Axis-aligned Filtering for Interactive Physically-based Rendering

by

Soham Uday Mehta

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ravi Ramamoorthi, Chair
Professor James O'Brien
Professor Marty Banks

Spring 2015

Axis-aligned Filtering for Interactive Physically-based Rendering

Abstract

Axis-aligned Filtering for Interactive Physically-based Rendering

by

Soham Uday Mehta

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Ravi Ramamoorthi, Chair

Computer graphics rendering is undergoing a renaissance, with physically-based rendering methods based on accurate Monte-Carlo image synthesis replacing ad-hoc techniques in a variety of applications including movie production. In interactive applications like product visualization or video games, physically-based lighting effects are increasingly popular. However, producing photo-realistic images at interactive speeds still remains a challenge.

In Monte-Carlo rendering, a pixel's color is computed by sampling and integrating over a high-dimensional space. This includes effects like (1) motion blur, due to objects moving during the time the camera shutter is open; (2) defocus blur, due to camera lens optics; (3) area and environment map lighting, which is direct illumination coming from many directions; (4) global illumination, due to light reflected from one surface to another. The color is sampled through ray- or path-tracing. With insufficient rays, the image looks noisy because the integrand has high variance, and 1000s of rays are needed (per pixel) for a pleasing image. Previous work has showed a Fourier analysis for some of these effects, deriving a compact double-wedge spectrum, and a *sheared* filter that aligns with the slope of the spectrum. This filter can remove noise from a very sparsely sampled Monte-Carlo image, but is very slow. In this thesis, we will extend the Fourier analysis for more general cases, and propose a less compact *axis-aligned* filter, that aligns with the frequency axes. The resulting spatial bandwidths are then used for image-space filtering, that is orders of magnitude faster than sheared filtering. The packing of the Fourier spectra also provides adaptive sampling rates that minimize noise in conjunction with the adaptive filter. These algorithms improve speed relative to converged ground-truth by about $30-60\times$, and we are able to demonstrate interactive speed with a GPU ray-tracer. We also demonstrate an application of our method to mixed reality with a Kinect camera.

Contents

Acknowledgments

I want to thank my advisor Prof. Ravi Ramamoorthi for providing invaluable guidance in the progress of my work. His deep insights in the field of rendering were indispensable for the completion of this work.

I thank Prof. Fredo Durand from MIT who was a co-advisor of many of my projects. In particular, his expertise in Fourier analysis was extremely helpful. I also want to thank Prof. James O'Brien for providing advice at various points, and for having me as a GSI for the undergraduate Computer Graphics course, which was a great learning experience for me. I would also like to thank Prof. Carlo Sequin and Prof. Maneesh Agrawala for welcoming me into their classes and helping my knowledge of graphics grow. I also thank Prof. Alysoha Efros and Prof. Marty Banks for providing constructive criticism as my qualifying exam and dissertation committee members. I want to thank my colleagues Dr. Dikpal Reddy, Dr. Jiamin Bai, Dr. Michael Tao who provided great advice on graduate school. I want to thank my collaborators and friends Ling-Qi Yan, Brandon Wang and Eric Yao for their inputs and contributions.

This work was supported in part by NSF grant CGV 1115242, an NVIDIA PhD fellowship, GPU donations from NVIDIA, and funding from the Intel Science and Technology Center for Visual Computing.

Chapter 1

Introduction

The central focus of this thesis is speeding up photo-realistic rendering in 3-dimensional computer graphics. Rendering refers to converting a 3-dimensional virtual collection of objects with different reflective properties, viewed through a camera illuminated by some light sources, into a 2-dimensional red-green-blue image. The scene, i.e. objects, materials, lights and camera, is assumed to be known. Rendering a single image requires computing the light reaching the camera at each pixel, and each image typically contains about a million pixels.

1.1 Motivation

Computing the color at each pixel requires evaluation of a complicated integral involving light energy bouncing around in the scene, in the steady state. Specifically, we deal with photo-realistic rendering effects, commonly termed "distribution effects", such as

1. Soft Shadows (integrating over an area light),
2. Indirect Illumination (integrating over light reflected from other objects or surfaces),
3. Depth of Field (Integrating over the all points on the camera lens),
4. Motion Blur (Integrating over time for which the camera shutter remains open),
5. Environment Lighting (Integrating over distant lighting coming from all directions).

The integrand consists of a high-dimensional "light field" that is a function of coordinates in camera, light, and/or directional space. In most scenarios, the integrand light fields at adjacent or nearby pixels overlap across multidimensional domains. But, in traditional rendering, computational results are not shared between pixels. These multidimensional effects are expensive, but also crucial for realism in high quality offline rendering. Many approximate techniques have been proposed for each effect, but they introduce bias and fail to produce the accurate result. The correct, physically-based solution is only possible through Monte-Carlo path-tracing.

As the complexity within a pixel's multidimensional domain increases, the computation time required to render an image using Monte-Carlo path-tracing also increases. For example, for shadows cast by an area light, as the light source becomes larger each pixel sees more complex geometry and the occlusion signal has more variance. Capturing this variance, i.e. reducing the noise using previous methods requires computing more samples for each pixel, which increases render times. Another observation is that as the complexity of these effects increases, the final image content is often smooth. Intuitively, for shadows, the larger the size of the light source, the blurrier (softer) the shadows. This blur in turn removes high frequencies from the final image. Putting these two observations together we come to an ironic conclusion. Using simple Monte-Carlo ray-tracing, as the light size increases, more rays are required, but the complexity and spatial frequencies in the final image actually decrease due to the blurring or filtering from the light source – and we end up devoting more and more resources to compute a simpler and simpler result. One of the key insights is that as complexity increases inside of a single pixel, it is often true that there is a corresponding increase in overlap between the integral domains of nearby pixels. For example, nearby pixels see the same blockers casting shadows from the light source. Similar arguments can be applied for other distribution effects. At an intuitive level it seems obvious that we should be able to share information to reduce the total computation in these cases. However, robustly deriving how much information to share and how to share it *fast* is a more difficult problem, and it is the problem addressed by this thesis.

1.2 Contributions

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. Previous work has extended image denoising techniques to Monte-Carlo images. Other work has showed a Fourier analysis for some of these effects, deriving a compact filter that aligns with the slope of the spectrum. Although these methods based on sheared filtering, statistical denoising or light field reconstruction can denoise very sparsely sampled images, their practical filtering algorithms are very slow, making them unsuitable for interactive use. In this thesis, we make a different set of tradeoffs. We use a simple filter to reduce the number of Monte-Carlo samples considerably compared to brute force, but less than some previous methods. However, we benefit in having an extremely simple filtering step, which 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 ray-tracer. The final algorithm is essentially an image denoiser which is very simple to implement and can operate with minimal overhead. Our analysis also provides adaptive sampling rates which reduce noise throughout the image. Therefore, we are able to achieve interactive frame rates while obtaining the benefits of high quality ray-traced distribution effects.

Along with our practical contribution, we also make important theoretical contributions, consisting of novel results in the Fourier analysis of light fields for soft shadows, indirect illumination, defocus blur and environment illumination, showing that many spectra have a consistent double-wedge Fourier power spectrum. In the case of indirect illumination, we show that the light field has the double wedge spectrum even for non-parallel reflectors and receivers. In the case of environment illumination, the spectrum is the convolution of sheared Gaussians and is shaped like an ellipsoid. In each case, we show how to band-limit this spectrum based on the lighting, BRDF (diffuse and Phong) geometry terms. In the case of multiple effects, more than one bandlimit applies, so we separate the shading into texture and irradiance; our novel factorization scheme allows pre-filtering noisy irradiance before multiplication with texture.

We now state our specific contributions succinctly:

1. We study the multidimensional signals listed in Sec.1.1 above in both the primal (for e.g., space-angle) and Fourier (frequency) domains. We show the structure of these signals in the Fourier domain is similar, and is captured inside a "double-wedge" determined by some minimum and maximum slope lines determined by the geometry of the scene. We then show how this signal is bandlimited (See Sec.2.3) during intgeration. Previous work has used a "sheared" or sloped filter to bound the double wedge, instead, our resulting filter is "axis-aligned", meaning it is a rectangle with edges aligned with the coordinate axes. The axis-aligned filter effectively provides a filter width in image-space, allowing us to filter noisy images to produce smooth but accurate images.

2. Based on the different bandlimits, we derive minimum sampling rates needed to accurately reconstruct the image after filtering. This adaptive sampling scheme adds more samples in image regions with small filter sizes and vice versa. We also provide higher-dimensional adaptive sampling for the case of multiple simultaneous effects.

3. We provide fast implementations of our adaptive sampling and filtering schemes using ray-tracing and filtering on a Graphics Processing Unit (GPU). Our implementation runs at 5-10 frames per second for simple effects and 2-5 seconds per image for complex effects. This is about an order of magnitude faster than other state-of-the-art methods, for about the same accuracy. Relative to equal visual quality ground truth, we are able to render images 30 to $60\times$ faster.

1.3 Overview

This thesis is organized as follows. Chapter 2 gives the basic background theory for radiometry, rendering, Fourier analysis and filtering. Chapters 3-6 discuss our algorithm for the distribution effects listed in Sec. 1.1, including a detailed discussion of relevant previous work. Chapter 3 presents our technique for accurately rendering images with area light soft shadows at interactive speeds. We examine the shadow light field in the frequency domain, showing that it is limited to a double-wedge shape, and propose a novel axis-aligned filter that reduces to filtering the shading directly in image space. Our filter also provides adaptive per-pixel sampling rates. Chapter 4 extends and expands these filtering ideas to reducing the number of rays when calculating indirect illumination at a pixel. The light field in this case requires a different parameterization but reduces to the same double-wedge model, allowing us to apply the axis-aligned filter for fast filtering. Chapter 5 considers a more complex case with simultaneous defocus(or motion) blur, and soft shadows and indirect illumination. We apply a two-level axis-aligned filter, in combination with two-level adaptive sampling to separate the effects of lens (time), and area light and BRDF, allowing us to reconstruct the image in a few seconds. Finally, in Chapter 6, we consider environment illumination. This requires a different curvature-dependent parameterization. The shading spectrum must take both illumination and visibility into account, and we show that it is no longer a double-wedge, but an ellipsoid. We demonstrate an application of our filter to a new domain in rendering, namely mixed reality. In mixed reality, virtual objects are added to a real-world video stream, with physically correct shading – both virtual and real objects affect each other's appearance. Chapter 7 summarizes our work, and suggests possible future work.

Chapter 2

Background

Rendering is the process of simulating how light travels (light transport) and is recorded on the camera film. In this thesis, we will consider scenes with solid surfaces and no atmospheric (volumetric) effects. The basic operations for light transport are travel through free space and reflection. Understanding light transport requires defining certain units of measurement, as explained below.

2.1 Radiometry

Radiometry is the study and quantification of light energy. Two of the most common measurements of light are radiant flux and radiance. **Flux** Φ is a measurement of power, which is simply energy per time (Watts). The energy of light is usually distributed unevenly across wavelengths, so Φ is also a function of wavelength λ, and if it varies with time, also a function of time t. All following radiometric quantities are also function of λ and t. For our analysis, this dependence is ignored since λ and t do not change.

Radiance L is a measure of flux across a differential surface area dA, across a differential solid angle $d\omega$, measured in $W \cdot m^{-2} \cdot Sr^{-1}$:

$$L = \frac{\Phi}{dA \cos\theta d\omega} \tag{2.1}$$

with θ being the angle between the normal $\mathbf{n}$ of the differential surface dA and the differential solid angle $d\omega$. This essentially measures light through a cylinder in the limit as the cylinder becomes infinitely thin. One of the most convenient properties is that radiance is constant along a ray in free space. Throughout this thesis, we will study the structure of the radiance **light field**, that is, the full 4-dimensional radiance field varying with space ($\mathbf{x}$ on a surface) and direction ω.

Irradiance E is the radiant flux received by a surface per unit area (measured in $W \cdot m^{-2}$ and can be computed from the radiance as:

$$E(\mathbf{x}) = \int_{H_+} L(\mathbf{x}, \omega_i) \cos\theta_i \, d\omega_i \tag{2.2}$$

Here H_+ specifies the upper visible hemisphere of possible incoming light directions around the surface normal $\mathbf{n}$.

How light interacts with surfaces is modeled by the bidirectional reflectance distribution function, or **BRDF** [69]. The BRDF $\rho(\omega_i, \omega_o)$ is classically defined to be a 4D function that relates the differential incoming irradiance E_i along a direction ω_i to the differential outgoing radiance L_o along a direction ω_o, at a given spatial location:

$$\rho(\omega_i, \omega_o) = \frac{dL_o(\omega_o)}{dE_i(\omega_i)} = \frac{dL_o(\omega_o)}{L_i(\omega_i)\cos\theta_i \, d\omega_i}. \tag{2.3}$$

One of the most commonly used BRDF models is the diffuse or Lambertian BRDF, $\rho(\omega_i, \omega_o) = k_d$, where k_d is a spatially varying RGB texture. That is, the reflectance is independent of incoming and outgoing directions. A commonly used Glossy (shiny) BRDF model is the Phong BRDF

$$\rho(\omega_i, \omega_o) = k_s(\mathbf{r} \cdot \omega_o)^n \tag{2.4}$$

where $\mathbf{r}$ is the vector ω_i reflected about the normal $\mathbf{n}$, and n is a numeric exponent. Both models produce realistic-looking images, but are not physically-based. Modern renderers use more sophisticated models such as the Torrance-Sparrow BRDF.

2.2 The Rendering Equation

The radiance reflected from a surface is defined by the reflection equation. This equation computes the outgoing radiance L_o by integrating the product of the BRDF ρ and incoming radiance L_i:

$$L_o(x, \omega_o)) = \int_{H_+} \rho(\omega_i, \omega_o) L_i(x, \omega_i) \cos\theta_i \, d\omega_i \tag{2.5}$$

As an approximation we can assume that the incoming light L_i only comes from light emitters in the scene (ignoring reflections from other surfaces). Simulating this subset of light paths is called direct illumination, and can be computed directly from the reflection equation. Note that visibility of the light source in any direction ω_i is unknown (scene-dependent).

In the real world, light may reflect any number of times from surfaces before reaching the camera pixel. Fully simulating this is more expensive, and is called global illumination. In this case the outgoing light L_o and incoming light L_i are related by $L_i(\mathbf{x}, \omega_i) = L_o(R(\mathbf{x}, \omega_i), -\omega_i)$, where $R(\mathbf{x}, \omega_i)$ is the closest surface location looking from point x towards the ω_i direction. By also including light emitters L_e we can then write the full rendering equation [44] as follows:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{H+} \rho(\omega_i, \omega_o) L_o(R(\mathbf{x}, \omega_i), -\omega_i) \cos\theta_i \, d\omega_i \tag{2.6}$$

Note that unknown surface-outgoing radiance L_o appears on both sides of the equation. Closed form solutions to this equation can only be computed for the simplest of scenes.

For general scenes, solutions to equations 2.5 and 2.6 can be found by **path-tracing**. Specifically, we trace rays that simulate light paths and determine light-source visibility in eqn. 2.5 and closest visible surface in eqn. 2.6. Usually, the set of feasible directions is infinite, so in theory infinite rays would be required to estimate the integrals. In practice, **Monte-Carlo** ray-tracing is used, wherein we only sample a random subset of directions. For more details on Monte-Carlo ray-tracing for different problems in rendering, see [77].

2.3 Fourier Analysis and Filtering

In this thesis, we study the properties of the radiance light-field $L(\mathbf{x}, \omega_i)$ for various situations. This light field is structured and sparse (low-dimensional), so we find it suitable to study it in the Fourier or frequency domain. We define the **Fourier Transform** of a 1D function $f(x)$ to be:

$$\hat{f}(\Omega) = (2\pi)^{-1} \int_{-\infty}^{\infty} f(x) e^{-jx\Omega} dx, \tag{2.7}$$

where Ω is the frequency. In general, $\hat{f}$ is a complex-valued function. Fourier transforms of higher-dimensional functions are defined analogously. Note that the original function f can also be recovered from its Fourier transform $\hat{f}$:

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\Omega) e^{jx\Omega} d\Omega \tag{2.8}$$

Here j is the complex unit $\sqrt{-1}$. The power-spectral-density **PSD** of a signal in the Fourier domain is simply the absolute value of its fourier transform squared, such as $||\hat{f}(\Omega)||^2$. Most light fields and other signals encountered in rendering are bandlimited, that is the PSD is non-zero only in a small range of frequencies. Specifically, we say f has a bandlimit B if

$$\int_{-B}^{B} ||\hat{f}(\Omega)||^2 d\Omega > 0.99 \int_{-\infty}^{\infty} ||\hat{f}(\Omega)||^2 d\Omega \tag{2.9}$$

.

If an unknown signal has bandlimit B, it can be accurately reconstructed throughout its range from discrete samples obtained at intervals of $(2B)^{-1}$. This is the famous Shannon-Nyquist sampling theorem and the minimum sampling rate is termed **Nyquist rate**. We will often try to determine the Nyquist rate for sampling the unknown light field to reduce computation.

A **filter** is a function that multiplies a Fourier spectrum usually to isolate a part of the frequency range. In rendering, we typically use a low-pass filter, that is non-zero within some frequency range $[-B, +B]$. Mathematically, applying a filter $\hat{h}$ to an input signal spectrum $\hat{g}$ gives:

$$\hat{f}(\Omega) = \hat{g}(\Omega)\hat{h}(\Omega). \tag{2.10}$$

In practice, we apply the filter directly in spatial domain, since it doesn't require us to compute any Fourier transform explicitly. Spatial-domain filtering is a convolution operation:

$$f(x) = \int g(x-u)h(u)\,du. \tag{2.11}$$

For more details on Fourier theory, refer to a basic text such as [31].

Chapter 3

Soft Shadows

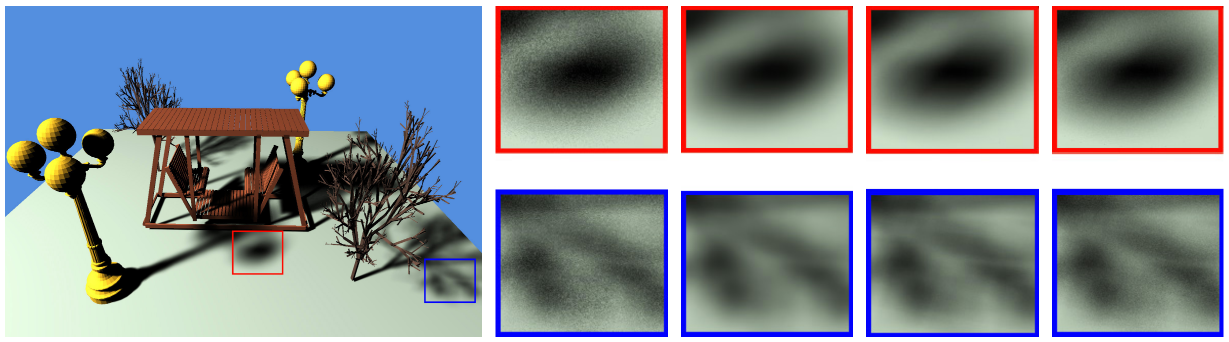
3.1 Introduction

This chapter focuses on accurate and efficient rendering of soft shadows from planar area lights. Soft shadows are a key effect in photorealistic rendering, and important to set the tone of a scene, but are expensive because every location on the area light must be considered and sampled. While real-time techniques [40, 43] 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 [76]. However, the number of samples per pixel for soft shadows remains too large for interactive use.

We build on [25] to significantly reduce the number of Monte Carlo samples needed, while still keeping the benefits of accurate raytraced occlusion. [25] 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 thesis, 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 $4\times$-$10\times$.

The main benefit is that overhead is minimal (about 5 milliseconds), even in GPU raytracers,



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

Figure 3.1: (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.

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. 3.1).

3.2 Previous Work

Real-Time and Accelerated Soft Shadows: The shadow mapping method [107] can be extended to soft shadow maps that consider occlusion from the entire area source [34, 3]. As noted in [43], these methods make various tradeoffs of speed and accuracy. Soler and Sillion [95] provide an analytic solution, but only for geometry in parallel planes. Shadow volumes [17] can also be extended to soft shadows using geometric ideas like penumbra wedges [6, 53] An analytic approach based on beam tracing is proposed by [73], but is not yet fast enough for real-time use, especially on complex scenes. Another body of work is precomputed relighting [93], 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.

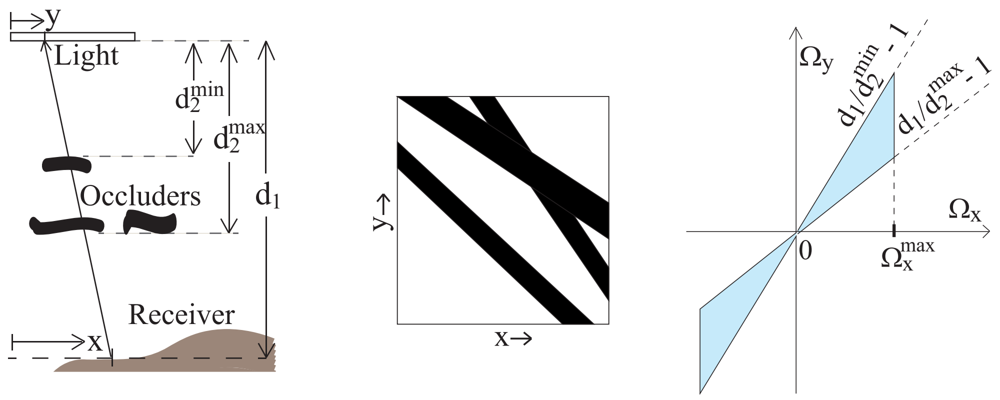
Monte Carlo and Ray-Traced Shadows: A number of accelerations that exploit coherence have been proposed [38, 2], as well as methods to separate near and far-field effects [5], prefilter visibility [52] and accelerate ray packets [11]. In contrast, we directly leverage a GPU raytracing framework in Optix [76] but add an adaptive image-space filter in post-processing. A few recent works have explored GPU acceleration of occlusion queries [27, 28] and sampled visibility [92]. 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 [35], and more recently [37] and [72]. We build most closely on recent approaches for frequency analysis of shadows and light transport [13, 23, 54]. In particular, Egan et al. [25] 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 [26, 96, 24]. 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. [56] 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 [87, 62, 108]. Recently, [88] 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 [18]. 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.

3.3 Background

We now briefly introduce the basic spatial and frequency domain analysis of the occlusion function. The ideas are summarized in Fig. 3.2. 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.



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

Figure 3.2: (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.

Assumptions: We use y for light coordinates and x for coordinates on a parameterization plane parallel to the light source as shown in Fig. 3.2(a); Our goal is to compute the result $h(x)$,

$$h(x) = r(x) \int f(x,y) l(y)\, dy, \tag{3.1}$$

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 [95, 25]. 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 [24], 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. 3.11(a).

Occlusion Function: From the geometry of Fig. 3.2(a), we can derive the 2D occlusion function $f(x,y)$ in terms of a 1D visibility ($g(\cdot)$ is defined along the occluder plane parallel to the light plane),

$$f(x,y) = g\left(\frac{d_2(x-y)}{d_1} + y\right), \tag{3.2}$$

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 3.2; the slope of the

[1]Note that [25] 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.

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. 3.2(b).

Fourier Analysis: The Fourier spectrum for equation 3.2 lies on a line with slope $s = (d_1/d_2) - 1$ (orthogonal to the spatial domain slope in Fig. 3.2(b)). In terms of Fourier spectra F and G,

$$F(\Omega_x, \Omega_y) = \frac{d_1}{d_2}\delta(\Omega_y - s\Omega_x)G\left(\frac{d_1}{d_2}\Omega_x\right). \tag{3.3}$$

As noted in [13, 25], 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. 3.2(c).[2] Maximum and minimum slopes respectively are

$$s_{\max} = \frac{d_1}{d_2^{\min}} - 1 \qquad s_{\min} = \frac{d_1}{d_2^{\max}} - 1. \tag{3.4}$$

3.4 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. 3.6).

3.4.1 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 $\Omega_{\text{g}}^{\max}$ in our analysis. In particular, we first introduce a parameter α, so that

$$\alpha = \frac{\Omega_{\text{g}}^{\max}}{\Omega_{\text{pix}}^{\max}} \in (0, 1], \tag{3.5}$$

[2]As discussed in [25], 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.

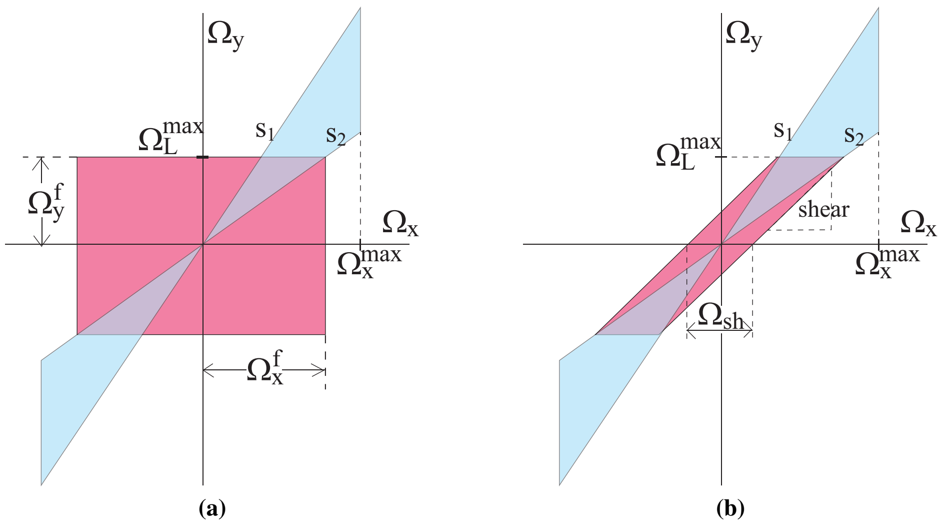


Figure 3.3: Schematic of (a) Axis-Aligned Filter, (b) Sheared Filter.

where Ω_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 $\Omega_g^{max} = \Omega_{pix}^{max}$. From equation 3.3,

$$\Omega_x^{max} = \alpha \frac{d_2^{max}}{d_1} \Omega_{pix}^{max} = \alpha(1 + s_{min})^{-1}\Omega_{pix}^{max}. \tag{3.6}$$

It is also useful to know the Ω_y bandlimit of the occlusion function. From the geometry of Fig. 3.2(c), it is clear that $\Omega_y = s\Omega_x$, which leads to $(d_1/d_2 - 1)\alpha(d_2/d_1)\Omega_{pix}^{max}$, and

$$\Omega_y^{max} = \alpha \left(1 - \frac{d_2^{min}}{d_1}\right) \Omega_{pix}^{max}. \tag{3.7}$$

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 Ω_L^{max} will be filtered (Fig. 3.3(a)).

We assume the light is a Gaussian of standard deviation σ meters, so $\Omega_L^{max} = (1/\sigma)\ m^{-1}$. 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 work 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. 3.11(b)).

3.4.2 Frequency Extent of Axis-Aligned Filter

We define the axis-aligned filter in the frequency domain as $[-\Omega_{\mathrm{x}}^{f}, \Omega_{\mathrm{x}}^{f}] \times [-\Omega_{\mathrm{y}}^{f}, \Omega_{\mathrm{y}}^{f}]$. As shown in Fig. 3.3(a),

$$\Omega_{\mathrm{x}}^{f} = \min\left[\frac{\Omega_{\mathrm{L}}^{\max}}{s_{\min}}, \Omega_{x}^{\max}\right] \qquad \Omega_{\mathrm{y}}^{f} = \Omega_{\mathrm{L}}^{\max}. \tag{3.8}$$

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. 3.3(a) is larger than the sheared filter in Fig. 3.3(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.

3.4.3 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 $\beta(x) = 1/\Omega_{\mathrm{x}}^{f}$.

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 3.1 (we also omit $r(x)$ in equation 3.1 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,

$$h(x) = \int_{x'} \int_{y} w(x - x'; \beta(x)) \bar{f}(x', y) w(y; \gamma(y))\, dy\, dx', \tag{3.9}$$

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

[4]As described in equation 3.13, we actually find it better to use a value $\beta(x) = k^{-1}\Omega_{\mathrm{x}}^{f}$ where k accounts for the Gaussian energy being spread over multiple standard deviations, and we usually set $k = 3$.

simple Gaussian filter,

$$\begin{aligned}\bar{h}(x') &= \int_y \bar{f}(x',y)l(y)\,dy \\ h(x) &= \int_{x'} \bar{h}(x')w(x-x';\beta(x))\,dx'.\end{aligned} \tag{3.10}$$

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

$$w(x-x';\beta) = \frac{1}{\sqrt{2\pi}\beta}\exp\left[-\frac{\| x-x' \|^2}{2\beta^2}\right], \tag{3.11}$$

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,

$$\| \mathbf{x}-\mathbf{x}' \|^2 = | \mathbf{x}-\mathbf{x}' |^2 - \left[\mathbf{n}\cdot(\mathbf{x}-\mathbf{x}')\right]^2. \tag{3.12}$$

Discussion: The simplicity of equation 3.10 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. 3.6).

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 3.4 and 3.8 as,

$$\boxed{\beta(x) = \frac{1}{k}\cdot\frac{1}{\mu}\max\left[\sigma\left(\frac{d_1}{d_{2,\max}}-1\right), \frac{1}{\Omega_{\mathrm{x}}^{\max}}\right],} \tag{3.13}$$

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. 3.5.2.

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. 3.7(a,c)). Also, the lower frequency the light (the larger σ is), the more we can filter.

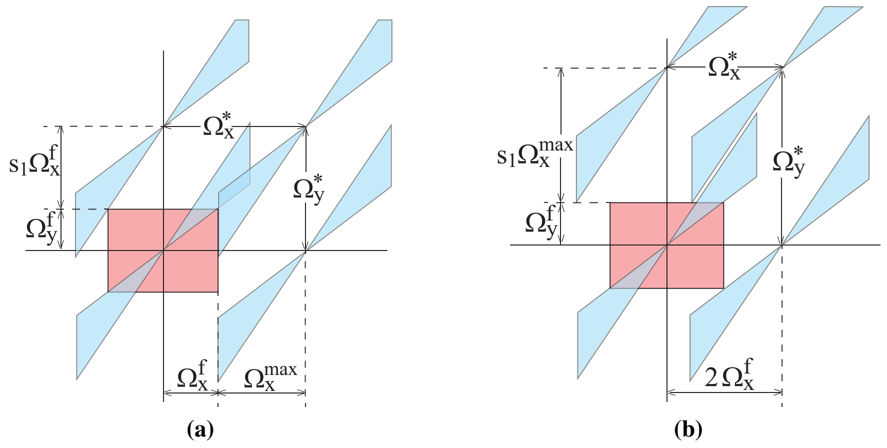


Figure 3.4: Packing of spectra for axis-aligned filtering. The packing in (a) is denser, and will be used in the rest of our analysis.

3.5 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. 3.4.

Packing (Fig. 3.4)	Ω_x^*	Ω_y^*
(a)	$\Omega_x^f + \Omega_x^{\max}$	$\Omega_y^f + \min(s_{\max}\Omega_x^f, \Omega_y^{\max})$
(b)	$2\Omega_x^f$	$\Omega_y^f + s_{\max}\Omega_x^{\max}$

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

$$(\Omega_{\rm x}^{*} \times \Omega_{\rm y}^{*})(a) - (\Omega_{\rm x}^{*} \times \Omega_{\rm y}^{*})(b) = (\Omega_{\rm x}^{\max} - \Omega_{\rm x}^{f})(\Omega_{\rm y}^{f} - s_{\max}\Omega_{\rm x}^{f}) \leq 0 \tag{3.14}$$

since $\Omega_{\rm x}^{\max} \geq \Omega_{\rm x}^{f}$ and $s_{\max}\Omega_{\rm x}^{f} = s_{\max}\Omega_{\rm y}^{f}/s_{\min} \geq \Omega_{\rm y}^{f}$. So, (a) represents a tighter packing than (b). It also corresponds intuitively to sampling at pixel resolution in x (if $\Omega_{\rm x}^{\max} \approx \Omega_{\rm pix}^{\max}$). In fact, as we shall see next, we sample at each pixel as in standard Monte Carlo, and use the analysis above to set per-pixel sampling rates.

3.5.1 Per-Pixel Sampling Rate

Extending the 2D analysis to 4D, the sampling rate is given by $(\Omega_{\rm x}^{*})^2(\Omega_{\rm y}^{*})^2$. 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),

$$n = (\Omega_{x}^{*})^2(\Omega_{y}^{*})^2 \times A_p \times A_l. \tag{3.15}$$

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

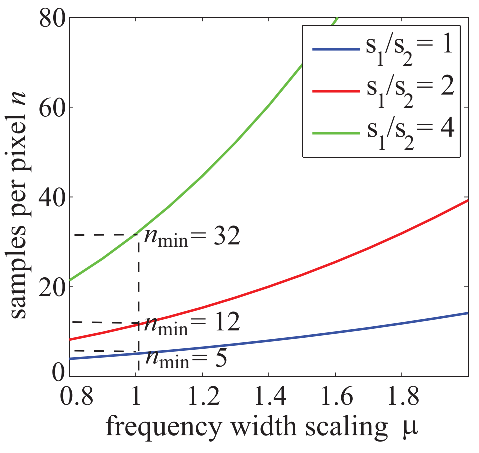
In the common case (where we need not consider the min/max expressions), we know that $\Omega_{\rm x}^{f} = \Omega_{\rm L}^{\max}/s_{\min}$, $\Omega_{\rm y}^{f} = \Omega_{\rm L}^{\max}$ (equation 3.8), and $\Omega_{\rm x}^{\max}$ is given by equation 3.6. Hence, the sampling rates for packing scheme (a) in Fig. 3.4 are

$$\begin{aligned}
(\Omega_{\rm x}^{*})^2 \times A_p &= \left[\frac{\Omega_{\rm L}^{\max}}{s_{\min}} + \alpha(1+s_{\min})^{-1}\Omega_{\rm pix}^{\max}\right]^2 \times A_p \\
&= \left(\frac{2}{s_{\min}}\sqrt{\frac{A_p}{A_l}} + \alpha(1+s_{\min})^{-1}\right)^2 \\
(\Omega_{\rm y}^{*})^2 \times A_l &= \left[\Omega_{\rm L}^{\max}\left(1+\frac{s_{\max}}{s_{\min}}\right)\right]^2 \times A_l = 4\left(1+\frac{s_{\max}}{s_{\min}}\right)^2,
\end{aligned} \tag{3.16}$$

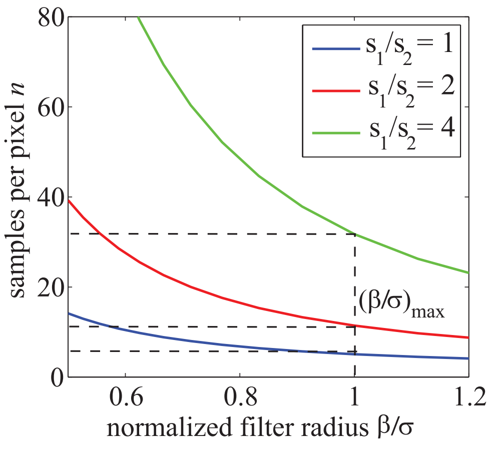
from which we can derive the final sampling rate

$$\boxed{n = 4\left(1+\frac{s_{\max}}{s_{\min}}\right)^2\left(\frac{2}{s_{\min}}\sqrt{\frac{A_p}{A_l}} + \alpha(1+s_{\min})^{-1}\right)^2.} \tag{3.17}$$

Note that the sampling rate depends on $s_{\max}$ and $s_{\min}$ and is therefore spatially-varying (adaptive for each pixel), as shown in Fig. 3.7(b,d).



