometry causing the occlusion.[3] There is no definite way to quantify this bandlimit, so we will introduce it as a parameter $\Omega_{\text{x}}^{\max}$ in our analysis. In particular, we first introduce a parameter α, so that

$$\alpha = \frac{\Omega_{\text{g}}^{\max}}{\Omega_{\text{pix}}^{\max}} \in (0,1], \tag{5}$$

where $\Omega_{\text{g}}^{\max}$ is the (unknown) bandlimit on the occluding geometry. The conservative approach (which we use in our images) would simply set $\alpha = 1$, i.e., use $\Omega_{\text{g}}^{\max} = \Omega_{\text{pix}}^{\max}$. From equation 3,

$$\Omega_{\text{x}}^{\max} = \alpha\frac{d_2^{\max}}{d_1}\Omega_{\text{pix}}^{\max} = \alpha(1+s_2)^{-1}\Omega_{\text{pix}}^{\max}. \tag{6}$$

[2] As discussed in [Egan et al. 2011b], there are unusual configurations that violate these assumptions, but they do not arise in our practical tests.

[3] In practice, sharp edges could produce infinite frequencies, but we limit ourselves here to pixel resolution for practical purposes.

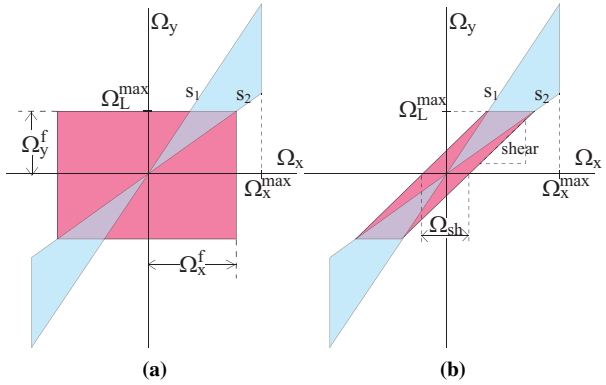


Figure 3: *Schematic of (a) Axis-Aligned Filter, (b) Sheared Filter.*

It is also useful to know the Ω_y bandlimit of the occlusion function. From the geometry of Fig. 2(c), it is clear that $\Omega_{\text{y}} = s\Omega_{\text{x}}$, which leads to $(d_1/d_2 - 1)\alpha(d_2/d_1)\Omega_{\text{pix}}^{\max}$, and

$$\Omega_{\text{y}}^{\max} = \alpha\left(1 - \frac{d_2^{\min}}{d_1}\right)\Omega_{\text{pix}}^{\max}. \tag{7}$$

Finally, since effective visibility is the integration of the light internsity with the occlusion function, frequencies in the occlusion function outside the light's bandlimits $\Omega_{\text{L}}^{\max}$ will be filtered (Fig. 3(a)).

We assume the light is a Gaussian of standard deviation σ meters, so $\Omega_{\text{L}}^{\max} = (1/\sigma)\ m^{-1}$. Of course, Gaussian lights will not perfectly bandlimit, and the cutoff can also be scaled by a small constant (e.g., using 2σ instead of σ) without materially affecting the derivations. The numerical constants used in our paper are consistent, and agree well with empirical observations. We have also experimented with constant (non-Gaussian) lights. These become Fourier domain sincs without a strict frequency cutoff, but they do work in practice, with more conservative bandlimits (see Fig. 11(b)).

4.2 Frequency Extent of Axis-Aligned Filter

We define the axis-aligned filter in the frequency domain as $[-\Omega_{\text{x}}^f, \Omega_{\text{x}}^f] \times [-\Omega_{\text{y}}^f, \Omega_{\text{y}}^f]$. As shown in Fig. 3(a),

$$\Omega_{\text{x}}^f = \min\left[\frac{\Omega_{\text{L}}^{\max}}{s_2}, \Omega_x^{\max}\right] \qquad \Omega_{\text{y}}^f = \Omega_{\text{L}}^{\max}. \tag{8}$$

The Ω_y bandlimit simply comes from the light, while the Ω_x bandlimit is induced by the light. Ω_{x}^f must also be clipped to $\Omega_x^{\max}$.

Even though the size of the axis-aligned filter in Fig. 3(a) is larger than the sheared filter in Fig. 3(b), one important advantage is the decoupling of filtering over the spatial x and light y dimensions. This reduces to a simple screen-space filter, as we will see next.

4.3 Towards filtering in Screen-Space

Spatial Domain Filters: The primal domain analogue of a frequency domain box filter is a sinc filter, which decays slowly. Hence, we approximate both the fequency and primal domain filters with Gaussians. In the primal or pixel domain, the standard deviation[4] is $\beta(x) = 1/\Omega_{\text{x}}^f$.

[4] As described in equation 13, we actually find it better to use a value $\beta(x) = k^{-1}\Omega_{\text{x}}^f$ where k accounts for the Gaussian energy being spread over multiple standard deviations, and we usually set k = 3.

Similarly, the standard deviation in the y-dimension $\gamma(y) = 1/\Omega_{\mathrm{y}}^{\mathrm{f}}$. However, $\Omega_{\mathrm{y}}^{\mathrm{f}} = \Omega_{\mathrm{L}}^{\max} = (1/\sigma)$, and $\gamma(y) = \sigma$, which is simply the standard deviation of the original light source (in meters). Thus, the filter in the y dimension simply integrates against the light source $l(y)$. *This integral can be performed first and simply gives the standard noisy Monte Carlo result.* Thereafter, we can apply the x filter only in the spatial dimensions.

Shading Equations: Note that the x and y dimensions are treated separately in equation 1 (we also omit $r(x)$ in equation 1 for simplicity since it just multiplies the final result). The y dimension involves an integral of the light and the visibility, which can be performed in any orthonormal basis (either spatial or frequency domain). On the other hand, the x dimension is related to the final image, and we must apply the axis-aligned filter (a spatial convolution) to remove replicas from sampling. Putting this together,

$$h(x) = \int_{x'} \int_{y} w(x - x'; \beta(x)) \bar{f}(x', y) w(y; \gamma(y))\, dy\, dx', \quad (9)$$

where $w()$ are the spatial domain Gaussian filters, and $\bar{f}$ is the sampled (noisy) visibility. To simplify further, note per the earlier discussion that $w(y; \gamma(y)) = l(y)$ is just the light source itself. Hence, we can pre-integrate the lighting to obtain a noisy result $\bar{h}(x)$ to which we then apply a simple Gaussian filter,

$$\begin{aligned} \bar{h}(x') &= \int_{y} \bar{f}(x', y) l(y)\, dy \\ h(x) &= \int_{x'} \bar{h}(x') w(x - x'; \beta(x))\, dx'. \end{aligned} \quad (10)$$

