Frequency Analysis and Sheared Filtering for Multidimensional Effects in Rendering

Kevin Egan

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2012

ABSTRACT

Frequency Analysis and Sheared Filtering for Multidimensional Effects in Rendering

Kevin Egan

Many of the most expensive effects in rendering are those that require integrating complex multidimensional signals. Computation for a single pixel can require hundreds of samples, and standard methods do not provide a mathematically sound way to share samples between pixels with overlapping integrands. This thesis first analyzes the underlying signals for motion blur and occlusion and identifies the sparse structure of these signals in the Fourier domain. We then leverage this information to design a sheared filter that is customized to each pixel's frequency content. We finally present practical algorithms that share samples between pixels, reduce sampling requirements by an order of magnitude, and provide significant speedups for many of the most expensive computations in computer graphics.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgments

I would first like to acknowledge the invaluable guidance, insight, and encouragement given to me by my research advisor Ravi Ramamoorthi. I would also like to thank all of my co-authors for their tireless efforts and their contributions to this thesis. Specifically, Frédo Durand who has been a constant source of sound advice and good cheer, Nicolas Holzschuch who helped me with some of the earliest Fourier derivations, as well as Yu-Ting Tseng and Florian Hecht who both did an amazing job writing software, suggesting improvements, running experiments, and all the other things required to bring a project to fruition. I would like to thank Eitan Grinspun, and all the members of the graphics lab for their helpful discussions and good humor. I would like to thank Mark Meyer and John Anderson who both contributed to early discussions relating the shadow method presented in this thesis. Finally, I would like to thank those brave souls on my thesis committee who have volunteered to help and guide my work on this document.

On a personal note I would like to thank my family and friends (Val, Etienne, Miklos, David, Maritza, Breannan, the CPC, Ryan, Manu, Trish, Charlie, Loreal, Bill, Kit, JP, Taylor and Kristina to name a few) for enriching the rare times that were not devoted to furious research.

Dedicated to Mike, Debbie, and Hugh.

Chapter 1

Introduction

Rendering a single image requires computing the light transport for millions of pixels. In many common scenarios nearby pixels do expensive integrals across overlapping multidimensional domains, but computational results are not shared between pixels. This is the case for both motion blur (integrating across time), and soft shadows (integrating over an area light or distant lighting). These multidimensional effects are expensive, but also crucial for realism in high quality offline rendering.

As the complexity within a pixel's multidimensional domain increases the computation time required to render an image also increases using previous techniques. The faster an object moves in a motion blurred image, the more information that gets packed into a single pixel's integration domain, in turn increasing complexity and the amount of high frequency information. Similarly for shadows cast by an area light, as the light source becomes larger each pixel sees more complex geometry, and the occluding signal has more energy in the high frequencies. Capturing these higher frequencies using previous methods requires computing more samples for each pixel, which slows down render times.

Another observation is that as the complexity of these effects increases, the final image content is often stripped of high frequency information. Intuitively for motion blur the faster an object moves in a motion blurred image the larger the effective blur. This blur in turn removes high frequencies from the final image.

Putting these two observations together we come to a somewhat frustrating conclusion. Using previous methods as an object increases in velocity more computation is required, but the the

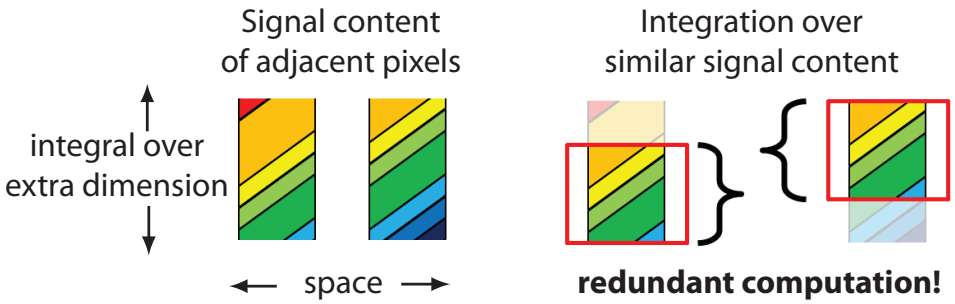

Figure 1.1: When integrating across other dimensions such as time or incoming light source, there is often correlation in the signal content between nearby pixels.

complexity and spatial frequencies in the final image actually *decrease* due to the blurring or filtering from motion. In effect as velocity increases we devote 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 (see Figure 1.1). Two adjacent pixels that view a fast moving object will end up integrating over functions that are very similar. 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 is a more difficult problem, and it is the problem addressed by this thesis.

This thesis makes the following contributions:

We use Fourier analysis to show how the multidimensional signals such as motion blur and occlusion transform during rendering. For motion blur this involves looking at how that signal changes with velocity in both the primal and Fourier space-time domains. For shadows from complex occluders we show that a very similar formulation can be exists in the lightfield between light and receiver. In both cases we show that most of the Fourier energy is captured inside a wedge determined by the minimum and maximum velocity (motion blur), or occluder depth (shadows).

