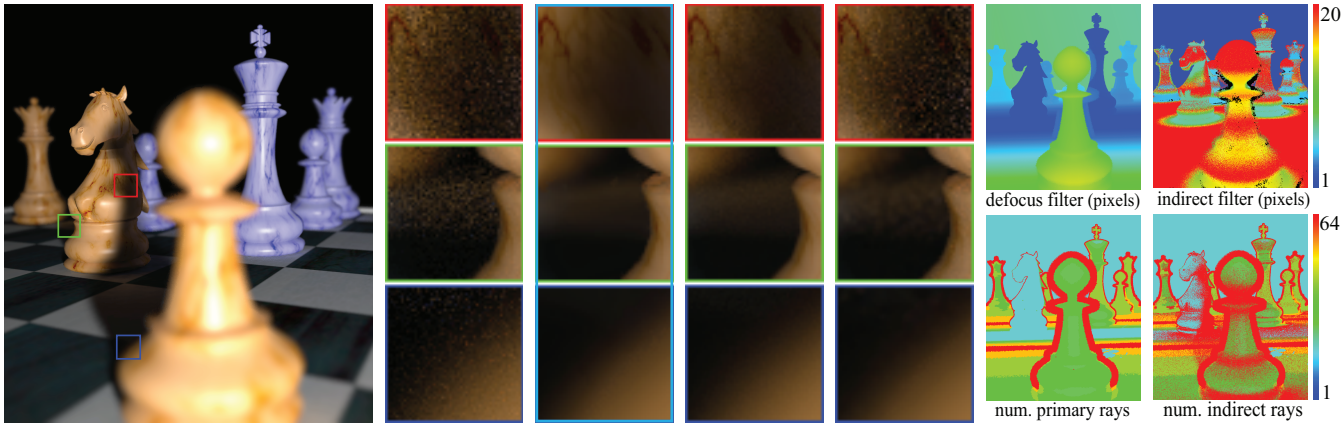
Factored Axis-Aligned Filtering for Rendering Multiple Distribution Effects

Soham Uday Mehta[1] JiaXian Yao[1] Ravi Ramamoorthi[1] Fredo Durand[2]

[1]University of California, Berkeley [2]MIT CSAIL



(a) Our Method, 138 rpp, 3.14 sec **(b)** Equal time, 176 rpp, 3.11 sec **(c)** Our method, 138 rpp, 3.14 sec **(d)** Eq. quality, 5390 rpp, 130 sec **(e)** No factoring, 138 rpp, 3.10 sec **(f)** Primary filter and rpp **(g)** Indirect filter and rpp

Figure 1: *(a) The CHESS scene, with defocus blur, area light direct and indirect illumination, rendered at 900×1024 with an average 138 atomic rays per pixel (rpp), in 3.14 sec on an NVIDIA Titan GPU. We compare to different methods in the insets. Readers are encouraged to zoom into the PDF to examine the noise. The top row inset is a sharp in-focus region while the other two regions are defocus blurred; all insets include noisy direct and indirect illumination. In (b) we compare to equal time stratified Monte Carlo (MC) sampling with 176 rpp; in (c), Our method; and in (d), Equal quality MC with 5390 rpp is 40× slower. One of the key contributions of our method is factoring of texture and irradiance, so that irradiance can be pre-filtered before combining with texture. Without factoring, the defocus filter cannot remove noise for in-focus regions as shown in (e), top inset. In (f) and (g) top row, we show filter size for texture and indirect irradiance. Black pixels indicate that factoring cannot be used and irradiance cannot be pre-filtered. In the bottom row we show number of primary rays and indirect samples respectively. Our method uses separate sampling rates and filters for primary and secondary effects which makes it more effective.*

Abstract

Monte Carlo (MC) ray-tracing for photo-realistic rendering often requires hours to render a single image due to the large sampling rates needed for convergence. Previous methods have attempted to filter sparsely sampled MC renders but these methods have high reconstruction overheads. Recent work has shown fast performance for individual effects, like soft shadows and indirect illumination, using axis-aligned filtering. While some components of light transport such as indirect or area illumination are smooth, they are often multiplied by high-frequency components such as texture, which prevents their sparse sampling and reconstruction.

We propose an approach to adaptively sample and filter for simultaneously rendering primary (defocus blur) and secondary (soft shadows and indirect illumination) distribution effects, based on a multi-dimensional frequency analysis of the direct and indirect illumination light fields. We describe a novel approach of factoring texture and irradiance in the presence of defocus blur, which allows for pre-filtering noisy irradiance when the texture is not noisy. Our approach naturally allows for different sampling rates for primary and secondary effects, further reducing the overall ray count. While the theory considers only Lambertian surfaces, we obtain promising results for moderately glossy surfaces. We demonstrate 30× sampling rate reduction compared to equal quality noise-free MC. Combined with a GPU implementation and low filtering overhead, we can render scenes with complex geometry and diffuse and glossy BRDFs in a few seconds.

CR Categories: Computing Methodologies [Computer Graphics]: Rendering—Ray Tracing

Keywords: sampling, filtering, diffuse, global illumination

Links: DL PDF WEB VIDEO

1 Introduction

Monte Carlo distribution ray-tracing is an accurate way to render multiple distribution effects such as depth-of-field, soft shadows in direct illumination, and indirect (global) illumination. But the convergence to a noise-free image is slow, often requiring over a thousand rays per pixel. Fortunately, there is significant coherence in the color between pixels, making adaptive sampling and filtering an obvious solution to reducing the ray-tracing cost. However, these existing methods are usually memory intensive and have high reconstruction overheads, even though they reduce the number of samples in ray-tracing by one to two orders of magnitude.

The recent method of sheared filtering ([Egan et al. 2009; Egan et al. 2011]) utilized Fourier-domain sparsity of motion-blurred or visibility light fields, to reduce sampling rate and filter in light-field space. To reduce the filtering overhead, more recent

work on axis-aligned[1] filtering ([Mehta et al. 2012; Mehta et al. 2013]) for soft shadows and indirect illumination proposes a method for adaptive sampling and image-space filtering that has very low overhead and makes fast render-times possible. But these methods are limited to rendering single effects rather than full distribution ray-tracing with multiple effects, and extending them to a higher-dimensional rendering domain is a challenge. Specifically, coupling between the irradiance and texture components of light transport (due to motion or defocus blur) hinders sparse sampling and reconstruction in proportion to the bandwidth of each individual component.

In this paper, we provide an axis-aligned method for image-space filtering that can handle a combination of different effects, namely defocus blur (primary), soft shadows and indirect illumination (secondary). We do not consider motion blur in the main paper since our GPU ray-tracer doesn't support it, but we provide a proof-of-concept description with results, in the Appendix. We derive a multi-dimensional end-to-end frequency analysis that handles a combination of effects. Our analysis differs from previous works in that it provides filtering bandwidths separately for both the noisy texture and illumination using a simple geometric approach. We introduce factoring of the radiance integral to more efficiently sample and filter each component. Previous methods either (i) filter the full high-dimensional light field [Lehtinen et al. 2011; Lehtinen et al. 2012], resulting in large overhead, or (ii) use expensive atomic computation [Belcour et al. 2013] for each ray interaction to compute the overall bandwidth for the noisy pixel color without factoring texture and irradiance. Our primary contributions are:

Combined frequency analysis for primary and secondary effects: Our main theoretical contibution is the geometric and Fourier analysis of both direct and indirect illumination under lens (defocus) blur for diffuse surfaces. We extend the flatland 2D Fourier analysis of [Egan et al. 2011; Mehta et al. 2013], to the 3D light-field in a position-lens-light space for direct or position-lens-angle space for indirect. We derive how the light field is bandlimited due to integration over the lens, light, and angle. Our approach is end-to-end unlike the atomic operations of [Durand et al. 2005; Belcour et al. 2013]. This makes our parameters easier to evaluate, without requiring complex frequency analysis along light paths.

Factoring texture and irradiance: Our main practical contribution is a method to approximate the integral of color (radiance) at each pixel as a product of texture and irradiance integrals. In the absence of defocus blur and spatial anti-aliasing, the primary hit is a single location per-pixel, making factorization trivial, as assumed by previous irradiance filtering (e.g. irradiance caching) methods. However, due to defocus blur, the texture and irradiance are coupled. We propose to use the texture-irradiance factoring approximation when the error is below a threshold. For example, an image region with a high-frequency texture (Fig. 1(e) top row) cannot be filtered without factorization, since we do not want to blur the texture. Hence, a large number of secondary rays would previously need to be traced to reduce the noise in soft shadows and indirect illumination.

Two-level adaptive sampling strategy: Our method is based solely on ray-tracing for all rendering effects. Instead of naively path tracing each pixel, we allocate rays to primary and secondary effects in proportion to their frequency content while maintaining physical correctness, as depicted in Fig.1(f,g) lower row. For example, at an in-focus pixel with soft shadows we allocate a single lens ray but multiple light shadow rays. At an out-of-focus pixel with no soft shadows, we have an adequate number of primary rays, and a single shadow ray per primary ray. Similarly, we predict the sampling rate for indirect illumination taking into account both the local illumination frequency, and defocus. Previous adaptive sampling methods like [Belcour et al. 2013] only provide a single sampling rate at each pixel and hence are inefficient in reducing ray-tracing cost.

We have integrated our method into NVIDIA's Optix ray-tracer [Parker 2010] and implemented our reconstruction method on the GPU. We achieve about $4\times$ ray count reduction compared to equal RMS error MC (quantitative) and about $30\times$ compared to equal visual quality ground truth MC (qualitative). Figure 12 emphasizes this point. Our method has a low reconstruction overhead of under 200 ms, and can be easily combined with a GPU ray-tracer. We demonstrate render times of 3-10 seconds[2] for a variety of scenes.

2 Previous Work

[Cook et al. 1984] and [Kajiya 1986] first introduced Monte Carlo distribution ray and path tracing for evaluating pixel radiance using the rendering equation. Building on the basic framework of physically based rendering, we use adaptive sampling and filtering to efficiently produce physically-based renderings with depth of field, and area light direct and indirect illumination.