(a)



(b)

Figure 3.5: 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.

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 $\Omega_y^* = 2\Omega_y^{max}$; substituting equation 3.7,

$$\begin{aligned} n_{\text{nofilter}} &= 4\left[\alpha\left(1-\frac{d_2^{\min}}{d_1}\right)\Omega_{\text{pix}}^{\max}\right]^2 \times A_l \\ &= 4\left[\alpha\left(1-\frac{d_2^{\min}}{d_1}\right)\right]^2 \frac{A_l}{A_p}, \end{aligned} \tag{3.18}$$

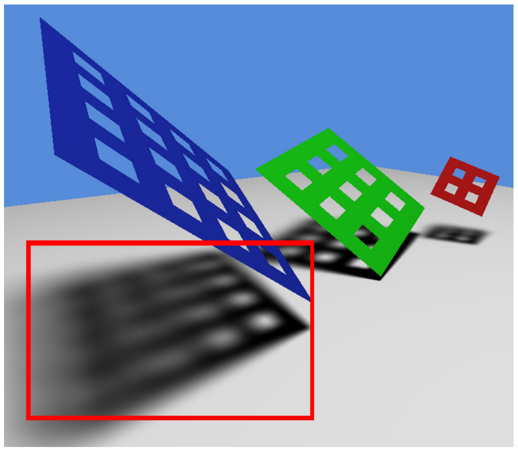
which can be very large because of the A_l/A_p factor, since the light source area is usually much larger than the area seen by a single image pixel. While Monte Carlo analysis is typically in terms of statistical noise reduction (although see [22] for a Fourier view), this frequency analysis also helps 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 [25] but not used in their actual algorithm) under the same conditions, is

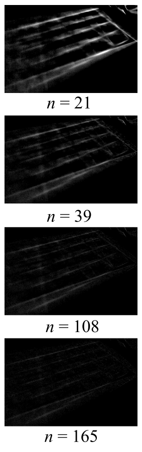
$$n_{\text{shear}} \approx 4\left(1-\frac{s_{\min}}{s_{\max}}\right)^2 \left(\Omega_{\text{sh}} \cdot d + \alpha(1+s_{\min})^{-1}\right)^2, \tag{3.19}$$

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 3.17,

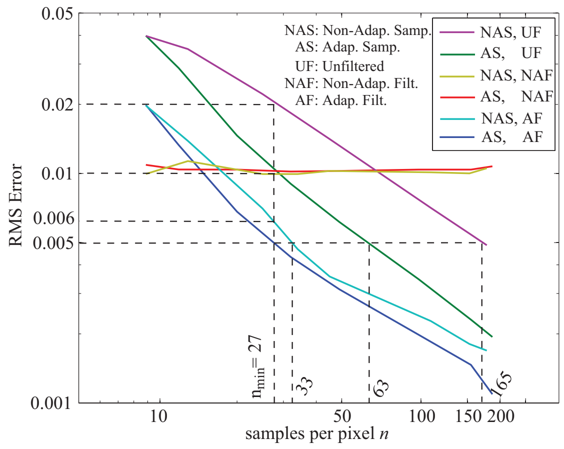
$$n_{\text{axis}} = 4\left(1+\frac{s_{\max}}{s_{\min}}\right)^2 \left(\Omega_x^f \cdot d + \alpha(1+s_{\min})^{-1}\right)^2, \tag{3.20}$$



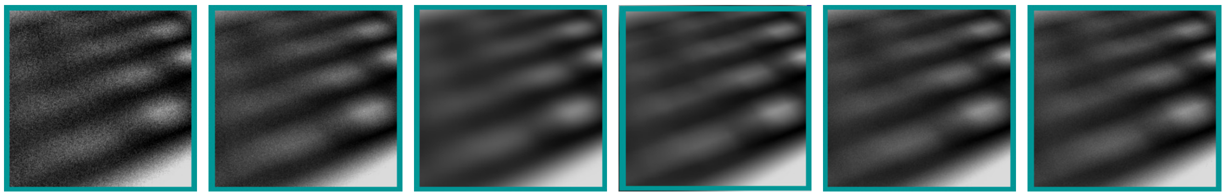
(a) our method, 27 samples per pixel (spp)



(b) error vs. spp



(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)** gr. truth **(h)** equal error, 63 spp, AS, UF **(i)** equal error, 165 spp, NAS, UF

Figure 3.6: (a) The 'Grids' scene, (b) Difference images (scaled up $20\times$*) 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).*

It can be seen that $n_{\text{shear}} < n_{\text{axis}}$ as expected, since the sheared filter scale $\Omega_{\text{sh}} < \Omega_x^f$ and $s_{\min} < s_{\max}$. 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.

3.5.2 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 $\beta \sim 1/\Omega_{\mathrm{x}}^{f}$. We will denote $\Omega_{\mathrm{x}}^{f} = \mu \Omega_{\mathrm{x}0}^{f}$, where μ measures how much the critical spatial filter $\Omega_{\mathrm{x}0}^{f}$ is scaled. Making the parameter μ explicit in equation 3.17,

$$\boxed{n(\mu) = 4\left(1+\mu\frac{s_{\max}}{s_{\min}}\right)^2 \left(\mu\frac{2}{s_{\min}}\sqrt{\frac{A_p}{A_l}} + \alpha(1+s_{\min})^{-1}\right)^2.} \tag{3.21}$$

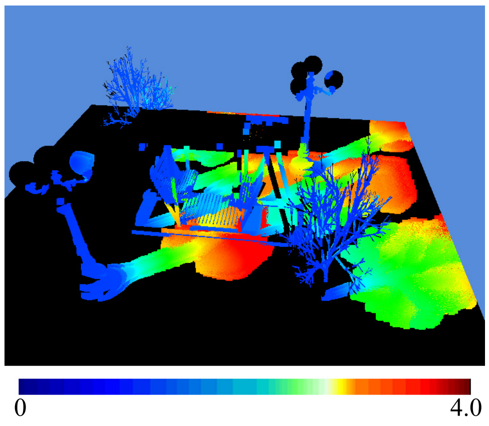
Given a desired value of μ, we can find the number of samples n (we typically use $\mu = 2$). Note that the number of samples is a quadratic function of μ, as shown in Fig. 3.5(a).

The variation of $\beta = 1/\Omega_{\mathrm{x}}^{f}$ with the number of samples per pixel n is shown in Fig. 3.5(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.

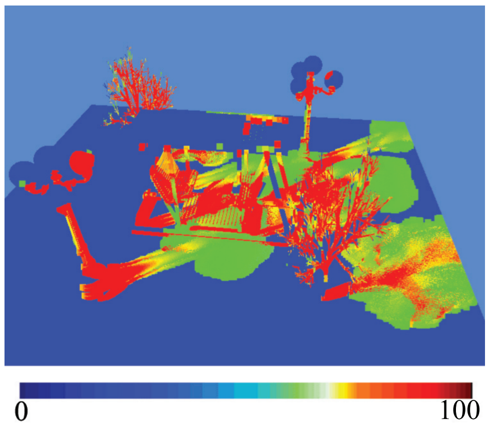
3.5.3 Evaluation and Discussion

Figure 3.6 evaluates our method on the grids scene (taken from [25]). While the geometry is simple, the shadows are intricate. Figure 3.6(a) shows that our method produces smooth results without artifacts, while Fig. 3.6(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. 3.6(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. 3.6).

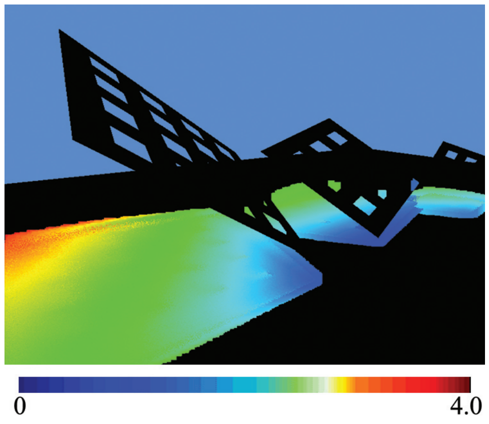
First, consider unfiltered (UF) results, both with standard Monte Carlo (non-adaptive sampling NAS) and adaptive sampling AS. We clearly see that our adaptive sampling method significantly reduces error. We can also run NAS and AS with a fixed non-adaptive filter (NAF), with the filter width chosen to 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. 3.4. 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. 3.6(b)).



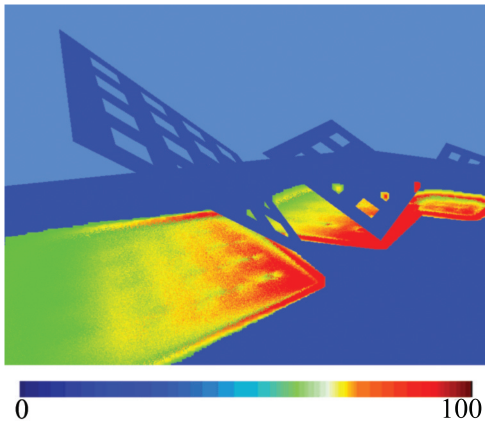
(a) ‘Bench’, scale β



(b) ‘Bench’, spp n



(c) ‘Grids’, scale β



(d) ‘Grids’, spp n

Figure 3.7: 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.

Finally, we show equal sample and equal error images in the bottom row of Fig. 3.6. Our method has almost no noise, and is visually superior even to equal error comparisons that exhibit some noise.

3.6 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

			Base Raytraced Occlusion			Our Algorithm			Total	
scene	tris	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 3.1: 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.*

equations 3.13 and 3.21 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 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 3.13, visualized in Fig. 3.7(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 3.21, visualized in Fig. 3.7(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. 3.6(e)). We then filter to obtain the final image h per equation 3.10 (such as Fig. 3.6(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 3.11 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 discontinu-

[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 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 [25].

ities are handled automatically by the Gaussian filter in equation 3.11. 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 3.11 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.

3.7 Results

We tested our method on four scenes of varying complexity, including some used in previous papers [74, 25]. 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 3.1 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 [56] and 4-5 orders of magnitude faster than the minutes of overhead in [25]. 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 ver-

[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.

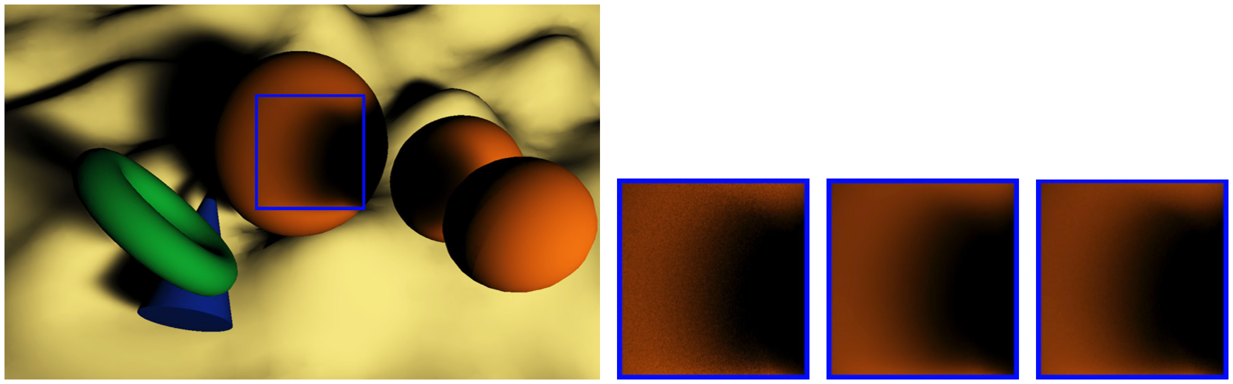

(a) our method, 30 spp **(b)** equal time, unfiltered, 30 spp **(c)** our method, 30 spp **(d)** equal error, 200 spp

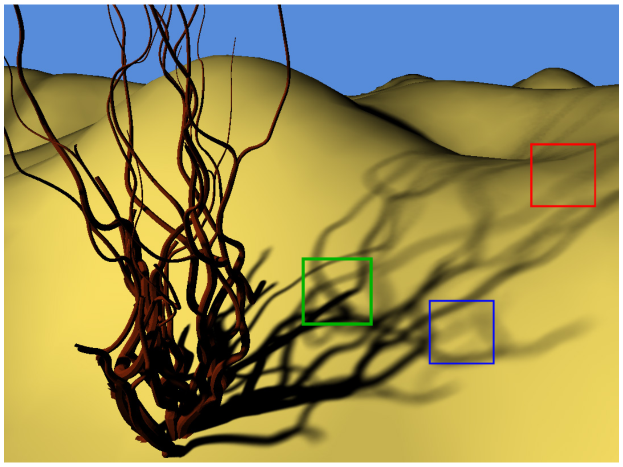
Figure 3.8: Interactive soft shadows from a curved occluder onto a curved receiver, with an average of 30 samples per pixel.

tices. This is an order of magnitude faster than what brute-force OptiX raytracing can achieve, and enables raytraced soft shadows at interactive rates.

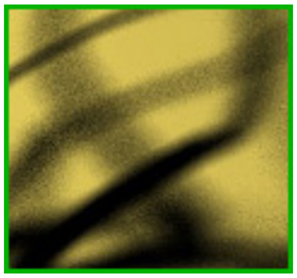
Accuracy: Our scenes are chosen to show a variety of soft shadow effects. First, Fig. 3.8 shows that we can accurately capture curved objects casting shadows on other curved objects, and that an equal sample standard raytrace is considerably noisier. We have already evaluated the grids scene, which has soft shadows of varying sizes, in Fig. 3.6. Our method does not under or overblur, and reproduces accurate soft shadows across the entire image. Figure 3.1 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. 3.9 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.

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 [92].

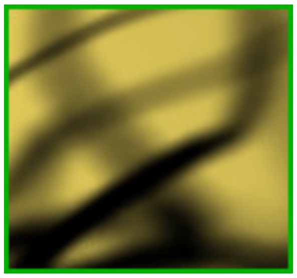
Comparison to De-Noising: Our final algorithm is essentially a denoising filter on the base Monte Carlo image. Therefore, we include a comparison with image de-noising strategies in Fig. 3.10. The fixed-width gaussian and bilateral filters (with parameters chosen by trial and error to produce the best visual results) cannot achieve the accuracy of our adaptive filter. In particular, a



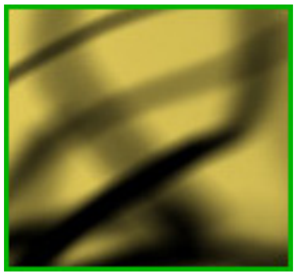
(a) our method, 23 spp



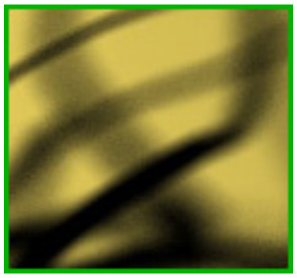
(b) equal time, unfilt., 23 spp



(c) our method, 23 spp



(d) ground truth, 5000 spp



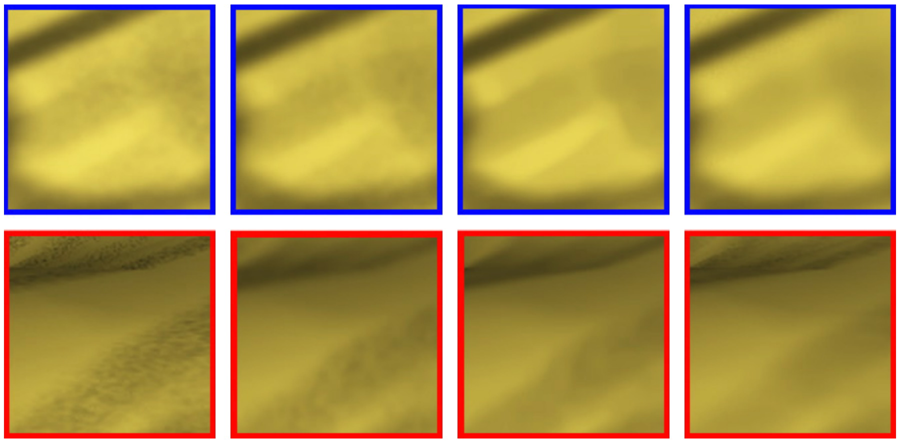
(e) equal error, 200 spp

Figure 3.9: 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.

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 [18] 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. 3.10. Moreover, that method (like most other previous works) is not designed for efficiency, taking 8 seconds.

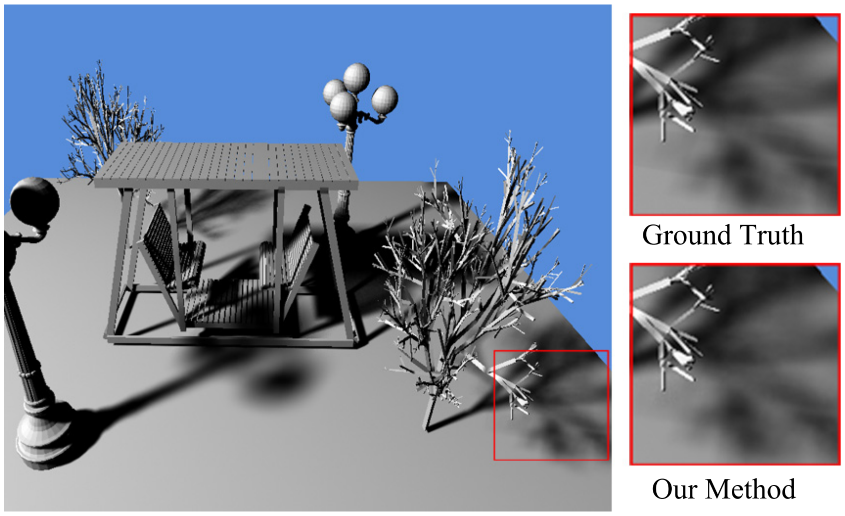
Extensions: Initial tests indicate the method extends in practice to cosine-falloff shading and non-Gaussian lights, without any modifications to the adaptive sampling or filtering steps. In Fig. 3.11(a), we show the bench scene rendered with cosine falloff (the scene is shown in grayscale to avoid masking subtle shading effects). In Fig. 3.11(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.



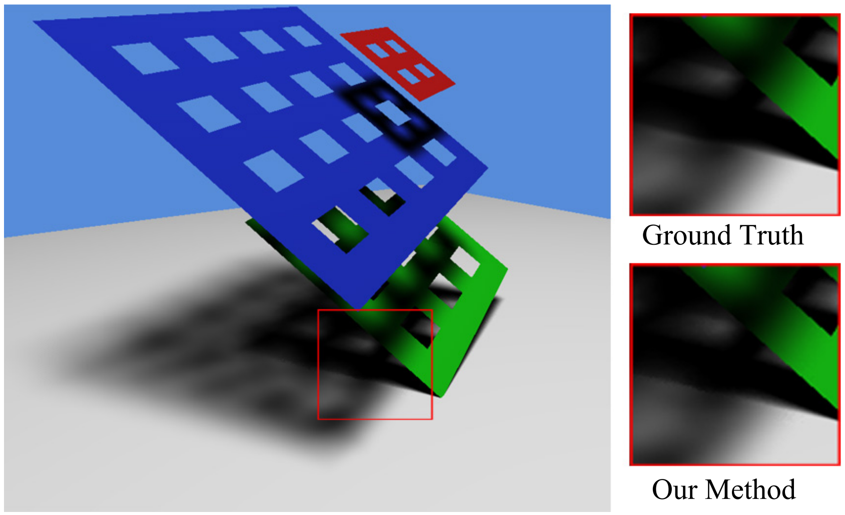
(a) fixed blur, 0.003 sec **(b)** bilateral filter, 200 sec **(c)** bm3d, 8 sec **(d)** our method, 0.005 sec

Figure 3.10: 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.

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. 3.6), converging rapidly to ground truth. The framerates in table 3.1 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.



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



(b) uniform intensity square light, 30 spp

Figure 3.11: 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).

Chapter 4

Diffuse Indirect Illumination

4.1 Introduction

The previous chapter focused on interactive rendering of *direct* illumination from an area light. Interactive rendering of *indirect* illumination is also one of the main challenges of computer graphics. In this chapter, we take an important step towards solving this problem for diffuse inter-reflections, with both Lambertian and Phong receivers, based on physically-accurate Monte Carlo ray or path tracing, followed by image-space filtering. Monte Carlo integration at each pixel has long been regarded as the gold standard for accuracy—but not suitable for interactive use, with hundreds of samples needed and slow render times. This has led to a number of real-time but approximate alternatives, such as point-based gathering [104, 58] or voxel-based cone tracing [16]. We seek to obtain the best of both worlds; physically accurate and interactive.

We are inspired by recent work on sheared filtering for motion blur and soft shadows by Egan et al. [26, 24, 25], which has demonstrated dramatically reduced sample counts. In the previous chapter (also Mehta et al. [63]), we developed an axis-aligned filtering method for area light soft shadows on diffuse surfaces (axis-aligned or sheared refers to the pixel-light space, rather than the image domain, although the method also uses an axis-aligned image filter). This approach trades off a somewhat increased sample count for a much simpler filter, that reduces to an adaptive 2D image-space gaussian blur, does not require storage or search over an irregular 4D domain, and allows for adaptive sampling and adjustment of filter sizes to guarantee convergence with more samples. However, none of these methods is designed for global illumination.

In this chapter, we extend the axis-aligned filtering method for indirect illumination, based on an analysis of the frequency-domain structure of the indirect light field. Our theory considers diffuse inter-reflections, but receiving surfaces can have general BRDFs. In practice, since our approach is based on accurate path tracing and always converges in the limit, our method works for diffuse and moderately glossy objects. Specific theoretical and practical contributions include:

Fourier Analysis of Indirect Illumination: Our main theoretical contribution (Sec. 4.3) is a frequency analysis of indirect lighting. We provide an exact derivation (Secs. 4.3.1, 4.3.2), without first-order assumptions inherent in previous works. This is essential for handling arbitrarily oriented surfaces—unlike the constant velocity assumption in motion blur [26], we cannot assume parallel receivers and reflecting surfaces. Oriented reflectors involve a nonlinear transformation in the Fourier analysis, but surprisingly the spectrum of the indirect light field still lies in a double wedge, bounded by the minimum and maximum depths of the reflector; this result enables us to leverage much of the filtering theory in previous work. We also extend the theory to glossy receivers (Sec. 4.3.3).

Bandlimits for Axis-Aligned Filtering of Indirect Lighting: In the previous chapter, we used the light source (and its size) to bandlimit the occlusion function. However, there is no single light source in global illumination; we show that the geometry/parameterization and BRDF play the role of the bandlimit instead, and derive axis-aligned filter sizes, as well as adaptive sampling rates for diffuse and glossy surfaces (Sec. 4.4).

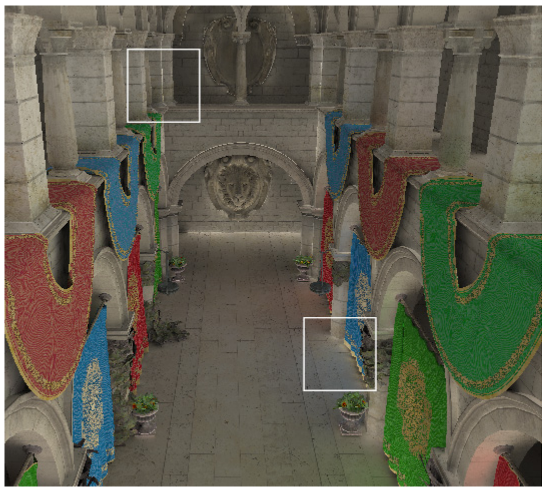
Interactive Sampled Global Illumination: We demonstrate interactive global illumination with one or more indirect bounces—with adaptive sampling and accurate Monte Carlo path tracing using NVIDIA's Optix GPU raytracer, followed by adaptive image filtering; an example is shown in Fig. 4.1, and later in Figs. 4.6, 4.7, 4.8, 4.11.

4.2 Previous Work

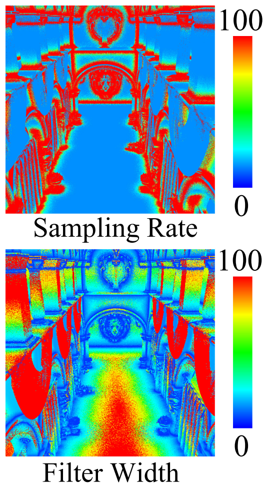
Our method builds on Monte Carlo ray and path tracing, introduced in seminal papers of Cook [14] and Kajiya [44]; these are still usually regarded as the gold standards for physically accurate rendering; we leverage their accuracy, while achieving interactivity.

Interactive Global Illumination: A number of brute-force and approximate methods exist for interactive global illumination [84], but do not usually guarantee physical accuracy or convergence. These include interactive raytracing [102, 101]; we use the fast GPU raytracer in NVIDIA's Optix, but focus on reducing sample count and filtering. Our approach is orthogonal to GPU raytracer accelerations [4]. Another method is approximate voxel-based cone tracing [16] on the GPU. Point-based approaches include micro-rendering [83]. [104] (with refinements in [58]) raytrace shading points and partition them into coherent shading clusters using adaptive seeding followed by k-means, and then apply final gather to evaluate the irradiance using GPU-based photon mapping. In contrast, we sample every pixel using Monte Carlo.

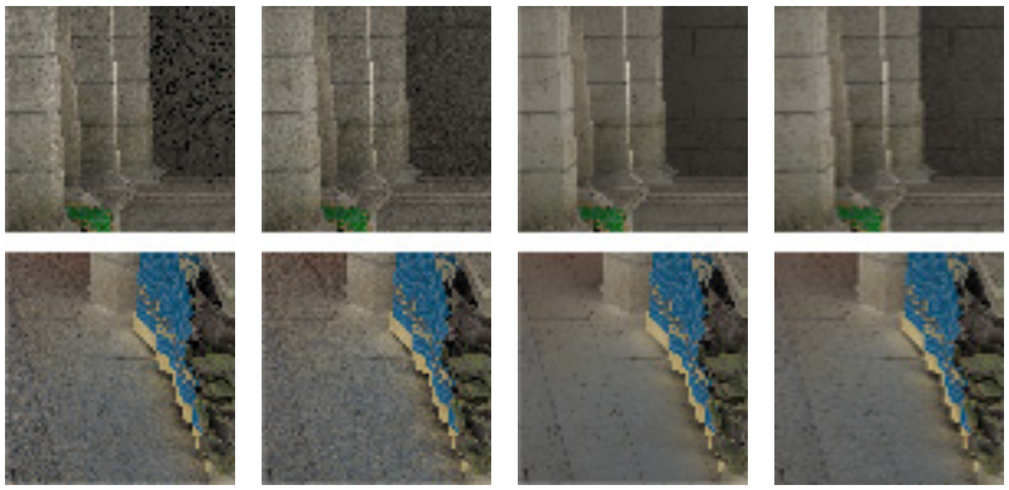
Precomputation: Precomputation-based methods [93] can be used for indirect illumination [39] in relighting static scenes. Our approach does not require precomputation, and can be used with dynamic geometry.



(a) 1-bounce indirect, Our Method avg. 63 samples per pixel (spp)



(b) Adaptive Sampling and Filtering



(c) unfiltered 63 spp adaptively sampled

(d) unfiltered 63 spp uniformly sampled

(e) Our Method 63 adap. sample, filter

(f) Equal error 324 spp

Figure 4.1: (a) We render the Sponza Atrium with 262K triangles, textures and 1-bounce physically-based global illumination at about 2 fps on an NVIDIA GTX 690 graphics card, with an average of 63 Monte Carlo (adaptive) samples per pixel (spp) raytraced on the GPU with Optix, followed by adaptive image filtering. (b) Adaptive sampling rates and filter widths (in pixels) derived from our novel frequency analysis of indirect illumination. (c) Insets of the unfiltered result. Adaptive sampling produces lower noise in high-frequency regions with small filter sizes (see right of bottom inset), with greater noise in low-frequency regions, that will be filtered out. Compare to (d) uniform standard stratified Monte Carlo sampling with uniformly distributed noise. Our method (e) after adaptive sampling and filtering is accurate at 63spp. (f) Equal error at 324 spp, which is still noisy. Overhead in our algorithm is minimal, and we provide a speedup vs equal error of 5×. Readers are encouraged to zoom into the PDF for this and all subsequent figures, to more clearly see the noise and image quality.

Adaptive Sampling: Irradiance caching (IC) and gradients [106, 105] attempt to extrapolate irradiance on diffuse surfaces from neighboring pixels, tracing a pixel only if the error is high. An extension to general low-frequency radiance is given by [51]. [35] provides a more general adaptive sampling heuristic. Note that IC is used to determine where to put caches, not to compute a sampling rate, and is not an interactive technique. Some IC heuristics approximate our results, but our bandlimits are based on fundamental Fourier analysis. Closer to ourapproach, [50] use the harmonic mean of occluder distances as a heuristic to set a filter width at each pixel, and use the total filter weight around each pixel to estimate its sampling rate. Like IC, they do not guarantee convergence either. In contrast, our adaptive filtering and sampling follows directly from the novel Fourier analysis derivations. We also do not require their 3D tree of pixel locations to search for potential contributing pixels at each point. Avoiding this search, and providing a simple gaussian filter, dramatically reduces overhead and enables interactivity.

Adaptive Filtering: Recently, several adaptive filtering and reconstruction methods have been proposed, but they are all designed for offline use. Building on [37, 72], Lehtinen et al. [56, 55] demonstrate GPU-accelerated reconstruction for temporal effects and indirect light fields, but their methods take several minutes. Similarly, we are inspired by recent work on iterative filtering of [88], and the anisotropic statistical filtering of [57] and [85], as well as simple image denoising in graphics [87, 62]. But these are all offline methods, which enables more complex filtering and adaptive sampling. In contrast, past work on fast global illumination has involved simple depth-space heuristic filters [91], edge-avoiding wavelets [19], or filtering secondary scene attributes [8]. We differ in using frequency analysis to develop a spatially-varying image-space gaussian filter.

Frequency Analysis: We are inspired by Chai et al. [13] and Durand et al. [23], who introduce the basic space-angle and pixel-light Fourier theory on which we build, as do many previous works in this area [26, 96, 24, 25, 9]. We build most directly on the axis-aligned filtering approach of the previous chapter ([63]), which reduces to simple 2D image-space filtering rather than irregular reconstruction from the 4D light field, and can therefore be implemented very efficiently, with minimal time or memory overhead. We extend it non-trivially from soft shadows to global illumination, where there is no single light source, and we consider non-parallel geometry of receivers and reflectors as well as glossy receivers.

4.3 Fourier Analysis of Indirect Illumination

In this section, we perform a Fourier analysis of the indirect illumination light field (the incoming light from other objects, as a function of spatial location and incident angle). We assume that the direct lighting is computed separately, and focus on the global component, using the geometry in Fig. 4.2(a). We derive the wedge shape of the Fourier spectrum (Figs. 4.3b,c), which allows us (Sec. 4.4) to apply a suitable axis-aligned filter, with adaptive sampling rates.

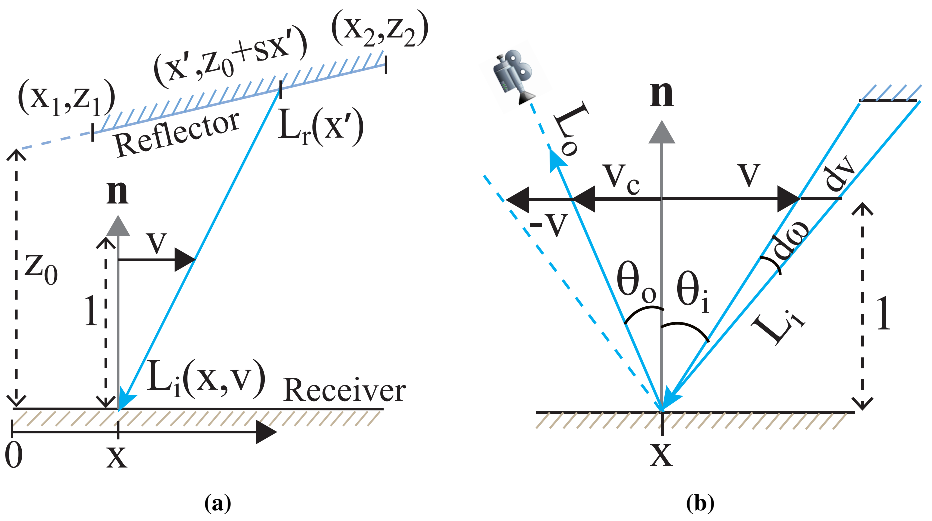
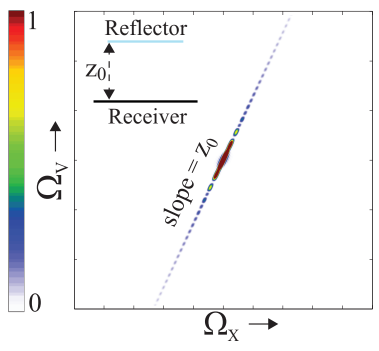


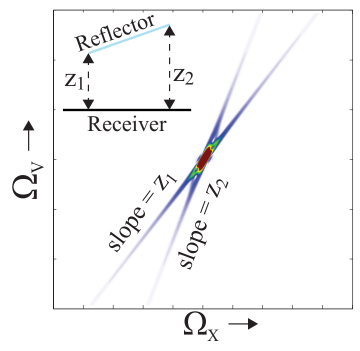
Figure 4.2: (a) The (x,v) parameterization for the indirect illumination light field, and associated notation. (b) Showing how the indirect illumination is reflected towards the camera.

We first consider a receiver, which is the surface seen at a pixel, illuminated by a reflector, which is the nearest surface in a particular direction. We will later see that the theory extends naturally to multiple reflectors at a range of distances and orientations. In essence, we are considering final gather, which adds up the full indirect light (including multiple bounces) from reflectors. Indirect bounces of light will generally be lower frequency, since the BRDF acts as a low-pass filter at each bounce. This diffuse property of global illumination has been exploited in previous works [67, 10]. We also utilize it in our theory, letting the receiver have a general BRDF, but assuming reflected light from the reflector is diffuse—we set filter sizes and sampling rates based on this analysis. The practical algorithm always uses radiance values from accurate path tracing and converges in the limit (but may slightly under or over-blur for non-diffuse indirect light with fewer samples, as in Fig. 4.9). Our results show we can handle diffuse and moderately glossy objects (Fig. 4.8).