We look at bandlimits and sampling rates imposed during rendering. This analysis allows us to derive required sampling rates to enable adaptive sampling. We also derive other interesting relationships, such as that using a conventional (axis-aligned) aliasing and shutter filter, for any uniform velocity, the product of spatial and temporal sampling rates is essentially constant for motion blurred

images.

We further demonstrate that we can sample more sparsely and pack frequency replicas much more tightly if we use a new sheared (not axis-aligned) reconstruction filter, which conforms to the frequency wedge, and follows the first-order direction of motion or occlusion in the primal domain. Although the analysis is in the frequency domain, the filter is simple to compute and implement directly in the primal domain. For the large angular domain of spherical distant lighting we give a rotationally invariant filter that is based on the sheared filter.

Finally we give a practical rendering method for both motion blur and complex occlusion. The algorithm sparsely samples the scene, estimates frequency bounds for the underlying signal, computes per-pixel sheared filters, and reconstructs multidimensional effects without requiring any explicit computation of Fourier spectra.

1.1 Overview

This thesis is organized as follows:

Chapter 2 gives background for rendering, motion blur, shadows, and reconstruction.

Chapter 3 presents a new method for accurately producing motion blurred images. We examine the image signal in space-time and show that in many cases the spectrum is contained in a double wedge shape. We propose a novel sheared filter that reduces the number of image space samples, and shares information across pixels to accelerate rendering.

Chapter 4 extends and expands these filtering ideas to reducing the number of shadow rays at a given shading point when calculating shadows from an area light source. We first analyze the frequencies of the 4D shadow light field for a single receiver. We then store all of the samples in a 4D ray database that is parameterized independent of any single receiver, allowing us to extend sheared filtering to receivers at many different depths.

Chapter 5 extends applies sheared filtering to a large angular domain for calculating shadows and spherical harmonic occlusion. We also allow for receivers with high frequency normal maps and general BRDFs. We first analyze the occlusion frequencies for distant lighting. We then propose a new rotationally invariant filter that can smoothly handle the large angular extent of the problem.

Chapter 6 summarizes our work, and suggests possible future work.

Chapter 2

Background

2.1 Definitions

Rendering is the process of simulating how light travels and is recorded. For our purposes we will consider scenes with solid surfaces and no atmospheric effects. The basic operations for light transport are travel through free space and reflection. At times it can also be convenient to consider occlusion to be a separate operation.

Two of the most common measurements of light are flux and radiance. Flux Φ is a measurement of power, which is simply energy per time. Radiance is a measure of flux across a differential surface area dA, across a differential solid angle $d\omega$.

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

with θ being the angle between the normal 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 (no intersecting surfaces and no atmospheric effects).

How light interacts with surfaces is modeled by the bidirectional reflectance distribution function, or *BRDF* [Nicodemus *et al.*, 1977]. 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:

$$\rho(\omega_i, \omega_o) = \frac{\mathrm{d}L_o(\omega_o)}{\mathrm{d}E_i(\omega_i)} = \frac{\mathrm{d}L_o(\omega_o)}{L_i(\omega_i)\cos(\theta_i)\mathrm{d}\omega_i} \quad , \tag{2.2}$$

where θ_i is the angle between ω_i and the surface normal n. We can additionally parameterize the BRDF with other variables, such as spatial coordinates x, time t, and light wavelength λ.

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 $\rho()$ and incoming lighting L_i:

$$L_o(x, \omega_o, \lambda, t) = \int_{\mathcal{H}^+} \rho(x, \omega_o, \omega_i, \lambda, t) L_i(x, \omega_i, \lambda, t) \cos(\theta_i) d\omega_i \quad . \tag{2.3}$$

Here $\mathcal{H}^+$ specifies the upper hemisphere of possible incoming light directions around the surface normal n at point x. 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*.

In the real world this restriction does not exist, light may reflect any number of times before being recorded. 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 $L_i(x, \omega_i, \lambda, t) = L_o(r(x, \omega_i), -\omega_i, \lambda, t)$, where $r(x, \omega_i)$ is the surface location found by tracing away from point x towards the $\omega_i$direction. By also including light emitters L_e we can then write the full rendering equation as follows [Kajiya, 1986]:

$$L_o(x, \omega_o, \lambda, t) = L_e(x, \omega_o, \lambda, t) + \int_{\mathcal{H}^+} \rho(x, \omega_o, \omega_i, \lambda, t) L_o(r(x, \omega_i), -\omega_i, \lambda, t) \cos(\theta_i) d\omega_i \tag{2.4}$$

Note that L_o appears on both sides of the equation. Specifically the rendering equation is a Fredholm integral of the second kind, and closed form solutions can only be computed for the simplest of scenes. For a deeper mathematical analysis of light transport and the rendering equation see [Veach, 1997]. For an introduction to rendering theory and practice see [Pharr and Humphreys, 2004].