Image Filtering and Noise-guided Reconstruction: Image filtering has been a popular approach to remove noise in images generated by MC ray-tracing, because of its simplicity and efficiency. Geometric information such as normals, textures, and depths, can play an important role for predicting noise in rendered images. Methods that use a denoising approach based on noise estimation from geometric parameters include the use of the cross bilateral filter [Ritschel et al. 2009], the A-Trous wavelet transform [Dammertz et al. 2010], adaptive wavelet rendering [Overbeck et al. 2009] and the filtering of stochastic buffers [Shirley et al. 2011]. Other examples are random parameter filtering (RPF, [Sen and Darabi 2012]) which has a high computation and storage cost, [Li et al. 2012] that uses Steins unbiased risk estimator for sampling and bandwidth selection for anisotropic filters, and [Rouselle et al. 2012] which uses non-local means filtering and residual errors to guide sampling density. [Kalantari and Sen 2013] give a noise estimation metric to locally identify the amount of noise in different parts of the image, and adaptively sample and filter using standard denoising techniques. Another novel approach is that of [Sen et al. 2011] which uses compressed sensing. [Delbracio et al. 2014] use ray color histograms to guide filtering. These methods do not exploit the geometric and Fourier structure of the light field and require either high sampling rates or high reconstruction overheads. Our method is also different from 2D post-processing solutions [Max and Lerner 1985; Potmesil and Chakravarty 1981] that blur a 2 or 2.5D image with spatially-varying filters according to depth or motion, since we filter an image obtained by accurate ray and path tracing. Further, using a primary filter given by the circle of confusion or motion vector does not filter noisy secondary effects.

Light Field Analysis: These methods attempt to reconstruct the full high-dimensional light field. Multi-dimensional adaptive sampling and reconstruction [Hachisuka et al. 2008] improves upon [Mitchell 1991], and uses sample contrast to guide adaptive sampling, followed by anisotropic reconstruction of the light field. [Lehtinen et al. 2011] and [Lehtinen et al. 2012] proposed a reconstruction method for motion and defocus blur from sparse sampling of the 3D/5D (spatial position, lens and time) light field that uses velocity and depth information to reproject samples into each pixel, but with a high memory and computation overhead.

Fourier-guided Adaptive Sampling and Filtering: We build on recent approaches that have studied the frequency aspects of light

[1] Axis-aligned refers to the Fourier light-field rather than image-space, although the image-space filters as implemented are also axis-aligned.

[2] We are slower than [Mehta et al. 2013] since we need more rays for depth of field, and our images are at higher resolution.

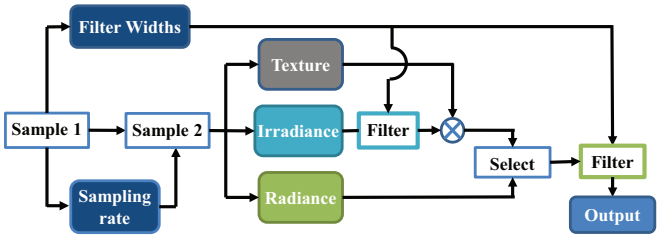


Figure 2: *A simplified flow chart of the algorithm. In two sampling (ray-tracing) passes, we adaptively sample the noisy per-pixel texture, irradiance and radiance. These are then filtered and combined to produce an accurate radiance value.*

transport in static scenes, e.g. [Chai et al. 2000; Ramamoorthi and Hanrahan 2001; Durand et al. 2005]. They presented an a-priori frequency analysis to perform adaptive sampling and appropriate reconstruction. [Egan et al. 2009; Egan et al. 2011] applied such an analysis and sheared reconstruction to motion-blurred images and soft shadows utilizing the space-time and space-light Fourier spectra respectively. [Soler et al. 2009] proposed to adaptively sample primary rays by predicting image bandwidth and per-pixel variance of incoming light, to efficiently ray-trace and reconstruct images with depth of field. They used a sampled representation of the spectrum which is expensive and prone to noise. 5D covariance tracing [Belcour et al. 2013] uses a covariance representation that is compact and stable, addressing the full 5D (space-angle-time) light field, and focuses on atomic operations to achieve generality. We instead use an image-space axis-aligned filter similar to [Mehta et al. 2012; Mehta et al. 2013] combined with adaptive sampling for both primary and secondary effects in a single framework.

Our method is simpler and faster than covariance tracing and generalizes axis-aligned filtering to combine primary and secondary effects. Moreover, our texture-irradiance factorization improves on methods that only filter the final pixel radiance, without filtering irradiance. Those methods are inefficient for in-focus regions. Methods that retain the full light field (individual samples) such as [Lehtinen et al. 2011; Lehtinen et al. 2012; Sen and Darabi 2012] can filter the noisy irradiance separately but have a high storage and reconstruction overhead. The concurrent work of [Vaidyanathan et al. 2014] demonstrates a much faster, interactive formulation of sheared filtering for depth of field. They separate the image into depth layers, and simplify the 4D filter into splatting and screen-space convolution steps.

3 Overview

We describe our algorithm in brief to motivate the theoretical analysis and highlight the main contributions. A block diagram is shown in Fig 2. We first sparsely path trace each pixel to identify frequency bandwidths for each of the effects under consideration, namely the defocus blur, area light direct illumination and indirect illumination. Through this sparse sampling, we can predict the local Fourier structure of the high-dimensional light field for both direct and indirect illumination. Sections 4-6 derive this structure and corresponding reconstruction bandwidths.

For both the direct and indirect components at a pixel, we need to decide if approximating the radiance integral by factoring into a product of texture and irradiance integrals is possible. Knowing the Fourier structure of the local light field gives us the required sampling rates for each component, as derived in Section 7. In a second path-tracing pass, we trace the minimum adequate number of primary rays to sample world location and texture, and then for each primary ray, trace an appropriate number of secondary rays to compute the irradiance. Then, in the first filtering pass, if the factorization was determined valid, we filter the factored direct and indirect irradiance. In a second filtering pass, we take the combined

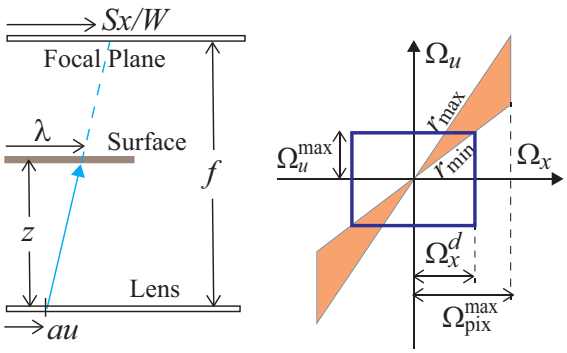


Figure 3: *(a) Ray-tracing geometry for defocus blur. (b) Fourier spectrum and axis-aligned filter for defocus blur.*

color at a pixel and apply the defocus filter. This gives the final noise free image. Implementation details are provided in Section 8.

Assumptions: The key assumption underlying our analysis is that surfaces are diffuse. Lambertian BRDFs allow a simple end-to-end equation for multiple effects since there is no angle-dependence. However, our practical method works well for moderately glossy surfaces, and all of our results include glossy direct and indirect illumination. Note that we always filter samples obtained by accurate ray or path tracing using the full BRDF. We also assume Gaussian lens aperture transmission and area light intensity, like previous methods based on frequency analysis. We evaluate the direct illumination form factor for a surface at the center of the light to simplify analysis and implementation, as in previous work.

4 Defocus Blur

We now describe our Fourier light field analysis which guides our bandwidth prediction. We first consider defocus blur only, assuming diffuse surfaces and a thin lens model. Secondary effects are discussed in the next two sections. Defocus blur is a distribution effect produced by primary (eye) rays, traced from the camera lens of finite aperture out to the world focal plane.

Our derivation is in flatland, but the extension to the 3D world is straightforward. The set up is shown in Fig 3(a). The screen resolution is $2W$, lens aperture is $2a$ and focal distance is f (in world space; we do not explicitly need to involve the focal length). Pixel coordinates x (pixel units) range in $[-W, W]$; lens coordinates u (dimensionless) range in $[-1, 1]$. A primary ray (x, u) is traced from world location $(au, 0)$ to $(Sx/W, f)$, where $2S$ is the world space size (meters) of the focal plane. Consider a parallel surface at depth z from the lens, parametrized by λ, i.e. λ is the world x-coordinate along the object. Then, the intersection with the object, of the primary ray (x, u) is

$$\lambda = au + \frac{z}{f}\left(\frac{Sx}{W} - au\right) = \frac{Sz}{Wf}\left(x + auW\frac{f-z}{Sz}\right). \quad (1)$$

Let us denote

$$r(x,u) = \frac{aW}{S}\left(\frac{f}{z(x,u)} - 1\right) \quad (2)$$

as the width of the circle of confusion (in pixel units) for the ray hitpoint (x, u). We also define the magnification $\ell_p = (Sz/Wf)$ measured in meters per pixel, which transforms pixel-space x to world space λ. For a non-parallel surface, $r(x, u)$ changes for each ray, since the depth is not constant. Since the surface is pure diffuse, the light field at the camera sensor is

$$L(x,u) = L_o(\lambda) = L_o(\ell_p \cdot (x + ru)). \quad (3)$$

Here $L_o(\lambda)$ is the intensity reflected by the receiver with argument λ in meters. The Fourier transform of the light field is

$$\hat{L}(\Omega_x, \Omega_u) = \frac{1}{\ell_p}\delta(\Omega_u - r\Omega_x)\hat{L_o}\left(\frac{\Omega_x}{\ell_p}\right). \tag{4}$$

All frequencies on the spatial axis are in pixel^{-1} units. Each parallel receiver surface contributes a line in Fourier space (Ω_x, Ω_u), with slope given by its circle of confusion. Due to sloped or multiple receivers, the spectrum is a double wedge, as shown in Fig 3(b). The final color c at pixel x is

$$c(x) = \int L(x,u)A(u)\,du, \tag{5}$$

where $A(u)$ is the lens aperture function. The Fourier transform of the pixel-domain color is

$$\hat{c}(\Omega_x) = \int \hat{L}(\Omega_x, \Omega_u)\hat{A}(-\Omega_u)\,d\Omega_u. \tag{6}$$

The lens aperture function bandlimits $\hat{L}(\Omega_x, \Omega_u)$, on the lens frequency axis, so that we can apply a simple filter to $\hat{c}$ as shown in Fig 3(b). The spatial bandwidth of the axis-aligned filter is

$$\Omega_x^d = \min\left\{\Omega_{\text{pix}}^{\max},\, \Omega_{\text{u}}^{\max}/r_{\min}\right\}. \tag{7}$$

Here $\Omega_{\text{pix}}^{\max} = 0.5$ is the maximum allowed pixel bandlimit (corresponding to 1 sample per pixel) and $\Omega_{\text{u}}^{\max}$ is the bandlimit of $A(u)$. This becomes an image-space filter implemented as a Gaussian with std. deviation Ω_x^d. We use the superscript d to denote defocus filter width, since we have different filters for different effects. We assume the lens to be a Gaussian with a 2σ width over $u \in [-1, 1]$, so $\sigma = 1$, implying a standard deviation of $\hat{\sigma} = 1$ in Fourier space and $\Omega_{\text{u}}^{\max} = \hat{\sigma} = 1$. Since the primal-domain defocus filter width R_x^d (in pixels) is inversely proportial to the Fourier domain filter width, eqn. 7 implies:

$$R_x^d = \max\left\{2,\, r_{\min}\right\}. \tag{8}$$

The minimum width of 2 pixels corresponds to the filter weight for the adjacent pixel going to zero. As intuition would suggest, for diffuse surfaces, the defocus filter width is simply the smallest circle of confusion at a pixel. The actual bandlimit can be somewhat smaller for glossy highlights, but the difference (from using a filter width derived for diffuse) is usually not noticeable.