The Gaussian filter $w(x - x')$ is given in the standard way by

$$w(x - x'; \beta) = \frac{1}{\sqrt{2\pi}\beta} \exp\left[-\frac{\| x - x' \|^2}{2\beta^2} \right], \quad (11)$$

where in the 3D world, we set $\| x - x' \|$ to be the distance between two world-space locations, but measured along the plane parallel to the light (motion normal to the light is excluded). In other words, if $|\cdot|$ measures Euclidean distance and $\mathbf{n}$ is the light normal,

$$\| \mathbf{x} - \mathbf{x}' \|^2 = | \mathbf{x} - \mathbf{x}' |^2 - \left[\mathbf{n} \cdot (\mathbf{x} - \mathbf{x}') \right]^2. \quad (12)$$

Discussion: The simplicity of equation 10 is key for our algorithm. In essence, we are just computing the standard noisy visibility $\bar{h}(x)$ followed by a denoising or filtering step to obtain $h(x)$. Our analysis is simply telling us exactly how much to filter (giving the value of $\beta(x)$) for each spatial location, and $\beta(x)$ is spatially-varying, depending on the local geometry and occluders. So far, all of the discussion has been in world-space, but the filtering can be reduced to image-space as discussed in our implementation (Sec. 6).

Unlike sheared filtering, we do not need to keep track of the full occlusion light field, which is a major memory savings. Moreover, filtering happens only on the image, and we do not need to search the light field for samples that fall inside an irregular sheared filter.

Finally, $\beta(x) \sim 1/\Omega_{\mathrm{x}}^{\mathrm{f}}$ and is given from equations 4 and 8 as,

$$\boxed{\beta(x) = \frac{1}{k} \cdot \frac{1}{\mu} \max\left[\sigma\left(\frac{d_1}{d_{2,\max}} - 1 \right), \frac{1}{\Omega_{\mathrm{x}}^{\max}} \right]}, \quad (13)$$

where $\sigma = 1/\Omega_{\mathrm{L}}^{\max}$ is the standard deviation of the Gaussian for the light, and the factor of k (we use $k = 3$) corrects for the spread of energy from the Gaussian filter across multiple standard deviations. For now, $\mu = 1$; it is a parameter we introduce later to allow the frequency (and hence spatial) width of the filter to adapt to the number of samples, as described in Sec. 5.2.

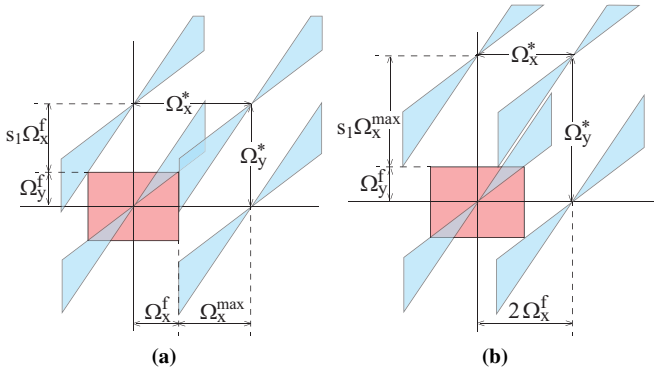


Figure 4: *Packing of spectra for axis-aligned filtering. The packing in (a) is denser, and will be used in the rest of our analysis.*

This equation simply expresses the intuitive notion that shadows from close occluders can be sharper and should be filtered less, while those from further occluders (smaller $d_{2,\max}$) can be filtered more aggressively (see filter widths in Fig. 7(a,c)). Also, the lower frequency the light (the larger σ is), the more we can filter.

5 Adaptive Sampling Rates

Besides adaptive (spatially-varying) filtering, a second component of our method is to adaptively choose the number of samples per pixel, and we now derive this sampling rate from the frequency analysis. So far, we have considered the "critical" sampling and filtering, which is just adequate to produce antialiased results. In practice, because of non-idealities in sampling and reconstruction (including the use of Gaussian filters without perfect bandlimits), we would like to increase the number of samples beyond the minimum required, just as in standard Monte Carlo rendering. Our next contribution is to show how the image filter size can be decreased with inreasing sampling rate, in the limit reducing to the standard pixel filter and guaranteeing a result consistent with ground truth.

To determine the minumum sampling rates, we must pack the spectra such that adjacent copies do not overlap the axis-aligned filter, to avoid aliasing error. The resulting frequency space separations are denoted Ω_{x}^{*} and Ω_{y}^{*}. Note that the occlusion spectra themselves can overlap, but not in the region of the filter—only frequencies in the axis-aligned filter are relevant for the final image; higher frequencies are filtered out by the light source. The two possible compact packings of the repeated spectra are shown in Fig. 4.

Packing (Fig. 4)	Ω_{x}^{*}	Ω_{y}^{*}
(a)	$\Omega_{\mathrm{x}}^{f} + \Omega_{\mathrm{x}}^{\max}$	$\Omega_{\mathrm{y}}^{f} + \min(s_1\Omega_{\mathrm{x}}^{f}, \Omega_{\mathrm{y}}^{\max})$
(b)	$2\Omega_{\mathrm{x}}^{f}$	$\Omega_{y}^{f} + s_1\Omega_{\mathrm{x}}^{\max}$

The difference in the sampling rates $\Omega_{\mathrm{x}}^{*} \times \Omega_{\mathrm{y}}^{*}$ between the two is:

$$(\Omega_{\mathrm{x}}^{*} \times \Omega_{\mathrm{y}}^{*})(a) - (\Omega_{\mathrm{x}}^{*} \times \Omega_{\mathrm{y}}^{*})(b) = (\Omega_{\mathrm{x}}^{\max} - \Omega_{\mathrm{x}}^{f})(\Omega_{\mathrm{y}}^{f} - s_1\Omega_{\mathrm{x}}^{f}) \leq 0 \quad (14)$$

since $\Omega_{\mathrm{x}}^{\max} \geq \Omega_{\mathrm{x}}^{f}$ and $s_1\Omega_{\mathrm{x}}^{f} = s_1\Omega_{\mathrm{y}}^{f}/s_2 \geq \Omega_{\mathrm{y}}^{f}$. So, (a) represents a tighter packing than (b). It also corresponds intuitively to sampling at pixel resolution in x (if $\Omega_{\mathrm{x}}^{\max} \approx \Omega_{\mathrm{pix}}^{\max}$). In fact, as we

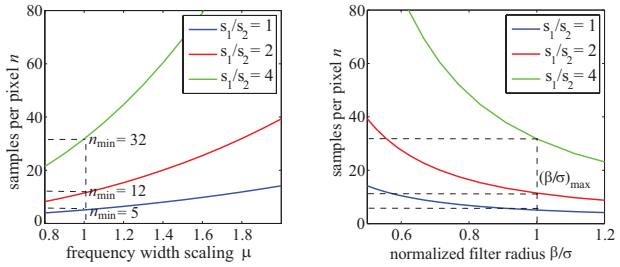


Figure 5: *Sampling beyond the minimum rate. (a) The number of samples varies quadratically with μ. The values of n_{min} are shown for $\mu = 2$. (b) The filter width β varies inversely with μ, in the limit approaching a single pixel as in standard Monte Carlo.*

shall see next, we sample at each pixel as in standard Monte Carlo, and use the analysis above to set per-pixel sampling rates.