Physical cameras must open and close the camera shutter for a finite period of time during the film exposure. If the recorded light changes during the film exposure the final image can be blurry due to nearby pixels integrating over similar time domains. This effect is called *motion blur*, and it is very common for fast moving objects, such as a speeding car. Motion blur is often very expensive

to simulate for fast moving phenomenon. To create a motion blurred image we must capture how the recorded light changes over time and integrate across the time dimension. For an exposure of length 1 with $t' \in [t, t+1]$ we have:

$$L_o(x, \omega_o, t) = \int_t^{t+1} \int_{\mathcal{H}^+} \rho(x, \omega_o, \omega_i, t') L_i(x, \omega_i, t') \cos(\theta_i) d\omega_i dt' \quad , \tag{2.5}$$

where we have dropped the dependence on wavelength λ for simplicity.

For direct lighting with area lights we will compute *soft shadows*, meaning shadows that may either be fully occluded, partially occluded, or fully unoccluded. For direct lighting we can rewrite the reflection equation to integrate directly over the surfaces of light sources. With A representing the surface of all light sources, and y a point on this surface we have:

$$L_o(x, \omega_o, t) = L_e(x, \omega_o, t) + \int_A \rho(x, \omega_o, x \to y, t) L_e(y, y \to x, t) g(x, y) \cos(\theta_y) \cos(\theta_i) dy \quad , \tag{2.6}$$

where $g(x, y)$ is a geometry term that is 0 if the path between light point y and surface point x is occluded, and 1 otherwise. The θ_y parameter is the angle between the emitting surface normal and $y \to x$, and the θ_i parameter is the angle between the reflecting surface normal and $x \to y$.

We can also consider distant lighting, that is lighting from a distant source that is defined across the sphere of all directions. For direct lighting with a distant light source we can define occlusion in a similar manner:

$$L_o(x, \omega_o, t) = L_e(x, \omega_o, t) + \int_{\mathcal{H}^+} \rho(x, \omega_o, \omega_i, t) g(x, -\omega_i) L_d(-\omega_i, t) \cos(\theta_i) d\omega_i \quad , \tag{2.7}$$

where L_d is the distant light source that is assumed to be emitted from an infinite distance away, $g(x, -\omega_i)$ is a geometry term that is 0 if there is any surface that intersects the ray that starts at point x along direction $-\omega_i$ and 1 otherwise.

A *light field* is a parameterization of all rays of light crossing through some volume. One common parameterization is to define rays by their intersection between two parallel planes. One slight variation is to define a ray by it's offset between the first and second plane (parameterizing by position and angle).

Chapter 3

Motion Blur

3.1 Introduction

Motion blur is important for creating synthetic images that match physical cameras, and for eliminating temporal aliasing in animations. As the velocity increases, more samples are usually required to render motion-blurred images. This is frustrating since the complexity and spatial frequencies in the final image actually *decrease* due to the blurring or filtering from motion (see Figures 3.1 and 3.2).

We seek to accelerate the rendering of motion-blurred scenes by a combination of adaptive sampling and a new sheared filter. Our main contribution is an analysis of the frequency content of scenes in space-time. This theoretical analysis enables us to derive the bandwidth, required sampling rate, and reconstruction filters for accurate rendering. We make the following contributions:

Space-Time Fourier Theory: We develop our frequency analysis in Sec. 3.3 with three key visual effects: movement of objects and surface texture, rotations of the BRDF and lighting, and moving shadows. We find similar mathematical forms in all cases: the final motion-blurred signal undergoes a *shear* in space-time and a corresponding shear in the frequency domain. For a given range of velocities, the Fourier spectrum can be approximated by a wedge.

Spatial and Temporal Bandlimits and Sampling Rates: This analysis allows us to derive required spatial and temporal sampling rates (Sec. 3.4), enabling adaptive sampling. In fact, we show