Discussion: [Belcour et al. 2013] arrive at a similar result for the lens matrix that tranforms the light field's covariance matrix. Although they are also able to handle glossy surfaces and occlusions in the same framework, their approach is also more complex and requires additional data structures for occlusion testing. [Soler et al. 2009] also give a light field analysis for defocus blur, using a series of atomic shears, and use the predicted shape of the power spectrum to guide sampling and filtering. However, our approach is end-to-end unlike the approach of concatenating atomic operators used in both [Belcour et al. 2013] and [Soler et al. 2009]. Neither of these papers explicitly equates the slope of the defocused light field to the local circles of confusion. The concurrent work of [Vaidyanathan et al. 2014] does derive a very similar frequency analysis for defocus blur, but does not explore the connections with direct and indirect illumination that we study next.

5 Direct Illumination with defocus blur

While texture samples at a pixel are functions of the random lens position alone, incident radiance samples (direct and indirect) are functions of two random parameters: the lens position and the illumination direction. Hence, reconstruction can be more efficient if the pixel irradiance can be pre-filtered to remove noise due to incoming direction. To avoid the overhead of storing the full

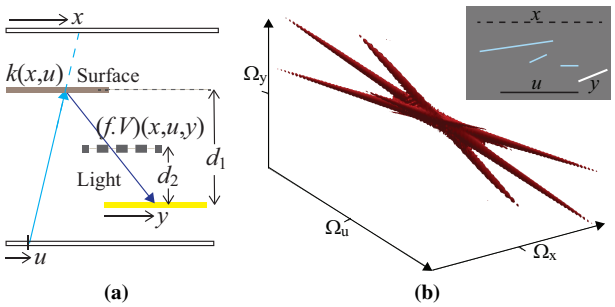


Figure 4: *(a) Path tracing geometry for defocus blur and soft shadows (area light shown in yellow). (b) Power Spectrum of V, $|\hat{V}(\Omega_x, \Omega_u, \Omega_y)|^2$, for a simple flatland scene with non-parallel geometry, showing our double-cone model holds for this case.*

light field (individual ray samples), we propose a novel method to factor the integrated texture and irradiance. This allows us to filter efficiently and work with lower sampling rates than [Belcour et al. 2013] who consider all effects but only derive a single filter width at each pixel. To motivate our filtering scheme with factored irradiance, we first study the illumination integral, and perform a frequency analysis of the incident light field. Sections 5 and 6 treat direct and indirect illumination respectively, in combination with defocus blur.

Consider a flatland scene illuminated with an area light with intensity function $I(y)$ which is a Gaussian over a support $[-\ell_I, \ell_I]$. Similar to the lens function, the light bandlimit is $\Omega_{\text{y}}^{\max} = 1/\ell_I$. As before, we sample the lens u, for each screen (pixel) coordinate x, and each sample corresponds to a world location $\lambda(x,u)$. We now trace secondary shadow rays from λ to y on the light, as depicted in Fig 4(a). For the differential geometry at λ, we parametrize the texture and illumination in (x,u,y) coordinates, and define $k(x,u) = k(\lambda)$ as the diffuse texture, $f(x,u,y)$ as the form factor (two cosine terms divided by distance-squared) and $V(x,u,y)$ as the light visibility for the ray pair (x,u,y). We use the form factor evaluated at the center of the light $f_c(x,u) = f(x,u,0)$ (as in [Egan et al. 2011; Mehta et al. 2012]) because this simplifies both analysis and implementation. The pixel radiance due to direct illumination from the area light is given by

$$\begin{aligned} L_{\text{dir}}(x) &= \int_u k(x,u)\left(\int_y f(x,u,y)V(x,u,y)I(y)\,dy\right)A(u)\,du \\ &= \int_u k(x,u)\,f_c(x,u)\left(\int_y V(x,u,y)I(y)\,dy\right)A(u)du. \end{aligned} \tag{9}$$

Factoring Texture and Irradiance: Equation 9 suggests that the irradiance (the inner integral) is integrated with both the light and the lens functions, while the texture term is integrated with the lens function. To pre-filter the irradiance term, we must factor the radiance into a product of integrated texture and irradiance. We first define the expectation of a function b over a kernel A as $\mathbf{E}_A[b] \equiv \int b(u)A(u)du$. Here $A(\cdot)$ is chosen to be the lens function satisfying $\int A(u)du = 1$. Then we invoke the standard identity from statistics,

$$\mathbf{E}_A[b_1 \cdot b_2] = \mathbf{E}_A[b_1]\cdot\mathbf{E}_A[b_2] + \mathbf{E}_A[(b_1 - \mathbf{E}_A[b_1])(b_2 - \mathbf{E}_A[b_2])]. \tag{10}$$

The second term is negligibly small if either b_1 or b_2 is almost constant, or if they are uncorrelated over the support of $A(u)$. Thus, when the texture $k(x,u)$ and the incident light intensity $f_c(x,u)\int V(x,u,y)I(y)\,dy$ are either constant or uncorrelated w.r.t. u, we can approximate equation 9 as

$$\begin{aligned} L_{\text{dir}}(x) \approx \int k(x,u)A(u)\,du \cdot \\ \int f_c(x,u)A(u)\left(\int V(x,u,y)I(y)\,dy\right)du. \end{aligned} \tag{11}$$

We can rewrite this last approximation as

$$L_{\text{dir}}(x) \approx k_{\text{dir}}(x) \cdot E_{\text{dir}}(x), \tag{12}$$

where we have defined the integrated texture term as

$$k_{\text{dir}}(x) = \int k(x,u)A(u)\,du \tag{13}$$

and the integrated irradiance term as

$$E_{\text{dir}}(x) = \int f_c(x,u)A(u)\left(\int V(x,u,y)I(y)\,dy\right)du. \tag{14}$$

Almost all image-space global illumination methods ([Gershbein et al. 1994; Ward and Heckbert 1992; Mehta et al. 2013], etc.) factor out the texture term and work with the irradiance. However, methods that deal with defocus blur cannot do this directly. If only the pixel radiance is filtered in image space (e.g. [Belcour et al. 2013], [Li et al. 2012]), in a region with high frequency texture and small or no defocus, the shadow filter cannot be used (else the texture will be incorrectly blurred), and the light visibility will have to be sampled densely to remove noise. Our proposed factorization allows us to pre-filter the irradiance term by the light bandlimit separately, before multiplication by texture and applying the defocus blur filter. Without factoring, the pixel color frequency is only determined by defocus blur magnitude, since the texture term is not filtered by the light. To derive the filter and sampling rate for the irradiance, we perform a frequency analysis of the visibility $V(x,u,y)$.

Frequency analysis of light visibility in (x,u,y) space: We first perform a frequency analysis of the light visibility in (x,u,y) space, considering a parallel plane of occluders. Let $g(\cdot)$ be the one-dimensional visibility in the occluder plane. The set up is as shown in Fig 4(a). Let $\rho = d_2/d_1$, where d_1 and d_2 are distances of receiver and occluder from the light respectively. Then, the shadow light field on the receiver surface, following [Egan et al. 2011] and [Mehta et al. 2012], is[3] $g(\rho\lambda + (1-\rho)y)$. Hence, we have

$$\begin{aligned} V(x,u,y) = V(\lambda,y) = g(\rho\lambda + (1-\rho)y) \\ = g(\rho\ell_p(x+ru) + (1-\rho)y) \equiv g(\alpha x + \beta u + \gamma y). \end{aligned} \tag{15}$$

In the last step we have introduced the parameters α, β, γ to simplify the representation. Performing a 3D Fourier transform gives:

$$\begin{aligned} \hat{V}(\Omega_x,\Omega_u,\Omega_y) = \int\int\int g(\alpha x + \beta u + \gamma y) \\ \exp(-j(x\Omega_x + y\Omega_y + u\Omega_u))dx\,du\,dy \\ = \frac{1}{\alpha}\hat{g}\left(\frac{\Omega_x}{\alpha}\right)\delta\left(\Omega_u - \frac{\beta}{\alpha}\Omega_x\right)\delta\left(\Omega_y - \frac{\gamma}{\alpha}\Omega_x\right). \end{aligned} \tag{16}$$

This is a line through the origin in 3D frequency space, normal to the isosurface plane defined by eqn. 15. Due to the integration with aperture and light in eqn. 14, this line is bandlimited (clipped) by the planes $|\Omega_u| \le \Omega_{\text{u}}^{\max}$ and $|\Omega_y| \le \Omega_{\text{y}}^{\max}$. The two slopes given by the delta functions in eqn. 16 imply that the line is clipped along Ω_x to $|\Omega_x| \le \Omega_{\text{u}}^{\max}/r$ and $|\Omega_x| \le \ell_p\Omega_{\text{y}}^{\max}/(\rho^{-1}-1)$. Let

$$s = \rho^{-1} - 1 = (d_1/d_2) - 1. \tag{17}$$

s is analogous to r defined in eqn. 2. For non-parallel and multiple receivers and occluders, the visibility spectrum becomes a bandlimited double-cone in frequency space. As illustrated in Fig 4 (b) with a flatland simulation, this is a good approximation for arbitrarily oriented surfaces and area lights. The filter width (in per-pixel units) that can be used to filter E_{dir} according to the light bandlimit then becomes

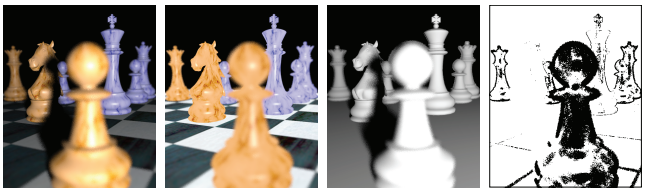
$$\Omega_x^s = \min\left\{\Omega_{\text{pix}}^{\max},\ \ell_p\Omega_{\text{y}}^{\max}/s_{\min},\ \Omega_{\text{u}}^{\max}/r_{\min}\right\} \tag{18}$$

Equivalently, the primal-domain filter size in pixels is

$$R_x^s = \max\left\{2,\ \ell_I s_{\min}/\ell_p,\ r_{\min}\right\}. \tag{19}$$

Filtering using factoring: In practice, we can use factoring (eqn. 12) wherever the error[4] $||L_{\text{dir}} - k_{\text{dir}} \cdot E_{\text{dir}}||$ (obtained from a first sparse sampling pass) is small enough. For the pixels where the factorization is valid, we can filter E_{dir} separately, by the shadow filter Ω_x^s, then multiply by k_{dir} and filter the product by the defocus filter Ω_x^d. Pre-filtering allows lower sampling rates for the light visibility, thus saving on expensive ray-tracing. For pixels where the factorization is not valid, we filter the pixel radiance $L_{\text{dir}}(x)$ for defocus only. Usually these are pixels with a large defocus width, and hence can gather radiance information from many neighboring pixels, implying a lower sampling rate. Hence, most pixels can work with low sampling rates which are derived in Section 7.