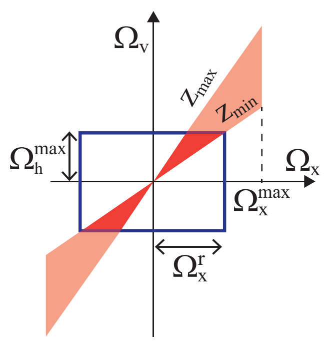
We first derive the equations in flatland or 2D, using the (x,v) light field parameterization, as in [23], and then show the extension of bandlimits to 3D in Sec. 4.4. x is measured along the local plane of the receiver (globally, the receiver could of course be curved, as could the reflectors) and v is measured on a plane parallel to the receiver plane, and a unit distance from it. We make minimal assumptions on spatial properties of the global illumination; reflectors (and receivers) could have complex textures or high-frequency lighting. As we will see, the shape of the Fourier spectrum is determined by key features of the geometry and incoming light. The distance(s) of the reflector will determine the slope (or range of slopes) of the indirect light field. The BRDF of the receiver, and the parameterization, determine bandlimits of the Fourier spectrum.



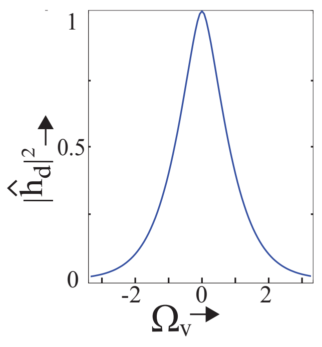
(a) Parallel Reflector $\hat{L}_i$



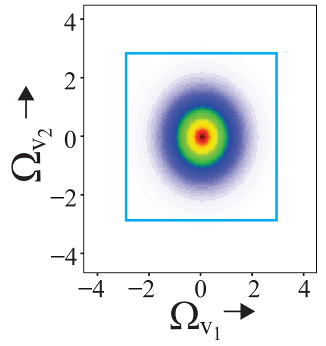
(b) Sloping Reflector $\hat{L}_i$



(c) Axis-Aligned Filter



(d) Flatland h_d PSD



(e) 3D h_d PSD

Figure 4.3: (a) Spectrum of the indirect light field for a diffuse reflector at a single depth z_0 is a line, (b) Spectrum of the indirect light field for a diffuse reflector in the depth range $[z_1, z_2]$ is a double wedge. Because of discontinuities (high derivatives) at reflector end-points, significant energy is concentrated at extreme slopes z_1 and z_2. (c) shows a schematic of the double wedge and our axis-aligned filter determined by the BRDF/transfer bandlimit Ω_h^{max} (d) Power spectral density of our flatland diffuse receiver transfer function h_d. $\Omega_h^{max} = 2.0$ captures 95% energy and (e) the same PSD in 3D, where we need a slightly higher bandlimit of 2.8 (the blue box) to capture the same fraction.

4.3.1 Indirect Light from the Reflector

We first consider the indirect light from the reflector, and then show how this is integrated for global illumination at the receiver.

Assume the reflector slope is s relative to the receiver, as shown in Fig. 4.2a. We parameterize the indirect light field as $L_i(x, v)$ for a receiver point x in a direction v. Similarly, we parameterize a reflector point by x', so that its coordinates are $(x', z') = (x', z_0 + sx')$, where z_0 is the intercept of the reflector line at $x' = 0$. The reflected light is then given by $L_r(x')$—this can include high-frequency illumination, multiple bounces, and texture on the reflector, but has no directional information; we focus on diffuse interreflections. However, the receiver can be glossy as discussed later in Sec. 4.3.3. The reflector lies between (x_1, z_1) and (x_2, z_2), so that $L_r(x') = 0$ unless $x_1 \leq x' \leq x_2$.

For a receiver point x in a direction v, the reflecting point is given by $x' = x + zv$, where z is the distance perpendicular to the receiver, as shown in Fig. 4.2(a). Simple geometry[1] dictates that $z = z_0 + sx'$ from which it follows that

$$\begin{aligned} z = z_0 + s(x+zv) \Rightarrow \quad z &= \frac{z_0 + sx}{1 - sv} \\ x' = x + zv \Rightarrow \quad x' &= \frac{x + z_0 v}{1 - sv}, \end{aligned} \tag{4.1}$$

which implies that the indirect light field $L_{\mathrm{i}}(x,v) = L_{\mathrm{r}}(x')$ is

$$\boxed{L_{\mathrm{i}}(x,v) = L_{\mathrm{r}}\left(\frac{x + z_0 v}{1 - sv}\right).} \tag{4.2}$$

This is a simple relation of the indirect light field to the outgoing reflected light. In the special case that $s = 0$, when receiver and reflector are parallel, it reduces to $L_{\mathrm{i}}(x,v) = L_{\mathrm{r}}(x + z_0 v)$. In that case, the 2D indirect light field L_{i} is the 1D reflected or outgoing light L_{r}, sheared by an amount proportional to the distance z_0 of the reflecting surface. This first-order simplification is analogous to the free space light propagation discussed in [23], and also mathematically similar to the relation for the visibility function in [24]. Note however that there is no "visibility" term in our case; rather we are considering the indirect illumination field. There is also no separate lighting term, as there is for shadows. In essence, we consider the rendering equation, rather than the reflection equation, and integrate over the reflector surface. Moreover, we generalize many previous light field analyses, in explicitly considering a general slope s for the reflector, which leads to the more general rational form above (with denominator of $1 - sv$).

4.3.2 Fourier Spectrum of Incident Light Field

We now conduct a frequency-space analysis.[2] We denote Fourier domain quantities with a hat, and arguments using Ω, and with the symbol $j = \sqrt{-1}$. Equation 4.2 now becomes

$$\hat{L}_{\mathrm{i}}(\Omega_x, \Omega_v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L_{\mathrm{r}}\left(\frac{x + z_0 v}{1 - sv}\right) e^{-j(x\Omega_x + v\Omega_v)}\, dx\, dv. \tag{4.3}$$

We first do the integral along x to compute the partial Fourier transform (denoted with a tilde on top). If v is held fixed, the argument to L_{r} is simply a scale and shift of x, with the Fourier transform being given by standard Fourier scale[3] and shift theorems,

$$\begin{aligned} \tilde{L}_{\mathrm{i}}(\Omega_x, v) &= e^{jz_0 v\Omega_x}(1 - sv)\hat{L}_{\mathrm{r}}\left[(1 - sv)\Omega_x\right] \\ \hat{L}_{\mathrm{i}}(\Omega_x, \Omega_v) &= \int e^{-jv(\Omega_v - z_0\Omega_x)}(1 - sv)\hat{L}_{\mathrm{r}}\left[(1 - sv)\Omega_x\right] dv. \end{aligned} \tag{4.4}$$

[1]These relations do not apply when $x_1 = x_2$, so that $s = \infty$. Parameterizing L_{r} by z instead of x' in that case will lead to a similar result.

[2]For readability, we omit numerical constant factors to normalize the Fourier transforms, that do not affect the insights or the final results.

[3]Technically, the Fourier scale theorem requires the multiplicative factor $\mid 1 - sv \mid$. Since the reflector has finite extent and is in front of the receiver to reflect on to it, we take the positive sign without loss of generality.

First, consider the special case of a parallel reflector with $s = 0$. In this case, $\hat{L}_{\mathrm{r}}(\Omega_x)$ comes out of the integral, which reduces to a delta function with $\hat{L}_{\mathrm{i}}(\Omega_x, \Omega_v) = \hat{L}_{\mathrm{r}}(\Omega_x)\delta(\Omega_v - z_0\Omega_x)$. The Fourier spectrum is compact, essentially given by a shear of $\hat{L}_{\mathrm{r}}$, and restricted to the line $\Omega_v = z_0\Omega_x$ (Fig. 4.3a). This has the same mathematical form as the visibility function in [24, 63]. In those works, they address a range of depths simply by extending the frequency spectrum to a double wedge bounded by minimum and maximum depths. However, that extension is heuristic and not formally justified. Equation 4.4 is more general, explicitly considering a general sloped reflector; we proceed to formally derive the wedge spectrum in this case. We also explicitly consider the finite extent of the reflector, thereby treating (dis)occlusion (directions where there is no indirect light).

In equation 4.4, we make the substitutions $u = (1 - sv)\Omega_x$ so that $v = (1 - u/\Omega_x)/s$ and $dv = -du/(s\Omega_x)$ so that,

$$\begin{aligned} \hat{L}_{\mathrm{i}}(\Omega_x, \Omega_v) &= \int e^{-j(\Omega_v - z_0\Omega_x)(1-u/\Omega_x)/s}\, u\hat{L}_{\mathrm{r}}(u)\, \frac{-du}{s\Omega_x^2} \\ &= \frac{-\exp[-j(\Omega_v - z_0\Omega_x)/s]}{s\Omega_x^2} \int u\hat{L}_{\mathrm{r}}(u) e^{ju(\Omega_v - z_0\Omega_x)/(s\Omega_x)}\, du. \end{aligned} \tag{4.5}$$

The integral is now simply the inverse Fourier transform of $u\hat{L}_{\mathrm{r}}(u)$, evaluated at $(\Omega_v - z_0\Omega_x)/(s\Omega_x)$. Recall the Fourier transform of the derivative L'_{r} is $\hat{L}'_{\mathrm{r}}(u) = ju\hat{L}_{\mathrm{r}}(u)$, so that,

$$\boxed{\hat{L}_{\mathrm{i}}(\Omega_x, \Omega_v) = \frac{j\exp[-j(\Omega_v - z_0\Omega_x)/s]}{s\Omega_x^2} L'_{\mathrm{r}}\left(\frac{\Omega_v - z_0\Omega_x}{s\Omega_x}\right).} \tag{4.6}$$

This is a general result, with the Fourier transform of the indirect light field expressed in terms of the (derivative of) spatial content of the reflected light, and evaluated at a sheared and scaled argument. The first term is simply a phase offset and an Ω_x^{-2} falloff. The second term is more interesting. Since $L_{\mathrm{r}}(x)$ only takes non zero values for $x_1 \le x \le x_2$ (the extent of the reflector), the same holds for $L'_{\mathrm{r}}(x)$. Then, $\hat{L}_{\mathrm{r}}(\Omega_x, \Omega_v)$ takes non-zero values only when[4]

$$x_1 \le \frac{\Omega_v - z_0\Omega_x}{s\Omega_x} \le x_2 \quad \Rightarrow (z_0 + sx_1)\Omega_x \le \Omega_v \le (z_0 + sx_2)\Omega_x. \tag{4.7}$$

By definition $z_0 + sx_1 = z_1$ and $z_0 + sx_2 = z_2$. Therefore,

$$\boxed{z_1\Omega_x \le \Omega_v \le z_2\Omega_x.} \tag{4.8}$$

In other words, the *frequency spectrum lies in a double wedge*, with slopes z_1 and z_2 bounded by the minimum and maximum depths of the reflector, as shown in Fig. 4.3(b). (Note also the significant energy at the extreme slopes z_1 and z_2, since the derivative L'_{r} in equation 4.6 is large at the end-points of the reflector).

[4]The inequalities hold for $s > 0$. For $s < 0$, we must reverse the inequalities, but the same formula for the wedge in equation 4.8 is obtained as long as we adopt the convention that $z_1 < z_2$.

For the special case of a parallel reflector ($s = 0$), we have that $z_1 = z_2 = z_0$, and the spectrum is restricted to the single line $\Omega_v = z_0\Omega_x$, as shown in Fig. 4.3(a). By taking the limit of $s \to 0$ in equation 4.6, one can derive[5] $\hat{L}_{\rm i}(\Omega_x, \Omega_v) = \hat{L}_{\rm r}(\Omega_x)\delta(\Omega_v - z_0\Omega_x)$.

Finally, for multiple reflectors, we combine the spectra for individual reflectors[6], and use the double wedge bounded by the minimum and maximum depths of all reflectors, a schematic of which is shown in Fig. 4.3(c). Figure 4.4(c) verifies this numerically for a flatland scene with multiple reflectors that also occlude each other.

4.3.3 Outgoing Light from the Receiver

We now consider the actual image, corresponding to the outgoing light after it is reflected from the receiver. Let $f(v, v_c)$ be the receiver's BRDF. We do not explicitly consider texture in this section, which will simply modulate the reflected light. The reflected outgoing radiance towards camera c from x can be written (see Fig. 4.2(b)):

$$L_o(x, v_c(x)) = \int_{H^2} L_{\rm i}(x, v)\, f(v, v_c)\, \cos\theta_i\, d\omega. \tag{4.9}$$

In flatland, $\cos\theta_i d\omega = (1 + v^2)^{-3/2}\, dv$, as is derived in [23] and elsewhere (from Fig. 4.2, $\cos\theta_i = 1/\sqrt{1+v^2}$ and flatland 'solid angle' $d\omega = dv \cos\theta_i/\sqrt{1+v^2}$). Hence,

$$\begin{aligned} L_o(x) &= \int_{-\infty}^{\infty} L_{\rm i}(x, v) h(v, v_c) dv & (4.10) \\ h(v, v_c) &= \frac{f(v, v_c)}{(1+v^2)^{3/2}}. & (4.11) \end{aligned}$$

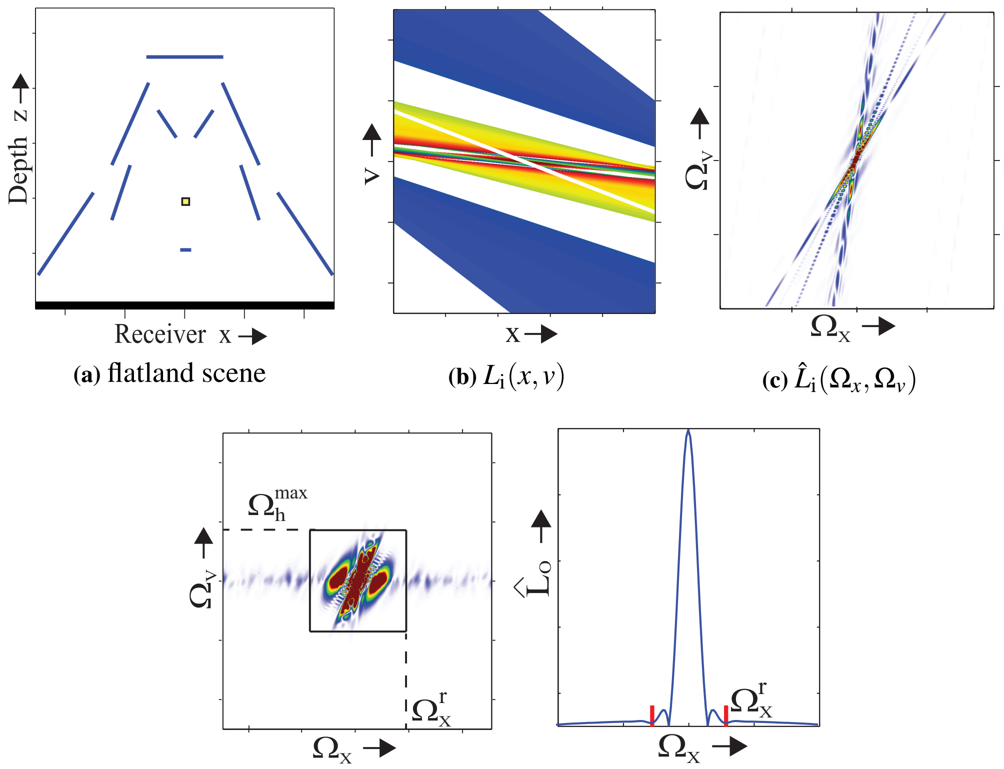
Here we have combined the BRDF, the cosine and the solid angle terms into a single transfer function h. The 3D extension of these results is straightforward, and discussed briefly in Sec. 4.4.2; we will see that we can use almost the same band-limits as derived from the flatland analysis. For a diffuse receiver with coefficient k_d,

$$h_d(v) = k_d \frac{1}{(1+v^2)^{\frac{3}{2}}}. \tag{4.12}$$

An analytic formula for the Fourier transform of h_d is known in terms of Bessel functions, and is plotted in Fig. 4.3(d).

[5] Clearly, as $s \to 0$, the $L'_{\rm r}$ term becomes a delta function of the form $\delta(\Omega_v - z_0\Omega_x)$. To find the multiplier, by definition of the delta function, we must integrate equation 4.6 over Ω_v. First, substitute that $w = (\Omega_v - z_0\Omega_x)/(s\Omega_x)$ with $d\Omega_v = s\Omega_x dw$. Then, we simply want $\int (j \exp[-jw\Omega_x]\Omega_x^{-1}) L'_{\rm r}(w)\, dw$. Simply integrating by parts, this reduces to $\int L_{\rm r}(w) \exp[-jw\Omega_x]\, dw$ which is simply the Fourier transform $\hat{L}(\Omega_x)$. Thus, as $s \to 0$, $\hat{L}_{\rm i}(\Omega_x, \Omega_v) = \hat{L}(\Omega_x)\delta(\Omega_v - z_0\Omega_x)$.

[6] As in most previous work on light field filtering [13, 25, 24, 63], the spectra do not strictly combine because of occlusion between different reflectors, and there can be some leakage outside the wedge for the multiple reflector case. As in previous work, the extent of leakage is empirically small and we neglect it.



(a) flatland scene **(b)** $L_i(x,v)$ **(c)** $\hat{L}_i(\Omega_x,\Omega_v)$

(d) $\hat{L}_i(\Omega_x,\Omega_v)\hat{h}_d(-\Omega_v)$ **(e)** $\hat{L}_o(\Omega_x)$

Figure 4.4: Schematic for a flatland scene: (a) The configuration, where blue shows reflecting surface and black shows receiving surface, and the white point is the light source; (b) The incident indirect light field L_i in false color; (c) The numerical double-wedge spectrum of (b); (d) The spectrum in (c) is multiplied by the Fourier transform of h_d; the black box is our filter, which captures 99% of the energy; (e) shows the Fourier transform of L_o obtained by integrating (d) corresponding to equation 4.14; the red bars mark our spatial domain filter.

Now, consider a receiver Blinn-Phong BRDF (with exponent m and specular coefficient k_s). The half-angle is $(\tan^{-1} v + \tan^{-1} v_c)/2$,

$$h_s(v,v_c) = k_s \frac{\cos^m\left(\frac{1}{2}\left[\tan^{-1} v + \tan^{-1} v_c\right]\right)}{(1+v^2)^{\frac{3}{2}}}. \tag{4.13}$$

In summary, Sec. 4.3 has analyzed the indirect light field in both the primal and Fourier domain. A key result is in formally deriving the wedge spectrum for an arbitrarily oriented reflector. This result is mathematically similar to (but more general than) previous analyses of spectra for soft shadows [24, 25, 63]. Our derivation is more accurate, applying to general configurations of reflector and receiver. Moreover, there is no single light source per se, unlike shadows. In Sec. 4.4, we build on these results to develop the Fourier space bandlimits and filters for the transfer function, and to select the adaptive sampling rates.

4.4 Axis-Aligned Filtering

We now develop the bandwidths for an axis-aligned filter (which reduces to image-space blurring) in a fashion similar to the previous chapter. Our filter in the Fourier domain is shown in Fig. 4.3(c).

We first write the Fourier equivalent of equation 4.10, which is simply a frequency domain integral for $\hat{L}_o$, the spectrum of outgoing light:

$$\hat{L}_o(\Omega_x) = \int \hat{L}_{\text{i}}(\Omega_x, \Omega_v)\hat{h}(-\Omega_v)\, d\Omega_v, \tag{4.14}$$

where we keep the camera direction v_c argument in h and $\hat{h}$ implicit (or alternatively, since $v_c(x)$ is a function only of the spatial pixel x, we can include it in the BRDF term).

4.4.1 Fourier and Spatial Reconstruction Filter

In the expression above, the transfer function $\hat{h}$ (which includes the BRDF and geometry terms) acts as a low-pass band-limiting filter on the indirect illumination light field. It plays much the same role as the low-frequency lighting in [24] or the area light width in the previous chapter. However, note the different representation that is necessary, compared to those papers. In essence, our indirect illumination field L_{i} is analogous to the visibility term in chapter 3, while our BRDF and geometry term is analogous to the light intensity term. Also note that we do not need to explicitly consider visibility. Of course, there will be directions where no reflector is present and we receive no indirect light contribution; indeed, our Fourier spectrum derivation in Sec. 4.3.2 explicitly handles finite extent reflectors. Consider the frequency bandlimits from equation 4.14. Assume $\hat{h}$ has a frequency bandlimit $\Omega_h^{\max}$. Then, frequencies in $\hat{L}_{\text{i}}$ higher than this value in Ω_v need not be considered during integration. This in turn induces a limit on the maximum spatial frequency of $\hat{L}_o$, as seen in Fig. 4.3(c). The width of our reconstruction filter Ω_x^r is:

$$\boxed{\Omega_x^r = \mu \cdot \min\left\{\frac{\Omega_h^{\max}}{z_{\min}}, \Omega_x^{\max}\right\}}, \tag{4.15}$$

where μ is a scale factor (for now $\mu = 1$) that enables over-sampling and convergence, as discussed in Sec. 4.4.3, $z_{\min}$ is the minimum world-space distance to any reflector, and $\Omega_x^{\max}$ is the maximum spatial frequency that is always a limit (even if $z_{\min}$ is small). Similar to previous work, we define $\Omega_{\text{pix}}^{\max}$ as the maximum frequency in pixel space, with $\Omega_x^{\max} = \alpha\Omega_{\text{pix}}^{\max} = \alpha d^{-1}$, where d is the projected distance per pixel[7] and $0 < \alpha < 1$ is a constant. We use $\alpha = 0.3$ in most of our renderings.

We integrate over the v dimension (or in Fourier space Ω_v). Therefore, axis-aligned filtering of the indirect light field in x-v space reduces to a spatial filter over the noisy indirect illumination,

[7]The lateral size in terms of actual distance that pixel covers. This term also naturally accounts for perspective effects and foreshortening.

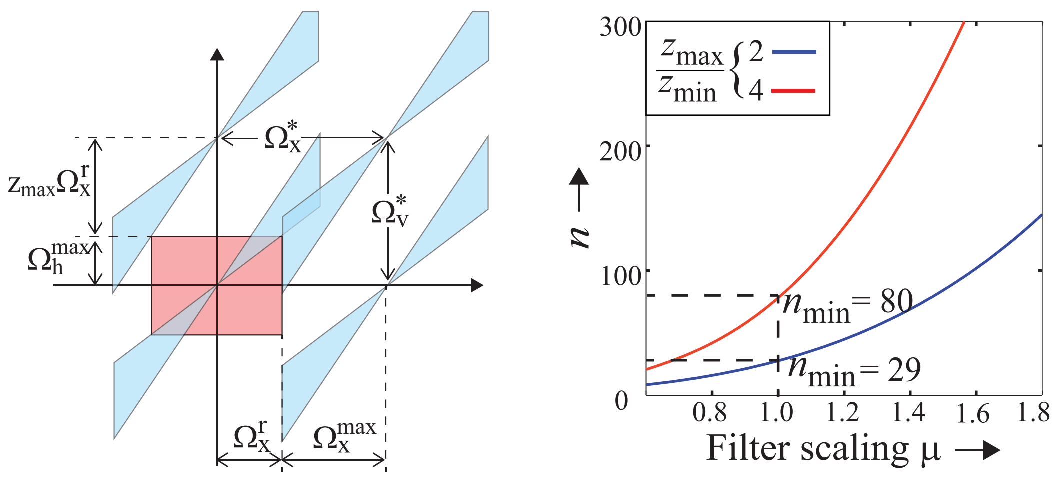


Figure 4.5: (a) Compact packing of spectra of the indirect light field in the Fourier domain. Ω_x^* and Ω_v^* are the minimum sampling rates. (b) the per-pixel minimum sampling rate vs. fourier-space filter scaling μ (image space filter size is inversely proportional to μ) for two different occluder depth ranges.

which is effectively a screen-space filter. In other words, we first compute the standard noisy Monte Carlo global illumination result $\bar{L}_o(x)$ using relatively few samples to evaluate equation 4.10. We then filter,

$$L_o(x) = \int N(x-y;\beta)\,\bar{L}_o(y)\,dy, \tag{4.16}$$

where N is the primal domain gaussian filter with standard deviation $\beta = 2/\Omega_x^r$, since the Fourier space standard deviation is taken to be $0.5 \times \Omega_x^r$ (in practice, we use Gaussian filters in spatial and frequency domains as in previous work, instead of sincs or boxes).

4.4.2 Bandlimit of the Transfer Function

The remaining question is to find the numerical bandlimit $\Omega_h^{\max}$ for the transfer function. In these calculations, we can ignore the diffuse and specular coefficients k_d and k_s. For a Lambertian receiver, given by equation 4.12, Fig. 4.3(d) shows the power spectral density (PSD) of the transfer function, $|\hat{h}_d(\Omega_v)|^2$. $\Omega_h^{\max} = 2.0$ captures 99% of the energy in $\hat{h}$. Moreover, as shown in Fig. 4.4(d,e), we see that $\Omega_h^{\max} = 2.0$ is usually sufficient to capture approximately 99% energy in $\hat{L}_o$ since $\hat{L}_\mathrm{i}$ also usually decays with frequency.

For the 3D case (4D light field), the transfer function depends on the $\mathbf{v} = (v_1, v_2)$ coordinates, and we denote this 3D extension as $H(\mathbf{v}, \mathbf{v_c})$. We still have $\cos\theta_i = 1/\sqrt{1+\mathbf{v}\cdot\mathbf{v}}$ and the solid angle is given by $d\omega = dv_1\,dv_2 \cos\theta_i/(1+v_1^2+v_2^2)$. Therefore,

$$H(\mathbf{v}, \mathbf{v_c}) = \frac{f(v_1, v_2, v_{c_1}, v_{c_2})}{(1+v_1^2+v_2^2)^2}. \tag{4.17}$$

Note that our final filter is only along the **x** dimension, and the PSD of H is symmetric in Ω_{v_1} and Ω_{v_2}. Figure 4.3(e) shows the PSD for the diffuse transfer function—we can use a bandlimit slightly higher than the flatland case, $\Omega_h^{\max} = 2.8$, to capture 99% energy.

For the Blinn-Phong BRDF, the bandlimit $\Omega_h^{\max}$ depends on $\mathbf{v_c}$ and the exponent m. There is no closed form analytic expression for the Fourier transform, but we can still obtain a simple numerical linear fit between the exponent m and the band-limit for moderately glossy BRDFs, as discussed in the Appendix for the 3D case. In summary, we use the numerical values,

$$\boxed{\Omega_{h,d}^{\max} = 2.8 \qquad \Omega_{h,s}^{\max}(m) = 3.6 + 0.084m.} \tag{4.18}$$

The appendix also gives bandlimits for the Phong BRDF.

4.4.3 Adaptive Sampling Rates

Discrete sampling of a continuous signal (here, the indirect 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.

As in the previous chapter, the most compact packing is that shown in Fig. 4.5(a), and the minimum sampling rates in the Ω_x and Ω_v axes are:

$$\begin{aligned} \Omega_x^* &= \quad \Omega_x^r + \Omega_x^{\max} &&= \frac{\Omega_h^{\max}}{z_{\min}} + \alpha\Omega_{\text{pix}}^{\max} \\ \Omega_v^* &= \quad \Omega_h^{\max} + z_{\max}\Omega_x^r &&= \Omega_h^{\max}\left(1 + \frac{z_{\max}}{z_{\min}}\right). \end{aligned} \tag{4.19}$$

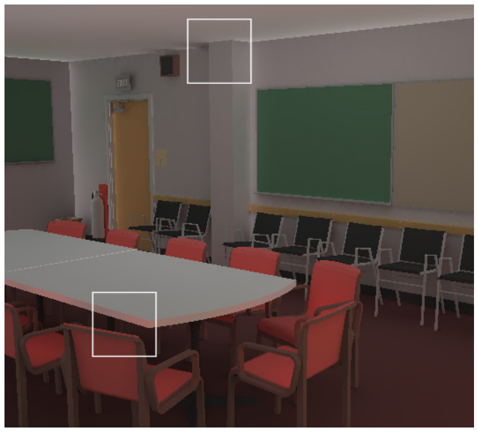
The per-pixel sampling rate is then given as:

$$\begin{aligned} n &= \left[(\Omega_x^*)^2 \times A_p\right] \times \left[(\Omega_v^*)^2\right] \\ &= \left(\Omega_h^{\max}\frac{\sqrt{A_p}}{z_{\min}} + \alpha\right)^2 \times (\Omega_h^{\max})^2\left(1 + \frac{z_{\max}}{z_{\min}}\right)^2, \end{aligned} \tag{4.20}$$

where A_p is the spatial area in world-space subtended by a pixel $A_p = d^2$, with $\sqrt{A_p}\Omega_{\text{pix}}^{\max} = 1$.

As in the previous chapter, we can also increase the sampling rate beyond the minimum required, and simultaneously reduce the filter size, so that our algorithm converges to Monte Carlo ground truth in the limit. In the Fourier domain, we increase the filter size by a factor of $\mu > 1$, such that $\Omega_x^r = \mu\Omega_{x0}^r$, where Ω_{x0}^r is the critical size given by equation 4.15. Then, the sampling rate increases to

$$\boxed{n(\mu) = \gamma \cdot \left(\mu\Omega_h^{\max}\frac{\sqrt{A_p}}{z_{\min}} + \alpha\right)^2 \times (\Omega_h^{\max})^2\left(1 + \mu\frac{z_{\max}}{z_{\min}}\right)^2,} \tag{4.21}$$



(a) 1-bounce indirect illumination, 60 spp

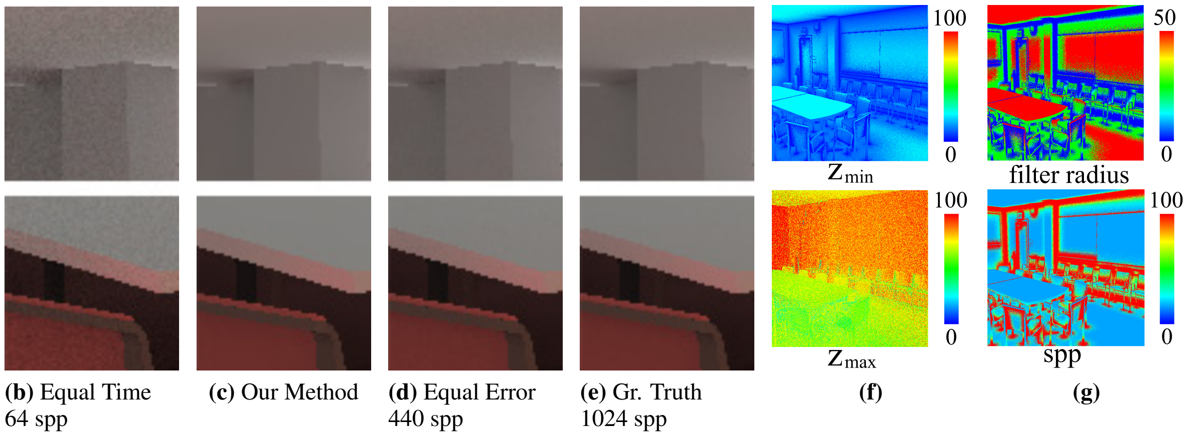
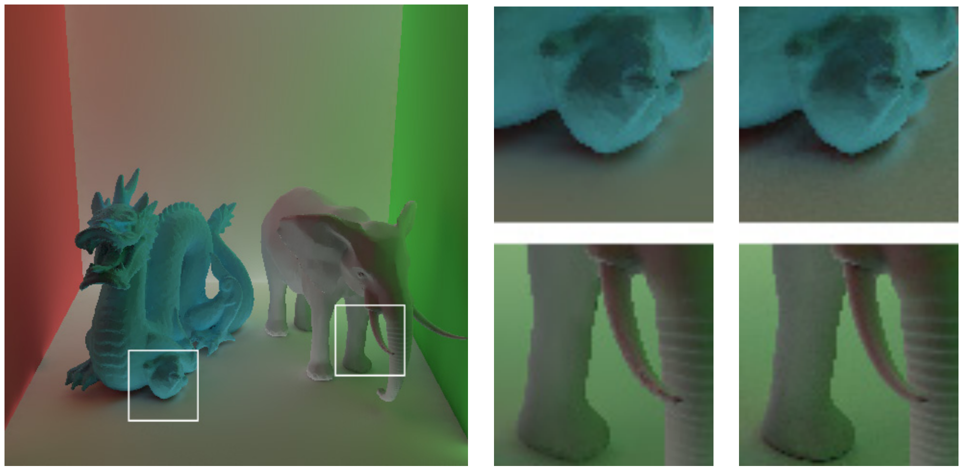


Figure 4.6: (a) The diffuse Conference scene with 331K triangles is rendered at 3 fps with our method and 60spp, using 1-bounce of indirect illumination. Insets show (b) Equal time uniform MC at 64 spp is still very noisy, (c) our method compared to (d) equal error uniform MC at 440 spp and (e) ground truth at 1024 spp; (f) minimum and maximum reflector distances $z_{\min}$, $z_{\max}$ (g) screen-space filter radius in pixels, and per pixel sampling rates. Our algorithm obtains accurate results $7\times$ faster that basic path tracing.

where γ is a scaling factor for importance sampling discussed shortly. We can also use equation 4.21 with $\mu < 1$; in practice, we find $\mu = 0.9$ adequate in most cases. Figure 4.5(b) shows how the effective samples per pixel vary with μ.

Importance Sampling Adjustment: The theory is derived assuming uniform angular sampling. In rendering, more efficient importance sampling is used, where samples in the important regions are already closer together. This enables us to adjust $\gamma < 1$ in equation 4.21 for lower sampling rates, where γ depends on the receiver BRDF. For the diffuse cosine lobe, the incident angle may vary in $[0, \pi/2]$ but about 80% of energy is in $[0, \pi/4]$, enabling $\gamma \approx 1/2$. In practice, we use $\gamma = 0.4$, which we find provides adequate quality with lower sampling counts. For the specular lobe, setting $c = \cos(\pi/4) = 1/\sqrt{2}$, about 80% of energy is concentrated in the range $[0, \cos^{-1}(c^{1/m})]$,



(a) 3-bounce indirect, 56 spp **(b)** Our Method, 56 spp **(c)** Equal error, 298 spp

Figure 4.7: A Cornell Box scene (a) 3-bounce Indirect illumination, our method using an average 56 spp; insets showing (b) our method compared, to equal error (c) at 298 spp, a $5\times$ *speedup.*

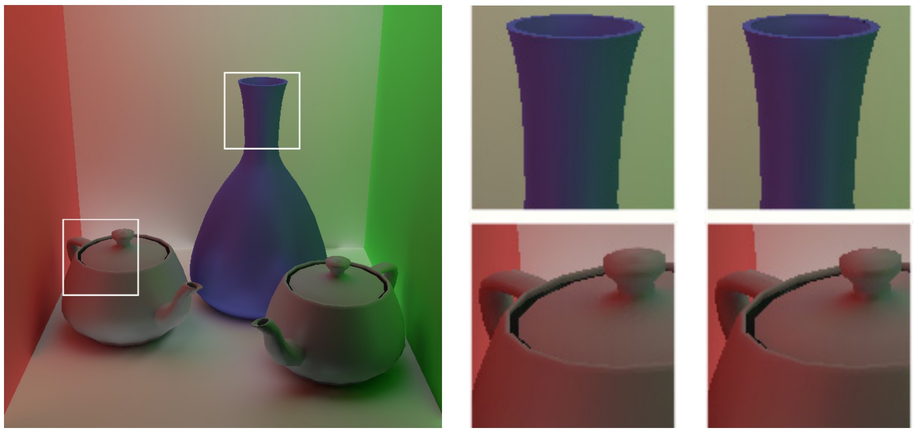
and so we set $\gamma = \cos^{-1}(c^{1/m})/(\pi/2)$. For Blinn-Phong exponent $m = 20$, this sets $\gamma = 0.11$.