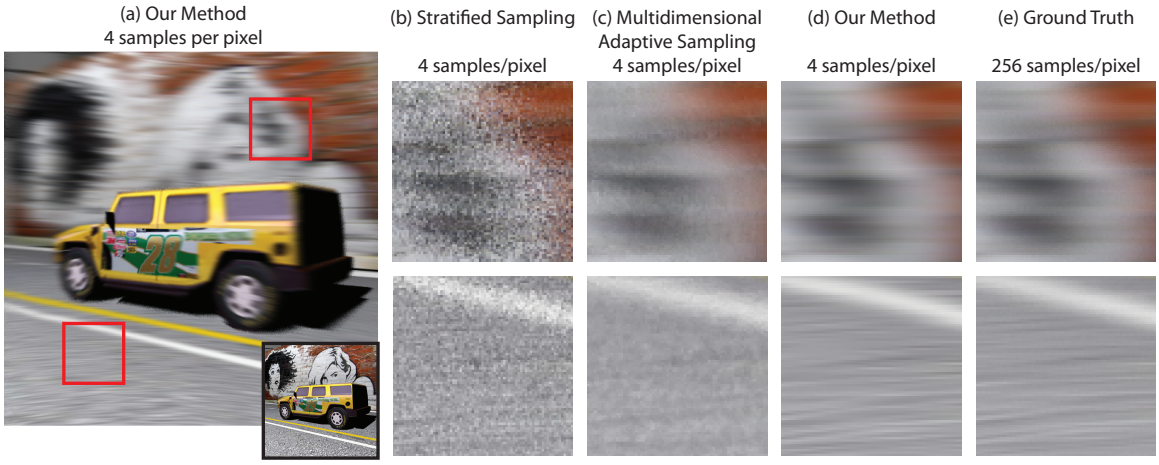

Figure 3.1: (a) Our method using an average of only 4 samples per pixel over the image. A static rendering of the scene is inset in the lower right and closeups are shown in (b-e). Stratified sampling in (b) is very noisy at this low sample count. Multidimensional Adaptive Sampling [Hachisuka et al. 2008] in (c) performs much better, but still has some noise, especially in fast-moving high-frequency textures, such as the mural (top) and ground (bottom closeup). Our technique in (d) produces a high-quality image with minimal noise that closely matches ground truth (e). Figure 3.7 shows details for our sheared filter.

that, using a conventional (axis-aligned) aliasing and shutter filter, and for uniform velocities, the product of spatial and temporal sampling rates is essentially constant, independent of the speed of motion.

Sheared Reconstruction Filter: We further demonstrate that we can sample more sparsely and pack frequency replicas much tighter if we use a new sheared (not axis-aligned across time) reconstruction filter, which conforms to the frequency wedge (Sec. 3.5), and gathers information from nearby pixels by following the first-order direction of motion in the primal domain.

Practical Space-Time Rendering Algorithm: Our motion-blur rendering method (Sec. 3.6) first estimates frequency bounds by sparsely sampling the scene. The algorithm then computes per-pixel sheared filters and sampling rates, without requiring any explicit computation of Fourier spectra. As shown in Figure 3.1, it can produce high-quality results with low sample counts.

3.2 Related Work

Motion Blur Rendering: Motion-blur rendering often relies on sampling the shutter interval, e.g. [Korein and Badler, 1983; Cook *et al.*, 1984; Haeberli and Akeley, 1990; Cammarano and Jensen, 2002; Akenine-Möller *et al.*, 2007] and high-quality sampling patterns can improve results [Mitchell, 1991]. The Reyes architecture [Cook *et al.*, 1987] reduces costs by shading at one time instant but densely sampling visibility through time. The Maya rendering system computes shading and visibility separately to capture changing illumination and reduce noise [Sung *et al.*, 2002].

Multi-Dimensional Adaptive Sampling (MDAS) is a general approach that adaptively samples based on contrast in the multi-dimensional integrand [Hachisuka *et al.*, 2008]. They approximate anisotropic filters with finite differences and a modified nearest-neighbor. In contrast, we predict local frequency information with each sample and utilize sheared reconstruction filters. A comparison of the practical results is made in Sec. 3.6.4; our method is somewhat better on fast-moving high-frequency signals, as in Figure 3.1. More recently papers have continued to develop new methods in this area, creating methods that automatically find and exploit coherence in the image signal [Lehtinen *et al.*, 2011; Sen and Darabi, 2011]. One advantage of our method is that the derivation predicts how wide to make the reconstruction filter, whereas many other methods that share information use a fixed sized filter.

Pixel tracing also looks at the Fourier domain, but only gives spectrum shapes for constant motion and is intended more for spatial anti-aliasing using multiple frames of an animation rather than computing motion blurred results within a single frame [Shinya, 1993]. Our paper makes important theoretical contributions by analyzing practical scenes with non-uniforms velocities in the frequency domain. This leads to key insights for sampling rates and anisotropic filters that may be relevant to MDAS and other methods as well.

Multi-dimensional lightcuts [Walter *et al.*, 2006] groups point light sources and shading samples, including samples in time for motion blur, into hierarchical graphs. This method is orthogonal to ours, since they reuse similar surface and lighting samples within one pixel, while our sheared reconstruction filters has a support that can span multiple pixels.

Image-space solutions blur based on the motion field at a single instant [Potmesil and Chakravarty, 1983; Max and Lerner, 1985; Neulander, 2007]. They can be efficient but often require segmentation

into layers, provide only an approximation, and are prone to artifacts. Our sheared reconstruction is related but operates on the full space-time domain and adapts to the content to yield accurate results.

Other methods have used modified filters for motion blur. Catmull [1984] suggests scaling the pixel anti-aliasing filter to match the motion, but it relies on analytic filtering of polygons. Anisotropic texture filtering has also been used in real-time rendering [Loviscach, 2005]. Both of these methods define a stretched space-only filter instead of our sheared space-time filter.