In Fig 5(a) we show the pixel radiance L_{dir} from the first sampling pass, and compare it to the integrated texture k_{dir} in (b) and the integrated irradiance E_{dir} in (c). The thresholded error is shown as a binary flag in (d). Object silhouettes or regions with high defocus blur cannot be factored, but those with soft shadows on smooth textures can. About 80% of pixels are separable for this scene, but the fraction is larger for indirect illumination, and other scenes. This makes our method much more efficient since we can pre-filter noisy irradiance.



(a) L_{dir} **(b)** k_{dir} **(c)** E_{dir} **(d)** *SepDir*

Figure 5: *(a) The direct radiance L_{dir} for the CHESS scene from the first sampling pass (16 spp), (b) The factored texture k_{dir} and (c) the separated irradiance E_{dir}. (d) The factorization error $||L_{\text{dir}} - k_{\text{dir}} \cdot E_{\text{dir}}||$ is below a threshold except at the pixels marked black (shown smoothed with a median filter).*

Glossy Surfaces: Our filter widths Ω_x^d and Ω_x^s are derived assuming diffuse surfaces. Even though we do not handle glossy surfaces explicitly in our theory, our approximations work well for glossy direct and non-caustic indirect illumination, as demonstrated in our renders, all of which have glossy surfaces. This is because our filter is axis-aligned, and can capture a lot of energy that leaks beyond the double-wedge model. We also filter an accurately path traced illumination for both diffuse and glossy components. To handle direct illumination on a glossy surface, suppose $\hat{\mathbf{e}}$ is the primary ray (x,u), and $\hat{\mathbf{r}}$ is the direction from the hitpoint of the primary ray to the light's center, reflected about the hitpoint normal. Then the Phong BRDF gloss factor evaluated at the light center is simply $(-\hat{\mathbf{e}}\cdot\hat{\mathbf{r}})^m \equiv p(x,u)$. We can separate $p(x,u)$ out into the texture term. Explicitly, $k(x,u)$ in eqn. 13 becomes

$$k(x,u) = k_d(x,u) + k_s(x,u)p(x,u) \tag{20}$$

where k_d and k_s are the diffuse and specular textures respectively.

[3] In general there will also be a constant offset in the argument of $g(.)$ here, if the origins of the x and y coordinates are not aligned. However, the constant offset does not affect the Fourier energy spectrum, so we ignore it.

[4] $||\cdot||$ is the standard euclidean distance between RGB colors.

6 Indirect Illumination

The total pixel radiance is the sum of the integrated direct and indirect radiance, i.e. $L(x) = L_{\mathrm{dir}}(x) + L_{\mathrm{ind}}(x)$, and we treat the two components independently. Many of the same arguments, including factorization, that apply in the direct case, also apply to indirect illumination.

We use the parametrization in [Mehta et al. 2013], where incident radiance is a function of linearized direction v measured in a plane parallel to the local receiver. The indirect radiance L_{ind} at pixel x is the integral of the texture $k(x,u)$, the BRDF and geometry term[5] $h(v)$ and incoming indirect radiance $l_i(x,u,v)$ reflected from the nearest surface in direction v. We can also factorize the texture and irradiance as follows:

$$\begin{aligned} L_{\mathrm{ind}}(x) &= \int k(x,u) \left(\int h(v) l_i(x,u,v)\, dv \right) A(u) du \\ &\approx \int k(x,u) A(u)\, du \cdot \int A(u) \left(\int h(v) l_i(x,u,v)\, dv \right) du \\ &\equiv k_{\mathrm{ind}}(x) \cdot E_{\mathrm{ind}}(x). \end{aligned} \tag{21}$$

This is similar to the direct lighting equation, with the light intensity I replaced by the transfer function h. If the factorization error $||L_{\mathrm{ind}} - k_{\mathrm{ind}} \cdot E_{\mathrm{ind}}||$ is small, we use the factored product $k_{\mathrm{ind}} \cdot E_{\mathrm{ind}}$. Factoring allows pre-filtering the E_{ind} term and reducing the sampling rate required. If factoring is not possible, the radiance L_{ind} can still be blurred by the defocus filter, and a moderate sampling rate suffices.

Assuming diffuse reflectors, the spectrum of the incident indirect radiance l_i is also similar to that of the light visibility V from the previous section. Extending the result of [Mehta et al. 2013], individual reflectors contribute lines in the Fourier space with slope along the $\Omega_x \times \Omega_v$ plane given by the reflector depth z at (x,u,v). The slope along the $\Omega_x \times \Omega_u$ plane is given by the circle of confusion r at (x,u). Similar to eqn. 18, the filter width for E_{ind} is given by

$$\Omega_x^i = \min \left\{ \Omega_{\mathrm{pix}}^{\max},\ \ell_p \Omega_{\mathrm{v}}^{\max} / z_{\min},\ \Omega_{\mathrm{u}}^{\max} / r_{\min} \right\}. \tag{22}$$

$\Omega_{\mathrm{v}}^{\max}$ is the bandlimit of the low-pass transfer function h; the numerical values for diffuse and glossy BRDFs can be found in [Mehta et al. 2013]. The primal domain filter size is

$$R_x^i = \max \left\{ 2,\ z_{\min} / (\ell_p \Omega_{\mathrm{v}}^{\max}),\ r_{\min} \right\}. \tag{23}$$

Glossy Surfaces: The diffuse and glossy transfer functions $h(v)$ are different (we need not know their exact forms), and the v dependence cannot be dropped to separate h from the E_{ind} integral as we did for $f(x,u,y)$ in direct illumination. In other words, an approximation like eqn. 20 cannot be made for indirect illumination. For the factorization $k_{\mathrm{ind}} \cdot E_{\mathrm{ind}}$ to work for a surface with both diffuse and glossy components, each of $L_{\mathrm{ind}}, k_{\mathrm{ind}}, E_{\mathrm{ind}}$ must be evaluated and stored separately for the diffuse and glossy components. In the filtering pass, both diffuse and glossy E_{ind} are filtered according to their own bandwidths given in equation 22 and then combined with the appropriate k_{ind}.

7 Sampling Rates

Point sampling of the high-dimensional light field can cause aliasing if the sampling rate is not sufficient, even if we subsequently use the proper axis-aligned reconstruction filter. The minimum sampling rate is that which just prevents adjacent copies of spectra from overlapping our baseband filter. As in [Egan et al. 2009; Mehta et al. 2012; Mehta et al. 2013], we derive the minimum distance between aliases of spectra, $\Omega_x^*, \Omega_u^*, \Omega_y^*, \Omega_v^*$, and multiply these to obtain the per-pixel sampling rates. However, the major difference is that we derive different sampling rates for both primary and secondary rays. We allocate rays in proportion to the frequency content of each effect, instead of using a fixed number of secondary rays per primary ray at each pixel as has been done in previous work.

In the first sampling pass, we trace a fixed number of rays per pixel to estimate bandwidths and sampling rates for the next pass. In our main (second) sampling pass, at each pixel, we first send n_p primary rays from the lens, then for direct illumination we trace a number of shadow rays for each of these n_p primary rays so that the total number is n_{dir}. For indirect illumination, for each primary ray a certain number of indirect radiance samples are obtained, so that the total number is n_{ind}. In addition, our secondary sampling rates vary depending on whether we use the exact radiance, or factored texture and irradiance. Superscripts 'c' (combined) and 'f' (factored) are used to denote these two different secondary sampling rates respectively.

For determining the primary sampling rate, we need only consider simple defocus blur (assuming diffuse surfaces). Our axis-aligned filter was described in Section 4. The minimum primary sampling rate is obtained considering aliasing in $\Omega_x \times \Omega_u$ space, as shown in Fig. 6(a). We have,

$$\begin{aligned} n_p &= (\Omega_x^*)^2 (\Omega_u^*)^2 \\ &= (\Omega_{\mathrm{pix}}^{\max} + \Omega_x^d)^2 (1 + r_{\max} \Omega_x^d)^2. \end{aligned} \tag{24}$$

Although [Soler et al. 2009] use power spectral energy and variance to determine their sampling rate, their overall sampling density in defocused regions looks similar to ours. However, their sampling method is quite different as they obtain image and lens samples in Fourier space.

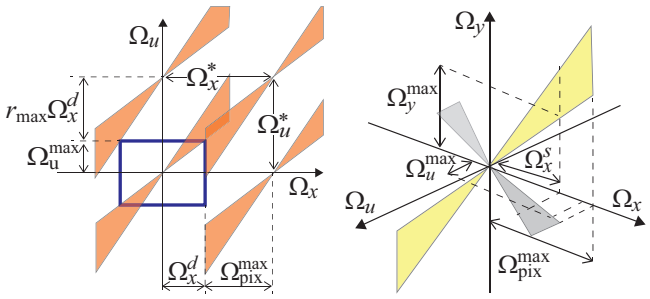


Figure 6: *(a) Packing of aliases for defocus only, showing the spatial and lens sampling rates Ω_x^* and Ω_u^* (b) Packing for aliases of the light visibility V. The yellow and grey double wedges are the projection on the $\Omega_x \times \Omega_y$ and $\Omega_x \times \Omega_u$ planes respectively. Aliases not shown for clarity. The minimum sampling rates Ω_x^*, Ω_y^* and Ω_u^* are analogous to those shown in (a).*

Secondary sampling rate with factored texture and irradiance: For area light direct illumination, the visibility in equation (7) must be sampled sufficiently to avoid aliasing in each of the (x,u,y) dimensions. At pixels with factored direct illumination, filtering E_{dir} clips it to a spatial Fourier width of Ω_x^s. This case is illustrated in Figure 6(b), with projections of the spectrum on two coordinate planes instead of the full volumetric spectrum for clarity. The minimum separation for no aliasing along each axis follows similarly from the 2D packing for defocus only; the per-pixel sampling rate

[5] The BRDF term is most generally a function of the world location also, i.e. $h(x,u,v)$. We assume that the surface visible in a small neighborhood around a current pixel (for all u) is flat, and drop the x,u dependence.

for $V(x,u,y)$ (secondary shadow rays) is

$$n^f_{\text{dir}} = (\Omega^*_x)^2(\Omega^*_u)^2(\Omega^*_y)^2\ell_I^2 = (\Omega^{\max}_{\text{pix}} + \Omega^s_x)^2(1 + r_{\max}\Omega^s_x)^2(1 + \ell_I s_{\max}\Omega^s_x/\ell_p)^2 \quad (25)$$