4.5 Implementation

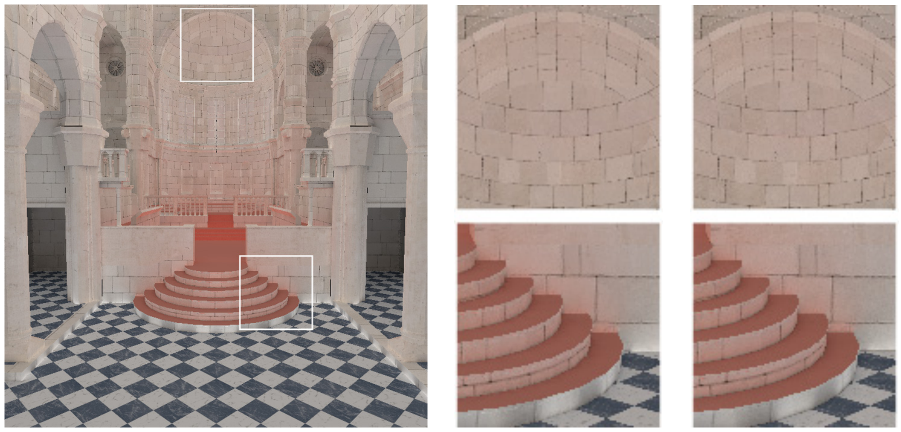
Given our derivation, the final implementation is simple. Our code uses the NVIDIA OptiX real-time raytracing framework [76] on a GTX 690 GPU. We only need to implement a simple extension for multi-bounce path tracing where desired, as well as the core of our adaptive sampling and filtering passes in the OptiX GPU framework. Our method runs entirely on the GPU in three passes, as described below:

Initial Raytracing for Filter Sizes and Adaptive Sampling: We first trace 16 stratified rays over the visible hemisphere from each pixel/shading point, to compute the per-pixel $z_{\min}$ and $z_{\max}$, which are simply min/max world-space distances to geometry (reflectors).[8] Example outputs are shown in Fig. 4.6(f). Direct lighting is also computed in the standard way in this pass (for this work, we used a single point light source in all examples), and we create buffers for pixel k_d and k_s values so we can separate textures from radiance computation. We then use $z_{\min}$ and $z_{\max}$ to compute the filter sizes and sampling rates according to equations 4.15 and 4.21 respectively, separately for the diffuse and glossy components of the pixel. We use $\mu = 0.9$ and $\alpha = 0.3$, with γ set as discussed earlier.

[8] As in most adaptive sampling approaches, using only 16 initial rays can give noisy or incomplete estimates of $z_{\min}$ and $z_{\max}$, but we never visualize them directly, and only use these to set filter sizes and sampling rates. As seen in our results, we found the final images to be high quality, with any additional accuracy not worth the cost of more rays. We also tried filtering the z estimates, but found this actually worsened image quality/speed, leading to overly conservative or aggressive sampling rates and filter sizes.



(a) 2-bounce indirect, 85 spp **(b)** our method **(c)** ground truth



(d) 2-bounce indirect, 86 spp **(e)** Our method **(f)** ground truth

Figure 4.8: (a) 2-bounce indirect illumination on a Cornell Box scene with glossy vase and teapots ($m = 12$), with average 85 spp. A close comparison to ground truth is shown in (b) and (c); in (d) we show 2-bounce indirect illumination on the Sibenik Cathedral scene, with a glossy textured floor and red carpet ($m = 20$), with average 86 spp. Insets in (e) and (f) show the accuracy of our filtering on glossy receivers.

For efficiency, we also use radiance values from the initial 16 samples, so we actually only need $n - 16$ samples in the second pass. Finally, we clamp the maximum sampling rate (the minimum must already be at least 16 because of initial sampling). We use a maximum of 100 samples, scaled up when $\mu > 1$, i.e., 100μ. Correspondingly, we also clamp the minimum $z_{\min}$ to about 2% of maximum scene dimension, since a $z_{\min} = 0$ implies no filtering and leaves corners and edges (such as wall intersections) noisy.

We visualize the resulting sampling rates and filter widths in Figs. 4.1(b) and 4.6(g). It is clear that smaller filter sizes and higher sampling rates are used for close-by geometry, while regions with further reflectors, such as the floor in Fig. 4.1, can use lower sampling rates and wider filters.

Path Tracing: We use OptiX to (stratified and importance) sample each pixel's hemisphere with the adaptive sampling rates. If more than one bounce is desired, we use standard path tracing for higher bounces. We compute the incoming radiance values and multiply with the BRDF (excluding textures). This gives a noisy estimate $\bar{L}_o$ at each pixel, as shown in Fig. 4.1(c). Notice that adaptive sampling ensures almost accurate results in high-complexity areas, such as the bottom of the curtain (right of second inset), while showing considerable noise (many fewer samples) on the floor. This is as desired with sampling optimized for filtering; larger filter widths on the floor ensure an accurate final image (Fig. 4.1(e)).

Screen-Space Filtering: In the final pass, we actually do Gaussian filtering on $\bar{L}_o$, using the spatial filter (equation 4.16). Like the previous chapter, we use a 2D screen-space filter, with a depth buffer giving the world-space coordinates and distances required. Moreover, we perform the standard separation of the Gaussian filter into two 1D filters along the image dimensions for efficiency.[9] Finally, we modulate filtered radiance by the textures for k_d and k_s.

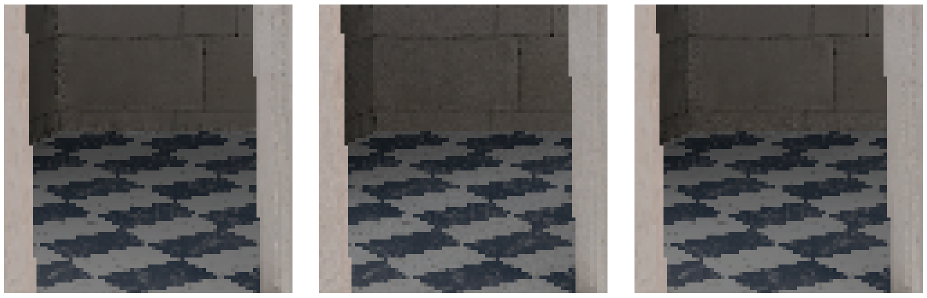
In summary, the method consists of path-tracing (the core optimized OptiX raytracer), coupled with initial estimates to set filter sizes, adjust sampling rates, and do screen-space filtering. The latter steps are simple to implement and extremely fast. Overhead is minimal compared to Optix (Table 4.1), and we achieve interactive speeds of 1-3 frames per second, often with nearly an order of magnitude fewer samples than basic GPU path tracing.

4.6 Results

We show results of interactive global illumination on five scenes in Figs. 4.1, 4.6, 4.7, 4.8 and 4.11. The accompanying video shows animations and screen captures with moving light source, viewpoint and examples of dynamic geometry (no precomputation is required; each frame is rendered independently). To focus on the global component, all images show indirect light only; direct illumination is added separately in the first step (initial raytracing in Sec. 4.5).

Visual Fidelity and Speedups: Our method is accurate in a range of different scenarios, with consistent reductions in sample counts over basic path tracing. Figure 4.1 shows an example of the Sponza scene (262K triangles) with 1-bounce diffuse indirect illumination (surfaces are Lambertian) and textures. Our image (Fig. 4.1a,e) is accurate with 63 average samples per pixel. Equal error with stratified Monte Carlo is only achieved for 324 samples, and is visually still noisy. Our method adds minimal overhead, so this is a speedup of $5\times$. Figure 4.6 shows similar results for the diffuse conference scene with 331K triangles. Our method requires only 60 samples per pixel. The overhead is minimal, with equal time Monte Carlo being only 64 samples. Equal error (still noisy) is only obtained for 440 samples, a speedup of about $7\times$. We demonstrate the extension to multi-bounce diffuse indirect illumination by path tracing in Fig. 4.7. Note the curved surfaces of

[9]As in Chapter 3, this separation is strictly accurate only if the gaussian width β is the same across all pixels in the filter but in practice we find almost no noticeable difference, since β usually varies slowly.



(a) Our Method, 86 spp **(b)** Ground Truth **(c)** Our method, 226 spp

Figure 4.9: Evaluating the theoretical approximation of diffuse indirect light. (a) Shows over-blurring (and hence darkening) of glossy-to-diffuse and glossy-to-glossy in an inset from the Sibenik Cathedral scene, when using $\mu = 1$ *at 86 spp. However, we match ground truth (b) closely using* $\mu = 2$ *at 226 spp, shown in (c).*

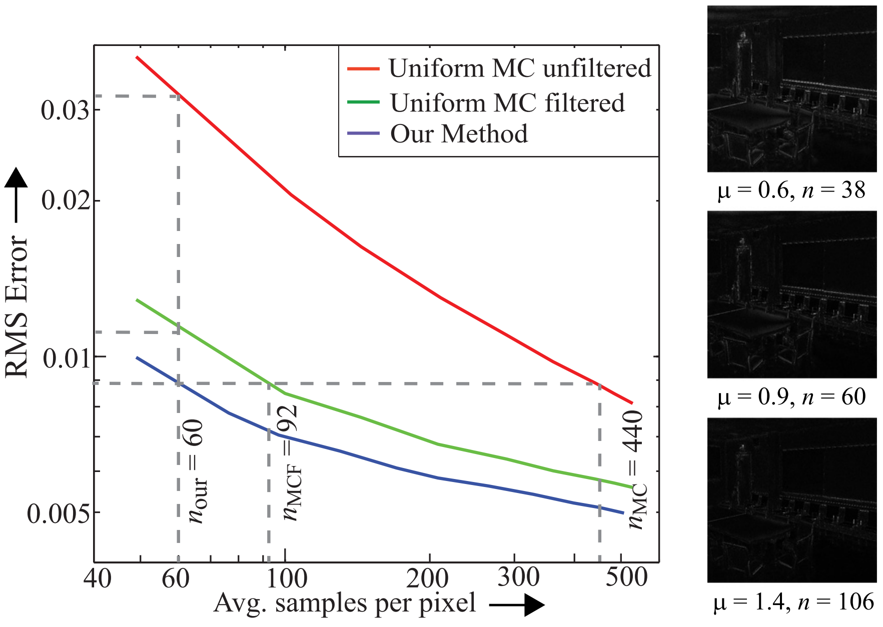
the dragon, and complex structures on the elephant's trunk in the insets, that our method renders accurately at 56 samples per pixel (close examination will show a slight overblur on the trunk).

Figure 4.8 shows that we can handle moderately glossy receivers for a modified Cornell box (vase and teapots are glossy with Blinn-Phong exponent 12), and for the Sibenik scene with textures and 75K triangles (floor and red carpet have Blinn-Phong exponent 20). Nevertheless, the method matches accurately, capturing the glossy reflections, and color-bleeding near edges. Again, a speedup of about $5\times$ is achieved, over simply using path-tracing in OptiX.

Figure 4.9 evaluates our use of the "diffuse interreflection" approximation to set sampling rates in Sec. 4.3, which does not fully consider high frequencies in glossy-to-diffuse or glossy-to-glossy interactions (however, receiver glossiness is fully handled as noted above). While our filter sizes depend on this approximation, note that our method operates on accurate path tracing input, and will therefore always converge in the limit with more samples. In Fig. 4.9 we see that we slightly overblur (and hence darken) for $\mu = 1$, but glossy-to-diffuse transfer is almost fully accurate for $\mu = 2$.

Quantitative Accuracy: We also evaluated our method quantitatively in Fig. 4.10, showing RMS errors vs average number of samples for the conference scene from Fig. 4.6. The error of our method (blue curve) is significantly below stratified Monte Carlo at all sample counts. Moreover, even just using filtering on uniform sampling (green curve) gives a substantial improvement. Note that our method is physically-based and consistent. As we increase the number of samples (higher μ), we do converge to ground truth and error decreases (also shown visually in insets), This is in contrast to most other solutions for fast, approximate global illumination.

Timings and Overhead: In table 4.1, we show timings for steps of our algorithm on different scenes, rendered at a resolution of 640×480. We obtain most of the benefits of axis-aligned filtering as in the previous chapter, even though our algorithm is somewhat more complex (involving additional texture buffers, and passes to handle diffuse and specular components). The total over-

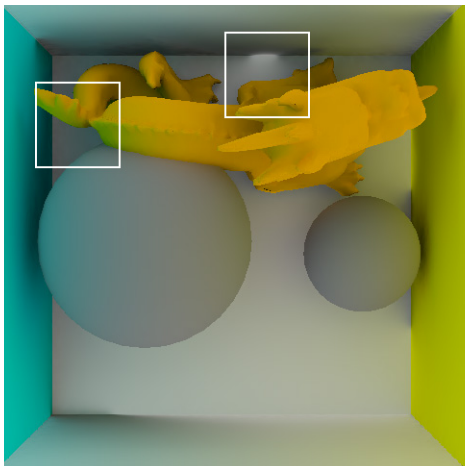


(a) **(b)**

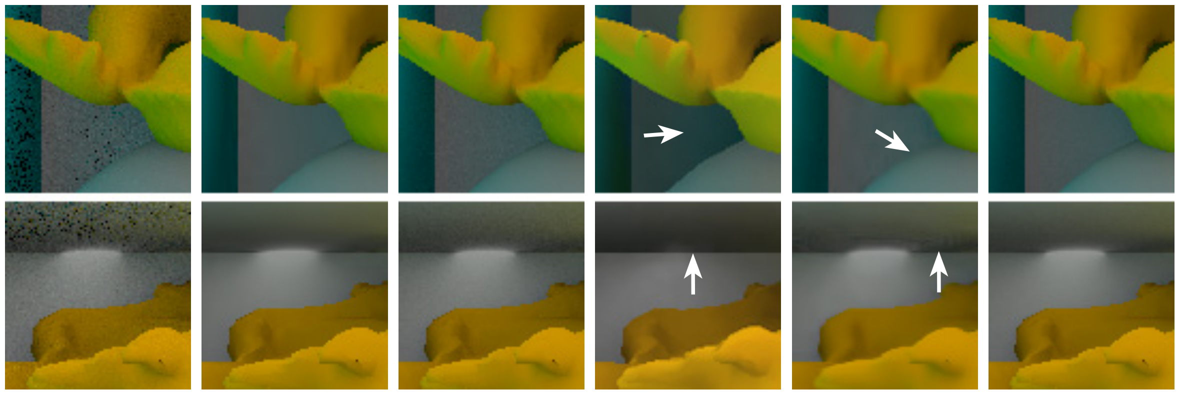
Figure 4.10: (a) Shows a log-log plot of the RMS pixel error vs average sampling rate for the Conference scene of Fig. 4.6. In (b) we show insets from the conference scene (showing error relative to ground truth magnified 10×*). It can be seen that our method converges to ground truth both numerically and visually with more samples.*

				Optix Ray/Path tracing			Our algorithm			Total	
scene	tris	avg. spp	num. bounces	1st pass (ms)	2nd pass (ms)	total (ms)	compute spp (ms)	adaptive filtering (ms)	total overhead (ms)	total time (ms)	fps (**with**/ without alg)
Sponza (Fig. 1)	262 K	63	1	181	580	761	10	55	**65**	826	**1.21** / 1.31
Conference (Fig. 6)	331 K	60	1	87	274	361	9	46	**55**	416	**2.40** / 2.77
Cor. Box (Fig. 7)	145 K	56	1	70	291	361	9	63	**72**	433	**2.31** / 2.77
			2	70	802	872	10	62	**72**	944	**1.06** / 1.14
			3	72	1423	1495	10	64	**74**	1569	**0.63** / 0.67
Sibenik (Fig. 8d)	75 K	86	1	110	440	550	10	50	**60**	610	**1.64** / 1.82
			2	108	1402	1510	12	52	**64**	1574	**0.64** / 0.67
Cor. Box (Fig. 11)	16.7 K	59	1	70	248	318	12	49	**61**	379	**2.64** / 3.14
			2	68	622	690	10	57	**67**	757	**1.32** / 1.44

Table 4.1: Timings of our scenes rendered at 640 × 480*. Our filtering overhead is small compared to the sampling time, and our impact on the fps is small.*



(a) 1-bounce indirect, 59 spp



(b) 59 spp unfilt., 0.32 sec **(c)** our method, 0.38 sec **(d)** Equal error 361 spp, 2 sec **(e)** IPP, 1 sec **(f)** BM3D, 5 sec **(g)** gr. truth 1024 spp, 6 sec

Figure 4.11: A Cornell Box scene (a) 1-bounce indirect illumination, our method using an average 59 spp at 2.7 fps; insets showing (b) our method unfiltered, (c) our method, filtered, (d) Equal error image with uniform MC 361 spp, (e) Importance Point Projection (IPP, arrows show high frequency details are lost and some regions are over-darkened) (f) Image denoising (BM3D, arrows show blurred geometry edges and artifacts due to noise in input) (g) Reference image with 1024 spp.

head in a frame is about 60-70 ms, which is small compared to the cost of OptiX path tracing, and results in only a marginal decrease in the performance of the real-time raytracer. Note that additional bounces in path tracing do cause a slowdown in the Optix raytracer, essentially linear in the number of bounces as expected, but do not affect our algorithm.

Note that our filter operates only in image-space and therefore has limited memory requirements (buffers for the noisy image, depth, and textures). We achieve a sample count reduction and speedup of $5-8\times$ on most scenes, with interactive frame rates of $1-3$ fps. Note that we are limited only by the speed of the real-time raytracer, and using further GPU raytracing accelerations would provide further speedups. To our knowledge, this is one of the first demonstrations of

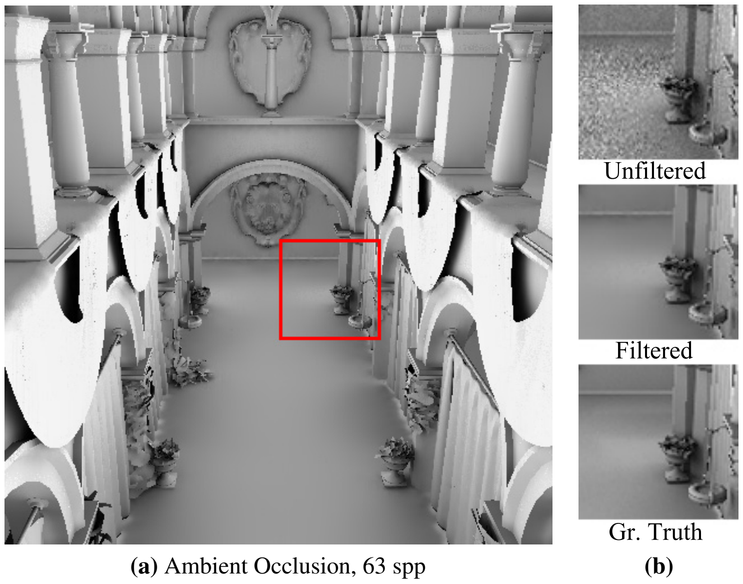


Figure 4.12: Ambient Occlusion on the Sponza scene using an average of 63 spp runs at 1.2 fps.

accurate interactive global illumination, based on principled Monte Carlo sampling. Alternative methods, discussed next, either sacrifice accuracy, or are offline, adding overheads of seconds to minutes.

Comparisons: In Fig. 4.11, we include comparisons to two alternative approaches. First, Fig. 4.11(e) considers the importance point-projection (IPP) method for global illumination ([58], building on [104]). We use the authors' code on their scene. Because only a few shading points are used, errors can occur in regions of complex interreflections, especially near edges, as shown the insets. Figure 4.11(f) compares to standard image denoising using a state of the art denoiser bm3d [18]; this still takes a few seconds, and can overblur or underblur (retain noise) in some regions, since it is not informed of the precise filter size from our frequency analysis. Newer denoisers for rendering [57, 85] have impressive results, but often report render times in the minutes, as does the indirect light field reconstruction approach of [55]. Thus, previous approaches aim for high quality or high accuracy; we differ in providing *accurate* results for *interactive* global illumination.

Extension—Ambient Occlusion: Our theory applies to any outgoing function on the reflectors. If we simply set it to 1, then we will directly compute the complement of ambient occlusion (rays that hit a reflector return 1, rays that miss return 0). This is an accurate interactive ambient occlusion calculation, several orders of magnitude faster than [24]. We show an example result in Fig. 4.12, computed at interactive rates of 1-2 fps.

Limitations: As shown in our numerical simulations and real experiments, the double wedge model is a good representation of the Fourier spectrum for indirect illumination. However, highly curved receivers and reflectors can lead to 'leaking' spectra outside theoretical predictions. This is not usually a significant practical problem (see accurate dragon and elephant insets in Fig. 4.7), and we also alleviate it in a number of ways. We implement a normal threshold (set to 10°), beyond which radiance values are not used in the filtering. This avoids filtering across very different orientations.[10] Our axis-aligned filter is looser than previous sheared filters to capture more of the spectrum leaks. And, we can always increase sample count (and μ) to converge to ground truth. Our average sample counts are low (~ 60) for global illumination, but still high enough to prevent fully real-time performance on complex scenes. However, we are 1-3 orders of magnitude faster than equally accurate Monte-Carlo based rendering methods, providing *interactive* frame rates. Finally, the theory is limited to diffuse interreflections on general BRDF receivers; in practice (Figs. 4.8, 4.9) it works for Lambertian and moderately glossy receivers/reflectors.

[10]This reduced filter size may however lead to fewer samples in the filter footprint than theory recommends. It is possible to adjust the adaptive sampling for this, but we found the overhead added by this additional pass not worth the marginal improvement in quality.

Chapter 5

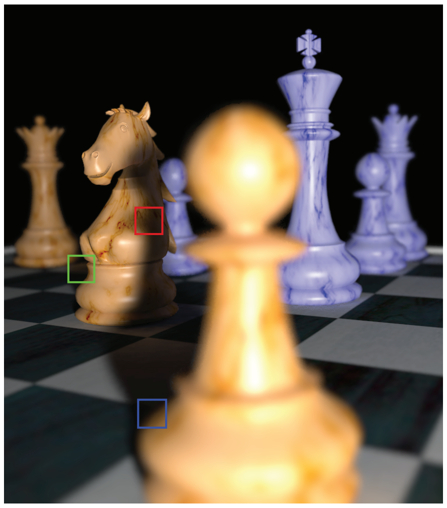
Multiple Effects

5.1 Introduction

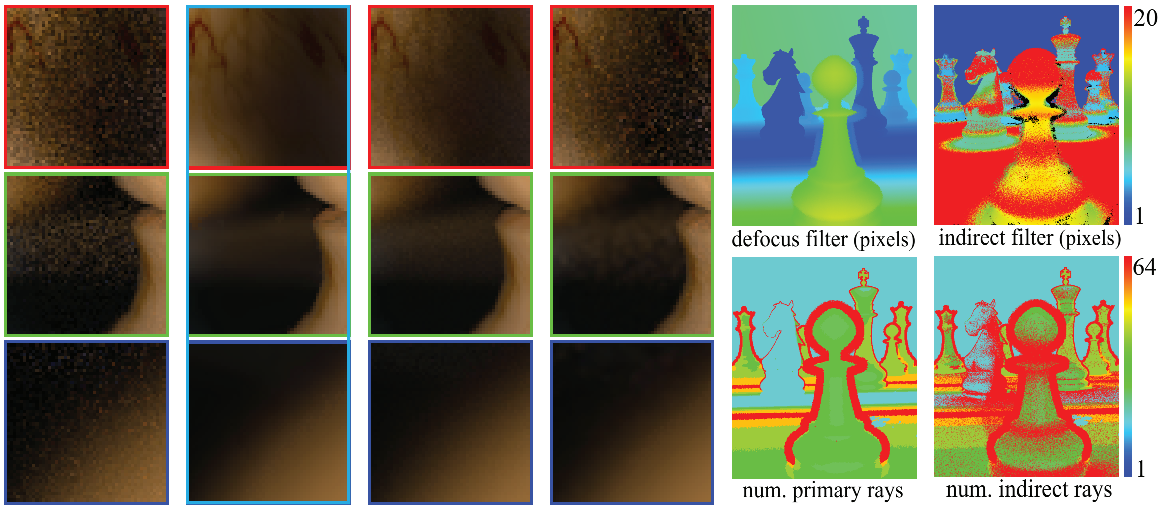
In the previous two chapters, we showed how to correctly filter noisy soft shadows and indirect illumination. These methods are limited to rendering single effects rather than full distribution ray-tracing with multiple effects, and extending them to a higher-dimensional rendering domain is a challenge. 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 chapter, 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 chapter since our GPU ray-tracer doesn't support it, but we provide a proof-of-concept description with results, in Appendix B. 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 [56, 55], resulting in large overhead, or (ii) use expensive atomic computation [9] 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 the previous chapters (also [25, 64]) 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 [23, 9]. This makes our parameters easier to evaluate, without requiring complex frequency analysis along light



(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 s. **(e)** No factoring, 138 rpp, 3.10 sec **(f)** Primary filter and rpp **(g)** Indirect filter and rpp

Figure 5.1: (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. The top row inset is a sharp in-focus region while the other two regions are defocus blurred. 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.

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 [9] only provide a single sampling rate at each pixel and hence are inefficient in reducing ray-tracing cost.

As before, we have integrated our method into NVIDIA's Optix ray-tracer [76] 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 5.12 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[1] for a variety of scenes.

5.2 Previous Work

[14] and [44] 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 pre-

[1]Multiple effects are slower than single effects (previous chapters) since we need more rays, and images are rendered at higher resolution for this chapter.

dicting 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 [83], the A-Trous wavelet transform [19], adaptive wavelet rendering [72] and the filtering of stochastic buffers [91]. Other examples are random parameter filtering (RPF, [88]) which has a high computation and storage cost, [57] that uses Stein's unbiased risk estimator for sampling and bandwidth selection for anisotropic filters, and [85] which uses non-local means filtering and residual errors to guide sampling density. [45] 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 [89] which uses compressed sensing. [21] 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 [61, 78] 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 [36] improves upon [65], and uses sample contrast to guide adaptive sampling, followed by anisotropic reconstruction of the light field. [56] and [55] 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: As in the previous two chapters, we build on recent approaches that have studied the frequency aspects of light transport in static scenes, e.g. [42, 80, 23]. They presented an a-priori frequency analysis to perform adaptive sampling and appropriate reconstruction. [96] 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 [9] 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 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 [56, 55, 88] can filter the noisy irradiance separately but have a high storage and reconstruction overhead. The concurrent work of [100] 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.

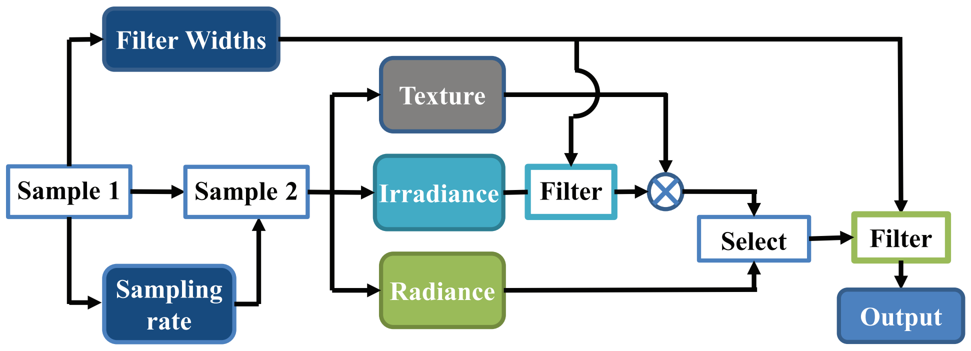


Figure 5.2: A simplified flow chart of the algorithm. In two sampling (ray-tracing) passes, we adaptively sample the noisy per-pixel texture, irradiance and radiance. These are then filtered and combined to produce an accurate radiance value.

5.3 Overview

We describe our algorithm in brief to motivate the theoretical analysis and highlight the main contributions. A block diagram is shown in Fig 5.2. 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 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.

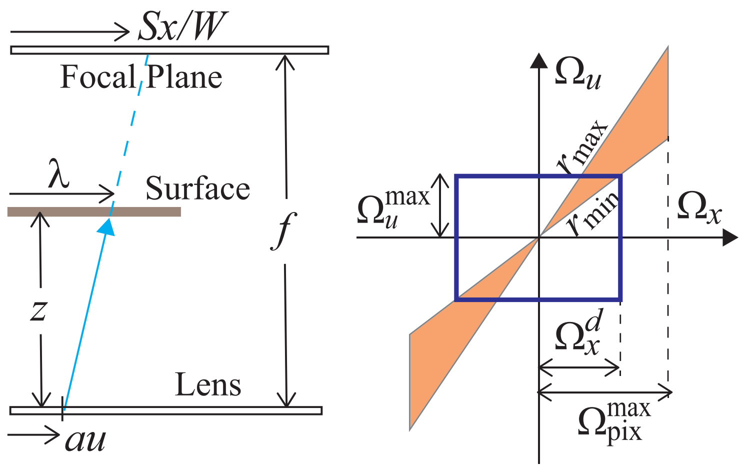


Figure 5.3: (a) Ray-tracing geometry for defocus blur. (b) Fourier spectrum and axis-aligned filter for defocus blur.

5.4 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 5.3(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

$$\lambda = au + \frac{z}{f}\left(\frac{Sx}{W} - au\right) = \frac{Sz}{Wf}\left(x + auW\frac{f-z}{Sz}\right). \tag{5.1}$$

Let us denote

$$r(x,u) = \frac{aW}{S}\left(\frac{f}{z(x,u)} - 1\right) \tag{5.2}$$

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

$$L(x,u) = L_o(\lambda) = L_o(\ell_p \cdot (x+ru)). \tag{5.3}$$

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

$$\hat{L}(\Omega_x,\Omega_u) = \frac{1}{\ell_p}\delta(\Omega_u - r\Omega_x)\hat{L_o}\left(\frac{\Omega_x}{\ell_p}\right). \tag{5.4}$$

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 5.3(b). The final color c at pixel x is

$$c(x) = \int L(x,u)A(u)\,du, \tag{5.5}$$

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

$$\hat{c}(\Omega_x) = \int \hat{L}(\Omega_x,\Omega_u)\hat{A}(-\Omega_u)\,d\Omega_u. \tag{5.6}$$

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 5.3(b). The spatial bandwidth of the axis-aligned filter is

$$\Omega_x^d = \min\left\{\Omega_{\text{pix}}^{\max}, \Omega_{\text{u}}^{\max}/r_{\min}\right\}. \tag{5.7}$$

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. 5.7 implies:

$$R_x^d = \max\{2, r_{\min}\}. \tag{5.8}$$

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: [9] 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. [96] 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 [9] and [96]. Neither of these papers explicitly equates the slope of the defocused light field to the local circles of confusion. The concurrent work of [100] 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.

5.5 Direct Illumination with defocus blur

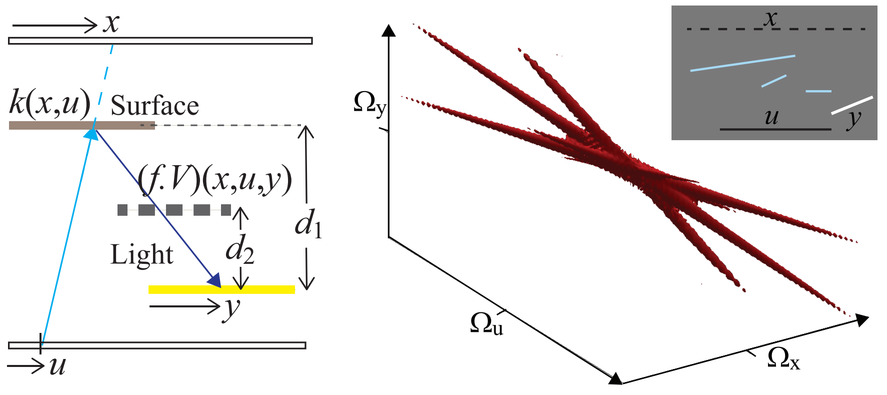
While texture samples at a pixel are functions of the random lens position alone, incident radiance samples (direct and indirect) are functions of two random parameters: the lens position and the illumination direction. Hence, reconstruction can be more efficient if the pixel irradiance can be pre-filtered to remove noise due to incoming direction. To avoid the overhead of storing the full 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 [9] 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 $\Omega_{\mathrm{y}}^{\max} = 1/\ell_I$. 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 5.4(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 [25, 63]) because this simplifies both analysis and implementation. The pixel radiance due to direct illumination from the area light is given by

$$\begin{aligned} L_{\mathrm{dir}}(x) &= \int_u k(x,u) \left(\int_y f(x,u,y) V(x,u,y) I(y)\, dy \right) A(u)\, du \\ &= \int_u k(x,u)\, f_c(x,u) \left(\int_y V(x,u,y) I(y)\, dy \right) A(u) du. \end{aligned} \tag{5.9}$$

Factoring Texture and Irradiance: Equation 5.9 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,

$$\mathbf{E}_A[b_1 \cdot b_2] = \mathbf{E}_A[b_1] \cdot \mathbf{E}_A[b_2] + \mathbf{E}_A[(b_1 - \mathbf{E}_A[b_1])(b_2 - \mathbf{E}_A[b_2])]. \tag{5.10}$$



(a) (b)

Figure 5.4: (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.

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 5.9 as

$$\begin{aligned} L_{\text{dir}}(x) \approx \int k(x,u)A(u)\,du \cdot \\ \int f_c(x,u)A(u)\left(\int V(x,u,y)I(y)\,dy\right)du. \end{aligned} \tag{5.11}$$

We can rewrite this last approximation as

$$L_{\text{dir}}(x) \approx k_{\text{dir}}(x) \cdot E_{\text{dir}}(x), \tag{5.12}$$

where we have defined the integrated texture term as

$$k_{\text{dir}}(x) = \int k(x,u)A(u)\,du \tag{5.13}$$

and the integrated irradiance term as

$$E_{\text{dir}}(x) = \int f_c(x,u)A(u)\left(\int V(x,u,y)I(y)\,dy\right)du. \tag{5.14}$$

Almost all image-space global illumination methods ([32, 105, 64], etc. and Chapter 3 and 4) 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. [9], [57]), 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. As previously discussed in Chapter 2, let $g(\cdot)$ be the one-dimensional visibility in the occluder plane. The set up is as shown in Fig 5.4(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, is[2] $g(\rho\lambda + (1-\rho)y)$. Hence, we have

$$\begin{aligned} V(x,u,y) = V(\lambda,y) &= g(\rho\lambda + (1-\rho)y) \\ &= g(\rho\ell_p(x+ru) + (1-\rho)y) \equiv g(\alpha x + \beta u + \gamma y). \end{aligned} \tag{5.15}$$

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

$$\begin{aligned} \hat{V}(\Omega_x,\Omega_u,\Omega_y) &= \int\int\int g(\alpha x + \beta u + \gamma y) \\ &\quad \exp(-j(x\Omega_x + y\Omega_y + u\Omega_u))dx\,du\,dy \\ &= \frac{1}{\alpha}\hat{g}\left(\frac{\Omega_x}{\alpha}\right)\delta\left(\Omega_u - \frac{\beta}{\alpha}\Omega_x\right)\delta\left(\Omega_y - \frac{\gamma}{\alpha}\Omega_x\right). \end{aligned} \tag{5.16}$$

This is a line through the origin in 3D frequency space, normal to the isosurface plane defined by eqn. 5.15. Due to the integration with aperture and light in eqn. 5.14, this line is bandlimited (clipped) by the planes $|\Omega_u| \le \Omega_{\mathrm{u}}^{\max}$ and $|\Omega_y| \le \Omega_{\mathrm{y}}^{\max}$. The two slopes given by the delta functions in eqn. 5.16 imply that the line is clipped along Ω_x to $|\Omega_x| \le \Omega_{\mathrm{u}}^{\max}/r$ and $|\Omega_x| \le \ell_p\Omega_{\mathrm{y}}^{\max}/(\rho^{-1}-1)$. Let

$$s = \rho^{-1} - 1 = (d_1/d_2) - 1. \tag{5.17}$$

s is analogous to r defined in eqn. 5.2. For non-parallel and multiple receivers and occluders, the visibility spectrum becomes a bandlimited double-cone in frequency space. As illustrated in Fig 5.4 (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

$$\Omega_x^s = \min\left\{\Omega_{\mathrm{pix}}^{\max},\, \ell_p\Omega_{\mathrm{y}}^{\max}/s_{\min},\, \Omega_{\mathrm{u}}^{\max}/r_{\min}\right\} \tag{5.18}$$

Equivalently, the primal-domain filter size in pixels is

$$R_x^s = \max\left\{2,\, \ell_I s_{\min}/\ell_p,\, r_{\min}\right\}. \tag{5.19}$$

Filtering using factoring: In practice, we can use factoring (eqn. 5.12) wherever the error[3] $||L_{\mathrm{dir}} - k_{\mathrm{dir}} \cdot E_{\mathrm{dir}}||$ (obtained from a first sparse sampling pass) is small enough. For the pixels where the

[2]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.