Light Transport Analysis: Our analysis builds on plenoptic sampling [Chai *et al.*, 2000; Isaksen *et al.*, 2000], and the frequency and gradient analysis of light transport [Durand *et al.*, 2005; Soler *et al.*, 2009; Ramamoorthi *et al.*, 2007]. In particular, we use the concept of light transport shears in the frequency domain [Durand *et al.*, 2005] and a wedge for the final spectrum [Chai *et al.*, 2000]. We extend these space-angle methods to consider non-uniform motion in space-time. Other work has touched on the sheared space-time spectra of translating signals [Shinya, 1993; Christmas, 1998; Levin *et al.*, 2008]. We go further in deriving explicit sampling rates, a theorem showing that the total sampling rate (in space and time) is approximately constant for axis-aligned filters, developing a sheared reconstruction filter, and in considering specularities and shadows.

3.3 Space-Time and Fourier Theory

We analyze the key visual effects in motion blur. We first examine the frequency content of a moving signal and show that it yields a space-time shear. General light transport involves shearing, convolution and other operations on spectra [Durand *et al.*, 2005], and it is beyond the scope of this paper to generalize all of them to the time domain. Instead, we focus on the three most common phenomena—object motion, BRDF reflection, and moving shadows. We show that, in space-time, all three effects have strikingly similar mathematical forms, which allows for a general treatment of motion blur as a *shear* in the space-time and Fourier domain. In Appendix D we give a detailed derivation of the general case considering all three effects.

For simplicity, most of the analysis is done for a 1D scanline, but the main insights carry over to 2D images and 3D space-time (with anisotropic shears following the direction of motion). An index of notation for the most important symbols is in Table 3.1.

$g(x, y)$	2D spatial signal (such as a planar texture)
$f(x, y, t)$	Time-Varying signal (moving object or texture)
$h(x, y, t)$	Time-Varying motion-blurred signal (image)
$f(x, t), h(x, t)$	1D time-varying signals for simplicity
$w(t)$	Temporal response of shutter
$\Omega_x^{\max}, \Omega_t^{\max}$	Max spatial, temporal frequencies (in $g(x)$,$w(t)$)
Ω_x^*, Ω_t^*	Spatial, temporal frequency bandlimit
Ω^*	Net frequency bandlimit (total samples needed)
$\mathcal{F}$	Fourier transform operator
$F(\Omega_x, \Omega_y, \Omega_t)$	Fourier transform of $f(x, y, t)$
$G(\Omega_x, \Omega_y), H(\Omega_x, \Omega_y, \Omega_t)$	Fourier transforms of $g(x, y)$, $h(x, y, t)$

Table 3.1: Notation for the key variables. The frequency analysis will use capital letters for Fourier transforms of the quantities shown here e.g., $F(\Omega_x, \Omega_t)$ denotes the Fourier transform of $f(x, t)$. Other notation is introduced in Secs. 3.3.2-3.3.3 to discuss BRDF effects and shadows.

3.3.1 Moving Object: Translating Signal

Consider a 2D signal $g(x, y)$, which can be thought of as a texture. The concept of "texture" here is general, and can also include geometric effects like silhouette boundaries. This signal is translated through time by $x_0(t)$ and $y_0(t)$,

$$f(x, y, t) = g(x - x_0(t), y - y_0(t)). \tag{3.1}$$

The motion-blurred signal or image is then given by

$$h(x, y, t) = \int_{-\infty}^{\infty} f(x, y, t')w(t - t')\, dt', \tag{3.2}$$

where $w(t)$ is the shutter response over time, responsible for motion blur. For Fourier analysis, it is useful to define all integrals over the infinite temporal domain. We consider h to be a continuous signal for analysis—in practice, the final rendering step will point-sample h in time to generate individual motion-blurred frames.

We first study the canonical case of translation with uniform velocities, so that $x_0(t) = at$ and