Similarly, for indirect illumination the indirect light field l_i in equation (12) must be sampled more to avoid aliasing in each of the (x,u,v) dimensions. When factored irradiance is used, the spectrum is clipped to a bandlimit Ω^i_x. The sampling rate is then,

$$n^f_{\text{ind}} = (\Omega^*_x)^2(\Omega^*_u)^2(\Omega^*_v)^2 = (\Omega^{\max}_{\text{pix}} + \Omega^i_x)^2(1 + r_{\max}\Omega^i_x)^2(\Omega^{\max}_{\text{v}} + z_{\max}\Omega^i_x/\ell_p)^2 \quad (26)$$

Secondary sampling rate without factoring: If the direct radiance cannot be factored into texture and irradiance, only the defocus filter must be applied to the radiance. The light field spectrum is then clipped to a bandlimit Ω^d_x. The direct illumination sampling rate is then

$$n^c_{\text{dir}} = (\Omega^{\max}_{\text{pix}} + \Omega^d_x)^2(1 + r_{\max}\Omega^d_x)^2(1 + \ell_I s_{\max}\Omega^d_x/\ell_p)^2 \quad (27)$$

Finally, without factorization, the indirect illumination sampling rate is

$$n^c_{\text{ind}} = (\Omega^{\max}_{\text{pix}} + \Omega^d_x)^2(1 + r_{\max}\Omega^d_x)^2(\Omega^{\max}_{\text{v}} + z_{\max}\Omega^d_x/\ell_p)^2 \quad (28)$$

Discussion: Observe from eqns. 7, 18 that $\Omega^s_x \le \Omega^d_x \le \Omega^{\max}_{\text{pix}}$. As expected, if the pixel is in focus (i.e., either $f = z$ or $a = 0$) the number of primary rays per pixel is $n_p = (2\Omega^{\max}_{\text{pix}})^2 = 1$ since $r_{\max} = r_{\min} = 0$. Similarly, $n^c_{\text{dir}} = n_p$ if we have a point light with $\ell_I = 0$. This means we only take one visibility sample if the light size shrinks to zero, verifying that our formulae work in the limit. Further, observe that $n^f_{\text{dir}} \le n^c_{\text{dir}}$, since the former uses a smaller spatial bandlimit. This is expected, since the ability to filter the irradiance E_{dir} allows for a lower secondary sampling rate. Also note that at a pixel that uses factored irradiance, the defocus is typically small, and then $n^f_{\text{dir}} \ge n_p$ since the secondary sampling rate has an extra $(\Omega^*_y)^2$ term in eqn. 25. Then we must trace $n^f_{\text{dir}}/n_p \ge 1$ secondary rays per primary ray[6].

We now qualitatively discuss our practical sampling rates shown in fig 8(e)-(g). First, in the region marked 'A', we have a high defocus and depth variance, and hence we provide more rays for all effects (i.e. $n_p, n_{\text{dir}}, n_{\text{ind}}$ are all large, but this type of region covers only a small part of the image). In the region marked 'B', which is in-focus, we need few primary rays but many indirect samples since it has nearby reflectors. In 'C', the wall is defocused but at a constant depth, so a few primary rays suffice, but more direct and indirect samples are needed.

Convergence with increasing sampling rate: As in [Mehta et al. 2012; Mehta et al. 2013] we can control our sampling rates with a user-defined parameter μ. To implement this, each reconstruction bandwidth is simply scaled by μ and the sampling rates are then computed as above. For example, the primary sampling rate as a function of μ becomes:

$$n_p(\mu) = (\Omega^{\max}_{\text{pix}} + \Omega^d_x(\mu))^2(1 + r_{\max}\Omega^d_x(\mu))^2, \quad (29)$$

where $\Omega^d_x(\mu) = \min\left\{\Omega^{\max}_{\text{pix}}, \mu \cdot \Omega^d_x\right\}$. Thus, we can smoothly control our speed and accuracy, and converge to ground truth with increasing ray count. We demonstrate this quantitatively as an error-vs-rpp plot in Fig. 12(e). Controlling sampling rate and filter size using μ can also be used to speed-up our method by starting with low μ and updating the image with increasing μ, and refreshing if the camera or light is changed. We demonstrate this interactive pre-view rendering system in our video.

[6] However, a pixel with constant texture and high defocus can also allow factorization. In this and some other cases, we may also have $n^f_{\text{dir}} < n_p$, but for physical correctness we trace one secondary ray per primary ray.

8 Implementation

A flow chart of our algorithm is shown in Figure 7. All quantities are concisely defined in Table 1 which also points to the relevant equations.

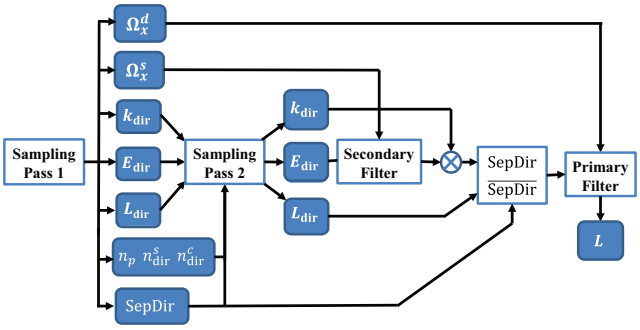


Figure 7: *Flow chart of the algorithm for the direct component only; the indirect component is handled similarly. Filled blocks are variables stored in memory, empty blocks are operations. Refer to table 1 for definitions of variables.*

Quantity	Description	Equation
L_{dir}	integrated color, direct component	9
k_{dir}	integrated texture, direct component	13
E_{dir}	integrated illumination, direct component	14
L_{ind}	integrated color, indirect component	21
k_{ind}	integrated texture, indirect component	21
E_{ind}	integrated illumination, indirect component	21
SepDir	Boolean, set if direct factorization error is small	30
SepInd	Boolean, set if indirect factorization error is small	30
$r_{\min}, r_{\max}$	min, max circle of confusion in pixels	2
$s_{\min}, s_{\max}$	min, max soft shadow slopes	17
$z_{\min}, z_{\max}$	min, max reflector distance for indirect	-
Ω^d_x	depth-of-field filter width	7
Ω^s_x	direct illumination filter width	16
Ω^i_x	indirect illumination filter width	19
n_p	Num. primary (lens) rays	24
n^c_{dir}	Num. light shadow rays if SepDir $= 0$	27
n^f_{dir}	Num. light shadow rays if SepDir $= 1$	25
n^c_{ind}	Num. indirect samples if SepInd $= 0$	28
n^f_{ind}	Num. indirect samples if SepInd $= 1$	26

Table 1: *A list of the variables we store in a per-pixel buffer. The defining equation for each variable is indicated in the last column.*

Our algorithm is implemented in multiple consecutive pixel shader passes in NVIDIA's Optix ray-tracer. The source code will be made available online upon publication.

1. **Sampling pass 1:** We first trace 16 paths per pixel into the scene. A single path is one primary lens ray, one secondary shadow ray, and a one-bounce indirect sample (separate for diffuse and glossy). We draw lens samples, light samples (for direct) and hemisphere samples (for indirect) from a 4×4 stratification each, and match the samples with random permutations [Shirley and Morley 2003]. At each pixel we accumulate the colors $L_{\text{dir}}, k_{\text{dir}}, E_{\text{dir}}, L_{\text{ind}}, k_{\text{ind}}, E_{\text{ind}}$. The direct parameters $s_{\min}, s_{\max}$, indirect parameters $z_{\min}, z_{\max}$ and defocus parameters $c_{\min}, c_{\max}$, and the lens-averaged world location, projected area A_p, and normal are computed. From these we compute the filter widths $\Omega^r_x, \Omega^s_x, \Omega^i_x$ and set flags:

$$\text{SepDir} = ||L_{\text{dir}} - k_{\text{dir}} \cdot E_{\text{dir}}|| < 0.01$$
$$\text{SepInd} = ||L_{\text{ind}} - k_{\text{ind}} \cdot E_{\text{ind}}|| < 0.01 \quad (30)$$

2. **Sampling pass 2:** The primary and secondary sampling rates $n_p, n^f_{\rm dir}, n^c_{\rm dir}, n^f_{\rm ind}$ and $n^c_{\rm ind}$ are as discussed in Section 7. We run a 3×3 max-filter on the ray counts, and a median filter on the factorization flags, to reduce noise and artifacts. Next, for pixels with $\mathrm{SepDir} = 1$, we trace n_p primary rays, and from each primary hit-point trace $n^f_{\rm dir}/n_p$ shadow rays when $\mathrm{SepDir} = 1$ and $n^c_{\rm dir}/n_p$ shadow rays otherwise. Samples are fully stratified[7] over each dimension (lens, light, hemisphere) and matched by random permutation as in the first pass. A similar sampling scheme applies for indirect illumination. Instead of importance sampling which gives noisier estimates of sampling rates and filter sizes, we apply explicit Gaussian weights to lens and light samples. This sampling pass updates the noisy color buffers, reducing the noise so that it can be removed by filtering.

3. **Irradiance Filtering:** In this pass, only the direct and indirect illumination $E_{\rm dir}$ and $E_{\rm ind}$ are filtered. We filter using world space distances and filter width Ω^s_x and Ω^i_x. Lens-averaged world space locations λ and normals are used since there is no single world space location per-pixel due to defocus. Explicitly, the weight we apply to the direct irradiance of a neighboring pixel j for a central pixel i is

$$w_i(j) = \exp\left\{-16||\lambda(i) - \lambda(j)||^2 \cdot (\Omega^s_x(i)/\ell_p(i))^2\right\} \tag{31}$$

Note that the projected pixel length ℓ_p converts Ω^s_x into meters. For adjacent pixels i and j, $||\lambda(i) - \lambda(j)|| \approx \ell_p(i)$; if $\Omega^s_x = 0.5$ then $w_i(j) = \exp(-4)$. Hence, the constant 16 is chosen so that sharp shadow edges are not blurred. We do not filter between pixels with normals differing by more than 10°.

4. **Defocus Filtering:** Finally, we choose between the exact radiance and factored product of texture and filtered irradiance. The direct and indirect components are added, as:

$$\begin{aligned} L(x) = {} & \mathrm{SepDir} \cdot (k_{\rm dir} \cdot E_{\rm dir}) + \overline{\mathrm{SepDir}} \cdot (L_{\rm dir}) \\ & + \mathrm{SepInd} \cdot (k_{\rm ind} \cdot E_{\rm ind}) + \overline{\mathrm{SepInd}} \cdot (L_{\rm ind}) \end{aligned} \tag{32}$$

The final pass filters the total pixel radiance $L(x)$ using a screen-space gaussian filter of width Ω^d_x to compute the final color; the weights are analogous to equation 31,

$$w_i(j) = \exp\left\{-16(i-j)^2 \cdot (\Omega^d_x(i))^2\right\}. \tag{33}$$

In both filtering passes, at current pixel i, a neighboring pixel j is rejected if $w_j(i) < 0.01$, i.e. if i does not fall in the filter radius of j. This mitigates errors due to noisy estimation of filter radii, and prevents bleeding of sharp regions into blurry regions.