5.1 Per-Pixel Sampling Rate

Extending the 2D analysis to 4D, the sampling rate is given by $(\Omega_x^*)^2(\Omega_y^*)^2$. Note that frequencies are in units of m^{-1}. To convert this to samples per pixel n, we multiply by the area of the pixel A_p as well as the area of the light source A_l (both in square meters),

$$n = (\Omega_x^*)^2(\Omega_y^*)^2 \times A_p \times A_l. \tag{15}$$

We will make the following simplifications
(1) For the pixel, $(\Omega_{\text{pix}}^{\max})^2 \times A_p = 1$.
(2) For the area light, $(\Omega_{\text{L}}^{\max})^2 \times A_l = 4$. ($\Omega_{\text{L}}^{\max} = \sigma^{-1}$ and we assume that the effective width of the light is 2σ, so that $A_l = (2\sigma)^2$.)

In the common case (where we need not consider the min/max expressions), we know that $\Omega_{\text{x}}^f = \Omega_{\text{L}}^{\max}/s_2$, $\Omega_{\text{y}}^f = \Omega_{\text{L}}^{\max}$ (equation 8), and $\Omega_{\text{x}}^{\max}$ is given by equation 6. Hence, the sampling rates for packing scheme (a) in Fig. 4 are

$$\begin{aligned}
(\Omega_{\text{x}}^*)^2 \times A_p &= \left[\frac{\Omega_{\text{L}}^{\max}}{s_2} + \alpha(1+s_2)^{-1}\Omega_{\text{pix}}^{\max}\right]^2 \times A_p \\
&= \left(\frac{2}{s_2}\sqrt{\frac{A_p}{A_l}} + \alpha(1+s_2)^{-1}\right)^2 \qquad (16) \\
(\Omega_{\text{y}}^*)^2 \times A_l &= \left[\Omega_{\text{L}}^{\max}\left(1+\frac{s_1}{s_2}\right)\right]^2 \times A_l = 4\left(1+\frac{s_1}{s_2}\right)^2,
\end{aligned}$$

from which we can derive the final sampling rate

$$\boxed{n = 4\left(1+\frac{s_1}{s_2}\right)^2\left(\frac{2}{s_2}\sqrt{\frac{A_p}{A_l}} + \alpha(1+s_2)^{-1}\right)^2}. \tag{17}$$

Note that the sampling rate depends on s_1 and s_2 and is therefore spatially-varying (adaptive for each pixel), as shown in Fig. 7(b,d).

Discussion: It is instructive to compare this sampling rate to that predicted by theory if no filtering were done (we sample each point x separately as in standard Monte Carlo rendering). In that case, the sampling rate $\Omega_{\text{y}}^* = 2\Omega_{\text{y}}^{\max}$; substituting equation 7,

$$\begin{aligned}
n_{\text{nofilter}} &= 4\left[\alpha\left(1-\frac{d_2^{\min}}{d_1}\right)\Omega_{\text{pix}}^{\max}\right]^2 \times A_l \\
&= 4\left[\alpha\left(1-\frac{d_2^{\min}}{d_1}\right)\right]^2 \frac{A_l}{A_p}, \qquad (18)
\end{aligned}$$

which can be very large because of the A_l/A_p factor, since the light source area is usually much larger than the area seen by a single image pixel. While Monte Carlo analysis is typically in terms of statistical noise reduction (although see [Durand 2011] for a Fourier view), this frequency analysis also helps explain the large number of samples often needed to obtain converged soft shadows.

The minimum sampling rate required for a sheared filter (a similar result was derived in the Appendix to [Egan et al. 2011b] but not used in their actual algorithm) under the same conditions, is

$$n_{\text{shear}} \approx 4\left(1-\frac{s_2}{s_1}\right)^2\left(\Omega_{\text{sh}} \cdot d + \alpha(1+s_2)^{-1}\right)^2, \tag{19}$$

where $d = \sqrt{A_p}$ is the linear size of the scene for one pixel. To make a comparison to axis-aligned filtering, we can make Ω_{x}^f explicit in the second factor in equation 17,

$$n_{\text{axis}} = 4\left(1+\frac{s_1}{s_2}\right)^2\left(\Omega_{\text{x}}^f \cdot d + \alpha(1+s_2)^{-1}\right)^2, \tag{20}$$

It can be seen that $n_{\text{shear}} < n_{\text{axis}}$ as expected, since the sheared filter scale $\Omega_{\text{sh}} < \Omega_{\text{x}}^f$ and $s_2 < s_1$. However, the reduction in Monte Carlo samples is more than offset by the additional complexity of implementing the sheared filter which can introduce overheads of minutes. Our method is intended to be a practical alternative where speed is of essence, somewhat increasing the needed samples per pixel, but making the filter very fast and simple to implement.

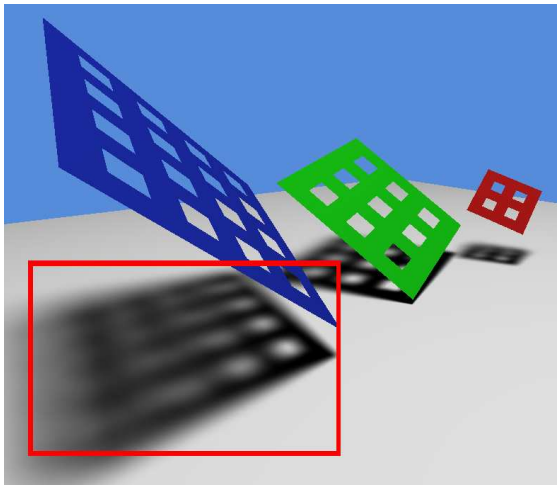
5.2 Sampling beyond the minimum sampling rate

We now show how we can adapt the spatial filter to the samples per pixel. As we increase the sampling rate, the replicas are separated by more than the minimum required for accurated antialiased images. We can take advantage of this by using a more conservative smaller filter in the primal domain (or a larger filter in the frequency domain), to provide more leeway for imperfect reconstruction. This approach has the desirable property that the image is consistent, improving with more samples and converging to ground truth (in the limit, our filter will be a single pixel).