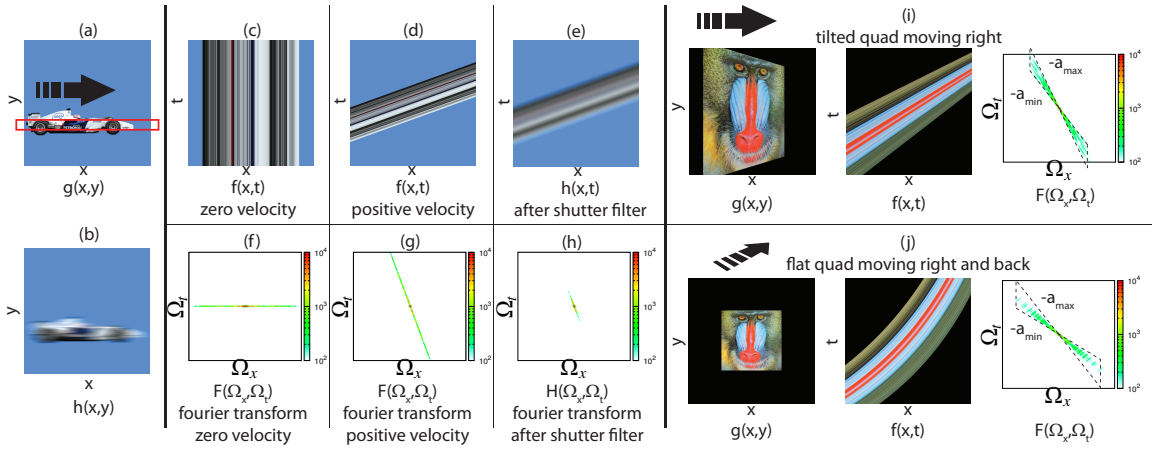

Figure 3.2: Space-Time and Fourier domain plots for a moving object. (a) Original signal $g(x, y)$; the scanline used for graphs (c), (d), and (e) is outlined in red. (b) (below (a)) $h(x, y, t)$ for a single instant in time; this is our final motion-blurred image. (c) A graph of $f(x, t)$ with zero velocity (a static image). In this case, there is no variation along the time or vertical axis. (d) $f(x, t)$ with positive uniform velocity, leading to a shearing along the spatial dimension. (e) $h(x, y, t)$ is obtained by applying a vertical blur along the time axis corresponding to the shutter filter. (f), (g) and (h) are the respective Fourier transforms of (c), (d) and (e). Note that (h) has frequencies in time restricted to $\Omega_t \in [-\Omega_t^{\max}, \Omega_t^{\max}]$ based on the shutter filter. (i) Because of perspective, the velocities change across space. (j) Because of perspective, velocities change across time. The frequency spectra span a wedge based on the minimum and maximum velocities.

$y_0(t) = bt$,

$$f(x, y, t) = g(x - at, y - bt). \tag{3.3}$$

For simplicity, consider a 1D scanline as in Figure 3.2(a):

$$\begin{aligned} f(x, t) &= g(x - at) \\ h(x, t) &= \int f(x, t')w(t - t')dt'. \end{aligned} \tag{3.4}$$

The basic setup is as shown in the top row of Figure 3.2, with Figure 3.2(b) being the final motion-blurred image. Figure 3.2(c) is a space-time diagram for a static scene ($a = 0$). In this case, there is no variation along the time (vertical) dimension. In Figure 3.2(d), we see the time-varying effects of motion. As is expected from Equations 3.3 and 3.4, this is a *shear* along the spatial x

direction. The effects of the shutter in Figure 3.2(e) are a blurring or filtering across the vertical time dimension. Figure 3.2(f-h), shows the corresponding frequency spectra, which we now derive analytically.

Fourier Analysis: To calculate the Fourier transform $\mathcal{F}(f(x, y, t))$, we first transform along x and y axes (denoted $\mathcal{F}_{x,y}$) to obtain an intermediate $F_t(\Omega_x, \Omega_y, t)$, and then transform along the time dimension. Therefore, we first calculate

$$F_t(\Omega_x, \Omega_y, t) = \mathcal{F}_{x,y}\left[g(x - x_0(t), y - y_0(t))\right]. \tag{3.5}$$

Since $x_0(t)$ and $y_0(t)$ depend only on time, they can be treated as constant shifts for the spatial Fourier transform above. By the standard theory of shifted Fourier transforms, F_t relates closely to $G(\Omega_x, \Omega_y)$ which is the Fourier transform of g,

$$F_t(\Omega_x, \Omega_y, t) = e^{-i2\pi(\Omega_x x_0(t) + \Omega_y y_0(t))} G(\Omega_x, \Omega_y). \tag{3.6}$$

Now consider translation with a uniform velocity a and b in the x and y directions, as per Equation 3.3. Applying the Fourier transform along the time axis

$$\begin{aligned} F(\Omega_x, \Omega_y, \Omega_t) &= G(\Omega_x, \Omega_y) \int e^{-i2\pi t(\Omega_x a + \Omega_y b + \Omega_t)}\, dt \\ &= G(\Omega_x, \Omega_y)\delta(\Omega_x a + \Omega_y b + \Omega_t). \end{aligned} \tag{3.7}$$

By translating the 2D signal (corresponding to a spatial shear in the space-time domain), we have sheared the signal along the temporal axis in the frequency domain (all non-zero frequencies lie on the plane $\Omega_x a + \Omega_y b + \Omega_t = 0$ in 3D Fourier space). This result also shows the coupling of spatial and temporal dimensions.