9 Results

We show results of distributed rendering with defocus blur, area light direct and one-bounce indirect illumination on five scenes with high-frequency textures and both diffuse and glossy surfaces. The accompanying video shows animations and screen captures with a moving light source and viewpoint, and examples of dynamic geometry. Our images are rendered on an Intel Xeon, 2.26GHz, 2 core desktop with a single Nvidia Titan GPU. Each frame is rendered independently, without any precomputation (except possibly the raytracer BVH). We report the total individual rays per pixel instead of the more common samples per pixel. In vanilla MC, a single 'sample' is 1 primary ray, 1 direct sample (1 shadow ray) and 1 one-bounce indirect sample (2 rays), i.e. a total of 4 rays.

[7] This requires that we round up n_p to p^2 and $n_{\rm dir}$ to $p^2 s^2$ where p, s are integers. We did not use low-discrepancy sampling based on (0,2) sequences since it requires rounding to a power of two, while other sequences caused some artifacts.

Our method is accurate in a range of different scenarios, with consistent reductions in sample counts over basic path tracing. Figure 1 shows the CHESS scene (21K triangles), with mid-depth focus. Our image (a,c) is noise-free with 138 average rays per pixel (rpp) in only 3.14 sec. Equal visual quality ground truth MC (d) requires 5390 rpp and 40× more time. Careful inspection reveals some noise even with 5390 rpp. Since our overhead is minimal, equal time MC (b) is only 176 rpp, and is very noisy due to the high dimensionality of the light field. We also compare to (e), obtained by simply filtering the radiance without factoring texture and irradiance. Noise from secondary effects is retained in the in-focus region in the top inset. Figure 8 shows similar results for the STILL LIFE scene with complex geometry and 128K triangles. Our method with only 178 rpp is perceptually comparable to stratified MC with 4620 rpp. Two more scenes, SIBENIK CATHEDRAL in Fig. 9 and TOASTERS in Fig. 11, with comparisons are discussed below. We also include ground truth insets, obtained at about 15000 rpp, in these figures. Figure 12, the ROOM scene, demonstrates that our method can produce fast results for scenes with complex light paths.

9.1 Timings

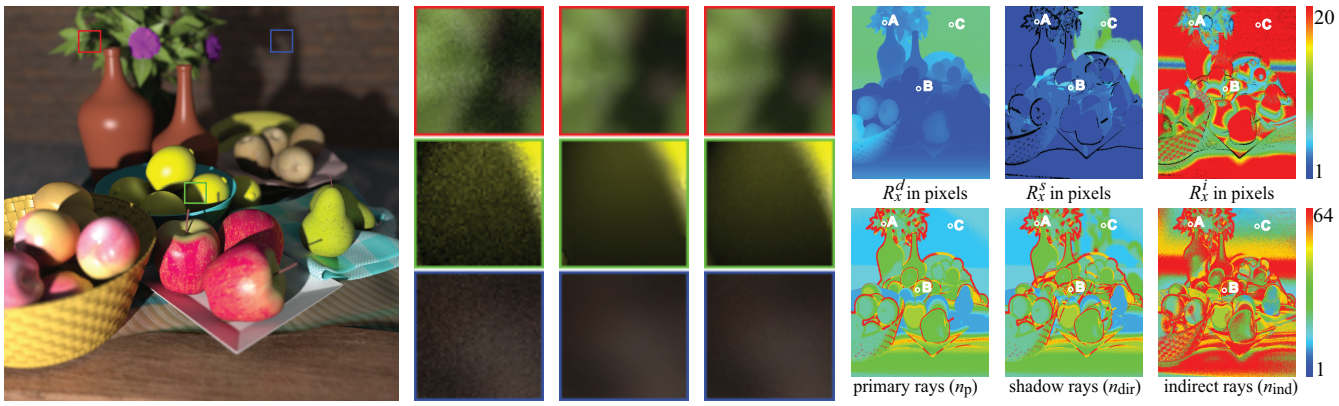
Scene	Tris	rpp	Sample (sec)	Filter (sec)	Total (sec)	Overhead
CHESS	21K	138	2.97	0.15	3.14	5.4%
STILL LIFE	128K	178	7.26	0.12	7.40	2.7%
SIBENIK	75K	192	6.10	0.11	6.23	3.2%
TOASTERS	2.5K	125	3.43	0.16	3.61	5.5%
ROOM	100K	181	8.08	0.15	8.25	2.4%

Table 2: *Render times for all scenes at 1024×1024. An extra overhead of 0.02 sec (20 ms) is incurred for determining filter sizes and sampling rates - this is not shown in the table above, but is included as part of total and overhead in the last two columns. Our overall render times are under 10 seconds, and the filtering overhead is very small compared to the ray-tracing time.*

As demonstrated in Fig. 10, to obtain the same visual quality and noise level as our method, MC path-tracing requires about 30× more rays, so we get a corresponding speed-up. In Table 2, we show timings for the sampling and filtering parts of our algorithm on our scenes, all rendered at 1024×1024. We obtain most of the benefits of axis-aligned filtering, as in [Mehta et al. 2012; Mehta et al. 2013], even though our algorithm is much more complex (with separate direct and indirect illumination, as well as separate radiance, irradiance and texture buffers). The total overhead in a frame is between 110 and 160 ms, which is small compared to the cost of OptiX path tracing (between 3 and 9 seconds), and results in only a marginal decrease in the performance of the real-time raytracer. Our current filter implementation uses only Optix; preliminary tests show a speed-up of over 4× on CUDA using image tiling. Although our overhead is currently about 5%, stratified MC still manages about 20% more rays (as seen in all our scenes which include equal-time rpp) in the same time, since our method produces an unbalanced GPU load. Our filter operates only in image-space and therefore has limited memory requirements (about 150 MB due to storing various buffers). Note that we are limited only by the speed of the raytracer, and using further GPU raytracing accelerations would provide further speedups. This is one of the first demonstrations of distributed rendering that runs in seconds and not minutes of time, based on principled Monte Carlo sampling. Alternative methods, discussed next, add overheads of 10 sec to 1 min.

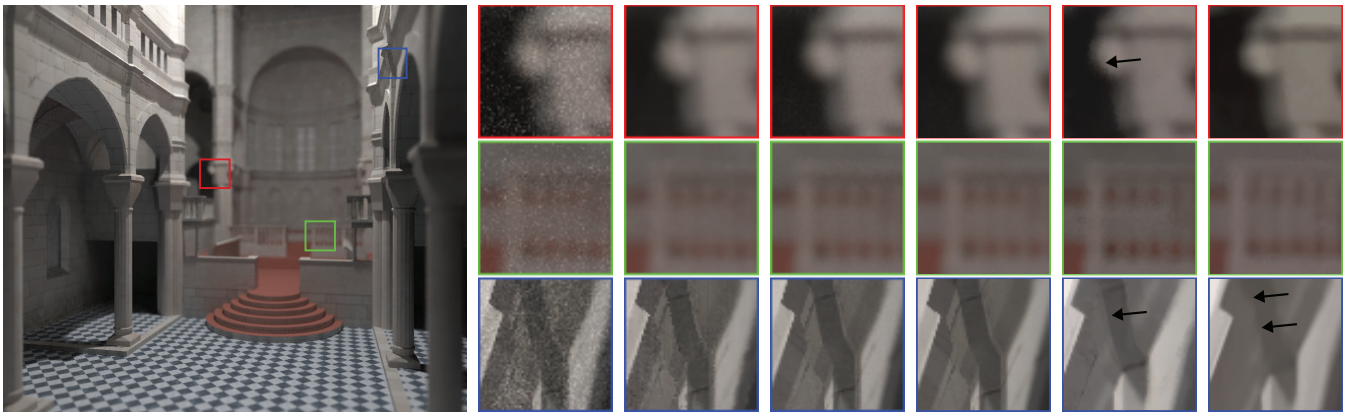
9.2 Quantitative Accuracy

We evaluated the accuracy of our method quantitatively; in Fig. 12(e) we show average per-pixel RMS error vs average number of rays for the ROOM scene. The error of our method (blue curve)



(a) Our method, 178 rpp, 7.40 sec | **(b)** Equal time, 220 rpp, 7.4 sec | **(c)** Our method, 178 rpp, 7.4 sec | **(d)** Eq. quality, 4620 rpp, 4.5 min | **(e)** Primary rpp and filter | **(f)** Direct rpp and filter | **(g)** Indirect spp and filter

Figure 8: *The STILL LIFE scene, with defocus blur, area light direct and indirect illumination, rendered in 7.40 sec with an average 178 rays per pixel (rpp). The insets compare (b) equal time stratified MC, (c) our method, and (d) equal quality stratified MC with 4620 rpp (275 sec). In (e)-(g), we show heatmaps for our three filter widths and sampling rates, namely for defocus, and direct and indirect illumination. A more detailed discussion is provided near the end of Sec. 7.*



(a) Our method, 192 rpp, 6.23 sec | **(b)** Equal time, 229 rpp, 6.3 sec | **(c)** Our Method, 192 rpp, 6.2 sec | **(d)** Eq. quality, 5400 rpp, 3.5 min | **(e)** Gr. Truth, 16400 rpp | **(f)** SURE, 200 rpp, 4 min | **(g)** AMLD, 200 rpp, 4 min

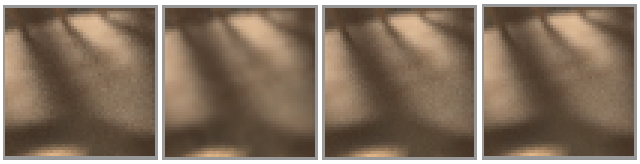
Figure 9: *(a) The SIBENIK CATHEDRAL scene, with area light direct and indirect illumination, and foreground defocus, with an average 192 rays per pixel (rpp) requires 6.23 sec; and insets showing (b) equal time stratified MC with 229 rpp, (c) Our method, (d) Equal quality with 5400 rpp and (e) Ground truth with 16400 rpp. We also compare to (f) SURE, with 200 rpp rendered in 4 min and (g) AMLD with 200 rpp in 4 min.*

is significantly below stratified Monte Carlo at all sample counts, and for the same error we require about 4× less rays. As we increase the number of rays (higher μ, eqn. 29), we do converge to ground truth and error decreases. This is in contrast to most previous solutions for filtering MC images which do not provide a simple solution to converge with increasing ray count. Since our method replaces some of the noise with some bias, equal perceptual error is achieved at over 30× fewer ray counts, as illustrated in Fig. 10.