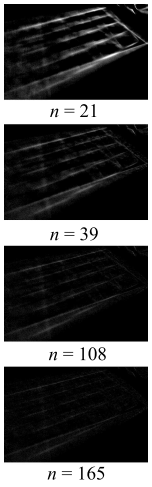
The free parameter is Ω_{x}^f, with the spatial filter width depending on $\beta \sim 1/\Omega_{\text{x}}^f$. We will denote $\Omega_{\text{x}}^f = \mu\Omega_{\text{x0}}^f$, where μ measures how much the critical spatial filter Ω_{x0}^f is scaled. Making the parameter μ explicit in equation 17,

$$\boxed{n(\mu) = 4\left(1+\mu\frac{s_1}{s_2}\right)^2\left(\mu\frac{2}{s_2}\sqrt{\frac{A_p}{A_l}} + \alpha(1+s_2)^{-1}\right)^2}. \tag{21}$$

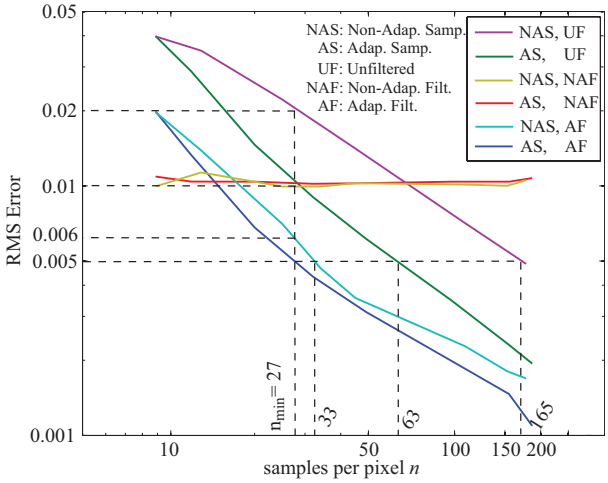
Given a desired value of μ, we can find the number of samples n (we typically use $\mu = 2$). Note that the number of samples is a quadratic function of μ, as shown in Fig. 5(a). The variation of $\beta = 1/\Omega_{\text{x}}^f$ with the number of samples per pixel n is shown in Fig. 5(b), and is inversely related to μ. We see that a small increase in sampling density enables a more conservative filter—that is faster to compute and reduces artifacts.



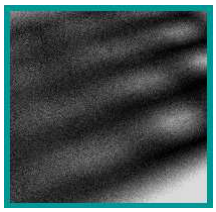
(a) *our method, 27 samples per pixel (spp)*



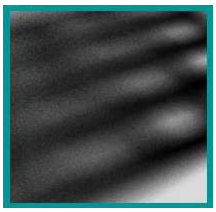
(b) *error vs. spp, our method*



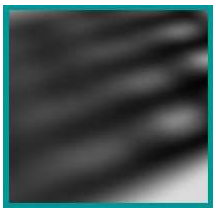
(c) *loglog plot of RMS error vs. spp for various schemes*



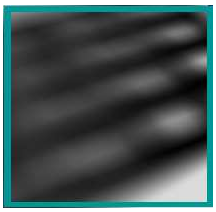
(d) *equal samp., 27 spp, NAS, UF*



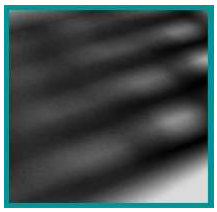
(e) *equal samp., 27 spp, AS, UF*



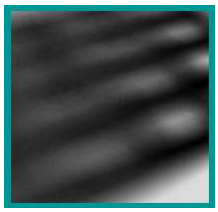
(f) *our method, 27 spp, AS, AF*



(g) *ground truth*



(h) *equal error, 63 spp, AS, UF*



(i) *equal error, 165 spp, NAS, UF*

Figure 6: *(a) The 'Grids' scene, (b) Difference images (scaled up 20×) show that our method converges to ground truth with increasing sampling rate, (c) The RMS error (with respect to ground truth) versus sampling rate. These plots show that our method is consistent, and that we obtain the best results by combining adaptive sampling and adaptive filtering. (d) through (i) show equal sample and equal RMS error comparisons for adaptive vs. non-adaptive sampling and filtering vs. no filtering. Note that our method produces visually higher quality results than the somewhat noisy equal error comparisons. (Readers are encouraged to zoom into the PDF to inspect the images and noise).*

5.3 Evaluation and Discussion

Figure 6 evaluates our method on the grids scene (taken from [Egan et al. 2011b]). While the geometry is simple, the shadows are intricate. Figure 6(a) shows that our method produces smooth results without artifacts, while Fig. 6(b) shows scaled-up error images with respect to ground truth. We see that the errors are small even numerically, and decrease rapidly with increasing sample count. Our algorithm converges in the limit, because we adjust the filter size to the number of samples. The graph in Fig. 6(c) plots numerical error vs the number of samples n for various combinations of sampling and filter. The minimum number of samples in these graphs is 9, which is our initial sampling pass (as described later in Sec. 6).

First, consider unfiltered (UF) results, both with standard Monte Carlo (non-adaptive sampling NAS) and adaptive sampling AS. We clearly see that our adaptive sampling method significantly reduces error. We can also run NAS and AS with a fixed non-adaptive filter (NAF), with the filter width chosen to best match the final image. A fixed-width filter cannot capture the complexity of the shadows however, and the error of the red and yellow curves remains flat and does not converge with increasing numbers of samples. Finally, consider the bottom two curves, where we apply our adaptive filtering (AF) from Sec. 4. The error is considerably reduced. Adaptive Sampling (AS) in addition to adaptive filtering further reduces error, though more modestly than without filtering. For a fixed error, our method (dark blue curve, bottom AS, AF) reduces the sample count by 6×, compared to standard unfiltered Monte Carlo (magenta curve, top NAS, UF). With only 27 samples, we are able to produce high quality renderings that closely match ground truth; more samples improve the results even further (Fig. 6(b)).

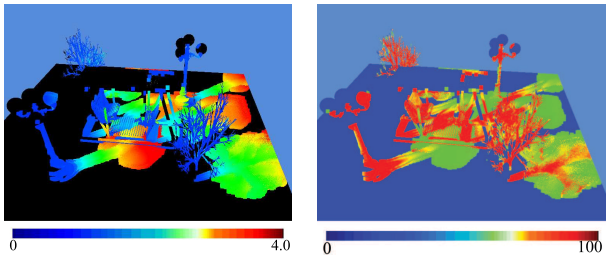
Finally, we show equal sample and equal error images in the bottom row of Fig. 6. Our method has almost no noise, and is visually superior even to equal error comparisons that exhibit some noise.