While our analysis applies fully to 2D signals, it is easier to expose with a single spatial dimension or a 1D signal per Equation 3.4,

$$\boxed{F(\Omega_x, \Omega_t) = G(\Omega_x)\delta(\Omega_x a + \Omega_t),} \tag{3.8}$$

restricting the frequency spectrum to a single line $\Omega_x a + \Omega_t = 0$, as seen in Figure 3.2(g). Note that Figure 3.2(g) is obtained by shearing the Fourier spectrum in Figure 3.2(f) along the time dimension, with the amount of shear given by the velocity a.

Finally, from Equation 3.4, we know that $h(x,t)$ is obtained from $f(x,t)$ simply by convolving with $w(t)$, which becomes a multiplication in the temporal frequency domain,

$$\boxed{H(\Omega_x, \Omega_t) = G(\Omega_x)\delta(\Omega_x a + \Omega_t)W(\Omega_t).} \tag{3.9}$$

As seen in Figure 3.2(h), the high temporal frequencies in Figure 3.2(g) are attenuated or removed, because W is the frequency spectrum of the low-pass shutter filter (in principle, only an infinite sinc function can be an exact low-pass filter, but most filters like gaussians allow one to define a practical threshold, such as capturing 99% of the energy).

Non-Uniform Velocities: For typical shutter speeds that cover a short time window, a uniform velocity is often a good approximation. However, there are cases where perspective, acceleration and occlusion effects cause variations in speed and spatially non-uniform velocities. An analytic Fourier transform cannot be obtained in these cases, but we can approximate its range, based on the non-negative minimum and maximum velocities $a \in [a_{\min}, a_{\max}]$.

Figure 3.2(i) shows a tilted quad moving to the right, where velocities change across space because of perspective. Analogously, Figure 3.2(j) shows a quad moving right and away from the camera, with velocities changing across time because of perspective. While the spectra are complicated, we find that most of the energy lies in the wedge bounded by shears corresponding to minimum $a_{\min}$ and maximum $a_{\max}$ velocities (Figures 3.2(i),3.2(j),3.6(a)). This is similar to the use of minimum and maximum depths to bound the frequency spectrum for image-based rendering [Chai *et al.*, 2000].

3.3.2 BRDF Effects and Shading

We now consider the motion of reflections (and shadows in Sec. 3.3.3). We will obtain very similar mathematical forms as those just seen for moving objects. This is illustrated in Figure 3.3, which shows the shearing in spatial and frequency domains, analogous to Figure 3.2. Some readers may wish to skip the derivations on a first reading, and can move directly to Sec. 3.4 without loss of continuity.

For simplicity, we consider flatland or 2D reflections, similar to [Durand *et al.*, 2005; Ramamoorthi *et al.*, 2007]. A diagram is shown in Figure 3.4. We write the standard reflection

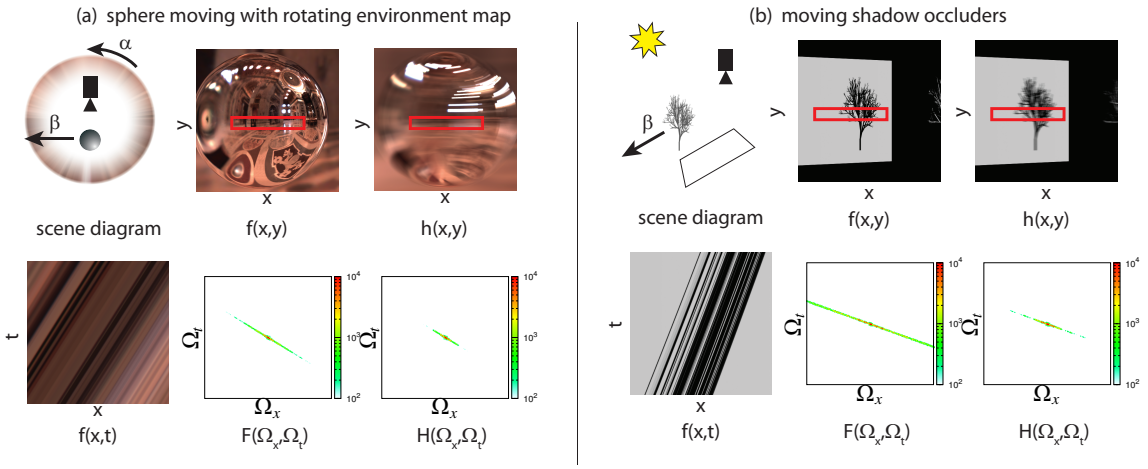

Figure 3.3: (a) A moving surface, in this case a sphere, with a rotating environment map. As the object moves, the specular reflections are motion-blurred. (b) A moving shadow from blockers, in this case a tree. As the occluder moves, so does the occluded region, leading to motion-blurred shadows on the receiver. We obtain space-time and frequency-domain shears based on the effective pixel velocities. (Note that since our analysis is local, curved global paths for specular highlights and shadows are not an issue.)

equation for $f(x,t)$, but extend it by considering its time-varying nature,

$$f(x,t) = \int l(\theta,t) r(2n(x,t) - \theta)\, d\theta, \tag{3.10}$$

where $l(\theta,t)$ is the (time-varying) incident lighting[1] and r is a radially symmetric BRDF (like Lambertian or Phong), including the cosine term. As shown in Figure 3.4, we consider a single overhead view, so that the angle between lighting and reflected directions is given by $2n - \theta$ where n is the normal.