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

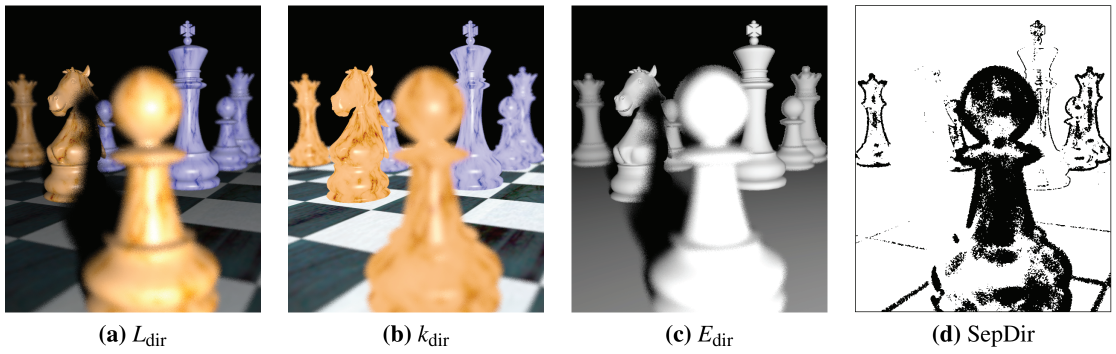

(a) L_{dir} **(b)** k_{dir} **(c)** E_{dir} **(d)** SepDir

Figure 5.5: (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).

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 5.5(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.

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. 5.13 becomes

$$k(x,u) = k_d(x,u) + k_s(x,u)p(x,u) \tag{5.20}$$

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

5.6 Indirect Illumination

The total pixel radiance is the sum of the integrated direct and indirect radiance, i.e. $L(x) = L_{\text{dir}}(x) + L_{\text{ind}}(x)$, 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 same parametrization as the previous chapter, 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[4] $h(v)$ 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:

$$\begin{aligned} L_{\text{ind}}(x) &= \int k(x,u)\,(\int h(v) l_i(x,u,v)\,dv\,) A(u) du \\ &\approx \int k(x,u) A(u)\,du \cdot \int A(u)\,(\int h(v) l_i(x,u,v)\,dv)\,du \\ &\equiv k_{\text{ind}}(x) \cdot E_{\text{ind}}(x). \end{aligned} \tag{5.21}$$

This is similar to the direct lighting equation, with the light intensity I replaced by the transfer function h. If the factorization error $||L_{\text{ind}} - k_{\text{ind}} \cdot E_{\text{ind}}||$ is small, we use the factored product $k_{\text{ind}} \cdot E_{\text{ind}}$. 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 the previous chapter, 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. 5.18, the filter width for E_{ind} is given by

$$\Omega_x^i = \min\left\{\Omega_{\text{pix}}^{\max},\, \ell_p \Omega_{\text{v}}^{\max}/z_{\min},\, \Omega_{\text{u}}^{\max}/r_{\min}\right\}. \tag{5.22}$$

$\Omega_{\text{v}}^{\max}$ is the bandlimit of the low-pass transfer function h; the numerical values for diffuse and glossy BRDFs can be found in Chapter 4 and Appendix A. The primal domain filter size is

$$R_x^i = \max\left\{2,\, z_{\min}/(\ell_p \Omega_{\text{v}}^{\max}),\, r_{\min}\right\}. \tag{5.23}$$

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. 5.20 cannot be made for indirect illumination. For the factorization $k_{\text{ind}} \cdot E_{\text{ind}}$ to work for a surface with both diffuse and glossy components, each of $L_{\text{ind}}, k_{\text{ind}}, E_{\text{ind}}$ must be evaluated and stored separately

[4]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 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 5.22 and then combined with the appropriate k_{ind}.

5.7 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. Similar to the previous chapters, 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. 5.6(a). We have,

$$\begin{aligned} n_p &= (\Omega_x^*)^2(\Omega_u^*)^2 \\ &= (\Omega_{\text{pix}}^{\max} + \Omega_x^d)^2(1 + r_{\max}\Omega_x^d)^2. \end{aligned} \tag{5.24}$$

Although [96] 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.

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 5.6(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

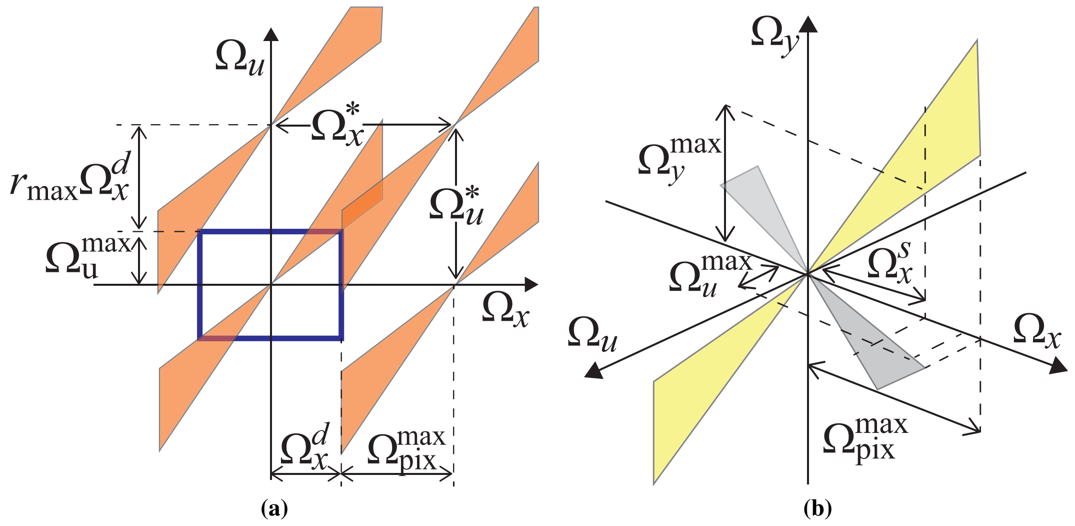


Figure 5.6: (a) Packing of aliases for defocus only, showing the spatial and lens sampling rates Ω_x^* and Ω_u^* (b) Packing for aliases of the light visibility V. The yellow and grey double wedges are the projection on the $\Omega_x \times \Omega_y$ and $\Omega_x \times \Omega_u$ planes respectively. Aliases not shown for clarity. The minimum sampling rates Ω_x^*, Ω_y^* and Ω_u^* are analogous to those shown in (a).

per-pixel sampling rate for $V(x,u,y)$ (secondary shadow rays) is

$$\begin{aligned} n_{\text{dir}}^f &= (\Omega_x^*)^2(\Omega_u^*)^2(\Omega_y^*)^2\ell_I^2 \\ &= (\Omega_{\text{pix}}^{\max} + \Omega_x^s)^2(1 + r_{\max}\Omega_x^s)^2(1 + \ell_I s_{\max}\Omega_x^s/\ell_p)^2 \end{aligned} \tag{5.25}$$

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_x^i. The sampling rate is then,

$$\begin{aligned} n_{\text{ind}}^f &= (\Omega_x^*)^2(\Omega_u^*)^2(\Omega_v^*)^2 \\ &= (\Omega_{\text{pix}}^{\max} + \Omega_x^i)^2(1 + r_{\max}\Omega_x^i)^2(\Omega_{\text{v}}^{\max} + z_{\max}\Omega_x^i/\ell_p)^2 \end{aligned} \tag{5.26}$$

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_x^d. The direct illumination sampling rate is then

$$n_{\text{dir}}^c = (\Omega_{\text{pix}}^{\max} + \Omega_x^d)^2(1 + r_{\max}\Omega_x^d)^2(1 + \ell_I s_{\max}\Omega_x^d/\ell_p)^2 \tag{5.27}$$

Finally, without factorization, the indirect illumination sampling rate is

$$n_{\text{ind}}^c = (\Omega_{\text{pix}}^{\max} + \Omega_x^d)^2(1 + r_{\max}\Omega_x^d)^2(\Omega_{\text{v}}^{\max} + z_{\max}\Omega_x^d/\ell_p)^2 \tag{5.28}$$

Discussion: Observe from eqns. 5.7, 5.18 that $\Omega_x^s \leq \Omega_x^d \leq \Omega_{\text{pix}}^{\max}$. As expected, if the pixel is in focus (i.e., either $f = z$ or $a = 0$) the number of primary rays per pixel is $n_p = (2\Omega_{\text{pix}}^{\max})^2 = 1$ 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. 5.25. Then we must trace $n^f_{\text{dir}}/n_p \ge 1$ secondary rays per primary ray[5].

We now qualitatively discuss our practical sampling rates shown in fig 5.8(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 discussed before, 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:

$$n_p(\mu) = (\Omega^{\max}_{\text{pix}} + \Omega^d_x(\mu))^2 (1 + r_{\max}\Omega^d_x(\mu))^2, \tag{5.29}$$

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. 5.12(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.

5.8 Implementation

A flow chart of our algorithm is shown in Figure 5.7. All quantities are concisely defined in Table 1 which also points to the relevant equations.
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

[5]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.

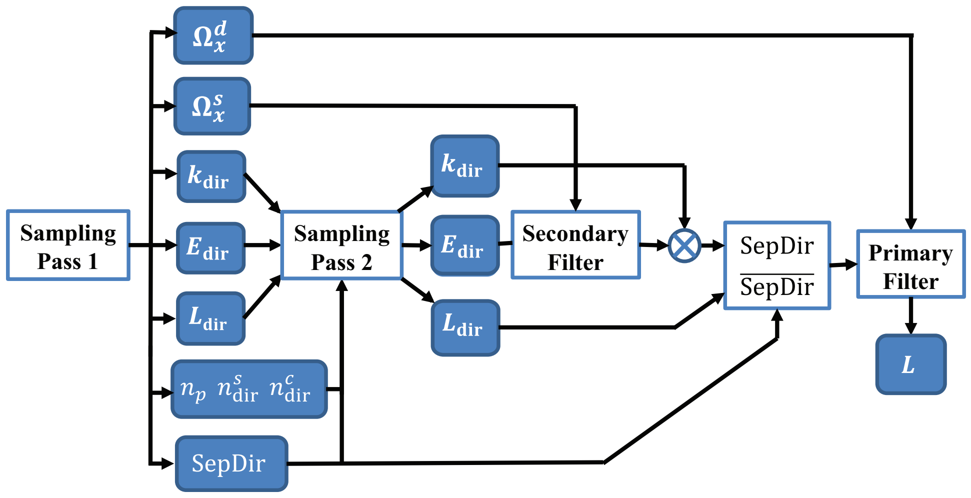


Figure 5.7: Flow chart of the algorithm for the direct component only; the indirect component is handled similarly. Filled blocks are variables stored in memory, empty blocks are operations. Refer to table 1 for definitions of variables.

Quantity	Description	Equation
L_{dir}	integrated color, direct component	5.9
k_{dir}	integrated texture, direct component	5.13
E_{dir}	integrated illumination, direct component	5.14
L_{ind}	integrated color, indirect component	5.21
k_{ind}	integrated texture, indirect component	5.21
E_{ind}	integrated illumination, indirect component	5.21
SepDir	Boolean, set if direct factorization error is small	5.30
SepInd	Boolean, set if indirect factorization error is small	5.30
$r_{\text{min}}, r_{\text{max}}$	min, max circle of confusion in pixels	5.2
$s_{\text{min}}, s_{\text{max}}$	min, max soft shadow slopes	5.17
$z_{\text{min}}, z_{\text{max}}$	min, max reflector distance for indirect	-
Ω_x^d	depth-of-field filter width	7
Ω_x^s	direct illumination filter width	16
Ω_x^i	indirect illumination filter width	19
n_p	Num. primary (lens) rays	5.24
n_{dir}^c	Num. light shadow rays if SepDir $= 0$	5.27
n_{dir}^f	Num. light shadow rays if SepDir $= 1$	5.25
n_{ind}^c	Num. indirect samples if SepInd $= 0$	5.28
n_{ind}^f	Num. indirect samples if SepInd $= 1$	5.26

Table 5.1: A list of the variables we store in a per-pixel buffer. The defining equation for each variable is indicated in the last column.

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 [90]. 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 $\Omega_x^r, \Omega_x^s, \Omega_x^i$ and set flags:

$$\begin{aligned}\text{SepDir} &= \|L_{\text{dir}} - k_{\text{dir}} \cdot E_{\text{dir}}\| < 0.01 \\ \text{SepInd} &= \|L_{\text{ind}} - k_{\text{ind}} \cdot E_{\text{ind}}\| < 0.01\end{aligned} \tag{5.30}$$

2. **Sampling pass 2:** The primary and secondary sampling rates $n_p, n^f_{\text{dir}}, n^c_{\text{dir}}, n^f_{\text{ind}}$ and n^c_{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 SepDir $= 1$, we trace n_p primary rays, and from each primary hit-point trace n^f_{dir}/n_p shadow rays when SepDir $=$ 1 and n^c_{dir}/n_p shadow rays otherwise. Samples are fully stratified[6] 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_{dir} and E_{ind} are filtered. We filter using world space distances and filter width Ω_x^s and Ω_x^i. 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

$$w_i(j) = \exp\left\{-16\|\lambda(i) - \lambda(j)\|^2 \cdot (\Omega_x^s(i)/\ell_p(i))^2\right\} \tag{5.31}$$

Note that the projected pixel length ℓ_p converts Ω_x^s into meters. For adjacent pixels i and j, $\|\lambda(i) - \lambda(j)\| \approx \ell_p(i)$; if $\Omega_x^s = 0.5$ 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:

$$\begin{aligned}L(x) = {} & \text{SepDir} \cdot (k_{\text{dir}} \cdot E_{\text{dir}}) + \overline{\text{SepDir}} \cdot (L_{\text{dir}}) \\ & + \text{SepInd} \cdot (k_{\text{ind}} \cdot E_{\text{ind}}) + \overline{\text{SepInd}} \cdot (L_{\text{ind}})\end{aligned} \tag{5.32}$$

[6]This requires that we round up n_p to p^2 and n_{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.

The final pass filters the total pixel radiance $L(x)$ using a screen-space gaussian filter of width Ω_x^d to compute the final color; the weights are analogous to equation 5.31,

$$w_i(j) = \exp\left\{-16(i-j)^2 \cdot (\Omega_x^d(i))^2\right\}. \tag{5.33}$$

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.

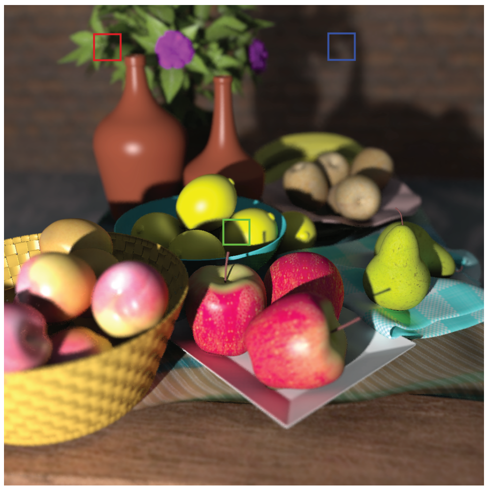
5.9 Results

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.

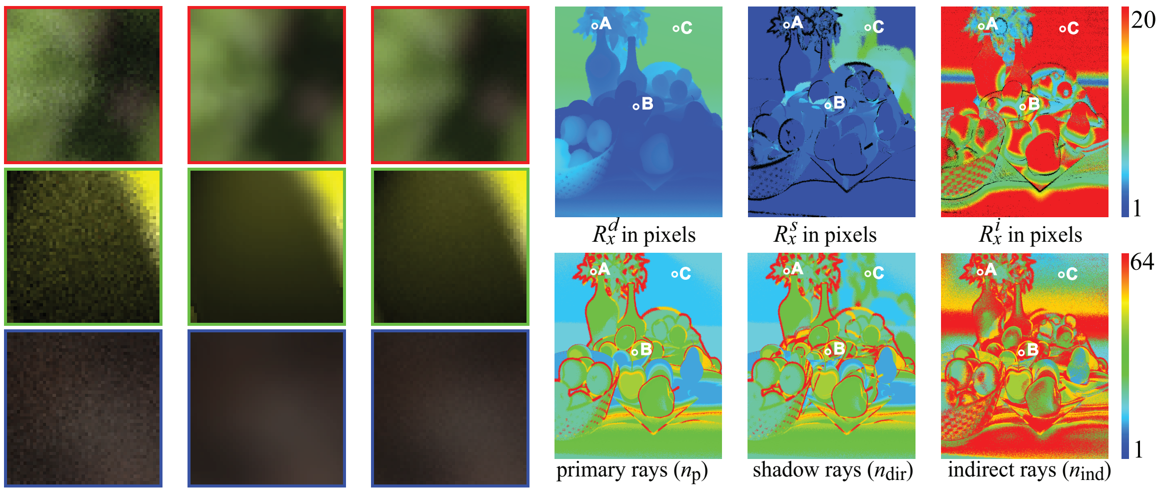
Our method is accurate in a range of different scenarios, with consistent reductions in sample counts over basic path tracing. Figure 1 shows the CHESS scene (21K triangles), with mid-depth focus. Our image (a,c) is noise-free with 138 average rays per pixel (rpp) in only 3.14 sec. Equal visual quality ground truth MC (d) requires 5390 rpp and $40\times$ 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 5.8 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. 5.9 and TOASTERS in Fig. 5.11, with comparisons are discussed below. We also include ground truth insets, obtained at about 15000 rpp, in these figures. Figure 5.12, the ROOM scene, demonstrates that our method can produce fast results for scenes with complex light paths.

5.9.1 Timings

As demonstrated in Fig. 5.10, to obtain the same visual quality and noise level as our method, MC path-tracing requires about $30\times$ more rays, so we get a corresponding speed-up. In Table 5.2, 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, even though our algorithm is

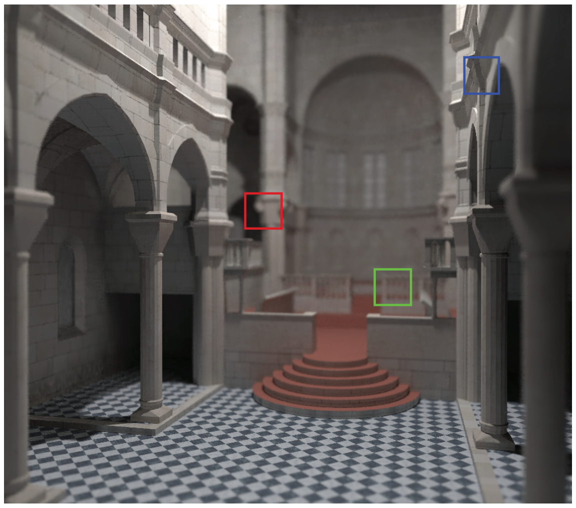


(a) Our method, 178 rpp, 7.40 sec

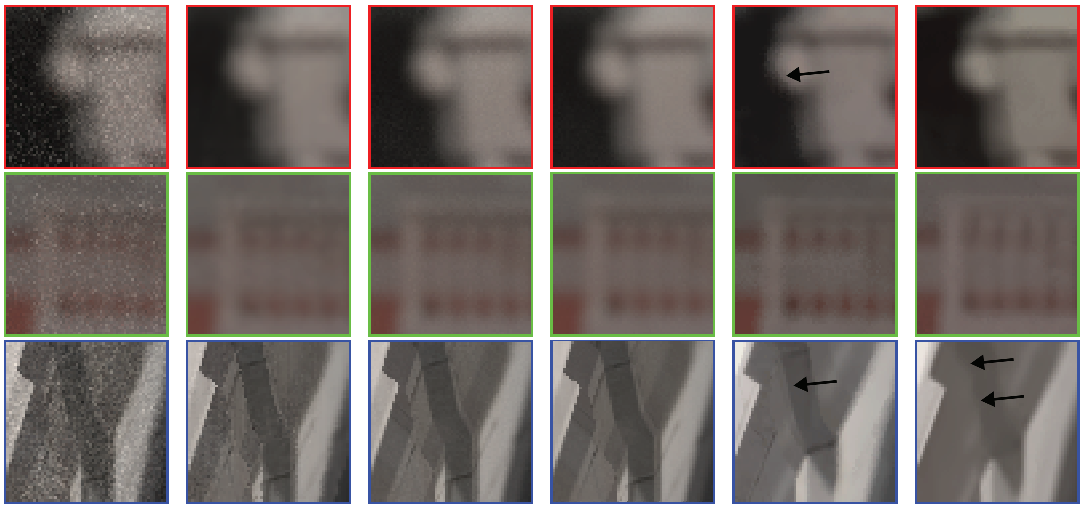

A
B
C
R_x^d in pixels
R_x^s in pixels
R_x^i in pixels
20
1
primary rays (n_p)
shadow rays (n_dir)
indirect rays (n_ind)
64
1

(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 5.8: 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 5.9: (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.

Scene	Tris	rpp	Sample (sec)	Filter (sec)	Total (sec)	Over-head
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 5.2: 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.

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.

5.9.2 Quantitative Accuracy

We evaluated the accuracy of our method quantitatively; in Fig. 5.12(e) we show average per-pixel RMS error vs average number of rays for the ROOM scene. The error of our method (blue curve) 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. 5.29), 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. 5.10.

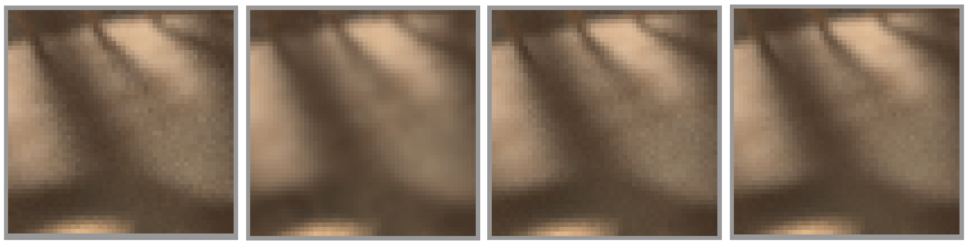


Figure 5.10: We compare stratified MC and our method with increasing sample count for an inset from the ROOM scene (Fig. 5.12). Stratified MC visually matches our method for about 5220 rpp. At 180 rpp, our method is very slightly over-blurred, but MC at 5220 rpp shows more noise (zoom in) in comparison.

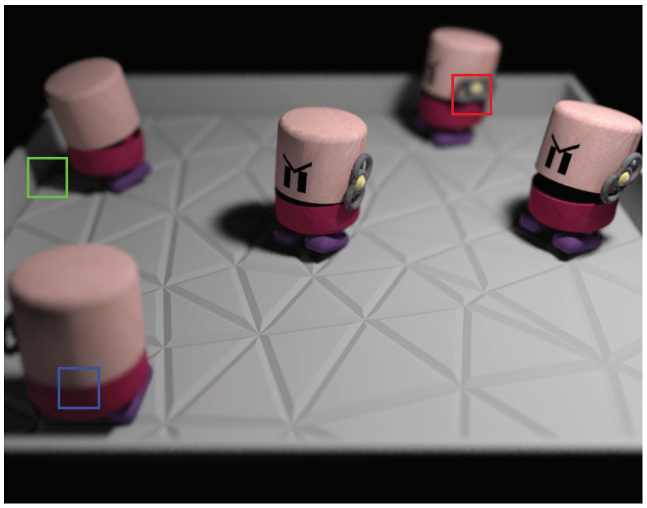
5.9.3 Comparisons

We have already discussed comparison to brute-force equal time and equal visual quality MC. In Figs. 5.9(f,g) and 5.11(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, [57], since for the same quality, they are faster than other recent approaches such as [88, 86], etc. The 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, [45]) demonstrated faster results using PBRT for adaptive sampling and BM3D [18] 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 [9] since the method has high overhead (reported as 2.5 min, total time 25 min on CPU) and implementation is complex.

5.10 Limitations

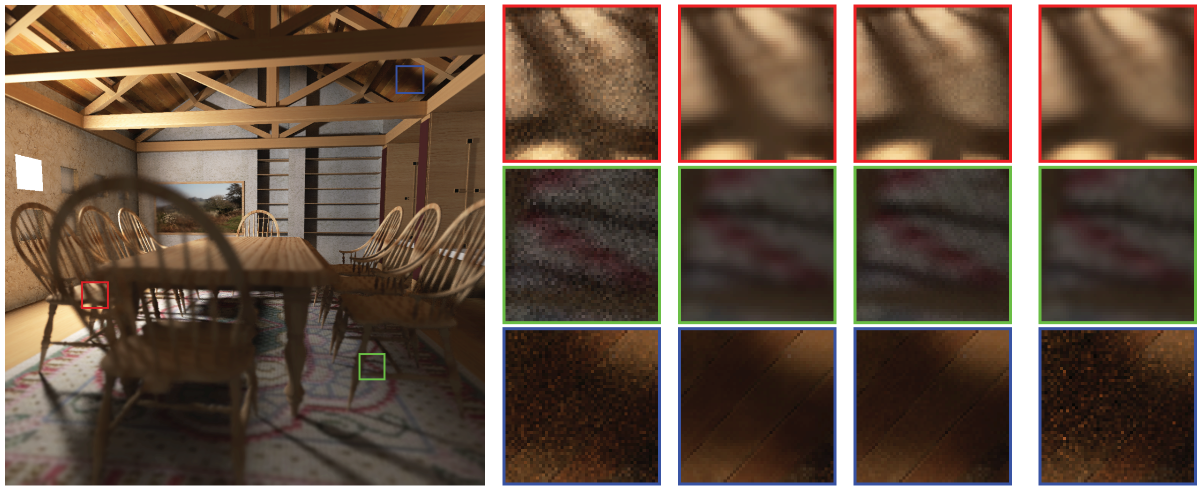
Direct and indirect illumination reflected from glossy surfaces cannot be treated with a simple approach like ours, and more complex analysis is required. Our diffuse bandwidths can result in some over-blurring of specular highlights. Like most image-space filtering methods, we need to allocate a lot 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. 5.9c lower inset). Since frequency analysis



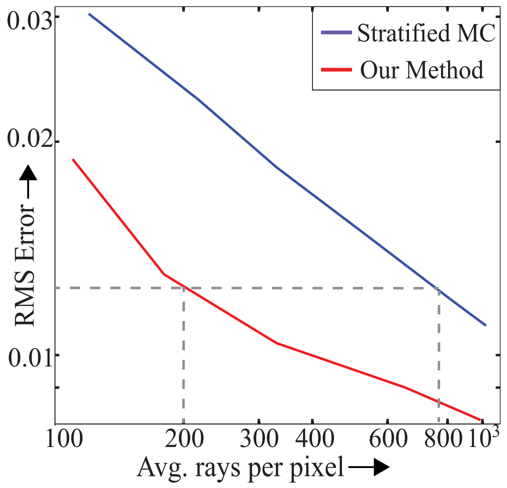
(a) Our method, 125 rpp, 3.61 sec

(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 5.11: (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 5.12: (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.

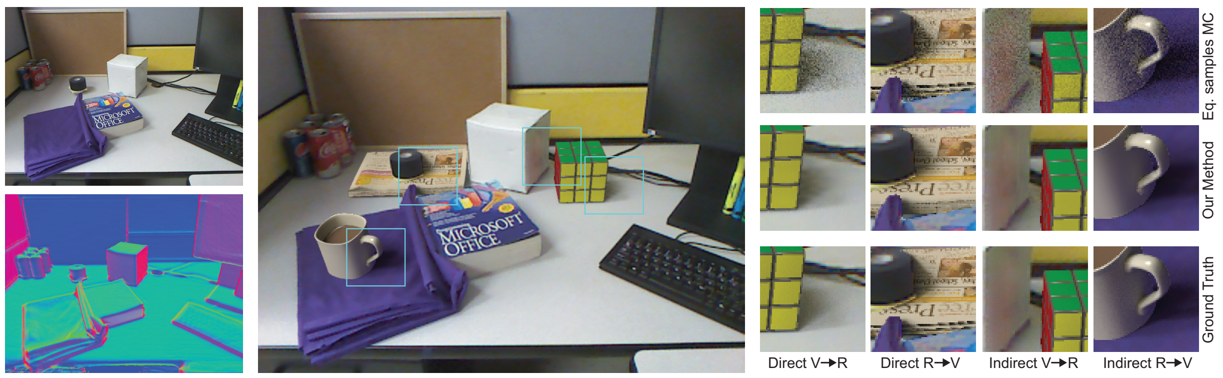
is only effective on locally smooth surfaces, high-frequency bump or normal mapping cannot be handled. Spatial anti-aliasing cannot be done directly because our factorization requires that only one image sample per pixel be taken. But it is easy to do indirectly, for example, for $4\times$ AA, render the image at $4\times$ resolution, with $\Omega_{\text{pix}}^{\text{max}} = 0.25$ (which requires fewer rays) and then downsample. An example of this is shown for the SIBENIK scene in the accompanying video.

Chapter 6

Environment Illumination and Application to Mixed Reality

6.1 Introduction

In chapter 3, we explained filtering direct illumination (soft shadows) from area lights. Another commonly used light source model in physically-based as well as real-time rendering is the environment light source, where a surface receives illumination from light sources located at infinity, i.e. from every direction. Indirect illumination from an environment light source can be handled



(a) Input RGB and normals **(b)** Our Method, 5.7 fps **(c)** Zoom-in Comparisons

Figure 6.1: We start with a stream of RGBD images from a Kinect camera, and use SLAM to reconstruct the scene. (a) shows the input RGB image and reconstructed normals. In this DESK scene, we insert a virtual rubik's cube, a newspaper, and a coffee mug, as shown in (b) where environment illumination (both direct and 1-bounce indirect) are computed using Monte Carlo (MC) path tracing followed by filtering. The full system runs at 5.7 fps. We show comparisons of our result with unfiltered MC with equal samples, which is very noisy, and reference, which takes 60× longer to render.

similar to chapter 4, however, the dominant direct illumination component behaves differently from area light soft shadows of chapter 3. In the past, various approximate techniques for environment lighting have been proposed, such as spherical harmonics and precomputed radiance transfer. Monte Carlo (MC) ray tracing is the physically accurate technique for rendering photo-realistic imagery with environment illumination. However, the number of rays needed to produce a visually pleasing noise-free image can be large, resulting in hours to render a single image.

As in the previous chapters, on smooth surfaces, shading from environment illumination is often slowly varying, and hence one approach for fast rendering is to exploit the smoothness by appropriately filtering a sparsely-sampled Monte Carlo result (see Fig. 5.1). We extend the axis-aligned filtering algorithm of the previous chapters, previously limited respectively to area light direct illumination and indirect illumination only, to filter environment illumination (direct and indirect) adaptively in screen-space. The filtering scheme is fast and in screen-space, and at the same time does not overblur the shading.

In Sec.6.6, we analyze the illumination, and resulting image shading in the Fourier domain. While previous work [23] has conducted such a frequency analysis, we provide new insights into the shape of spatio-angular radiance spectra. The 2D (flatland) light fields of incident illumination and visibility have different slopes. We show that convolution of the corresponding spectra in Fourier space, is an oriented ellipsoid, unlike previous double-wedge models (see Fig. 6.4). By understanding the nature of this spectrum, we derive an axis-aligned filter and compute the spatial shading bandwidth, for both diffuse and glossy cases. Using our Fourier analysis and bandwidth prediction, we derive Gaussian image space filters (Sec. 7) for environment map direct lighting. In addition, we make two minor changes to previous axis-aligned filtering methods – (i) Temporal filtering, to use the result of the previous frame as an input to our filter, which helps reduce noise and (ii) Anti-aliasing for primary visibility with 4 samples per pixel.

We demonstrate our technique applied to rendering mixed reality (MR). The fundamental objective of MR applications for immersive visual experiences is seamlessly overlaying virtual models into a real scene. The synthesis has two main challenges: (1) stable tracking of cameras (pose estimation) that provides a proper placement of virtual objects in a real scene, and (2) plausible rendering and post processing of the mixed scenes. For the first challenge, there are many existing techniques that can be leveraged. In this paper, we use dense simultaneous localization and mapping (SLAM) algorithms [68, 41] that provide the estimated 6DOF camera pose, as well as scene geometry in the form of per-pixel positions and normals. For the second challenge, many approaches use either rasterization with a dynamic but noisy real-world mesh obtained directly from a depth camera [49], or Monte Carlo ray-trace a fixed pre-defined real-world mesh [47]. We provide a two-mode path-tracing method that uses a denoised real-world vertex map obtained from the SLAM stage. As described in Sec. 6.5, a fast GPU ray-tracer is used for the virtual geometry, while real geometry is intersected with rays traced in screen space. Finally, we use our filtering scheme on a 16 samples/pixel noisy MC result to remove noise. This gives us the quality of a purely path-traced result at interactive rates of 5 frames/second. Note, we assume all real surfaces are diffuse, while virtual objects can be diffuse or glossy. The environment lighting is obtained by photographing a mirrored ball.

6.2 Previous Work

Environment Illumination: The most popular approach for real-time rendering with environment lighting is to use spherical harmonics [81] with pre-computed ambient occlusion, or more generally, precomputed radiance transfer [94]. This approach is limited to low-frequency lighting, and requires pre-computation that does not support dynamic MR.

Rendering in MR: A pioneering work in light transport for MR was presented by Fournier et al. [29], based on radiosity. It was later extended using image based lighting derived from a light probe [20]. Gibson and Murta [33] present a hardware-rendering approach using pre-computed basis radiance-maps and shadow maps. Cossairt et al. [15] synthesize inter-reflections without knowing scene geometry, using a controlled set-up to capture and re-project a 4D radiance light field. Recently, some work has adapted more modern techniques for MR, such as differential instant radiosity [49] using imperfect shadow mapping, and delta voxel cone-tracing [30]. While they show fast impressive results, they are not physically accurate.