6 Implementation

We implemented our method within OptiX, on an NVIDIA GTX 570 GPU. Scenes used Wavefront OBJ files and the BVH acceleration structure built into Optix. Therefore, our code only needs to implement the adaptive sampling and filtering steps. For all scenes, the parameters used in equations 13 and 21 were $k = 3$, $\alpha = 1$ and $\mu = 2$. The full source code is available from: `http://graphics.berkeley.edu/papers/UdayMehta-AAF-2012-12`

Sparse Sampling for Filter Size and Samples Per Pixel: In the first stage, we use a sparse sampling of 9 rays (stratified over the light source) at each pixel, from which we compute d_1 and d_2. Since the sampling is coarse, we can sometimes observe noise in these estimates (leading to some noise visible in animations in our video).[5] Most of the noise in the resulting filter width calculation

[5] Some occluders may also be missed with only 9 rays leading to inaccurate estimates of filter sizes; however, we are interested only in overall depth



(a) *'Bench', scale* β **(b)** *'Bench', spp* n

(c) *'Grids', scale* β **(d)** *'Grids', spp* n

Figure 7: *Visualization of filter width parameter* β *for bench (a) and grids (c) and samples per pixel* n *(b) and (d). Unoccluded pixels, shown in black, are not filtered. The unoccluded pixels have the minimum sample count of 9 from the first pass.*

arises from completely unoccluded pixels. Therefore, for these unoccluded pixels, we store the average values of d_1 and d_2 in a 5 pixel radius. We then compute the filter width β from equation 13, visualized in Fig. 7(a,c). Notice how the filter width is smaller in more complex shadow regions, such as those with close occluders. Finally, we compute the number of samples at each pixel n using equation 21, visualized in Fig. 7(b,d).[6] Notice how more samples are needed in regions with smaller filter width.

Adaptive Sampling and Filtering: We now cast n rays per pixel (including the original 9 rays); each pixel can have a different n. We may thus obtain the noisy visibility $\bar{h}$ (such as Fig. 6(e)). We then filter to obtain the final image h per equation 10 (such as Fig. 6(a,f)).

Screen-Space Adaptive Filter: A practical challenge in adaptive filtering is that β corresponds to object-space and is derived for a single spatial dimension x. But we need to develop an efficient 2D screen-space filter. We utilize the world-space distances between objects to compute the filter weights in equation 11 using a depth buffer. Our practical system also uses a check on normal variation to avoid filtering different objects or regions. (Spatial differences and depth discontinuities are handled automatically by the Gaussian filter in equation 11. If available, a per-pixel object ID check may also be used,[7] but is not required.)

Finally, greater efficiency can be achieved if we can use two 1D separable filters along the image dimensions. To do so, we write equation 11 as $w(x_{ij} - x_{kl}) = w(x_{ij} - x_{kj})w(x_{kj} - x_{kl})$, where ij and kl are pixel coordinates. This is a standard separation of the 2D distance metric along the individual coordinates and is common for gaussian convolutions. In our case involving spatially-varying filters, it is exact if the filter width β is the same within the pixels of interest (ij and kj). In practice, it is a good approximation since β varies slowly, and we found almost no observable differences between the 2D filter and our two separable 1D filters in practice.

Discussion: Our final implementation is entirely in screen space (with the additional information of a depth buffer to compute world-space distances). Unlike previous work, we do not need to store each ray sample individually, but rather operate directly on the integrated (noisy) occlusion values at each pixel. This enables a very much smaller memory footprint. Moreover, our linearly separable adaptive filter has an algorithmic complexity essentially equal to that of a typical gaussian blur, making the method very efficient.

7 Results

We tested our method on four scenes of varying complexity, including some used in previous papers [Overbeck et al. 2006; Egan et al. 2011b]. Besides moving viewpoint and light source, our method supports fully dynamic geometry since no precomputation is required. We show examples of animations and real-time screen captures in the accompanying video.

Sampling Rate and Timings: Table 1 has details on the performance for our scenes. In all cases, our theory predicted an average sampling rate of between 14 and 34 samples. Comparable images with brute force raytracing typically required at least 150-200 samples. The total overhead added by our algorithm averaged under 5 milliseconds, of which adaptive filtering took about 1.3 ms, and the time for determining the filter size and samples per pixel took about 3.6 ms. The timings for the base OptiX raytracer are scene-dependent but substantially larger in all cases.

Note that our filter operates in image-space, and is therefore insensitive to the geometric complexity of the scene; the memory requirements are also small. Our 5ms filtering time is 3 orders of magnitude faster than the reconstruction reported by [Lehtinen et al. 2011] and 4-5 orders of magnitude faster than the minutes of overhead in [Egan et al. 2011b]. This substantial speedup allows our method to be used in the context of a real-time raytracer, which has not previously been possible. We are able to achieve interactive performance—the simpler grids scene renders at over 30 frames per second, while we achieve a performance of 2-4 fps on the other scenes, which have between 72K and 309K vertices. This is an order of magnitude faster than what brute-force OptiX raytracing can achieve, and enables raytraced soft shadows at interactive rates.

Accuracy: Our scenes are chosen to show a variety of soft shadow effects. First, Fig. 8 shows that we can accurately capture curved objects casting shadows on other curved objects, and that an equal sample standard raytrace is considerably noisier. We have already evaluated the grids scene, which has soft shadows of varying sizes, in Fig. 6. Our method does not under or overblur, and reproduces accurate soft shadows across the entire image. Figure 1 shows intricate shadows cast by fairly complex geometry, and also includes visual equal time and equal error comparisons with standard Monte Carlo raytracing. It is clear that the noise is considerably reduced, and our images match closely with ground truth. Finally, Fig. 9 shows shadows from thin occluders being cast on a wavy ground plane. While the shadows form complex patterns, our method produces an accurate result. Moreover, our results are visually better than even the equal error basic comparisons, that still have noise, even with $4\times$ to $9\times$ more samples.