9.3 Comparisons

We have already discussed comparison to brute-force equal time and equal visual quality MC. In Figs. 9(f,g) and 11(f,g), we include comparison insets to two alternative recent approaches to MC denoising. Insets of the other methods use a similar average number of rays.

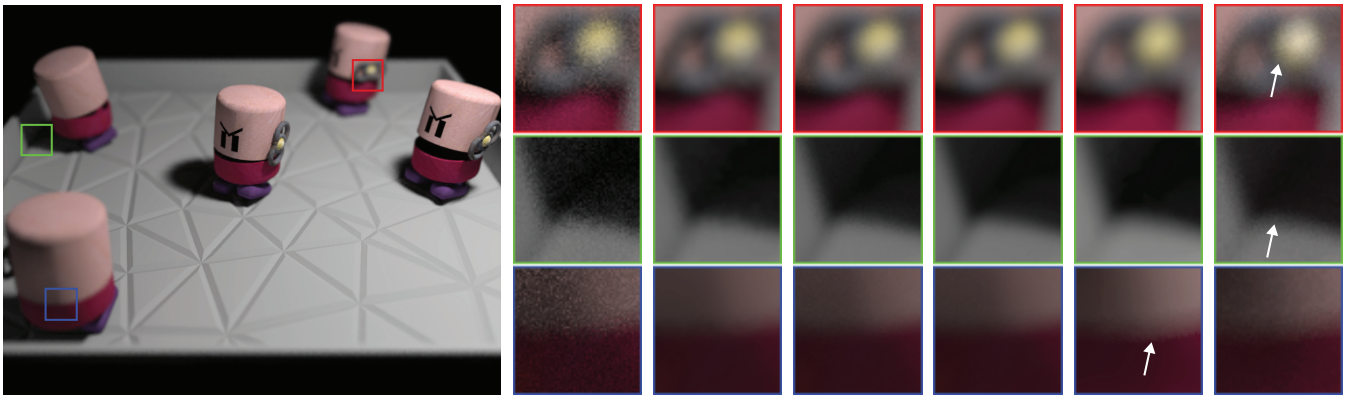
We compare to SURE, [Li et al. 2012], since for the same quality, they are faster than other recent approaches such as [Sen and Darabi 2012; Rousselle et al. 2011], etc. The comparison



MC, 3500 rpp | Our, 180 rpp | MC, 5220 rpp | MC, 7000 rpp

Figure 10: *We compare stratified MC and our method with increasing sample count for an inset from the ROOM scene (Fig. 12). Stratified MC visually matches our method for about 5220 rpp. At 180 rpp, our method is very slightly over-blurred, but MC at 5220 rpp shows more noise (zoom in) in comparison.*

insets show that SURE slightly over-blurs both in-focus and out-of-focus regions if the original image is very noisy. The authors' implementation with PBRT requires around 4 min, with a filtering overhead of 1 min due to its multiple filtering passes. It also requires a slightly higher sampling rate for the same quality. Adaptive Multi-Level Denoising (AMLD, [Kalantari and



(a) Our method, 125 rpp, 3.61 sec

(b) Equal time, 147 rpp, 3.7 sec

(c) Our Method, 130 rpp, 3.6 sec

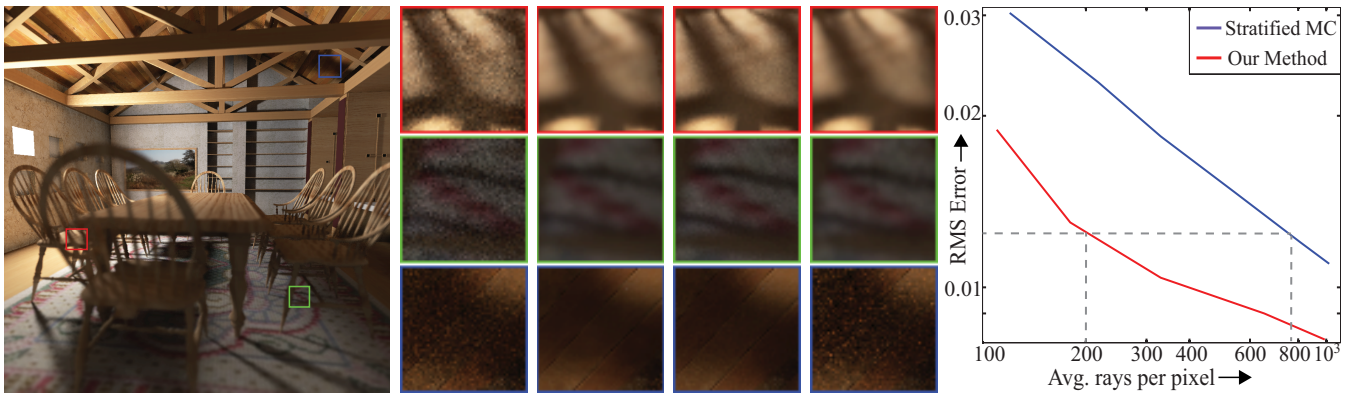
(d) Eq. quality, 4620 rpp, 3 min

(e) Gr. Truth 14800 rpp

(f) SURE, 150 rpp, 2.5 min

(g) AMLD, 128 rpp, 2 min

Figure 11: *(a) The TOASTERS scene, with area light direct and indirect illumination, and mid-depth focus, rendered with an average 125 rays per pixel in total 3.61 sec; Insets showing (b) equal time stratified MC with 147 rpp, (c) Our method, (d) Equal quality with 4620 rpp, (e) Ground Truth with 14800 rpp; and comparisons to (f) SURE, 128 rpp, 2.5 min and (g) AMLD, 128 rpp, 2 min.*



(a) Our method, 181 rpp, 8.25 sec

(b) Equal time, 219 rpp, 8.1 sec

(c) Our method, 181 rpp, 8.2 sec

(d) Eq. quality, 5220 rpp, 5 min

(e) No factoring, 181 rpp, 8.1 sec

(f) RMSE vs rpp

Figure 12: *(a) The ROOM scene, with area light direct and indirect illumination, and foreground defocus, rendered with an average 181 rays per pixel (rpp) in total 8.25 sec (b) Insets showing equal time stratified MC with 219 rpp, (c) Our method, (d) Equal quality MC with 5220 rpp (e) without factoring texture and irradiance, noise in in-focus regions is not filtered. In (f) we show RMS error relative to ground truth, for our method and stratified MC. Our method requires 4× fewer rays than stratified MC for the same RMS error.*

Sen 2013]) demonstrated faster results using PBRT for adaptive sampling and BM3D [Dabov et al. 2007] for denoising. We used the authors' code for producing the images. AMLD produces results of nearly the same quality as ours. There is some under-blur in the TOASTERS and over-blur in SIBENIK; however in some regions it can also perform better. Their overhead is still around 20 seconds (total time 3 min), making our method much faster. We do not compare to Covariance tracing [Belcour et al. 2013] since the method has high overhead (reported as 2.5 min, total time 25 min on CPU) and implementation is complex.

10 Limitations

Direct and indirect illumination reflected from glossy surfaces cannot be treated with a simple approach like ours, and more complex analysis is required. Our diffuse bandwidths can result in some over-blurring of specular highlights. Like most image-space filtering methods, we need to allocate a lot of samples to pixels where out-of-focus and in-focus objects overlap, since such pixels (generally few in number) cannot be filtered without blurring the in-focus object. Our approach also requires each light source to be handled separately, so scenes with many lights are an issue. Our sampling rates and filter sizes based on frequency analysis may not always be sufficient to eliminate noise completely at high-variance pixels (Fig. 9c lower inset). Since frequency analysis is only effective on locally smooth surfaces, high-frequency bump or normal mapping cannot be handled. Spatial anti-aliasing cannot be done directly because our factorization requires that only one image sample per pixel be taken. But it is easy to do indirectly, for example, for $4\times$ AA, render the image at $4\times$ resolution, with $\Omega_{\text{pix}}^{\max} = 0.25$ (which requires fewer rays) and then downsample. An example of this is shown for the SIBENIK scene in the accompanying video.

11 Conclusion and Future Work

In this paper we demonstrated a method to adaptively reconstruct a Monte Carlo ray-traced image. Most previous methods based on frequency analysis handle one effect at a time; our method can reconstruct images with depth of field, area light direct and indirect illumination. Our novel factorization scheme allows pre-filtering noisy irradiance before multiplication with texture. We also introduced a new sampling strategy wherein the number of primary and secondary rays traced can be controlled independently, allowing an overall lower sampling rate. By analyzing the Fourier spectra of

the direct and indirect illumination light fields, we derived image space bandlimits for both the radiance and irradiance. Our render times are around 5 seconds, and our overhead is more than $50\times$ lower than state-of-the-art.

It could also be possible to combine our approach with a many-lights framework [Walter et al. 2006] to handle more general lighting environments. Finally, it would also be interesting to see if the approach can be extended for noisy caustics from highly specular surfaces.

12 Acknowledgements

We thank the reviewers for their helpful suggestions. This work was supported in part by NSF grant CGV #1115242, 1116303, and the Intel Science and Technology Center for Visual Computing. We thank NVIDIA for graphics cards, and fellowship support.

References

BELCOUR, L., SOLER, C., SUBR, K., HOLZSCHUCH, N., AND DURAND, F. 2013. 5D covariance tracing for efficient defocus and motion blur. *ACM Transanctions on Graphics 32*, 3, 31:1–31:18.

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

COOK, R., PORTER, T., AND CARPENTER, L. 1984. Distributed Ray Tracing. In *Proceedings of SIGGRAPH 84*, 137–145.

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.

DAMMERTZ, H., SEWTZ, D., HANIKA, J., AND LENSCH, H. P. A. 2010. Edge-avoiding À-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics*, 67–75.

DELBRACIO, M., MUSÉ, P., BUADES, A., CHAUVIER, J., PHELPS, N., AND MOREL, J.-M. 2014. Boosting monte carlo rendering by ray histogram fusion. *ACM Transactions on Graphics 33*, 1, 8:1–8:15.

DURAND, F., HOLZSCHUCH, N., SOLER, C., CHAN, E., AND SILLION, F. 2005. A Frequency Analysis of Light Transport. *ACM Transactions on Graphics 24*, 3, 1115–1126.

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 28*, 3, 93:1–93:13.

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, 9:1–9:13.

GERSHBEIN, R., SCHRÖDER, P., AND HANRAHAN, P. 1994. Textures and Radiosity: Controlling Emission and Reflection with Texture Maps. In *Proceedings of SIGGRAPH 94*, 51–58.

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 27*, 3, 33:1–33:10.

KAJIYA, J. 1986. The Rendering Equation. In *Proceedings of SIGGRAPH 86*, 143–150.

KALANTARI, N. K., AND SEN, P. 2013. Removing the noise in Monte Carlo rendering with general image denoising algorithms. *Computer Graphics Forum (Proc. of Eurographics 2013) 32*, 2, 93–102.