Our approach is based purely on ray-tracing. Closest to our approach are differential progressive path tracing [47] and differential irradiance caching [46], which compute direct and indirect illumination using path-tracing. Both methods use pre-determined real-world mesh geometry. Differential progressive path-tracing uses only one sample/pixel/frame on the combined scene geometry. Since they do not filter their result, only a very noisy image is achievable in real-time. The user must wait without camera motion for the image to become noise-free. We overcome this limitation through the use of fast yet accurate filtering, and fast accumulation of 16 samples per pixel per frame. Further, we use screen-space ray-tracing for real objects. Methods like Kán and Kaufmann [48] demonstrate reflections, refractions and caustics through the use of differential photon mapping, which improves the realism. We can handle reflections and refractions, but we did not implement caustics, which would require photon mapping. We use pure ray-tracing and focus on low- to mid-frequency shading.

To improve the realism of inserted virtual objects, many works focus on post-processing techniques to match color palette and noise between pixels belonging to real and virtual objects. Our goal in this paper is primarily to provide a photorealistic rendering system for MR scenes. We do not focus on tracking/reconstruction quality, lighting estimation, and post-processing techniques.

Screen-space Ray-tracing (SSRT): We introduce a hybrid ray-tracing scheme that uses screen-space rays to find intersections with real geometry represented as a vertex map. Mark et al. [60] use depth information to re-render an image from a nearby viewpoint. This idea can be extended to screen-space ray-tracing, where rays traverse pixels and test depth for intersection – a technique that has been used for local specular reflections [97, 59]. We use SSRT to compute direct and indirect illumination as well. Like many previous works, the real-world mesh is represented as a dynamically updated vertex map of the current frame. Hence, real objects not visible in the current frame do not affect the shading of virtual objects. One could potentially use the global volumetric signed-distance function maintained by the SLAM back-end for ray-tracing; however, this would be extremely slow.

Fourier Analysis and Axis-aligned Filtering: As in the previous chapters, we are inspired by Chai et al. [12] and Durand et al. [23], who introduce the basic Fourier theory for space-angle and pixel-light light fields. The latter work, Fourier Analysis of Light Transport (FLT), models light-surface interactions atomically, and derives the Fourier equivalent for each interaction. Their Fourier spectra are parallelograms, while we show that the spectra can actually have an ellipsoidal shape; our bandwidths are more accurate. More recently, Belcour et al. [9] model the shading spectrum as a Gaussian covariance matrix that is the product of matrices for each atomic interaction, that is expensive and slow to compute. We use an end-to-end geometric approach that directly gives the object space bandwidth and is fast to compute. Bagher et al. [7] use bandwidth prediction to shade different materials under environment lighting via heirarchical shading, but do not consider occlusion.

Egan et al. [24] show that the Fourier spectrum for ambient occlusion in a position-angle space is a double wedge, and demonstrate a sheared filter that tightly fits the spectrum. They demonstrate good results for low-frequency environment lighting with 32 samples/pixel, although they do not take the interaction of BRDF and visibility into account, and their filter is offline.

The previous chapters 3 and 4 ([63] and [64]) are respectively useful only for area light soft shadows and indirect illumination. We extend this approach to handle environment lighting, which requires a different curvature-dependent parametrization, since the light sources are at infinity. While these previous works and chapters treat the spectrum to be strictly a double-wedge, the spectra in this chapter are not restricted to this model. The axis-aligned filter size is no longer dependent on only the BRDF or the lighting bandlimit, but combines the effect of both terms in a non-trivial way.

Denoising Monte-Carlo Images: Image filtering is a popular approach to remove noise in MC images, 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. The state of the art in this domain includes [45] (AMLD) that gives a noise estimation metric to locally identify the amount of noise in different parts of the image, with adaptive sampling and filtering using standard denoising techniques. Other approaches include use of Stein's unbiased risk estimator (SURE,[57]), ray histogram fusion [21] and adaptive local regression [66]. These approaches support general rendering effects, but have a high reconstruction overheads in seconds, and are offline. Recently, [99] (Fast-ANN) have shown approximate-nearest-neighbor collaborative filtering for general images at real-time speeds. We compare results to AMLD and Fast-ANN.

6.3 Differential Rendering

Our MR rendering system is based on the differential rendering method of Debevec [20]. The idea is to estimate the pixel color considering only real objects, and considering both real and virtual objects, and add the *difference* to the raw camera image. Let L_R be the (per-pixel, outgoing) radiance due to real objects only, and L_{RV} be the radiance due to both real and virtual objects (including indirect illumination between them). Then, differential rendering composites the final image per the equation:

$$L_{final} = (1-M) \cdot L_{RV} + M \cdot (L_{RV} - L_R + L_{cam}) \tag{6.1}$$

Here, M is the fraction of the pixel covered by real objects, and L_{cam} is the input radiance image.

Calculating the radiances L_R and L_{RV} implicitly requires knowledge of the real-object BRDF and texture. Under the assumption that all surfaces are diffuse, only the real-object RGB texture (albedo) k_R is unknown. We show that eqn. 6.1 can be written purely in terms of irradiances, without the need to explicitly estimate k_R.

We separate all outgoing radiances into a product of irradiance E and texture k. The first term in eqn. 6.1 corresponds to the contribution of a virtual object, so we replace L_{RV} with $k_V E_{RV}$ where k_V is known virtual object texture and E_{RV} is the pixel irradiance considering both real and virtual objects. The second term corresponds to the contribution of real objects, and we write $L_{RV} - L_R = k_R(E_{RV} - E_R)$. Like previous works, the estimate of k_R is $k_R = L_{\text{cam}}/E_R$. Substituting this into eqn. 6.1, we get:

$$\begin{aligned} L_{final} &= (1-M) \cdot k_V E_{RV} + M \cdot (k_R(E_{RV} - E_R) + L_{cam}) \\ &= (1-M) \cdot k_V E_{RV} + M \cdot \left(\frac{L_{cam}}{E_R}(E_{RV} - E_R) + L_{cam} \right) \end{aligned}$$

Simplifying,

$$L_{final} = (1-M) \cdot k_V E_{RV} + M \cdot \left(\frac{E_{RV}}{E_R} L_{cam} \right). \tag{6.2}$$

Thus, we have eliminated the unknown k_R completely. Our task now is to estimate E_R and E_{RV}.

Glossy Virtual Objects: Above, we consider only diffuse real and virtual objects, so that the radiance can be factored into texture and irradiance. However, we can easily handle glossy virtual objects. In the glossy case, the L_{RV} in the first term of eqn. 6.1 would be split into a spatial texture and outgoing radiance that depends on viewing angle. We may represent this outgoing radiance with the same symbol E_{RV}, without changing the resulting analysis. We follow this convention for glossy virtual objects throughout the paper for simplicity. In Sec. 6.6.3, our theory treats glossy BRDFs rigorously.

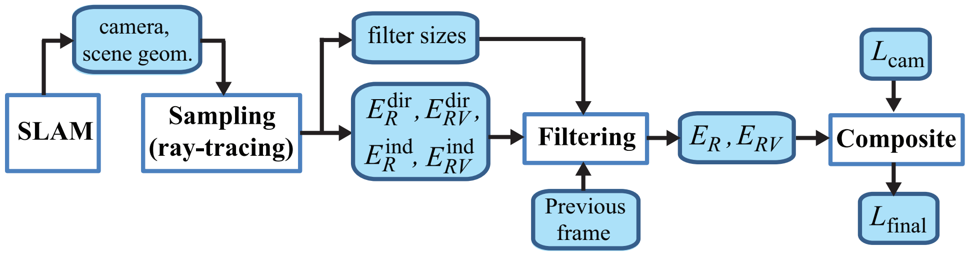


Figure 6.2: An abstract flowchart of our MR system. Stored input/output variables are enclosed in blue boxes, and computational steps are enclosed in white boxes. We start with dense SLAM to estimate camera pose and scene geometry from depth, followed by sparse Monte Carlo sampling to compute noisy estimates of four irradiance values (Refer to Sec. 6.4 for more details). We then filter the irradiances using our theory, also using the values from the previous frame for further noise reduction. The filtered values are then used to composite the final image, as explained in Sec. 6.3.

6.4 Overview

Figure 6.2 shows an overview of our MR system. Our system can be abstracted into four distinct stages; each stage is briefly explained below.

We take the RGBD stream from a Kinect camera and use the depth data to estimate camera pose and scene geometry. To realistically render virtual objects, the scene lighting in the form of an environment map must be known. We obtain the environment map from an image of a reflective metallic ball, using a camera looking down from above (the environment map may not be changed except for simple rotations). The entire pre-processing is done once for a single scene; each of the following stages runs per-frame at interactive speed.

1. Dense SLAM: The first step is camera tracking and scene reconstruction, to estimate per-pixel world-coordinates, normals as well as camera pose (rotation+translation), from the depth images. We use InfiniTAM [79], an open-source implementation of GPU-based voxel-hashing Kinect fusion [70]. It is fast, running at about 50 fps, and provides high quality scene reconstruction. Reconstructed normals for each scene are shown in the corresponding figures. While we can support dynamic geometry, both real and virtual, the SLAM backend is not robust to dynamic real objects. So, we only demonstrate dynamic virtual objects.

2. Sampling: We use two Monte Carlo path-tracing passes to estimate per pixel illumination without and with virtual objects. Each is the sum of direct illumination from an environment light source and 1-bounce indirect illumination, i.e., $E_R = E_R^{dir} + E_R^{ind}$, and $E_{RV} = E_{RV}^{dir} + E_{RV}^{ind}$. Thus, we compute four independent components: $E_{R|RV}^{dir|ind}$. The sampling algorithm is described in Sec. 6.5, with detailed pseudo-code in Appendix C.

3. Filtering: Obviously, the Monte Carlo sampled result is very noisy, and hence in the next stage, we filter each of the four irradiances using physically-based filters. The filter bandwidths are derived using Fourier analysis in Sec. 6.6. To reduce noise further, we also save the unfiltered irradiances from the previous frame, and use them as additional inputs to our filter. Details of this temporal filtering are discussed in Sec. 6.7.

4. Compositing: The last step involves compositing the final MR image, using the filtered values E_R and E_{RV} and the input RGB image, according to eqn. 6.2.

6.5 Two-mode Sampling Algorithm

In this section, we discuss our sampling algorithm (step 2 in the overview above). We aim to compute the pixel color using physically-based Monte Carlo sampling. The input is the camera position, per-pixel real object world positions and normals, and virtual object geometry as a triangle mesh. The output of the algorithm is per-pixel out-going illumination, namely E_R^{dir}, E_R^{ind}, E_{RV}^{dir}, and E_{RV}^{ind}, as explained in Sec. 3.

Previous works achieve this by tracing two kinds of rays: one that intersects only real geometry, and one that intersects both real and virtual geometry. Path-tracing using these two ray types is described in Kán and Kaufmann [47], but our method is slightly different. For the virtual geometry we use meshes, since most virtual models are mesh-based, and the NVIDIA OptiX [75] ray-tracer is very suitable to intersect mesh geometry. However, we do not assume a known mesh model of the real world, and using a mesh-based ray-tracer for real geometry (per-frame vertex map) is wasteful. A screen-space ray-tracer (SSRT) computes the same result much faster, by traversing the vertex map starting from the origin pixel, and returns the world position of the intersection. We use a hierarchical traversal method adapted from [98]. Since only the current frame is used, off-screen real objects will not affect the shading; the effects of this limitation are quite subtle for diffuse scenes. This can be addressed by using SSRT on a higher field-of-view vertex map rendered in the SLAM stage, at the cost of performance.

6.5.1 Algorithm

We propose a two-mode path-tracer that traces OptiX rays to intersect only virtual geometry, and screen-space rays to intersect only real geometry. Our sampling algorithm is explained in detail in Appendix C.

First, we compute 4 primary samples per pixel (spp) for anti-aliasing, and we determine whether a real or a virtual object is visible at the current sample and update the mask M (see Sec. 3). Next, we compute 4 secondary samples for each of the 4 primary samples, so we compute a total of 16 spp for each of direct and indirect illumination. For direct illumination, we importance sample the environment map (see Sec. 5.2), as this gives the least amount of noise for very little overhead. For indirect illumination, we sample the cosine hemisphere for diffuse surfaces (real and virtual) and a Phong lobe for glossy surfaces (virtual only).

In the sampling step, we also save the (average) world location, normal, virtual texture k_V. We also record the minimum hit distance for direct and indirect illumination; these are required

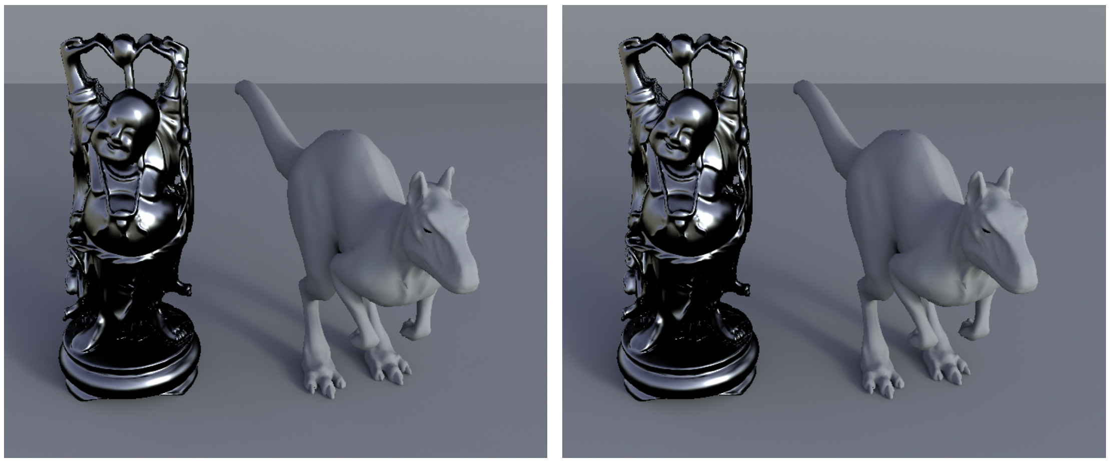

Our Pre-computed Importance Sampling, 1024 spp
BRDF Importance Sampling, 4096 spp (Ground Truth)

Figure 6.3: Our pre-computed sampling strategy introduces no visible difference.

for filtering. Since texture is multiplied with irradiance after filtering, we approximate $\langle k \cdot E \rangle \approx \langle k \rangle \cdot \langle E \rangle$, where $\langle \rangle$ denotes the mean of the quantity at a pixel. This approximation works on most pixels except on silhouettes, where the error is usually small.

6.5.2 Importance sampling the environment map

Traditional importance sampling[1] involves pre-computing a cumulative distribution function (CDF) of the 2D environment map and then generating samples at each pixel by finding the inverse CDF for a uniform stratified random sample. However, performing 16 such look-ups per-frame per-pixel is slow, so instead we pre-compute a large number of importance samples (4096) and store them to a buffer. Then, at each pixel we perform 16 random stratified look-ups into the buffer. This is somewhat analogous to a virtual-point-light approach, except we use a very large number of lights and sample randomly.

In Fig.6.3, we compare a 1024 spp image rendered with random sampling of our pre-computed importance samples, to a ground truth image with 4096 spp BRDF importance sampling. There is no visual difference, and the numeric (RMS, per channel) error is 10^{-5}.

6.6 Fourier Analysis for Environment Lighting

So far, we have computed the quantities E_R^{dir}, etc. as per the flowchart in Fig. 2, from our sampling phase above (stage 2). However, since we only use 16 samples per pixel, the results are noisy, and accurate rendering requires filtering (stage 3). We now discuss our filtering theory, that computes the required bandwidths; Sec. 7 discusses the actual filtering given these bandwidths.

[1]Complex importance sampling strategies such as [1] are more effective at importance sampling and could also be used.

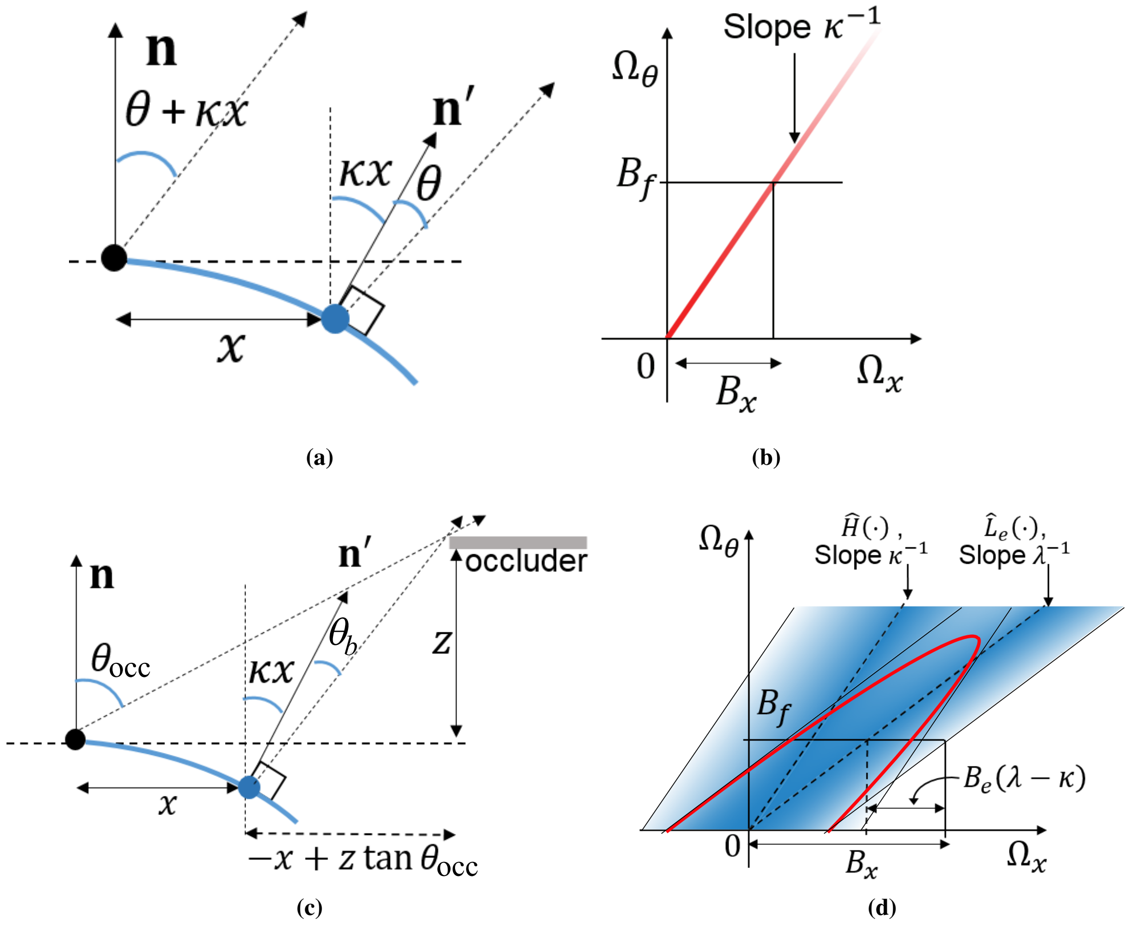
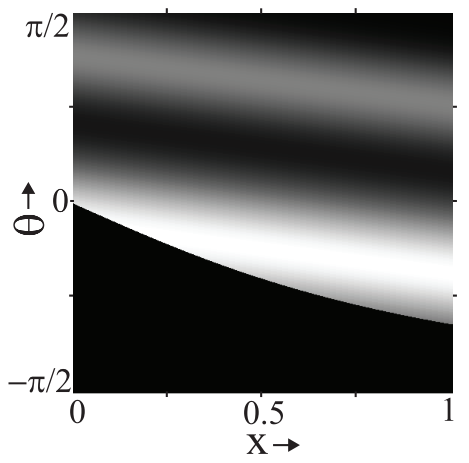


Figure 6.4: (a) Geometry for diffuse flatland case without considering visibility, (b) Power spectrum of L_i and axis-aligned filter. In (c) we show the flatland geometry with an occluder at depth z and (d) shows the power spectrum $\hat{G}$, defined in eqn. 6.11, which is the product of the two sheared Gaussians (shaded blue), and has an ellipsoidal shape (shaded red); B_x is our conservative estimate of its bandwidth.

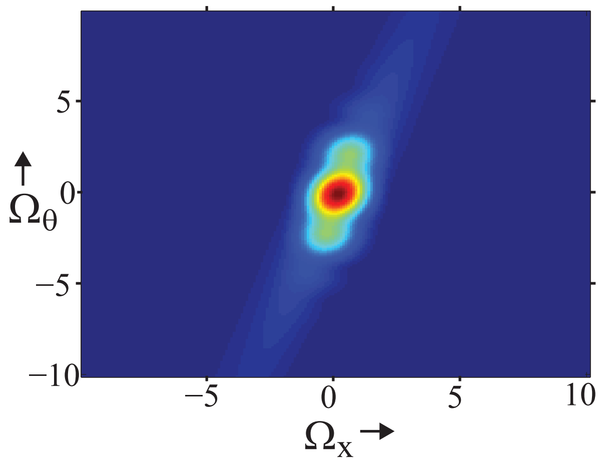
Our main contribution is the Fourier analysis of direct illumination from an environment map, considering occluders and visibility. We first perform a 2D Fourier analysis of the shading in a position-angle space, and then show that the shading is bandlimited by the BRDF in the angular dimension. The resulting axis-aligned filter provides a simple spatial bandwidth for the shading. In practice, the pre-integrated noisy per-pixel irradiance can be filtered using a Gaussian kernel of variance inversely related to the bandwidth, without altering the underlying signal.

6.6.1 Diffuse without visibility

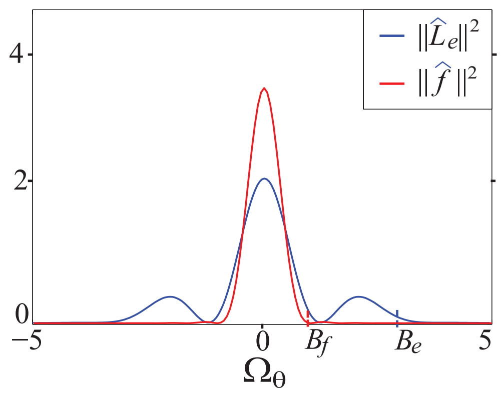
As in previous works, we perform our analysis in flatland (2D). As explained in Sec. 6.4, the 2D results provide a bound on the 3D result, even though there is no simple analytic formula for 3D. We begin with the simplest case. Parameterize a diffuse receiver surface of curvature κ by x along



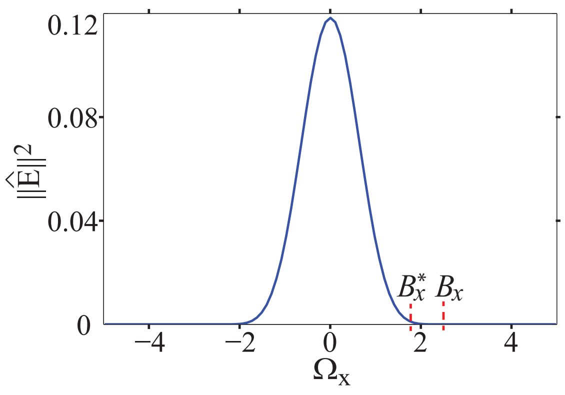
(a) $L_i \times V$



(b) $||\hat{G}||^2$



(c) $||\hat{L}_e||^2$, $||\hat{f}||^2$



(d) $||\hat{E}||^2$

Figure 6.5: Verification of eqn. 6.13 for the simple flatland setup of Fig. 3(c) with $\kappa = 0.5$ and one occluder at $\theta_{\text{occ}} = 0$ and $z = 2$, using a Gaussian-sinusoid product for illumination. (a) shows the product $L_i \times V(x, \theta)$ for this setup. (b) shows the power spectrum $||\hat{G}||^2$ of (a). In (c) we show the 1D power spectra of L_e and f, showing bandlimits $B_e = 3$ and $B_f = 1$. (d) shows the 1D power spectrum $\hat{E}$ of the surface irradiance. Eqn. 6.13 gives $B_x = 2.5$, while numerically the bandwidth of $\hat{E}$ is $B_x^* = 1.9$, thus showing that our estimate is conservative and reasonably tight. The FLT estimate in this case is $B_x = 1.0$, causing significant energy loss.

a tangent plane defined at the origin $x = 0$. Note that κ is defined as the change in the normal angle (relative to the origin normal) per unit change in x; it is assumed positive but the analysis extends easily for negative curvatures. Consider the set-up shown in Fig. 6.4(a). Let the environment illumination be $L_e(\cdot)$, with angles to the right of the normal being positive and to the left being negative. The illumination is assumed to have an angular bandwidth of B_e (i.e., 99% energy of $||\hat{L}_e(\Omega_\theta)||^2$ lies in $|\Omega_\theta| < B_e$). We now analyze the 1D surface irradiance given by the reflection equation in flatland:

$$E(x) = \int_{-\pi/2}^{\pi/2} L_i(x,\theta)\cos\theta\, d\theta = \int L_i(x,\theta) f(\theta)\, d\theta. \tag{6.3}$$

We have re-written the equation with a clamped cosine function $f(\theta) = \cos\theta$ for $\theta \in [-\pi/2, \pi/2]$ and 0 otherwise. The Fourier transform of eqn. 6.3 is straightforward,

$$\hat{E}(\Omega_x) = \int \hat{L}_i(\Omega_x, \Omega_\theta)\hat{f}(\Omega_\theta) d\Omega_\theta \tag{6.4}$$

The incoming direction θ at x corresponds to the direction $\theta + \kappa x$ at the origin. Then,

$$L_i(x,\theta) = L_e(\theta + \kappa x). \tag{6.5}$$

The 2D Fourier power spectrum of L_i is a single line through the origin , with slope κ^{-1} (see [12]). This line is bandlimited in the angular dimension by $\hat{f}$. Let this bandlimit be B_f. Then, as shown in Fig. 6.4(b), the bandwidth of $\hat{E}$ is $B_x = \kappa \cdot \min\{B_e, B_f\}$. In most interesting cases, we have $B_e > B_f$, so that

$$B_x = \kappa B_f. \tag{6.6}$$

This simple result (also derived in [23]) shows that higher curvature produces higher shading frequency. We now build upon this simple analysis to extend the result to include visibility and glossy BRDFs.

6.6.2 Diffuse with visibility

We now refine the above result by including an infinite occluder at depth z, as shown in Fig. 6.4(c). The occluder blocking angle θ_b for a point at a small $x > 0$ on the curved surface can be written in terms of the angle at the origin $\theta_{\text{occ}} = \theta_b(0)$:

$$\begin{aligned} \theta_b(x) &\approx \tan^{-1}\left(\frac{z\tan\theta_{\text{occ}} - x}{z + \kappa x^2}\right) - \kappa x \\ &\approx \tan^{-1}\left(\tan\theta_{\text{occ}} - \frac{x}{z}\right) - \kappa x \\ &\approx \theta_{\text{occ}} - x\frac{\cos^2\theta_{\text{occ}}}{z} - \kappa x \end{aligned} \tag{6.7}$$

In the second step, we ignore the offset κx^2 of the point below the plane of parametrization, since it is quadratic in x. In the last step we use Taylor approximation to expand the arctan: $\tan^{-1}(\alpha + x) \approx \tan^{-1}\alpha + x/(1+\alpha^2)$ for $x << 1$. This is similar to [82]. Thus, for small curvature and small displacement x, we get $\theta_b(x) = \theta_{\text{occ}} - \lambda x$ where

$$\lambda = \kappa + \cos^2\theta_{\text{occ}}/z \tag{6.8}$$

Finally, note that the visibility at x is

$$V(x,\theta) = H(\theta - \theta_b(x)), \tag{6.9}$$

where $H(\cdot)$ is a step function that takes value 0 when its argument is positive and 1 otherwise. Hence, the irradiance including visibility can be written as:

$$\begin{aligned} E(x) &= \int L_i(x,\theta)V(x,\theta)\,f(\theta)\,d\theta \\ &= \int L_e(\theta+\kappa x)H(\theta-\theta_{\text{occ}}+\lambda x)\,f(\theta)\,d\theta. \end{aligned} \tag{6.10}$$

To find the spatial bandwidth of $E(x)$, we find the Fourier transform (full derivation is provided in Appendix D):

$$\begin{aligned} \hat{E}(\Omega_x) &= \frac{1}{\lambda-\kappa}\int \hat{L}_e\left(-\frac{\Omega_x-\lambda\Omega_\theta}{\lambda-\kappa}\right)\hat{H}\left(\frac{\Omega_x-\kappa\Omega_\theta}{\lambda-\kappa}\right) \\ &\qquad\qquad e^{j(\dots)}\hat{f}(\Omega_\theta)d\Omega_\theta \\ &= \int \hat{G}(\Omega_x,\Omega_\theta)\hat{f}(\Omega_\theta)d\Omega_\theta \end{aligned} \tag{6.11}$$

The phase term $e^{j(\dots)}$ due to the θ_{occ} offset in H is ignored for brevity; we are only concerned with the magnitude of the integrand. Both terms $\hat{L}_e$ and $\hat{H}$ are 1-D functions sheared in 2-D along lines of slopes λ^{-1} and κ^{-1}, respectively. Since the respective 1-D functions are both low-pass (i.e. 99% energy lies in a small frequency range), the product $\hat{G}$ is shaped roughly like an ellipsoid. This is shown in Fig. 6.4(d). **The shape of the spectrum is no longer a simple line for a single depth occluder.**

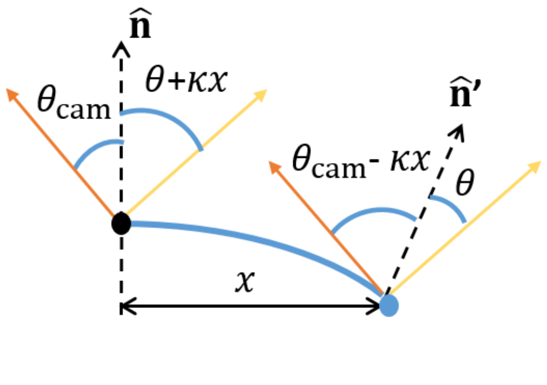
From eqn. 6.11, G is bandlimited in Ω_θ by the bandwidth of f, i.e. B_f. Since $\hat{L}_e$ has the smaller slope is λ^{-1}, the worst case spatial bandwidth of $\hat{E}$ is that of this term. Part of the bandwidth is from the center line, specifically $B_f\lambda$. The bandwidth has an additional component due to the non-zero spread of the $\hat{L}_e$ term in eqn. 6.11. Since the one-sided width of $\hat{L}_e(\Omega)$ is B_e, the width of this term is $B_e(\lambda-\kappa)$. Thus, our conservative estimate of the spatial bandwidth of $\hat{E}$ is

$$B_x = B_f\lambda + B_e(\lambda-\kappa) \tag{6.12}$$

This bandwidth is the sum of what one would get considering only visibility (first term), and the extra bandwidth due to the lighting (second term). However, it is not simply the sum of bandwidths due to illumination and visibility when considered separately, as one may have expected from the form of eqn. 6.10. Using the definition of λ (eqn. 6.8), we can re-write the filter width as:

$$\boxed{B_x = \kappa B_f + (\cos^2\theta_{\text{occ}}/z)(B_f+B_e)} \tag{6.13}$$

We verified that our predicted bandwidth holds for many flatland cases. One such set up is shown in Fig. 6.5. Observe the shape of the spectrum $\hat{G}$. The predicted bandwidth slightly overestimates the true bandwidth.



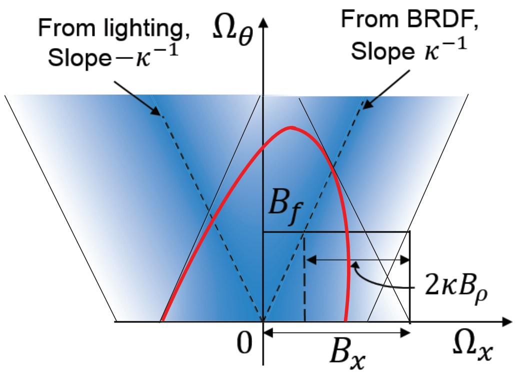

From lighting, Slope $-\kappa^{-1}$
Ω_θ
From BRDF, Slope κ^{-1}
B_f
$2\kappa B_\rho$
0
B_x
Ω_x
(b)

Figure 6.6: (a) Flatland geometry for shading with glossy BRDF. (b) The shading spectrum is the product of the two sheared Gaussians (shaded blue), and has an ellipsoidal shape (shaded red); B_x is our conservative estimate of its bandwidth.

The case with multiple occluders at different depths cannot be solved analytically; we find the most conservative bound by considering the closest occluder (smallest z). However, in this case, the spectrum is not a well-defined double-wedge as in previous works.

6.6.3 Glossy BRDF

We will now derive the bandwidth for shading on a curved glossy surface in flatland. Visibility is not considered, since it introduces an intractable triple product, but we intuitively add the effect of visibility at the end. As before, the surface is lit with a 1D environment illumination L_e relative to the origin, and the camera is at an angle θ_{cam} from the origin. The camera is assumed distant compared to the scale of surface being considered. The setup is shown in Fig. 6.6(a). The surface BRDF is assumed rotationally invariant, that is, only the difference of the incoming and outgoing angles determines the BRDF value. [2] Numerically, $\rho(\theta_i, \theta_o) = \rho(\theta_i + \theta_o)$, where the $+$ sign is due to angles being measured relative to the normal. Then, the radiance reflected to the camera is

$$\begin{aligned} L_0(x, \theta_{\text{cam}}) &= \int_{-\pi/2}^{\pi/2} L_i(x, \theta)\rho(\theta, \theta_o)\cos\theta \, d\theta \\ &= \int L_e(\theta + \kappa x)\rho(\theta_{\text{cam}} + \theta - \kappa x) f(\theta) \, d\theta \end{aligned} \tag{6.14}$$

This equation is similar to eqn. 6.10, except that the slopes of the two terms have opposite signs. The Fourier transform has the same form, with minor differences:

$$\begin{aligned} \hat{L}_o(\Omega_x, \theta_{\text{cam}}) = \frac{1}{2\kappa} \int \hat{L}_e\left(\frac{\Omega_x + \kappa\Omega_\theta}{2\kappa}\right) \hat{\rho}\left(\frac{\Omega_x - \kappa\Omega_\theta}{2\kappa}\right) e^{j(\ldots)} \\ \hat{f}(\Omega_\theta) d\Omega_\theta \end{aligned} \tag{6.15}$$

[2]This holds for the commonly used Phong glossy model, but similar to Durand et al. [23] our analysis can be extended for other models.