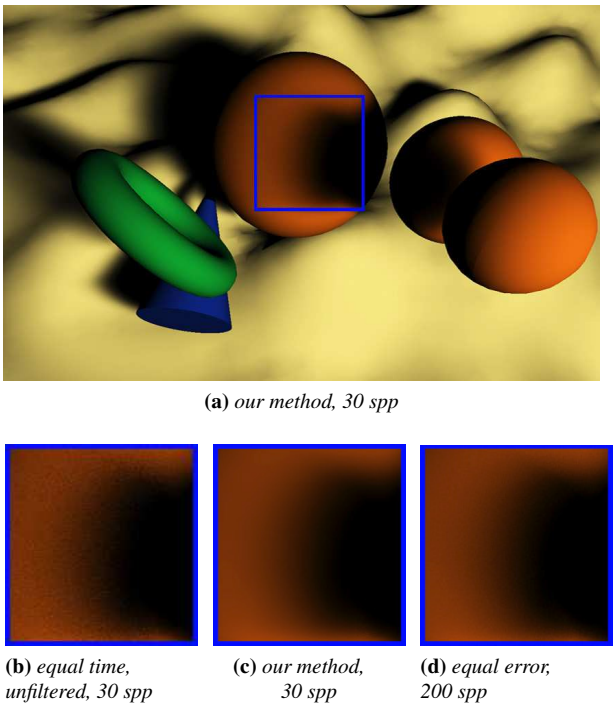
ranges. Note that this concern is shared by almost all adaptive methods. Our results include self-shadowing, bumpy occluders and receivers, and contact shadows, showing that the initial sparse sampling is robust.

[6] Unlike some previous works, β and n always come directly from our equations (our implementation includes appropriate $\min$ and $\max$ bandlimits where needed), with no need to identify special-case pixels that need brute force Monte Carlo, as in [Egan et al. 2011b].

[7] In the unusual case that a pixel is surrounded entirely by objects of different IDs, we will not filter and provide the standard Monte Carlo result.

			Base Raytraced Occlusion			Our Algorithm			Total	
scene	vertices	avg. spp	1st pass (ms)	2nd pass (ms)	total raytracing(ms)	compute β and n (ms)	adaptive filtering (ms)	total overhead (ms)	total time (ms)	fps (**with**/ without alg)
Grids	0.2 K	14.2	6.56	13.8	20.4	3.70	1.31	**5.01**	25.4	**39** / 49
Bench	309 K	28.0	50.9	374	425	3.51	1.27	**4.78**	430	**2.3** / 2.3
Tentacles	145 K	26.3	49.0	239	288	3.50	1.29	**4.79**	293	**3.4** / 3.5
Spheres	72 K	33.8	56.9	285	342	3.70	1.29	**4.99**	347	**2.9** / 2.9

Table 1: *Timings of our scenes rendered at* 640×480*. The last column shows that our precomputation and filtering overheads (a total of about 5ms) have negligible impact on the total rendering time or frames per second.*



(a) *our method, 30 spp*

(b) *equal time, unfiltered, 30 spp*

(c) *our method, 30 spp*

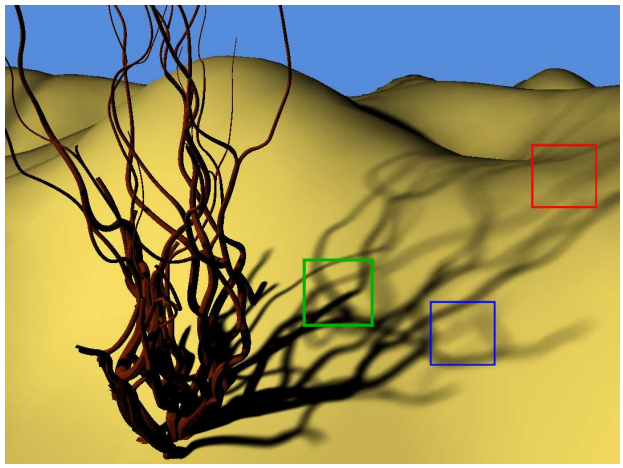
(d) *equal error, 200 spp*

Figure 8: *Interactive soft shadows from a curved occluder onto a curved receiver, with an average of 30 samples per pixel.*

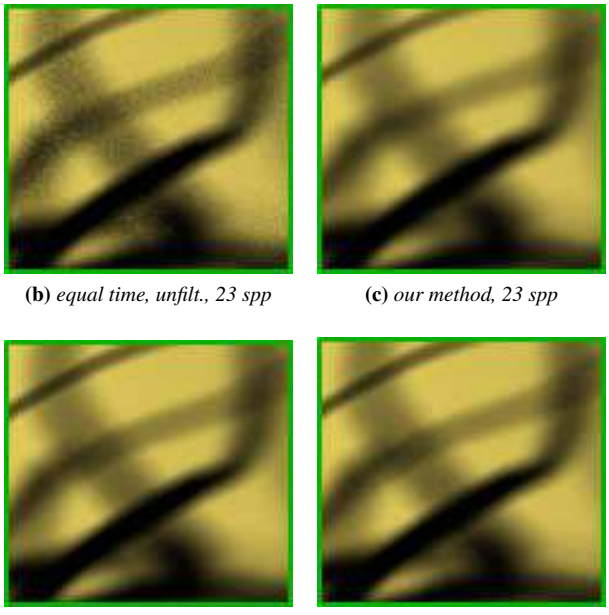
Note that we have only compared with standard Monte Carlo (the base OptiX raytracer), since our technique differs fundamentally from previous rendering methods—we are trying to achieve ray-traced image quality at close to real-time framerates. Our goal is accurate raytraced occlusion, as opposed to most previous approximations for real-time soft shadows. Moreover, we are an (many) order of magnitude faster than most prior work on offline soft shadows, since we build on a GPU-accelerated raytracer. In the future, further speedups could potentially be achieved by also GPU-accelerating the visibility sampling [Sintorn and Assarsson 2008].

Comparison to De-Noising: Our final algorithm is essentially a denoising filter on the base Monte Carlo image. Therefore, we include a comparison with image de-noising strategies in Fig. 10. The fixed-width gaussian and bilateral filters (with parameters chosen by trial and error to produce the best visual results) cannot achieve the accuracy of our adaptive filter. In particular, a fixed width gaussian overblurs or underblurs; the latter causes noise to remain as seen in the insets. Moreover, the fixed width gaussian is only slightly faster than our method, while the bilateral filter is considerably slower (although further optimizations are possible).

The bm3d algorithm of [Dabov et al. 2007] is more successful (although without additional information, it can blur across object boundaries). However, denoising comes at the price of some blocky artifacts, as seen in Fig. 10. Moreover, that method (like most other



(a) *our method, 23 spp*



(b) *equal time, unfilt., 23 spp*

(c) *our method, 23 spp*

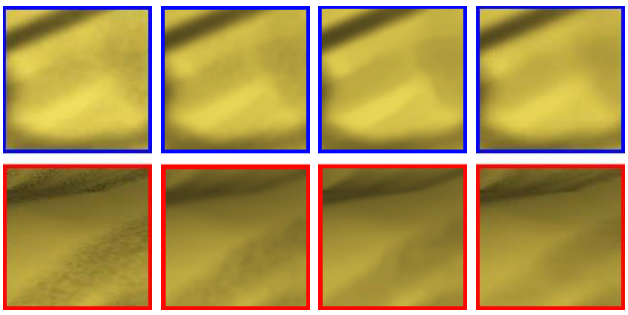
(d) *ground truth, 5000 spp*

(e) *equal error, 200 spp*

Figure 9: *Soft shadows for the 'tentacles' scene: (a) and (c) Our method, 23 spp (b) equal samples, no filtering, 23 spp (d) ground truth (e) equal RMS error for unfiltered 200 spp still has some noise.*

previous works) is not designed for efficiency, taking 8 seconds.