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, 55:1–55:12.

LEHTINEN, J., AILA, T., LAINE, S., AND DURAND, F. 2012. Reconstructing the indirect light field for global illumination. *ACM Transanctions on Graphics 31*, 4, 51:1–51:10.

LI, T.-M., WU, Y.-T., AND CHUANG, Y.-Y. 2012. SURE-based optimization for adaptive sampling and reconstruction. *ACM Transactions on Graphics 31*, 6, 186:1–186:9.

MAX, N. L., AND LERNER, D. M. 1985. A two-and-a-half-D motion-blur algorithm. In *Proceedings of SIGGRAPH 85*, 85–93.

MEHTA, S., WANG, B., AND RAMAMOORTHI, R. 2012. Axis-aligned filtering for interactive sampled soft shadows. *ACM Transactions on Graphics 31*, 6, 163:1–163:10.

MEHTA, S. U., WANG, B., RAMAMOORTHI, R., AND DURAND, F. 2013. Axis-aligned filtering for interactive physically-based diffuse indirect lighting. *ACM Transactions on Graphics 32*, 4, 96:1–96:12.

MITCHELL, D. 1991. Spectrally Optimal Sampling for Distribution Ray Tracing. In *SIGGRAPH 91*, 157–164.

OVERBECK, R., DONNER, C., AND RAMAMOORTHI, R. 2009. Adaptive Wavelet Rendering. *ACM Transactions on Graphics 28*, 5.

POTMESIL, M., AND CHAKRAVARTY, I. 1981. A lens and aperture camera model for synthetic image generation. In *Proceedings of SIGGRAPH 81*, 297–305.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. A Signal-Processing Framework for Inverse Rendering. In *Proceedings of SIGGRAPH 01*, 117–128.

RITSCHEL, T., ENGELHARDT, T., GROSCH, T., SEIDEL, H.-P., KAUTZ, J., AND DACHSBACHER, C. 2009. Micro-rendering for scalable, parallel final gathering. *ACM Transactions on Graphics 28*, 5, 132:1–132:8.

ROUSELLE, F., KNAUS, C., AND ZWICKER, M. 2012. Adaptive rendering with non-local means filtering. *ACM Transactions on Graphics 31*, 6, 195:1–195:11.

ROUSSELLE, F., KNAUS, C., AND ZWICKER, M. 2011. Adaptive sampling and reconstruction using greedy error minimization. *ACM Transactions on Graphics 30*, 6, 159:1–159:12.

SEN, P., AND DARABI, S. 2012. On filtering the noise from the random parameters in monte carlo rendering. *ACM Transactions on Graphics 31*, 3, 18:1–18:15.

SEN, P., DARABI, S., AND XIAO, L. 2011. Compressive rendering of multidimensional scenes. In *Proceedings of the 2010 International Conference on Video Processing and Computational Video*, Springer-Verlag, Berlin, Heidelberg, 152–183.

SHIRLEY, P., AND MORLEY, R. K. 2003. *Realistic Ray Tracing*, 2 ed. A. K. Peters, Ltd., Natick, MA, USA.

SHIRLEY, P., AILA, T., COHEN, J., ENDERTON, E., LAINE, S., LUEBKE, D., AND MCGUIRE, M. 2011. A local image reconstruction algorithm for stochastic rendering. In *ACM Symposium on Interactive 3D Graphics*, 9–14.

SOLER, C., SUBR, K., DURAND, F., HOLZSCHUCH, N., AND SILLION, F. 2009. Fourier depth of field. *ACM Transactions on Graphics 28*, 2, 18:1–18:12.

VAIDYANATHAN, K., MUNKBERG, J., CLARBERG, P., AND SALVI, M. 2014. Layered Light Field Reconstruction for Defocus Blur. *To appear in ACM Transactions on Graphics*. http://software.intel.com/en-us/articles/layered-light-field-reconstruction-for-defocus-blur.

WALTER, B., ARBREE, A., BALA, K., AND GREENBERG, D. 2006. Multidimensional lightcuts. *ACM Transactions on Graphics 25*, 3, 1081–1088.

WARD, G., AND HECKBERT, P. 1992. Irradiance Gradients. In *Eurographics Rendering Workshop 92*, 85–98.

A Appendix: Motion Blur

We now describe how the factored axis-aligned filtering and two-level adaptive sampling framework can be used for rendering motion blur with direct and indirect illumination. We assume no defocus blur; the combined analysis is left for future work.

Factoring: The equations for computing factored texture and irradiance are similar to eqn. 12. Lens coordinate u is replaced by time t, and the lens function is replaced by the shutter function. Instead of simply using the factoring error, for motion blur, factoring is enforced at all pixels with a single primary hit velocity. Pixels with two or more visible surfaces with different velocities are not factorizable.

Filtering: We follow the Fourier analysis for texture and irradiance under motion blur in [Egan et al. 2009]. At a given pixel, assume there is a single surface moving with image-space speed $v_p > 0$. The texture filter width is

$$\Omega^m_{x,p} = \min\left\{\Omega^{\max}_{\text{pix}}, \Omega^{\max}_{\text{t}}/v_p\right\} \tag{34}$$

The superscript 'm' denotes motion blur. The extra subscript 'p' indicates that this is a primary texture filter width. $\Omega^{\max}_{\text{t}}$ is the shutter bandwidth, inversely related to shutter open time. Similarly, if a static surface receives a shadow moving with image-space speed $v_s > 0$, then the irradiance (E_{dir}) filter width is

$$\Omega^m_{x,s} = \min\left\{\Omega^{\max}_{\text{pix}}, \Omega^{\max}_{\text{t}}/v_s\right\} \tag{35}$$

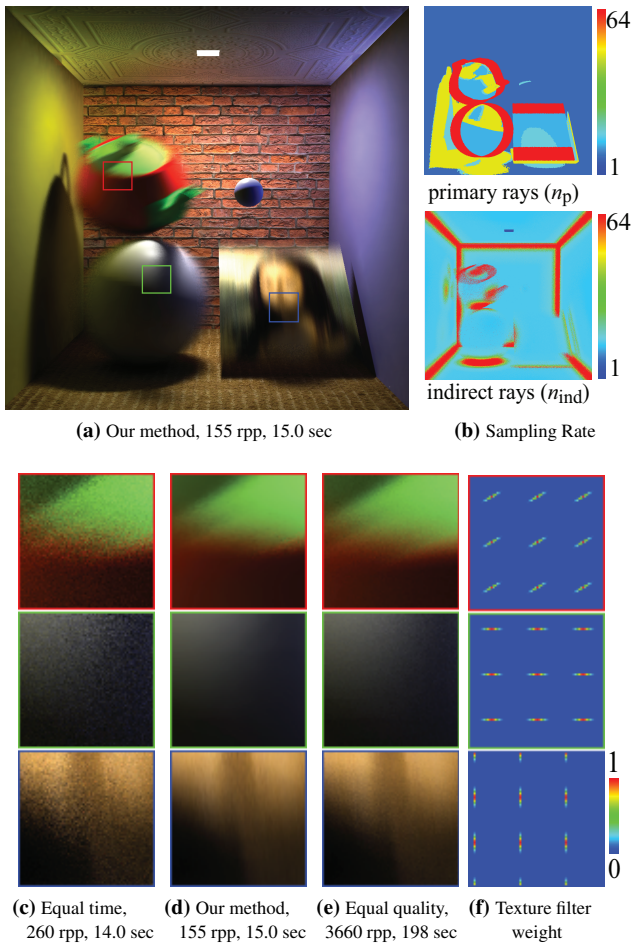
The subscript 's' indicates that this is a secondary, or irradiance filter width. These equations are analogous to eqn. 7 for defocus blur. Note that these filters are 1-D Gaussians in image space, oriented in the direction of the velocity $\mathbf{v}$, unlike the 2-D symmetric Gaussian defocus filter. The irradiance is also filtered according to the area-light filter width given in eqn. 18. For indirect illumination, we only apply the standard irradiance filter based on minimum reflector depth; this is found to filter out noise due to reflector motion with an adequate sampling rate. As stated before, factoring and filtering cannot be used at a pixel if there are two or more surfaces with different velocities. We apply a small 3×3 pixel-wide filter to the radiance to reduce noise at such pixels.
Recall that for defocus blur, the irradiance was pre-filtered and filtered again by the defocus filter after combining with texture. For motion blur, the motion blur filters are applied independently to texture and irradiance, since a moving surface may receive shadows that are not motion-blurred.

Sampling: In the first pass, we trace 9 paths per pixel. Since we only filter motion-blurred texture at a pixel with a single moving surface, the number of primary rays (second pass) is similar to eqn. 24:

$$n_p = (\Omega^{\max}_{\text{pix}} + \Omega^t_x)^2(1 + v\Omega^t_x)^2 \tag{36}$$

This equation is used only at pixels with a single moving visible surface and/or a single moving shadow. At pixels with more than one velocity for the primary hit or the shadow, we enforce a large constant primary sampling rate (64 rays), and apply a small 3×3 pixel-wide filter. The number of secondary shadow rays is similar to eqn. 25. The sampling rate for indirect can be computed similar to eqn. 26, but we found that the $(\Omega^*_t)^2$ term can be ignored.

Results: We implemented our algorithm for motion blur with soft shadows and indirect illumination on Intel's CPU-parallel Embree ray-tracer (since the Optix ray-tracer does not support motion-blur ray-tracing). In Fig. 13, we show a cornell box scene with textures and moving objects. The insets (c, d, e), from top to bottom, show a moving teapot, shadow of the moving teapot on a moving sphere, and shadow of a static sphere on a moving textured plane. We are able to filter noise from all kinds of motion blur effects while reducing ray-count $23\times$ and rendering time by $14\times$ compared to equal-quality stratified MC. In Fig. 13 (b) we show heatmaps of the primary and indirect sampling rate, and in (f) we show the texture filter weights used by specific pixels in the insets, to show how they filter using neighboring pixel values. These filters are aligned in the motion direction, unlike the isotropic defocus filter.



(a) Our method, 155 rpp, 15.0 sec **(b)** Sampling Rate

(c) Equal time, 260 rpp, 14.0 sec **(d)** Our method, 155 rpp, 15.0 sec **(e)** Equal quality, 3660 rpp, 198 sec **(f)** Texture filter weight

Figure 13: *A Cornell Box scene, with motion blur and area light direct and indirect illumination, rendered at* 1024×1024 *with an average 155 rays per pixel (rpp) in total 15.0 sec. The insets compare (c) equal time stratified MC with 260 rpp (d) our method, and (e) equal quality stratified MC with 3660 rpp (198 sec). In (b) we show the per-pixel primary and indirect sampling rates; (f) shows the filter weights used by certain pixels for their neighboring pixels.*