As before, the phase term $e^{j(\cdots)}$ is ignored for brevity. The terms $\hat{L}_e$ and $\hat{\rho}$ are low-pass, and sheared along lines of slopes $-\kappa^{-1}$ and κ^{-1}, respectively. Their product is visualized in Fig. 6.6(b). Thus, the conservative bandwidth estimate for $\hat{L}_o$ is

$$\begin{aligned} B_x &= \kappa B_f + 2\kappa \min\left\{B_e, B_\rho\right\} \\ &= \kappa\left(B_f + 2B_\rho\right) \end{aligned} \tag{6.16}$$

Here B_ρ is the angular bandwidth of the BRDF $\rho(\cdot)$. Comparing to eqn. 6.13, we see that the angular bandlimit has effectively increased from B_f to $B_f + 2B_\rho$. Thus, we can modify eqn. 6.13 to include the effect of a visibility discontinuity and rewrite the generalized bandwidth as

$$\boxed{B_x = \kappa\left(B_f + 2B_\rho\right) + (\cos^2\theta_{\text{occ}}/z)\left(B_f + 2B_\rho + B_e\right)} \tag{6.17}$$

We provide a numerican evaluation of this bandwidth estimate in Appendix E, similar to Fig. 6.5.

6.6.4 Extension to 3D

The flatland results above can be extended to 3D. Directions in 3D can be parameterized in spherical coordinates (θ, ϕ); however, there is no simple linear form in terms of curvature in 3D which makes the analysis tedious. However, we can restrict to a fixed ϕ – along the direction of maximum curvature – and perform our analysis. The resulting bandwidth is a conservative bound for the true bandwidth considering the full hemisphere of directions, since the normal angle θ changes most rapidly along the maximum curvature direction. In practice, computing the maximum curvature per pixel is difficult, and we instead determine screen-space curvatures κ_X, κ_Y, which bound the bandwidth along the image X, Y axes. The filter size is the reciprocal of the bandwidth.

In Fig. 6.7, using a purely virtual and untextured scene under an outdoor environment map (direct illumination only), we show that our flatland analysis works well in 3D using screen space curvatures. In (b), we show the mean of X and Y filter size. Note how the filter size depends on curvature, BRDF and occluder distance.

6.6.5 Indirect Illumination

The above analysis derives filter bandwidths for the direct illumination terms $E_R^{\text{dir}}, E_{RV}^{\text{dir}}$. We must also filter the sparsely-sampled noisy indirect illumination terms, $E_R^{\text{ind}}, E_{RV}^{\text{ind}}$. For indirect illumination, we use the axis-aligned filter derived in chapter 4 (also [64]). For any configuration of reflectors at a minimum distance $z_{\min}$ from the receiver, with BRDF bandlimit B_h, the bandwidth formula is:

$$\boxed{B_x^{\text{ind}} = B_h / z_{\min}} \tag{6.18}$$

For the diffuse case, $B_h \approx 2.8$. For a Phong BRDF with exponent m, $B_h \approx 4.27 + 0.15m$ (see chapter 4).

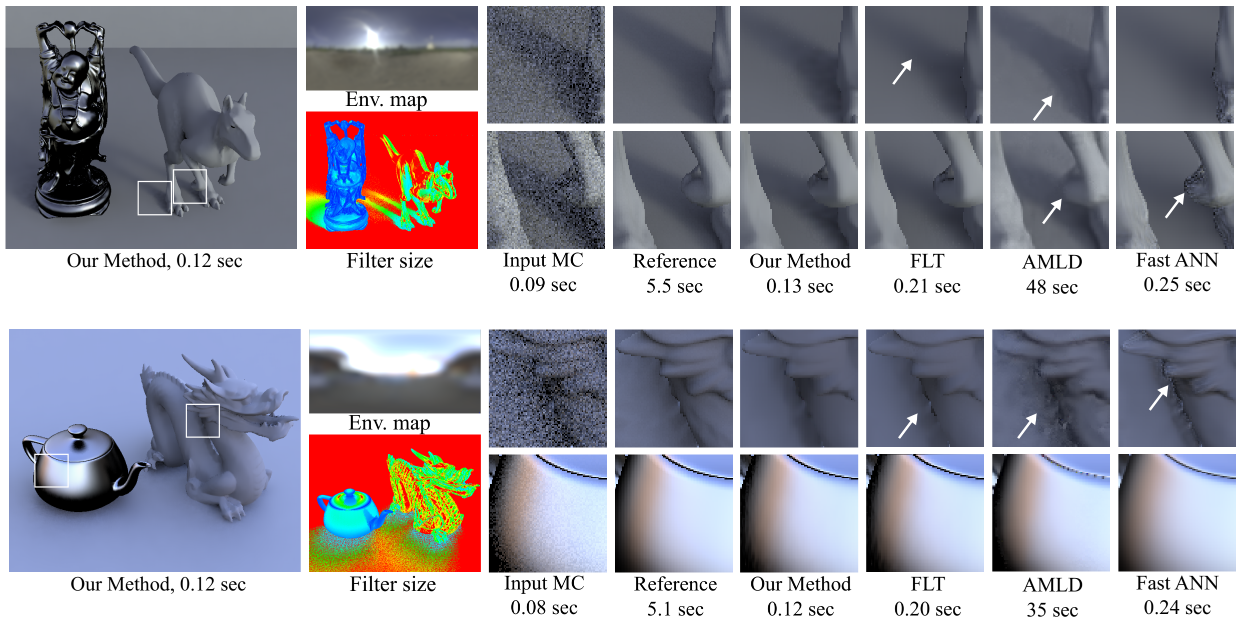


Figure 6.7: We demonstrate our filtering method on two purely virtual untextured scenes, with a diffuse and a phong (exponent 64) object on a diffuse plane. We show our result with an input of 16 spp (using temporal filtering, Sec. 7), that runs at about 8 fps (resolution 720×720). We also show the environment map used, and warm-to-cool filter size (inverse bandwidth). In the insets, we compare to unfiltered 16 spp MC input, reference with 1024 spp (60× slower). We also compare to three other methods with 32 spp input: FLT[23] which blurs shadows, AMLD[45] which is offline and blurs geometric edges, and Fast ANN[99], which is only 2× slower than ours, but produces artifacts. All methods work well on glossy objects (bottom row insets) since the noise is small.

6.6.6 Discussion

We discuss the novelty of and compare our contributions against previous works. We first emphasize that chapter 3 which treats area lights at finite distances and chapter 4 which treats indirect illumination from nearby reflectors, are both not applicable to environment lighting. An important difference from FLT [23] is the consideration of the source illumination bandwidth. They combine the effect of visibility and BRDF without doing a full derivation; their eqn. 21 gives the following bandwidth in object space (ignoring the scaling of $d/(\mathbf{n}\cdot\mathbf{v})$):

$$B_x^{FLT} = (2\kappa + z^{-1})B_\rho, \tag{6.19}$$

Comparing to our bandwidth, eqn. 6.17, the FLT approach ignores the B_e term, which arises from our exact evaluation of the convolution of illumination and visibility, and the ellipsoidal shape of the spectrum. Thus, FLT underestimates the shading bandwidth of high frequency lighting with nearby occluders, resulting in blurring high frequency shading effects when used for filtering.

6.7 Practical Filtering

We now discuss how the bandwidths derived above can be used to filter noisy irradiances. Before the filtering stage, we run a CUDA kernel to compute per-pixel screen-space curvatures κ_X, κ_Y as

$$\kappa_X(x,y) = \frac{\text{angle}\,(\mathbf{n}(x,y), \mathbf{n}(x+1,y))}{||\mathbf{p}(x,y) - \mathbf{p}(x+1,y)||} \tag{6.20}$$

where $\mathbf{n}$ is the normal and $\mathbf{p}$ is the world position[3]. The sampling pass stores the minimum occluder distances for direct (we ignore occluders which block samples with intensity below a threshold of 0.1) and indirect illumination. Then, filter bandwidths along each axis, B_X, B_Y are computed – using eqn. 6.17 for direct illumination and 6.18 for indirect illumination ($B_X = B_Y$ for indirect). B_e is taken to be the 99% energy bandwidth of the 2D Fourier spectrum of the environment map in lat-long coordinates.

A naive implementation of a 2D Gaussian filter of radius R has $\Theta(R^2)$ complexity. Analogous to previous axis-aligned filtering papers, the filter is separable into two stages, aligned along the image X and Y axes, reducing the complexity to $\Theta(R)$. We now provide the formulae for the 2-step separated filter. Let $E(x,y)$ denote a raw noisy irradiance value at the pixel (x,y), and $\bar{E}$ denote the filtered value. Then,

$$E_X(x,y) = \frac{\sum_{|i|<R} w_{xy}(x+i,y)E(x+i,y)}{\sum_{|i|<R} w_{xy}(x+i,y)} \tag{6.21}$$

E_X denotes the intermediate value resulting from filtering only in the X-direction. The filtered result is given as:

$$\bar{E}(x,y) = \frac{\sum_{|j|<R} w_{xy}(x,y+j)E_X(x,y+j)}{\sum_{|j|<R} w_{xy}(x,y+j)}. \tag{6.22}$$

The filter kernel is a Gaussian:

$$w_{xy}(x+i,y) = \exp(-2\bar{B}_X^2 ||\mathbf{p}(x,y) - \mathbf{p}(x+i,y)||^2) \tag{6.23}$$

Since the bandwidth estimates are also noisy, we use the average square bandwidth of the source and target pixel $\bar{B}_X^2 = 0.5(B_X^2(x,y) + B_X^2(x+i,y))$. Similarly, $w_{xy}(x,y+j)$ uses the bandwidth B_Y.

Temporal filtering: Since we only use 16 samples per pixel, the result from the sampling stage is very noisy, and the filtering can still leave some residual noise in temporal sequences, leading to distracting flickering. Hence, we do temporal filtering where the filter also extends to the previous frame. This scheme is physically accurate, assuming the illumination does not change (at a given world location) between two consecutive frames – which is a good assumption in most situations except rapid geometry or light source motion. Let E' be the irradiance value from the previous

[3]The curvature sign is the sign of the dot-product between the vector joining world coordinates and $\mathbf{n}(x,y)$

frame, and (x',y') be the pixel in the previous frame with the closest world location to pixel (x,y), i.e., $\mathbf{p}(x,y) \approx \mathbf{p}'(x',y')$. First, filter along X:

$$E'_X(x,y) = \frac{\sum_{|i|<R} w'_{xy}(x'+i,y')E'(x'+i,y')}{\sum_{|i|<R} w'_{xy}(x'+i,y')} \tag{6.24}$$

The weights are modified to

$$w'_{xy}(x'+i,y') = \exp(-2\bar{B}_X^2||\mathbf{p}(x,y) - \mathbf{p}'(x'+i,y')||^2) \tag{6.25}$$

Note that the center of the kernel is offset to (x',y'), unlike eqn. 6.23. To see why this is important, imagine that there is camera motion between the two frames only along Y. Then, if the filter were to be centered at (x,y), there may be no pixel $x+i$ where $||\mathbf{p}(x,y) - \mathbf{p}'(x+i,y)||$ is small resulting in few or no useful values of E' and artifacts in the final result.

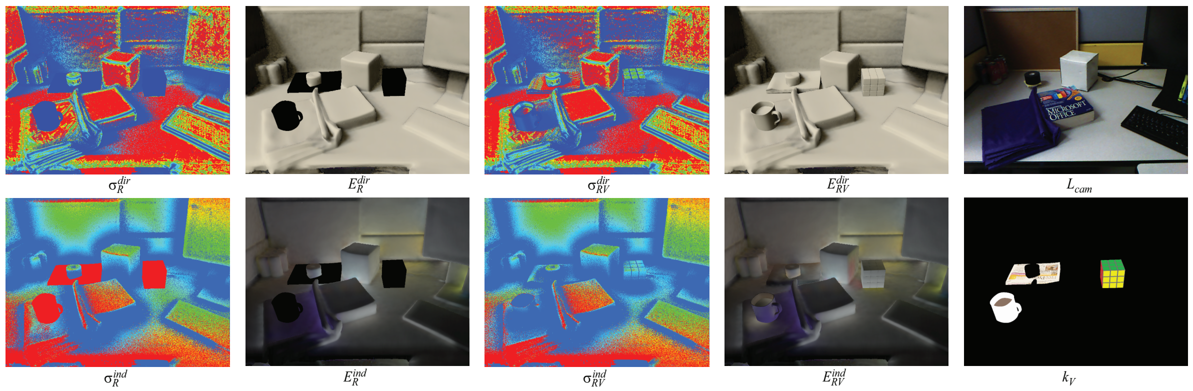
We can now combine the results of eqns. 6.21 and 6.24, and filter along the Y-axis to produce the final filtered value:

$$\bar{E}(x,y) = \frac{\left(\begin{array}{r}\sum\limits_{|j|<R} w_{xy}(x,y+j)E_X(x,y+j)+ \\ w'_{xy}(x',y'+j)E'_X(x',y'+j)\end{array}\right)}{\sum_{|j|<R} w_{xy}(x,y+j) + w'_{xy}(x',y'+j)}. \tag{6.26}$$

6.8 Results

We show four mixed reality scenes with environment map direct and indirect illumination, all rendered at the Kinect camera's VGA (640×480) resolution. Our results include a variety of real-life scenarios augmented with diffuse as well as glossy virtual objects that blend in seamlessly. The accompanying video shows animations and screen captures demonstrating temporal stability. Our images and video are rendered on an Intel Core i7, 3.60GHz desktop with a single NVIDIA Titan GPU, using CUDA v6.5 and OptiX v3.5.

In Fig.6.1, we show a simple DESK scene about (1 meter)^3 in size. A diffuse Rubik's cube, coffee mug and newspaper are inserted into the real image/video, and they blend in plausibly. The insets show the 4 primary modes of interaction. Direct illumination shadows cast from virtual to real objects are shown in the left-most column 1, and from real to virtual objects are shown in column 2. Similarly indirect illumination color bleeding from virtual to real objects is captured in column 3, and virtual to real color bleeding is shown in column 4. In addition, corresponding insets from the unfiltered, and converged reference images are also shown for comparison. In Fig. 6.8, we show intermediate steps of our system for the DESK scene, including each of the four irradiances E_R^{dir}, E_R^{ind}, E_{RV}^{dir}, E_{RV}^{ind}, and their correspoding filter sizes, as obtained from our theory.



(a) Real-only filters **(b)** E_R **(c)** Real+Virtual Filters **(d)** E_{RV} **(e)** L_{cam}, k_V

Figure 6.8: We show intermediate quantities in our filtering algorithm for the DESK scene of Fig. 1. Filter sizes are shown as color-coded standard deviations of the world-space Gaussians, $\sigma = 1/B_x$, using the appropriate bandwidths B_x. The filter sizes in (a) are used to filter the irradiance computed considering only real objects, i.e. $E_R^{\text{dir}}, E_R^{\text{ind}}$ using only real geometry; and the filtered results are shown in (b). Similarly, irradiances considering the full real-virtual geometry, $E_{RV}^{\text{dir}}, E_{RV}^{\text{ind}}$ are filtered using the filters in (c); the results are shown in (d). Observe how addition of virtual objects affects nearby filter sizes due to the introduced occlusions. The final result, computed using eqn. 6.2, requires two additional quantities – input radiance image L_{cam}, and the virtual object texture k_V – shown in (e).

Figure 6.9, FURNITURE, shows a larger scene of about (2 meter)^3 size consisting of furniture and a plastic mannequin. A diffuse sofa cushion and a wooden table, and a glossy (phong exponent 64, $B_\rho = 10$) trashcan are inserted. The insets show the 3 key regions of real-virtual interaction. Corresponding insets from the unfiltered image, and converged reference image are also shown for comparison. All of our scenes are captured in the same large room with a number of area lights (with a different set of lights used for each scene); the environment map for this scene is shown in (a). We used $B_e = 4$.

Figure 6.10 shows a kid's PLAYROOM scene, of about (2 meter)^3 size. We insert a diffuse doll, a toy robot and a stool (on the right) which matches the real stool (on the left). The insets show the parts of each virtual object. Corresponding insets from the unfiltered image, and converged reference image are also shown for comparison.

Figure 6.11, shows a multi-compartment SHELF scene about 2 meters wide, with some real or virtual objects in each compartment. We insert a diffuse gift box, a diffuse book and a glossy (phong exponent 64) metal bowl. The metal bowl reflects both the environment map and the local geometry, and is not over-blurred. Corresponding insets from the unfiltered image, and converged reference image are also shown for comparison.

Timings: We provide timings for each stage of our system in Table 6.1. Our overall speed is 5.7 fps for the fastest and 4.9 fps for the slowest scenes. For the SLAM step, compared to [79], we use a higher number of iterations for the ICP step for greater pose accuracy. We use a fixed

Scene	SLAM	Optix	SSRT	Filter	Total	FPS
DESK	22	85	44	24	175	**5.7**
FURNITURE	26	98	44	23	191	**5.2**
PLAYROOM	26	105	40	23	194	**5.2**
SHELF	27	110	41	25	203	**4.9**

Table 6.1: Detailed timings of our scenes (in milliseconds) rendered at 640×480*. Our filtering overhead is small compared to the rendering time. We achieve interactive frame rates on a variety of complex scenes.*

voxel size of (3 cm)^3; the timings differ by scene due to different voxel grid sizes. The sampling stage is split into an Optix pass and a CUDA screen-space pass, as explained in Appendix C; each sub-stage takes roughly equal time, and the entire stage accounts for three-quarters of the computational cost. The filtering stage, using a 30×30 neighborhood around each pixel, runs under 25 msec (including temporal filtering), and is under 15% of the total cost. To reduce read/write overheads, we store colors with 8 bits per channel and other quantities as half-precision (16 bits) floats. The compositing stage takes negligible time, and is hence not reported.

6.8.1 Comparisons

Figure 6.7 compares insets of our result with the result of filtering with the bandwidth from FLT[23] (see Sec 6.6). Their bandwidth ignores the illumination bandlimit B_e resulting in over-blurred shadows.

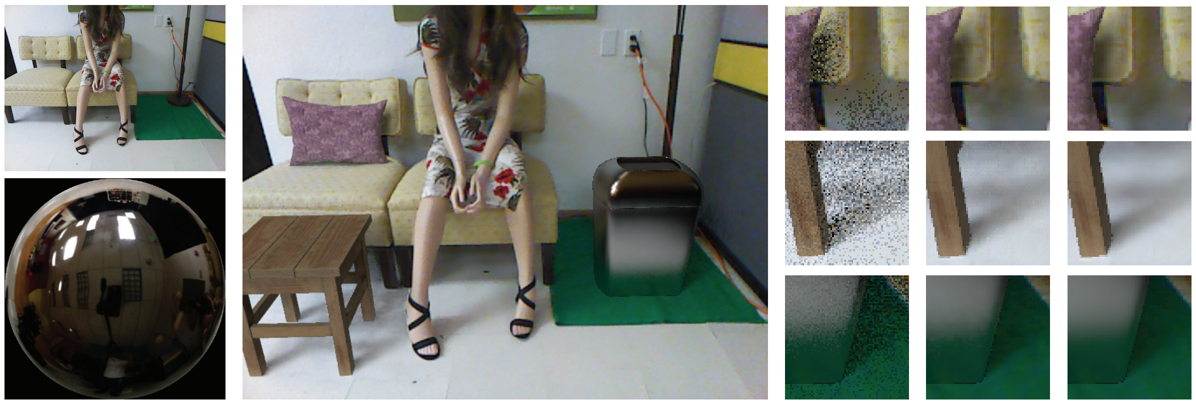
We also compare to a state-of-the-art Monte Carlo adaptive sampling and multi-level denoising algorithm, AMLD [45], and Fast-ANN image denoising [99], each with 32 spp input. AMLD is offline, with 10 sec filtering overhead, and preserves harder shadow edges but blurs geometric edges slightly since the method is not explicitly aware of geometry. Our method also slightly blurs shadow edges due to our bandlimited light assumption, but runs at real-time (40 ms overhead). Fast-ANN is real-time with 80 ms filtering overhead ($2\times$ of ours), but produces artifacts on smooth untextured images since it relies on finding similar patches using normals.

We do not compare against real-time but approximate MR rendering methods (e.g., Knecht et al.[49]), since their goal is not to produce physically-accurate renderings. They also use different input (marker-based tracking or pre-defined real meshes) which makes it difficult to produce identical images. Delta voxel cone-tracing [30] uses only point light sources. In general, these methods produce biased images (not physically accurate), or retain noise [47].

6.8.2 Limitations and Future Work

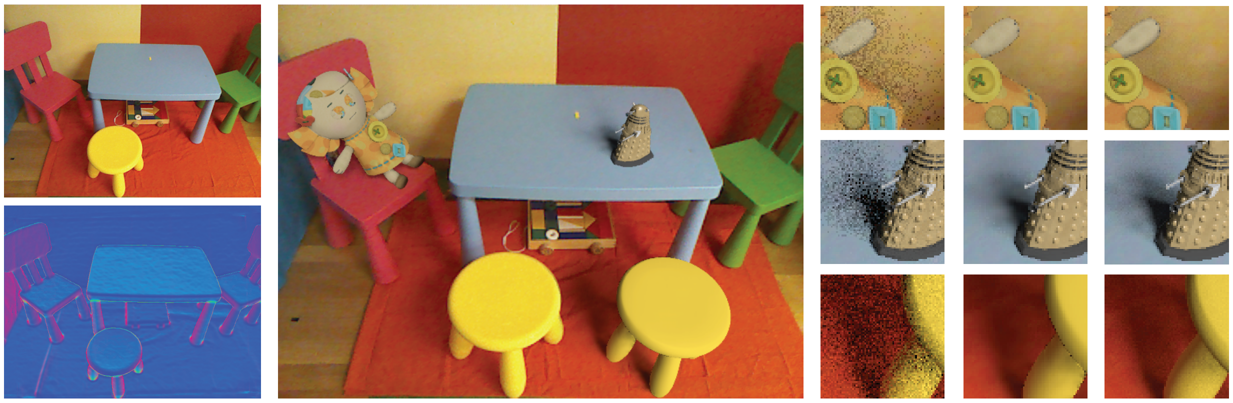
We describe some limitations of our work that are excellent avenues for future work. Our filtering-based approach assumes that the environment illumination is band-limited, and hence cannot handle high frequency components such as small bright lights. Simply using a large B_e will result in small filter size leaving visible residual noise, while using a large filter size would result in over-blurring. As in Nowrouzezahrai et al. [71], this can be treated by separating out high frequency illumination into a small set of point lights, then using our approach for the low frequency component.

For our MR system, since we use SLAM with a coarse noisy input depth, the reconstructed geometry is often not perfectly aligned with the RGB image, which causes artifacts. Although our theory supports dynamic real objects, we do not demonstrate it since our SLAM backend cannot handle moving geometry robustly. All these issues can be mitigated using a more robust SLAM and higher accuracy input depth. We assume that all real surfaces are diffuse, since estimating the true BRDF even with a simple model is difficult at interactive speed. This can be addressed in future work. Further, we do not handle caustics, since our method is based on ray-tracing, and the filtering theory does not treat specular to diffuse light transport.



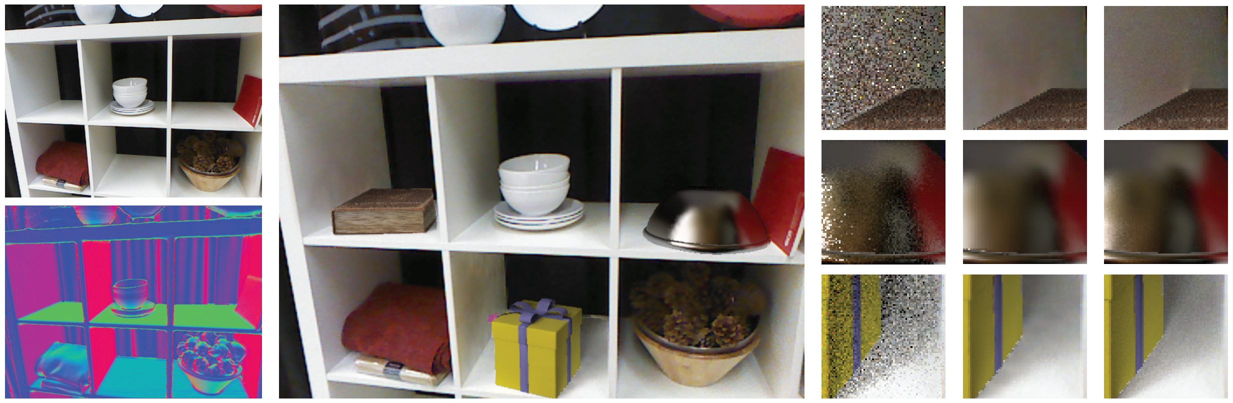
(a) Input RGB, env. map **(b)** Our Method, 5.2 fps **(c)** Input MC **(d)** Our **(e)** Reference

Figure 6.9: For this FURNITURE scene, the input image is shown in (a) (top), along with the captured environment illumination (bottom). The augmented image with a virtual cushion, wooden table and a glossy trashcan with physically correct illumination, is shown in (b). We compare our result (16 spp, 0.19 sec) with unfiltered Monte Carlo (16 spp), which is very noisy, as well as reference (1024 spp, 12 sec) which is 60× slower.



(a) Input RGB and normals **(b)** Our Method, 5.2 fps **(c)** Input MC **(d)** Our **(e)** Reference

Figure 6.10: The PLAYROOM scene with input image and 3D reconstruction are shown in (a). We insert a virtual doll, toy robot, and a stool (the one on the right), with physically correct illumination, as shown in (b). We compare our result (16 spp, 0.19 sec) with unfiltered Monte Carlo (16 spp), which is very noisy, as well as reference (1024 spp, 12 sec) which is 60× slower.



(a) Input RGB and normals **(b)** Our Method, 4.9 fps **(c)** Input MC **(d)** Our **(e)** Reference

Figure 6.11: The SHELF scene input image and 3D reconstruction are shown in (a). We insert a diffuse gift-box and a book, and a glossy bowl, as shown in (b). We compare our result (16 spp, 0.20 sec) with unfiltered Monte Carlo (16 spp), which is very noisy, as well as reference (1024 spp, 13 sec) which is 60× slower.

Chapter 7

Conclusion

Computer graphics rendering is undergoing a renaissance, with physically-based rendering methods based on accurate Monte Carlo (MC) image synthesis replacing ad-hoc techniques in a variety of applications including movie production. In interactive applications like product visualization or video games, physically-based lighting effects are increasingly popular. Photo-realistic rendering is classically achieved through Monte-Carlo path-tracing, which requires tracing many thousands of rays at each pixel in a high-resolution image. Ray-tracing is computationally expensive and therefore Monte-Carlo rendering takes minutes of time per image. In the last decade or so, computer graphics researchers have developed techniques that dramatically reduce the number of rays needed in Monte-Carlo rendering by sharing information between neighboring pixels. However, many of these methods based on sheared filtering, statistical denoising or light field reconstruction require an expensive post-processing step, and end up taking tens of seconds per image.

In this thesis, we proposed to use simple axis-aligned filters to reduce the number of Monte Carlo samples considerably compared to brute force, but less than some previous methods. However, because our filtering step is very simple, essentially a spatially-varying image-space blur, and it can be performed extremely fast if implemented appropriately. We utilize the powerful parallel computing capabilities of GPUs by implementing our filter as a per-pixel parallel kernel, and also by using a GPU-accelerated ray-tracer. In our algorithm, the filtering step has minimal cost, and takes only about 5% of the total time – the rest being taken up by the sampling or ray-tracing step. Previous methods utilize much larger fractions of the render time for filtering.

For each distribution effect (except chapter 6), we provided adaptive sampling rates which help in reducing noise throughout the image. Therefore, we are able to achieve interactive frame rates while obtaining the benefits of high quality ray-traced distribution effects. We demonstrated a variety of results for each effect using complex textured scenes with diffuse and glossy BRDFs, running at a 5-10 frames per second for simple effects and 2-5 seconds per image for complex effects. This is about an order of magnitude faster than other state-of-the-art methods, for about the same accuracy. Relative to equal visual quality ground truth, we are able to render images 30 to 60$\times$ faster.

We also made several theoretical contributions in the Fourier analysis of light fields for soft shadows, indirect illumination, defocus blur and environment illumination. We derived the Fourier spectrum for indirect illumination for the case of non-parallel reflectors and receivers. For environment illumination, we showed that the spectrum is the convolution of sheared Gaussians and is shaped like an ellipsoid. We showed how to band-limit the various spectra based on the lighting, BRDF (diffuse and Phong) geometry terms. To be able to handle multiple effects, we presented two different bandwidths for separated texture and irradiance. Another important contribution of our work was to precisely identify the minimum required per-pixel sampling rates, since regions with small filter sizes need more samples to eliminate noise. We also showed how to adjust the filter with the sampling rate, to enable convergence.

7.1 Future Work

There are several cases our set of algorithms cannot currently handle (as described in the preceding chapters) which make excellent avenues for future work. Extending our approach to a wider range of rendering problems would be crucial for its wide adoption. We do not handle specular-to-diffuse indirect illumination, namely caustics. Caustics are difficult to parametrize into simple light fields, and they are also hard to render with ray-tracing and typically require photon-mapping or Metropolis light transport. Although we showed filtering environment illumination, we do not handle the case of many discrete light sources in the scene – it could also be possible to combine our approach with a many-lights framework [103] to handle more general lighting environments. Handling more general BRDF models beyond the simple diffuse or Phong in our assumptions, such as the popular Cook-Torrance, is another possibility. In this work, we do not consider volume rendering effects such as multiple scattering through fog or fluids, but many of the smoothness assumptions apply to volumetric scattering effects as well, and hence our approach could be extended to these cases.

Our algorithms exploit spatial coherence or similarity within a single image. Very often, users wish to render a sequence of images of the same scene with some sort of animation or camera motion. As one may expect, there is a lot of coherence in the temporal domain as well, that is, the color at a given world location changes slowly across neighboring frames. Clearly, information can be shared in the temporal domain as well, but one needs to define a precise temporal filter for accurate reconstruction. We are investigating the frequency behavior and filtering of shading across time, considering a fixed spatial location. A combined spatio-temporal filter could reduce sampling rates by about $100\times$ relative to equal quality Monte-Carlo.

Another avenue for future work is to look at slightly more complex filter designs than our simple axis-aligned filter. This filter should still be almost as fast in implementation but also reduce the required sampling rate by a factor of two or so. One possibility is to simplify the 4D sheared filter into four 1D sheared filters [109]. This requires storing all the samples at each pixel (as for the previous sheared filters), but is almost as fast as axis-aligned filtering, and works with about 3 times fewer samples. Another idea is to use a combination of two smaller axis-aligned filters, one in the first quadrant of the Fourier plane, and another in the third quadrant. These filters would individually capture half of the double wedge spectrum, thus preserving the signal's frequency content, but also allow closer packing of aliases due to their smaller size. This could allow for a two to three times lower sampling rate for the same quality and higher speed compared to our method.

We believe the "sample and denoise" paradigm to be quite powerful as a unifying approach to achieve real-time physically-based distribution effects and we see our approach being integrated into production renderers and into real-time video games once it is more robust and general. However, this integration is non-trivial since production and game rendering systems have a different set of constraints and requirements, making this an exciting area for future work. Our novel application of axis-aligned filtering to mixed reality also opens up new directions for research in interactive physically-based rendering for mixed and augmented reality.

Appendix A

Bandlimits for Glossy BRDFs

The 2D Blinn-Phong BRDF in the v parameterization was described in equation 4.13. Here, we discuss the extension to 3D and resulting bandlimit $\Omega_h^{\max}$. Referring to Fig. A.1(a), the half vector in the v plane is

$$\mathbf{v_h} = \frac{l\mathbf{v_c} + l_c\mathbf{v}}{l + l_c} \tag{A.1}$$

where $l = \sqrt{1 + v_1^2 + v_2^2}$ and $l_c = \sqrt{1 + v_{c_1}^2 + v_{c_2}^2}$. Then the 3D Blinn-Phong transfer function is

$$H_s(\mathbf{v}, \mathbf{v_c}) = \frac{\cos^m \theta_h}{l^4} = \frac{1}{l^4(1 + v_{h_1}^2 + v_{h_2}^2)^{m/2}} \tag{A.2}$$

The Fourier transform of H_s will depend on the direction $\mathbf{v_c}$ and has no simple analytic form. However, since the choice of the v_1, v_2 axes is arbitrary (and we never actually compute using this parametrization), we simply use $\Omega_h^{\max}$ as an upper bound on the bandlimit of H_s for any value of $\mathbf{v_c}$ (given fixed m). For 99% energy, we can plot a curve of $\Omega_h^{\max}$ vs m, as shown in Fig. A.1(b), which can be fit with a simple linear approximation in the range $4 < m < 50$,

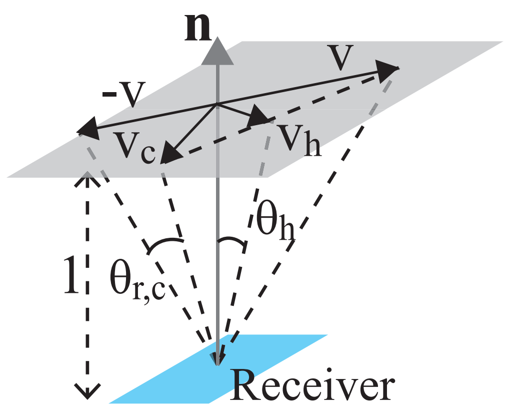
$$\Omega_h^{\max}(m) = 3.6 + 0.084m. \tag{A.3}$$

Similarly, in the Phong case, the angle of interest $\theta_{r,c}$ is between the reflection direction $(1, -\mathbf{v})$ and the camera direction $(1, \mathbf{v_c})$. Hence, the 3D Phong transfer function is

$$H_s(\mathbf{v}, \mathbf{v_c}) = \frac{\cos^m \theta_{r,c}}{l^4} = \frac{(1 - v_1 v_{c_1} - v_2 v_{c_2})^m}{l^4\, l^m\, l_c^m} \tag{A.4}$$

The bandlimit now closely fits the approximation,

$$\Omega_h^{\max}(m) = 4.27 + 0.15m. \tag{A.5}$$



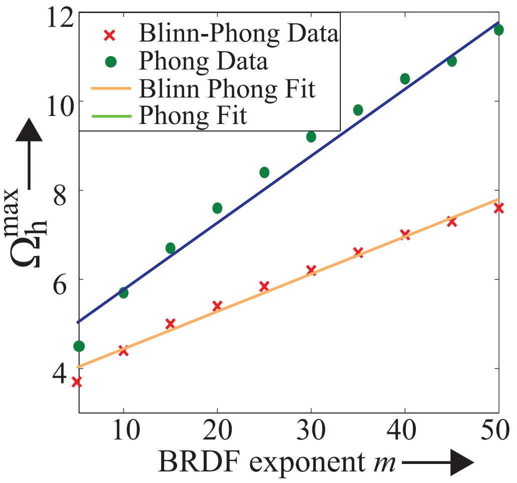


Figure A.1: (a) Geometry for Phong and Blinn-Phong BRDFs in our parameterization. (b) Our linear approximation for the transfer function bandlimit $\Omega_h^{\max}$ for Blinn-Phong and Phong BRDFs, as a function of the exponent m, is a close match to the numerical data.