Extensions: Initial tests indicate the method extends in practice to cosine-falloff shading and non-Gaussian lights, without any modifications to the adaptive sampling or filtering steps. In Fig. 11(a), we show the bench scene rendered with cosine falloff (the scene is shown in grayscale to avoid masking subtle shading



(a) *fixed blur, 0.003 sec* **(b)** *bilateral filter, 200 sec* **(c)** *bm3d, 8 sec* **(d)** *our method, 0.005 sec*

Figure 10: *Comparison of accuracy and overheads of denoising methods for the 'tentacles' scene, all using the same base 23 samples per pixel unfiltered image. (a) filtering with a constant kernel gaussian takes slightly less time but overblurs contact shadow edges while retaining noise in low-frequency shadows, (b) bilateral filtering cannot remove noise effectively in slow-varying shadows, and is computationally more expensive, (c) bm3d performs quite well, but produces low frequency artifacts, and takes longer, (d) our method performs best while being significantly faster.*

effects). In Fig. 11(b) we use a more challenging grids scene (where they also shadow each other) and a uniform light. Although light band-limits no longer strictly apply, and we do slightly overblur the shadows as expected, the method still performs well. However, we did need to use more conservative settings $\mu = 3$ in both examples.

Artifacts and Limitations: One of the main benefits of Monte Carlo rendering is that artifacts (noise at low sample counts) are well understood and easily reduced (use more samples). A benefit of our method is that it shares many of these properties: the main limitation is slight overblur at some sharp shadow boundaries, and some flicker in video sequences (as in standard Monte Carlo), both of which are easily addressed by more samples (higher μ). We have not undertaken a full comparison to alternative rasterization and GPU-based soft shadow methods; it is possible that for some scenes, they may produce comparable results to our system with smaller μ. However, our method is consistent (Fig. 6), converging rapidly to ground truth. The framerates in table 1 are at least an order of magnitude faster than most previous work on accurate shadow raytracing. Since we introduce minimal overhead, that is the performance of the base OptiX raytracer. However, our sample counts per pixel (an average of about $n = 25$), while low, are still too high for real-time performance on very complex scenes; nevertheless we do demonstrate interactive sessions in the video.

8 Conclusions and Future Work

A large body of recent work has indicated that the number of rays in Monte Carlo rendering can be dramatically reduced if we share samples between pixels. However, methods based on sheared filtering or light field reconstruction have included an expensive post-processing step. In this paper, we make a different set of trade-offs. We use axis-aligned filters to reduce the number of Monte Carlo samples considerably compared to brute force, but less than using sheared filtering. However, we benefit in having an extremely simple filtering step, that just reduces to a spatially-varying image-space blur. This can be performed extremely fast, and enables our method to be integrated in a GPU-accelerated raytracer. The final algorithm is essentially an image de-noiser which is very simple to implement and can operate with minimal overhead. Therefore, we are able to achieve interactive frame rates while obtaining the benefits of high quality ray-traced soft shadows. A key contribution of our work is to precisely identify the necessary sampling rates, and show how to adjust the filter with the sampling rate, to enable convergence. Of course, soft shadows are only one effect in real-time rendering and we believe similar methods could be employed for motion blur, depth of field, specularities and indirect illumination.

Acknowledgements

We thank the reviewers for their helpful suggestions. This work was supported in part by NSF grant CGV #1115242 and the Intel Science and Technology Center for Visual Computing. We acknowledge equipment and funding from Adobe, NVIDIA and Pixar.

References

AGRAWALA, M., RAMAMOORTHI, R., HEIRICH, A., AND MOLL, L. 2000. Efficient Image-Based Methods for Rendering Soft Shadows. In *SIGGRAPH 2000*, 375–384.

ANNEN, T., DONG, Z., MERTENS, T., BEKAERT, P., AND SEIDEL, H. 2008. Real-time all-frequency shadows in dynamic scenes. *ACM Transactions on Graphics (SIGGRAPH 08) 27*, 3, Article 34, 1–8.

ARIKAN, O., FORSYTH, D., AND O'BRIEN, J. 2005. Fast and detailed approximate global illumination by irradiance decomposition. *ACM Transactions on Graphics (SIGGRAPH 05) 24*, 3, 1108–1114.

ASSARSSON, U., AND MÖLLER, T. 2003. A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics (SIGGRAPH 03) 22*, 3, 511–520.

BENTHIN, C., AND WALD, I. 2009. Efficient ray traced soft shadows using multi-frusta tracing. In *High Performance Graphics 2009*, 135–144.

CHAI, J., CHAN, S., SHUM, H., AND TONG, X. 2000. Plenoptic Sampling. In *SIGGRAPH 00*, 307–318.

CROW, F. 1977. Shadow algorithms for computer graphics. In *SIGGRAPH 77*, 242–248.

DABOV, K., FOI, A., KATKOVNIK, V., AND EGIAZARIAN, K. 2007. Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Transactions on Image Processing 16*, 8, 2080–2095.

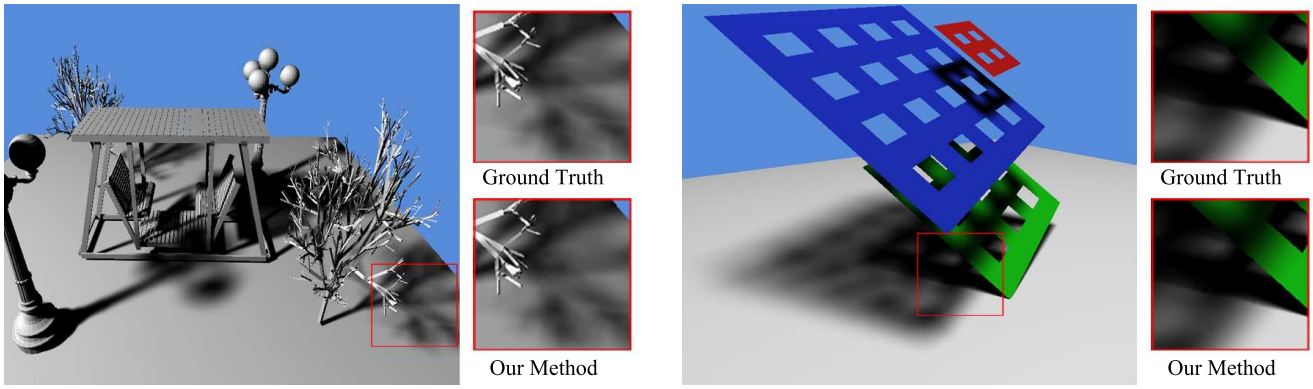
DURAND, F., HOLZSCHUCH, N., SOLER, C., CHAN, E., AND SILLION, F. 2005. A Frequency Analysis of Light Transport. *ACM Transactions on Graphics (Proc. SIGGRAPH 05) 25*, 3, 1115–1126.

DURAND, F. 2011. A frequency analysis of monte-carlo and other numerical integration schemes. Tech. Rep. MIT-CSAIL-TR-2011-052 http://hdl.handle.net/1721.1/67677, MIT CSAIL.

EGAN, K., TSENG, Y., HOLZSCHUCH, N., DURAND, F., AND RAMAMOORTHI, R. 2009. Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Transactions on Graphics (SIGGRAPH 09) 28*, 3.

EGAN, K., DURAND, F., AND RAMAMOORTHI, R. 2011. Practical filtering for efficient ray-traced directional occlusion. *ACM Transactions on Graphics (SIGGRAPH ASIA 09) 30*, 6.

EGAN, K., HECHT, F., DURAND, F., AND RAMAMOORTHI, R. 2011. Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Transactions on Graphics 30*, 2.



(a) *visibility filtered with BRDF cosine term, 50 spp*

(b) *uniform intensity square light, 30 spp*

Figure 11: *Extensions of our method to filtering visibility combined with the diffuse BRDF cosine term, for the 'Bench' scene in (a); and to non-Gaussian (uniform intensity) square lights for a modified 'Grids' scene in (b). We can handle these cases well with a slightly higher sampling rate ($\mu = 3$). In contrast, Ground Truth requires approximately 8,000 samples per pixel in (a) and 4,000 spp in (b).*

EISEMANN, E., AND DECORET, X. 2007. Visibility sampling on GPU and applications. *Computer Graphics Forum (EG 07) 26*, 3, 535–544.

FOREST, V., BARTHE, L., AND PAULIN, M. 2008. Accurate shadows by depth complexity sampling. *Computer Graphics Forum 27*, 2, 663–674.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2006. Real-time soft shadow mapping by backprojection. In *EGSR 06*, 227–234.

GUO, B. 1998. Progressive radiance evaluation using directional coherence maps. In *SIGGRAPH 98*, 255–266.

HACHISUKA, T., JAROSZ, W., WEISTROFFER, R., DALE, K., HUMPHREYS, G., ZWICKER, M., AND JENSEN, H. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Transactions on Graphics (SIGGRAPH 08) 27*, 3.

HART, D., DUTRÉ, P., AND GREENBERG, D. 1999. Direct illumination with lazy visibility evaluation. In *SIGGRAPH 99*, 147–154.

HASENFRATZ, J., LAPIERRE, M., HOLZSCHUCH, N., AND SILLION, F. 2003. A survey of real-time soft shadow algorithms. *Computer Graphics Forum 22*, 4, 753–774.

JOHNSON, G., HUNT, W., HUX, A., MARK, W., BURNS, C., AND JUNKINS, S. 2009. Soft irregular shadow mapping: fast, high-quality, and robust soft shadows. In *I3D 2009*, 57–66.

LACEWELL, D., BURLEY, B., BOULOS, S., AND SHIRLEY, P. 2008. Raytracing prefiltered occlusion for aggregate geometry. In *IEEE Symposium on Interactive Raytracing 08*.

LAINE, S., AILA, T., ASSARSSON, U., LEHTINEN, J., AND MÖLLER, T. 2005. Soft shadow volumes for ray tracing. *ACM Transactions on Graphics (SIGGRAPH 05) 24*, 3, 1156–1165.

LANMAN, D., RASKAR, R., AGRAWAL, A., AND TAUBIN, G. 2008. Shield fields: modeling and capturing 3D occluders. *ACM Transactions on Graphics (SIGGRAPH ASIA 08) 27*, 5.

LEHTINEN, J., AILA, T., CHEN, J., LAINE, S., AND DURAND, F. 2011. Temporal light field reconstruction for rendering distribution effects. *ACM Transactions on Graphics 30*, 4.

MCCOOL, M. 1999. Anisotropic diffusion for monte carlo noise reduction. *ACM Transactions on Graphics 18*, 2, 171–194.

OVERBECK, R., BEN-ARTZI, A., RAMAMOORTHI, R., AND GRINSPUN, E. 2006. Exploiting Temporal Coherence for Incremental All-Frequency Relighting. In *EuroGraphics Symposium on Rendering*, 151–160.

OVERBECK, R., RAMAMOORTHI, R., AND MARK, W. 2007. A real-time beam tracer with application to exact soft shadows. In *EGSR 07*, 85–98.

OVERBECK, R., DONNER, C., AND RAMAMOORTHI, R. 2009. Adaptive Wavelet Rendering. *ACM Transactions on Graphics (SIGGRAPH ASIA 09) 28*, 5.

PARKER, S., BIGLER, J., DIETRICH, A., FRIEDRICH, H., HOBEROCK, J., LUEBKE, D., MCALLISTER, D., MCGUIRE, M., MORLEY, K., ROBISON, A., AND STICH, M. 2010. OptiX: A general purpose ray tracing engine. *ACM Transactions on Graphics (SIGGRAPH 10) 29*, 4, 66:1–66:13.

RUSHMEIER, H., AND WARD, G. 1994. Energy preserving non-linear filters. In *SIGGRAPH 94*, 131–138.

SEN, P., AND DARABI, S. 2012. On filtering the noise from the random parameters in monte carlo rendering. *ACM Transactions on Graphics 31*, 3.

SINTORN, E., AND ASSARSSON, U. 2008. Sample based visibility for soft shadows using alias-free shadow maps. *Computer Graphics Forum (EGSR 08) 27*, 4, 1285–1292.

SLOAN, P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Transactions on Graphics (SIGGRAPH 02) 21*, 3, 527–536.

SOLER, C., AND SILLION, F. 1998. Fast Calculation of Soft Shadow Textures Using Convolution. In *SIGGRAPH 98*, 321–332.

SOLER, C., SUBR, K., DURAND, F., HOLZSCHUCH, N., AND SILLION, F. 2009. Fourier depth of field. *ACM Transactions on Graphics 28*, 2.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *SIGGRAPH 78*, 270–274.

XU, R., AND PATTANAIK, S. 2005. A novel monte carlo noise reduction operator. *IEEE Computer Graphics and Applications 25*, 2, 31–35.