Appendix B

Motion Blur with Secondary Effects

In Chapter 5 we described a factored axis-aligned filtering scheme for defocus blur with area light direct and indirect illumination. We now explain 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. 5.12. 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 [26]. At a given pixel, assume there is a single surface moving with image-space speed $v_p > 0$. The texture filter width is

$$\Omega^m_{x,p} = \min\left\{\Omega^{\max}_{\text{pix}}, \Omega^{\max}_{\text{t}}/v_p\right\} \tag{B.1}$$

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

$$\Omega^m_{x,s} = \min\left\{\Omega^{\max}_{\text{pix}}, \Omega^{\max}_{\text{t}}/v_s\right\} \tag{B.2}$$

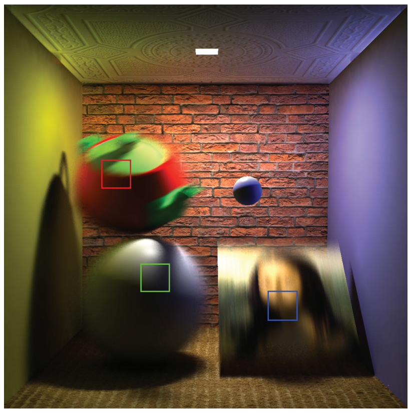
The subscript 's' indicates that this is a secondary, or irradiance filter width. These equations are analogous to eqn. 5.7 for defocus blur. Note that these filters are 1-D Gaussians in image space, oriented in the direction of the velocity $\mathbf{v}$, unlike the 2-D symmetric Gaussian defocus filter. The irradiance is also filtered according to the area-light filter width given in eqn. 5.18. 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. 5.24:

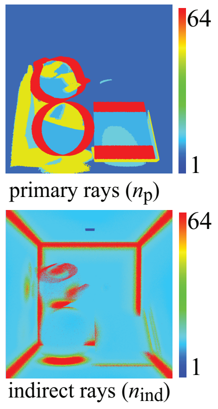
$$n_p = (\Omega_{\text{pix}}^{\max} + \Omega_x^t)^2 (1 + v\Omega_x^t)^2 \tag{B.3}$$

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. 5.25. The sampling rate for indirect can be computed similar to eqn. 5.26, but we found that the $(\Omega_t^*)^2$ term can be ignored.

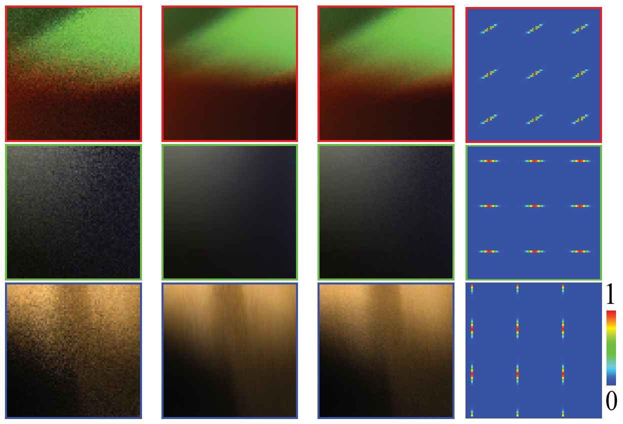
Results: We implemented our algorithm for motion blur with soft shadows and indirect illumination on Intel's CPU-parallel Embree ray-tracer (since the Optix ray-tracer does not support motion-blur ray-tracing). In Fig. B.1, we show a cornell box scene with textures and moving objects. The insets (c, 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. B.1 (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 B.1: 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.*

Appendix C

Two-mode Sampling Algorithm

In Chapter 6, we propose a two-mode path-tracer that traces OptiX rays to intersect only virtual geometry, and screen-space rays to intersect only real geometry. The pseudo-code for the sampling algorithm is given in Algorithm 1. For brevity, we omit obvious function arguments. Position and normal are abbreviated to 'pos' and 'n'; real and virtual quantities are separated with suffixes '_r' and '_v' respectively.

The main blocks in the code are explained below:
Lines 3-12: The outer loop consists of 4 samples per pixel (spp) anti-aliasing, and we determine whether a real or a virtual object is visible at the current sample and update the mask M.
Line 13: This loop computes 4 secondary samples for each of the 4 primary samples, so we compute a total of 16 spp for each of direct and indirect illumination.
Lines 14-25: For direct illumination, we importance sample the environment map, as this gives the least amount of noise for very little overhead. In line 15, an environment map importance sample is obtained, and in lines 18-19, the functions *trace_optix_sray* and *trace_ss_sray* return hit distances (-1 if no hit) for OptiX and screen-space shadow rays respectively.
Lines 26-40: For indirect illumination, we sample the cosine hemisphere for diffuse surfaces (real and virtual) and a Phong lobe for glossy surfaces (virtual only). Line 27 samples the BRDF to produce a sampling direction, and in lines 28-29, *trace_optix_iray* and *trace_ss_iray* return the indirect radiance and hit distance for OptiX and screen-space indirect rays respectively. In line 28, for screen-space rays, radiance from the secondary hit is computed by L_{cam} image look-up. In line 29, radiance from the secondary hit for OptiX rays (that intersect only virtual surfaces) is computed by tracing a secondary shadow ray to an environment map sample.

Although not shown in Algorithm 1, we also save the (average) world location, normal, virtual texture k_V. We also record the minimum hit distance for direct and indirect illumination; these are required for filtering. Since texture is multiplied with irradiance after filtering, we require the approximation $\langle k \cdot E \rangle \approx \langle k \rangle \cdot \langle E \rangle$, where $\langle\rangle$ denotes the mean of the quantity at a pixel.

This algorithm can be implemented in a single Optix pixel-shader kernel execution. However, Optix kernels are optimized only for ray intersection testing, and thread divergence reduces speed of this one-kernel approach significantly due to the screen-space ray-tracing component. Hence, we implement the algorithm in two passes. A first Optix pass traces only the Optix rays, storing the intersection results in a buffer. A second CUDA pixel shader pass traces the screen-space rays and also combines the result of screen-space and Optix ray-tracing, per lines 21-26 and 32-42 of the pseudo-code.

Algorithm 1

```
1: for each pixel do
2:     E_dir_r = E_dir_rv = E_ind_r = E_ind_rv = 0, M = 0
3:     for $i = 1:4$ do                                              ▷ Anti-aliasing
4:         is_real = false
5:         {pos_v ,n_v ,depth_v} = trace_primary_ray_optix(...)
6:         {pos_r ,n_r ,depth_r} = real_world_map(...)
7:         if depth_v = $-1$ OR depth_r $<$ depth_v then
8:             is_real = true, M += 0.25
9:             pos = pos_r, n = n_r
10:        else
11:            pos = pos_v, n = n_v
12:        end if
13:        for $j = 1:4$ do                                          ▷ Secondary Samples
14:            // Direct Illumination
15:            sample = get_importance_sample(...)
16:            L_env = env_map[sample.xy]
17:            ray_dir = {pos, sample_to_direction(sample.xy)}
18:            hit_dist_v = trace_optix_sray(ray_dir)
19:            hit_dist_r = trace_ss_sray(ray_dir)
20:            if is_real and hit_dist_r = $-1$ then
21:                E_dir_r += L_env * cos_theta / sample.pdf
22:            end if
23:            if hit_dist_r = $-1$ and hit_dist_v = $-1$ then
24:                E_dir_rv += L_env * cos_theta / sample.pdf
25:            end if
26:            // Indirect Illumination
27:            ray_ind = {pos, sample_BRDF(...)}
28:            {hit_dist_v, L_ind_v} = trace_optix_iray(ray_ind)
29:            {hit_dist_r, L_ind_r} = trace_ss_iray(ray_ind)
30:            if hit_dist_r $> 0$ and hit_dist_v$> 0$ then
31:                if hit_dist_r $>$ hit_dist_v then
32:                    E_ind_rv += L_ind_v
33:                else
34:                    E_ind_rv += L_ind_r
35:                end if
36:            end if
37:            // Cases where dist$= -1$ not shown
38:            if is_real and hit_dist_r $> 0$ then
39:                E_ind_r += L_ind_r
40:            end if
41:        end for
42:    end for
43:    E_{dir|ind}_{r|rv} /= 16 // normalize quantities
```

Appendix D

Derivation of Chapter 6, equation 11

Here we prove eqn. 6.11. Taking the 1D Fourier transform of eqn. 6.10 gives:

$$\begin{aligned}\hat{E}(\Omega_x) &= \iint L_e(\theta+\kappa x)H(\theta-\theta_{\text{occ}}+\lambda x)f(\theta)e^{-jx\Omega_x}\,d\theta\,dx \\ &= \int\Big(\iint L_e(\theta+\kappa x)H(\theta-\theta_{\text{occ}}+\lambda x)\,e^{-jx\Omega_x}e^{-j\theta\Omega_\theta}\,dx\,d\theta\Big)\hat{f}(-\Omega_\theta)\,d\Omega_\theta \\ &= \int \hat{G}(\Omega_x,\Omega_\theta)\hat{f}(\Omega_\theta)\,d\Omega_\theta\end{aligned} \tag{D.1}$$

In the second step, we have used the inverse Fourier transform $f(\theta)=\int\hat{f}(-\Omega_\theta)e^{-j\theta\Omega_\theta}\,d\theta$, and the Fourier transform of f satisfies $\hat{f}(-\Omega_\theta)=\hat{f}(\Omega_\theta)$.

Since both L_e and H are 1D functions sheared along constant slopes, their Fourier transforms are straight lines through the origin. The spectrum of the product $\hat{G}(\Omega_x,\Omega_\theta)$ is then a convolution of two lines of different slopes. We now derive $\hat{G}$ explicitly:

$$\begin{aligned}\hat{G}(\Omega_x,\Omega_\theta) &= \iint L_e(\theta+\kappa x)H(\theta-\theta_{\text{occ}}+\lambda x)\,e^{-jx\Omega_x}e^{-j\theta\Omega_\theta}\,dx\,d\theta \\ &= \mathcal{F}\{L_e(\theta+\kappa x)\} * \mathcal{F}\{H(\theta-\theta_{\text{occ}}+\lambda x)\} \\ &= \iint \hat{L}_e(\omega_\theta)\delta(\omega_x-\kappa\omega_\theta) \\ &\qquad \hat{H}(\Omega_\theta-\omega_\theta)\delta(\Omega_x-\omega_x-\lambda(\Omega_\theta-\omega_\theta))e^{-j\theta_{\text{occ}}\Omega_\theta}\,d\omega_x\,d\omega_\theta\end{aligned} \tag{D.2}$$

We use the property $\int f(x)\delta(ax-b)dx = f(b/a)/|a|$ to simplify the integral of the product of two delta functions.

$$\int \delta(\omega_x-\kappa\omega_\theta)\delta(\Omega_x-\omega_x-\lambda(\Omega_\theta-\omega_\theta))\,d\Omega_x = \delta(\Omega_x-\kappa\omega_\theta-\lambda(\Omega_\theta-\omega_\theta)) \tag{D.3}$$

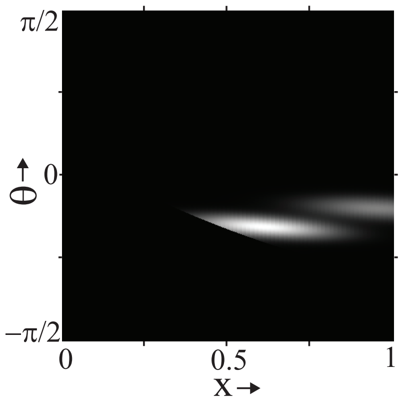
Substituting this into eqn. D.2 and applying the delta function integral property once again, we get:

$$\begin{aligned}
\hat{G}(\Omega_x,\Omega_\theta) &= e^{-j\theta_{\text{occ}}\Omega_\theta}\int \hat{L}_e(\omega_\theta)\hat{H}(\Omega_\theta-\omega_\theta)\delta(\Omega_x-\kappa\omega_\theta-\lambda(\Omega_\theta-\omega_\theta))\,d\omega_\theta \\
&= \frac{e^{-j\theta_{\text{occ}}\Omega_\theta}}{\lambda-\kappa}\hat{L}_e\left(\frac{-\Omega_x+\lambda\Omega_\theta}{\lambda-\kappa}\right)\hat{H}\left(\Omega_\theta-\frac{-\Omega_x+\lambda\Omega_\theta}{\lambda-\kappa}\right) \\
&= \frac{e^{-j\theta_{\text{occ}}\Omega_\theta}}{\lambda-\kappa}\hat{L}_e\left(-\frac{\Omega_x-\lambda\Omega_\theta}{\lambda-\kappa}\right)\hat{H}\left(\frac{\Omega_x-\kappa\Omega_\theta}{\lambda-\kappa}\right)
\end{aligned} \tag{D.4}$$

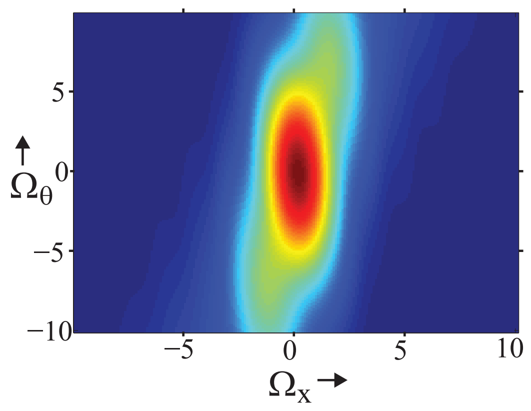
Appendix E

Verification of Chapter 6, equation 17

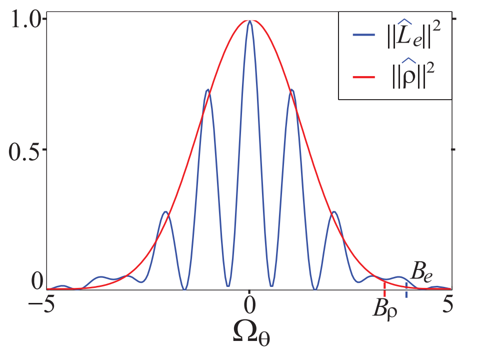
In Chapter 6, Fig.6.5 verifies eqn.6.13 for a particular diffuse flatland set-up. Here, we provide a similar verfication for eqn.6.17, the glossy case. As shown in Fig.E.1 below, eqn. 6.17 over-estimates the true bandwidth, since we simply combine eqn. 6.16 and eqn. 6.13. However, this estimate works well for filtering glossy surfaces as shown in Fig.6.7.



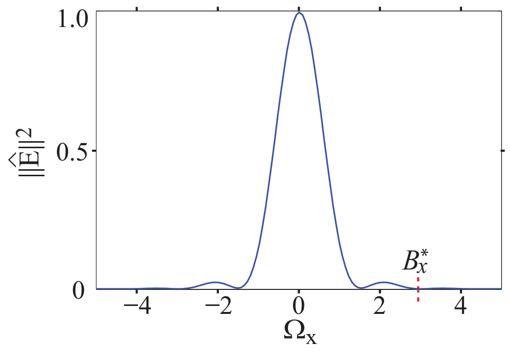
(a) $L_i \times V \times \rho$



(b) $||\hat{G}||^2$



(c) $||\hat{L}_e||^2$, $||\hat{\rho}||^2$



(d) $||\hat{E}||^2$

Figure E.1: Verification of eqn.6.17 for the simple flatland setup of Fig. 3(c) with a glossy ($n = 32$) surface with $\kappa = 0.5$ and one occluder at $\theta_{\text{occ}} = \pi/4$ and $z = 2$ ($\cos^2\theta_{\text{occ}}/z = 0.25$), under high-frequency illumination. (a) shows the product $L_i \times V(x,\theta)$ for this setup. (b) shows the power spectrum $||\hat{G}||^2$ of (a). In (c) we show the 1D power spectra of L_e and ρ, showing bandlimits $B_e = 4$ and $B_\rho = 3.5$. (d) shows the 1D power spectrum $\hat{E}$ of the surface irradiance, showing the true bandwidth $B_x^* \approx 3$. Eqn.6.17 ($B_f = 1$)gives $B_x = 4 + 0.25 \times 12 = 7$. Our estimate is conservative but not tight.

Bibliography

[1] Sameer Agarwal et al. “Structured Importance Sampling of Environment Maps”. In: *ACM Trans. Graph.* 22.3 (2003), pp. 605–612.

[2] M Agrawala et al. “Efficient Image-Based Methods for Rendering Soft Shadows”. In: *SIGGRAPH 2000*. 2000, pp. 375–384.

[3] T Annen et al. “Real-time all-frequency shadows in dynamic scenes”. In: *ACM Transactions on Graphics (SIGGRAPH 08)* 27.3 (2008), Article 34, 1–8.

[4] D. Antwerpen. “Improving SIMD Efficiency for Parallel Monte Carlo Light Transport on the GPU”. In: *High Performance Graphics*. 2011.

[5] O. Arikan, D. Forsyth, and J. O’Brien. “Fast and detailed approximate global illumination by irradiance decomposition”. In: *ACM Transactions on Graphics (SIGGRAPH 05)* 24.3 (2005), pp. 1108–1114.

[6] U. Assarsson and T. Möller. “A Geometry-Based Soft Shadow Volume Algorithm Using Graphics Hardware”. In: *ACM Transactions on Graphics (SIGGRAPH 03)* 22.3 (2003), pp. 511–520.

[7] Mahdi M. Bagher et al. “Interactive Rendering of Acquired Materials on Dynamic Geometry Using Bandwidth Prediction”. In: *Proceedings of the ACM SIGGRAPH I3D’12*. 2012, pp. 127–134.

[8] P. Bauszat, M. Eisemann, and M. Magnor. “Guided Image Filtering for Interactive High Quality Global Illumination”. In: *Computer Graphics Forum (EGSR 11)* 30.4 (2011), pp. 1361–1368.

[9] Laurent Belcour et al. “5D Covariance Tracing for Efficient Defocus and Motion Blur”. In: *ACM Trans. Graph.* 32.3 (2013), 31:1–31:18.

[10] A Ben-Artzi et al. “A Precomputed Polynomial Representation for Interactive BRDF Editing with Global Illumination”. In: *ACM Transactions on Graphics* 27.2 (2008), Article 13, 1–13.

[11] C. Benthin and I. Wald. “Efficient ray traced soft shadows using multi-frusta tracing”. In: *High Performance Graphics 2009*. 2009, pp. 135–144.

[12] Jin-Xiang Chai et al. “Plenoptic Sampling”. In: *SIGGRAPH ’00*. 2000, pp. 307–318.

[13] J Chai et al. “Plenoptic Sampling”. In: *SIGGRAPH 00*. 2000, pp. 307–318.

[14] R Cook, T Porter, and L Carpenter. “Distributed Ray Tracing”. In: *SIGGRAPH 84*. 1984, pp. 137–145.

[15] Oliver Cossairt, Shree Nayar, and Ravi Ramamoorthi. “Light Field Transfer: Global Illumination Between Real and Synthetic Objects”. In: *ACM Trans. Graph.* 27.3 (2008), 57:1–57:6.

[16] C. Crassin et al. “Interative indirect illumination using voxel cone tracing”. In: *Computer Graphics Forum* 30.7 (2011), pp. 1921–1930.

[17] F. Crow. “Shadow Algorithms for Computer Graphics”. In: *SIGGRAPH 77*. 1977, pp. 242–248.

[18] K. Dabov et al. “Image Denoising by sparse 3D transform-domain collaborative filtering”. In: *IEEE Transactions on Image Processing* 16.8 (2007), pp. 2080–2095.

[19] Holger Dammertz et al. “Edge-avoiding À-Trous Wavelet Transform for Fast Global Illumination Filtering”. In: *Proceedings of the Conference on High Performance Graphics*. Saarbrucken, Germany, 2010, pp. 67–75.

[20] Paul Debevec. “Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography”. In: *SIGGRAPH ’98*. 1998, pp. 189–198.

[21] Mauricio Delbracio et al. “Boosting Monte Carlo Rendering by Ray Histogram Fusion”. In: *ACM Transactions on Graphics* 33.1 (2014), 8:1–8:15. ISSN: 0730-0301. DOI: `10.1145/2532708`. URL: `http://doi.acm.org/10.1145/2532708`.

[22] F Durand. *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, 2011.

[23] F Durand et al. “A Frequency Analysis of Light Transport”. In: *ACM Transactions on Graphics (Proc. SIGGRAPH 05)* 25.3 (2005), pp. 1115–1126.

[24] K. Egan, F. Durand, and R. Ramamoorthi. “Practical Filtering for Efficient Ray-Traced Directional Occlusion”. In: *ACM Transactions on Graphics (SIGGRAPH Asia 11)* 30.6 (2011).

[25] K Egan et al. “Frequency Analysis and Sheared Filtering for Shadow Light Fields of Complex Occluders”. In: *ACM Transactions on Graphics* 30.2 (2011).

[26] K Egan et al. “Frequency analysis and sheared reconstruction for rendering motion blur”. In: *ACM Transactions on Graphics (SIGGRAPH 09)* 28.3 (2009).

[27] E. Eisemann and X. Decoret. “Visibility Sampling on GPU and Applications”. In: *Computer Graphics Forum (EG 07)* 26.3 (2007), pp. 535–544.

[28] V. Forest, L. Barthe, and M. Paulin. “Accurate Shadows by Depth Complexity Sampling”. In: *Computer Graphics Forum* 27.2 (2008), pp. 663–674.

[29] Alain Fournier, Atjeng S. Gunawan, and Chris Romanzin. “Common Illumination Between Real and Computer Generated Scenes”. In: *Graphics Interface*. May 1993, pp. 254–262.

[30] T.A. Franke. “Delta Voxel Cone Tracing”. In: *ISMAR*. Sept. 2014, pp. 39–44.

[31] Claude Gasquet and Patrick Witomski. *Fourier Analysis and Applications: Filtering, Numerical Computation, Wavelets*. Springer, 1998.

[32] R Gershbein, P Schröder, and P Hanrahan. “Textures and Radiosity: Controlling Emission and Reflection with Texture Maps”. In: *SIGGRAPH 94*. 1994, pp. 51–58.

[33] Simon Gibson and Alan Murta. “Interactive Rendering with Real-World Illumination”. In: *Eurographics Workshop on Rendering*. 2000, pp. 365–376.

[34] G. Guennebaud, L. Barthe, and M. Paulin. “Real-time Soft Shadow Mapping by Backprojection”. In: *EGSR 06*. 2006, pp. 227–234.

[35] B Guo. “Progressive Radiance Evaluation Using Directional Coherence Maps”. In: *SIGGRAPH 98*. 1998, pp. 255–266.

[36] T Hachisuka et al. “Multidimensional Adaptive Sampling and Reconstruction for Ray Tracing”. In: *ACM Transactions on Graphics* 27.3 (2008), 33:1–33:10.

[37] T Hachisuka et al. “Multidimensional adaptive sampling and reconstruction for ray tracing”. In: *ACM Transactions on Graphics (SIGGRAPH 08)* 27.3 (2008).

[38] D. Hart, P. Dutré, and D. Greenberg. “Direct illumination with lazy visibility evaluation”. In: *SIGGRAPH 99*. 1999, pp. 147–154.

[39] M Hasan, F Pellacini, and K Bala. “Direct to Indirect Transfer for Cinematic Relighting”. In: *ACM Transactions on Graphics (Proc. SIGGRAPH 06)* 25.3 (2005), pp. 1089–1097.

[40] J. Hasenfratz et al. “A survey of Real-Time Soft Shadow Algorithms”. In: *Computer Graphics Forum* 22.4 (2003), pp. 753–774.

[41] Shahram Izadi et al. “KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera”. In: *Symposium on User Interface Software and Technology*. 2011, pp. 559–568. ISBN: 978-1-4503-0716-1.

[42] X Tong J. Chai and H Shum. “Plenoptic Sampling”. In: *SIGGRAPH 00*. 2000, pp. 307–318.

[43] G. Johnson et al. “Soft irregular shadow mapping: fast, high-quality, and robust soft shadows”. In: *I3D 2009*. 2009, pp. 57–66.

[44] J Kajiya. “The Rendering Equation”. In: *SIGGRAPH 86*. 1986, pp. 143–150.

[45] Nima Khademi Kalantari and Pradeep Sen. “Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms”. In: *Computer Graphics Forum (Proc. of Eurographics 2013)* 32.2 (2013), pp. 93–102.

[46] Peter Kán and Hannes Kaufmann. “Differential Irradiance Caching for Fast High-Quality Light Transport Between Virtual and Real Worlds”. In: *ISMAR*. 2013, pp. 133–141.

[47] Peter Kán and Hannes Kaufmann. "Differential Progressive Path Tracing for High-Quality Previsualization and Relighting in Augmented Reality". In: *ISVC 2013, Part II, LNCS 8034*. Ed. by George Bebis. 2013, pp. 328–338.

[48] Peter Kán and Hannes Kaufmann. "High-Quality Reflections, Refractions, and Caustics in Augmented Reality and their Contribution to Visual Coherence". In: *ISMAR*. 2012, pp. 99–108.

[49] Martin Knecht et al. "Reciprocal Shading for Mixed Reality". In: *Computers and Graphics* 36.7 (2012), pp. 846–856.

[50] Janne Kontkanen, Jussi Räsänen, and Alexander Keller. "Irradiance Filtering for Monte Carlo Ray Tracing". In: *Monte Carlo and Quasi-Monte Carlo Methods 2004*. Springer, 2004, pp. 259–272.

[51] J. Krivanek et al. "Radiance Caching ofr Efficient Global Illumination Computation". In: *IEEE Transactions on Visualization and Computer Graphics* 11.5 (2005), pp. 550–561.

[52] D. Lacewell et al. "Raytracing prefiltered occlusion for aggregate geometry". In: *IEEE Symposium on Interactive Raytracing 08*. 2008.

[53] S. Laine et al. "Soft shadow volumes for ray tracing". In: *ACM Transactions on Graphics (SIGGRAPH 05)* 24.3 (2005), pp. 1156–1165.

[54] D Lanman et al. "Shield Fields: modeling and capturing 3D occluders". In: *ACM Transactions on Graphics (SIGGRAPH ASIA 08)* 27.5 (2008).

[55] Jaakko Lehtinen et al. "Reconstructing the Indirect Light Field for Global Illumination". In: *ACM Transanctions on Graphics* 31.4 (2012), 51:1–51:10.

[56] J. Lehtinen et al. "Temporal light field reconstruction for rendering distribution effects". In: *ACM Transactions on Graphics* 30.4 (2011).

[57] Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. "SURE-based Optimization for Adaptive Sampling and Reconstruction". In: *ACM Transactions on Graphics* 31.6 (2012), 186:1–186:9.

[58] D. Maletz and R. Wang. "Importance Point Projection for GPU-based Final gathering". In: *Computer Graphics Forum (EGSR 11)* 30.4 (2011), pp. 1327–1336.

[59] Michael Mara et al. *Fast Global Illumination Approximations on Deep G-Buffers*. Tech. rep. NVR-2014-001. NVIDIA Corp., 16, 2014, p. 16.

[60] William R. Mark, Leonard McMillan, and Gary Bishop. "Post-rendering 3D Warping". In: *Symp. on Interactive 3D Graph.* 1997, pp. 7–16.

[61] Nelson L. Max and Douglas M. Lerner. "A two-and-a-half-D motion-blur algorithm". In: *Proceedings of SIGGRAPH 85*. 1985, pp. 85–93.

[62] M. McCool. "Anisotropic diffusion for Monte Carlo noise reduction". In: *ACM Transactions on Graphics* 18.2 (1999), pp. 171–194.

[63] S. Mehta, B. Wang, and R. Ramamoorthi. “Axis-Aligned Filtering for Interactive Sampled Soft Shadows”. In: *ACM Transactions on Graphics* 31.6 (2012), 163:1–163:10.

[64] Soham Uday Mehta et al. “Axis-Aligned Filtering for Interactive Physically-Based Diffuse Indirect Lighting”. In: *ACM Transactions on Graphics* 32.4 (2013), 96:1–96:12.

[65] D Mitchell. “Spectrally Optimal Sampling for Distribution Ray Tracing”. In: *SIGGRAPH 91*. 1991, pp. 157–164.

[66] Bochang Moon, Nathan Carr, and Sung-Eui Yoon. “Adaptive Rendering Based on Weighted Local Regression”. In: *ACM Trans. Graph.* 33.5 (2014), 170:1–170:14.

[67] S Nayar et al. “Fast separation of direct and global components of a scene using high frequency illumination”. In: *ACM Transactions on Graphics (SIGGRAPH 2006)* 25.3 (2006).

[68] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. “DTAM: Dense tracking and mapping in real-time”. In: *International Conference on Computer Vision (ICCV)*. 2011, pp. 2320–2327.

[69] F E Nicodemus, J C Richmond, and J J Hsia. “Geometrical Considerations and Reflectance”. In: *National Bureau of Standards* (Oct. 1977).

[70] M. Nießner et al. “Real-time 3D Reconstruction at Scale using Voxel Hashing”. In: *ACM Trans. Graph.* 32.6 (2013), 169:1–169:10.

[71] Derek Nowrouzezahrai et al. “Light Factorization for Mixed-frequency Shadows in Augmented Reality”. In: *ISMAR*. 2011, pp. 173–179.

[72] R Overbeck, C Donner, and R Ramamoorthi. “Adaptive Wavelet Rendering”. In: *ACM Transactions on Graphics (SIGGRAPH ASIA 09)* 28.5 (2009).

[73] R. Overbeck, R. Ramamoorthi, and W. Mark. “A Real-time Beam Tracer with Application to Exact Soft Shadows”. In: *EGSR 07*. 2007, pp. 85–98.

[74] R Overbeck et al. “Exploiting Temporal Coherence for Incremental All-Frequency Relighting”. In: *EuroGraphics Symposium on Rendering*. 2006, pp. 151–160.

[75] Steven G. Parker et al. “OptiX: A General Purpose Ray Tracing Engine”. In: *ACM Trans. Graph.* 29.4 (2010), 66:1–66:13.

[76] S. Parker et al. “OptiX: A General Purpose Ray Tracing Engine”. In: *ACM Transactions on Graphics (SIGGRAPH 10)* 29.4 (2010), 66:1–66:13.

[77] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2004.

[78] Michael Potmesil and Indranil Chakravarty. “A lens and aperture camera model for synthetic image generation”. In: *Proceedings of SIGGRAPH 81*. 1981, pp. 297–305.

[79] V. A. Prisacariu et al. “A Framework for the Volumetric Integration of Depth Images”. In: *ArXiv e-prints* (2014). arXiv: `1410.0925`.

[80] R Ramamoorthi and P Hanrahan. “A Signal-Processing Framework for Inverse Rendering”. In: *SIGGRAPH 01*. 2001, pp. 117–128.

[81] Ravi Ramamoorthi and Pat Hanrahan. “An Efficient Representation for Irradiance Environment Maps”. In: *SIGGRAPH ’01*. 2001, pp. 497–500.

[82] Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. “A First-order Analysis of Lighting, Shading, and Shadows”. In: *ACM Trans. Graph.* 26.1 (2007).

[83] T. Ritschel et al. “Micro-rendering for Scalable, Parallel Final Gathering”. In: *ACM Transactions on Graphics* 28.5 (2009), 132:1–132:8.

[84] T. Ritschel et al. “The State of the Art in Interactive Global Illumination”. In: *Computer Graphics Forum* 31.1 (2012), pp. 160–188.

[85] F. Rouselle, C. Knaus, and M. Zwicker. “Adaptive Rendering with Non-Local Means Filtering”. In: *ACM Transactions on Graphics* 31.6 (2012), 195:1–195:11.

[86] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. “Adaptive Sampling and Reconstruction Using Greedy Error Minimization”. In: *ACM Transactions on Graphics* 30.6 (2011), 159:1–159:12.

[87] H. Rushmeier and G. Ward. “Energy preserving non-linear filters”. In: *SIGGRAPH 94*. 1994, pp. 131–138.

[88] P. Sen and S. Darabi. “On Filtering the Noise from the Random Parameters in Monte Carlo Rendering”. In: *ACM Transactions on Graphics* 31.3 (2012).

[89] Pradeep Sen, Soheil Darabi, and Lei Xiao. “Compressive Rendering of Multidimensional Scenes”. In: *Proceedings of the 2010 International Conference on Video Processing and Computational Video*. 2011, pp. 152–183.

[90] Peter Shirley and R. Keith Morley. *Realistic Ray Tracing*. 2nd ed. Natick, MA, USA: A. K. Peters, Ltd., 2003. ISBN: 1568811985.

[91] P. Shirley et al. “A local image reconstruction algorithm for stochastic rendering”. In: *ACM Symposium on Interactive 3D Graphics*. 2011, pp. 9–14.

[92] E. Sintorn and U. Assarsson. “Sample Based Visibility for Soft Shadows using Alias-Free Shadow Maps”. In: *Computer Graphics Forum (EGSR 08)* 27.4 (2008), pp. 1285–1292.

[93] P Sloan, J Kautz, and J Snyder. “Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments”. In: *ACM Transactions on Graphics (SIGGRAPH 02)* 21.3 (2002), pp. 527–536.

[94] Peter-Pike Sloan, Jan Kautz, and John Snyder. “Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments”. In: *ACM Trans. Graph.* 21.3 (2002), pp. 527–536.

[95] C Soler and F Sillion. “Fast Calculation of Soft Shadow Textures Using Convolution”. In: *SIGGRAPH 98*. 1998, pp. 321–332.

[96] C Soler et al. “Fourier depth of field”. In: *ACM Transactions on Graphics* 28.2 (2009).

[97] Tiago Sousa, Nick Kasyan, and Nicolas Schulz. “Secrets of CryENGINE 3 Graphics Technology”. In: *SIGGRAPH Courses* (2011).

[98] Art Tevs, Ivo Ihrke, and Hans-Peter Seidel. “Maximum Mipmaps for Fast, Accurate, and Scalable Dynamic Height Field Rendering”. In: *Symp. on Interactive 3D Graph.* 2008, pp. 183–190.

[99] Yun-Ta Tsai et al. “Fast ANN for High-Quality Collaborative Filtering”. In: *High-Performance Graphics*. 2014.

[100] Karthik Vaidyanathan et al. “Layered Light Field Reconstruction for Defocus Blur”. In: *To appear in ACM Transactions on Graphics* (2014). http://software.intel.com/en-us/articles/layered-light-field-reconstruction-for-defocus-blur.

[101] Ingo Wald et al. “State of the Art in Ray Tracing Animated Scenes”. In: *STAR Proceedings of Eurographics 07*. Ed. by Dieter Schmalstieg and Ji\vrí Bittner. The Eurographics Association, Sept. 2007, pp. 89–116.

[102] I. Wald et al. “Interactive Global Illumination using Fast Ray Tracing”. In: *Rendering Techiques (EGWR 02)*. 2002.

[103] B. Walter et al. “Multidimensional lightcuts”. In: *ACM Transactions on Graphics* 25.3 (2006), pp. 1081–1088.

[104] R. Wang et al. “An efficient GPU-based approach for interactive global illumination”. In: *ACM Transactions on Graphics* 28.3 (2009).

[105] G Ward and P Heckbert. “Irradiance Gradients”. In: *Eurographics Rendering Workshop 92*. 1992, pp. 85–98.

[106] G. Ward, F. Rubinstein, and R. Clear. “A ray tracing solution for diffuse interreflections”. In: *SIGGRAPH 88*. 1988, pp. 85–92.

[107] L. Williams. “Casting Curved Shadows on Curved Surfaces”. In: *SIGGRAPH 78*. 1978, pp. 270–274.

[108] R. Xu and S. Pattanaik. “A Novel Monte Carlo Noise Reduction Operator”. In: *IEEE Computer Graphics and Applications* 25.2 (2005), pp. 31–35.

[109] Ling-Qi Yan et al. *Fast 4D Sheared Filtering for Interactive Rendering of Distribution Effects*. Tech. rep. UCB/EECS-2014-174. EECS Department, University of California, Berkeley, Oct. 2014.