There are two sources of time-dependence or motion blur. First, the lighting may vary with time—for concreteness, we consider moving the lights. For distant illumination, this corresponds to a rotation, with α being the angular velocity. We can also linearize motions of local sources to a rotation and angular velocity,

$$l(\theta,t) = l(\theta - \theta_0(t)) = l(\theta - \alpha t). \tag{3.11}$$

[1] The lighting can canonically be thought of as a distant environment map, but can also correspond to the local environment at $x = 0$ (assuming the spatial variation of lighting is moderate, such as mid-range illumination).

Next, consider normal $n(x,t)$. If the object is translating,

$$n(x,t) = n(x - x_0(t)) = n(x - \beta t), \tag{3.12}$$

where we now use β for the velocity of motion (to distinguish from a used previously). Finally, the normal can be locally linearized so that $n(x) = \kappa x + \eta$, with κ related to the surface curvature,[2]

$$n(x - \beta t) = \kappa(x - \beta t) + \eta = \kappa x - \kappa\beta t + \eta. \tag{3.13}$$

Now, substituting Equations 3.11 and 3.13 into Equation 3.10 and using $\kappa' = 2\kappa$ and $\eta' = 2\eta$ to account for the factor of $2n(\cdot)$,

$$f(x,t) = \int l(\theta - \alpha t) r(\kappa' x - \beta\kappa' t - \theta + \eta')\, d\theta, \tag{3.14}$$

The above equation can be integrated by substituting $\omega = \theta - \alpha t$,

$$f(x,t) = \int l(\omega) r\left([\kappa' x - (\alpha + \beta\kappa')t + \eta'] - \omega\right)\, d\omega. \tag{3.15}$$

The right-hand side of the above equation is a convolution. Defining $\gamma = \alpha + \beta\kappa'$—where γ is the relative angular velocity of lighting and surface—and using $\otimes$ for convolution,

$$f(x,t) = (l \otimes r)(\kappa' x - \gamma t + \eta'), \tag{3.16}$$

where the result is evaluated at $(\kappa' x - \gamma t + \eta')$.

It is possible to bring Equation 3.16 into the same form as Equation 3.4, unifying two seemingly quite different phenomena—motion-blurred texture/geometry and specular reflections. To do so, we simply need to define $g = l \otimes r$, so that in analogy to Equation 3.4,

$$\begin{aligned} f(x,t) &= g\left(\kappa'\left[x - \frac{\gamma}{\kappa'}t + \frac{\eta'}{\kappa'}\right]\right) \\ h(x,t) &= \int f(x,t') w(t - t') dt'. \end{aligned} \tag{3.17}$$

In this case, the effective velocity a from Equation 3.4 is simply γ/κ', which is the effective spatial rate of motion (relative angular velocity divided by curvature). The η'/κ' term is only a constant offset, which will become a simple phase shift in Fourier space. The curvature κ' multiplies x to convert from spatial to angular coordinates.

[2]Since the surface may be tilted with respect to the image scanline along which the spatial dimension x is measured, κ is actually the screen-space curvature, and differs by a cosine factor from the geometric curvature.

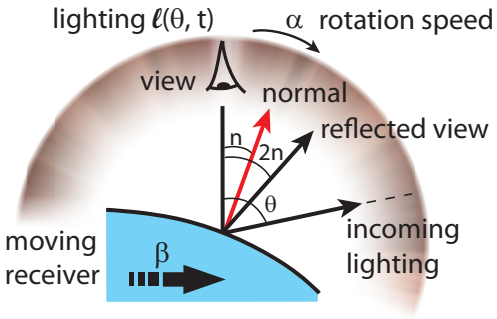

Figure 3.4: BRDF effects and shading with motion blur. The basic (planar or flatland 2D) setup shows a complex lighting environment $l(\theta, t)$ that can rotate with angular velocity α. The surface can also move with speed β.

Fourier Analysis: The convolution of lighting and BRDF in Equation 3.16 leads to a product in Fourier space,

$$F(\Omega_x, \Omega_t) = \frac{1}{|\kappa'|} L\left(\frac{\Omega_x}{\kappa'}\right) R\left(\frac{\Omega_x}{\kappa'}\right) e^{i2\pi\Omega_x\eta'/\kappa'} \delta\left(\Omega_x \frac{\gamma}{\kappa'} + \Omega_t\right). \tag{3.18}$$

The scale of κ' in the arguments of Equations 3.16 and 3.17 leads to the Fourier scale factors of $1/\kappa'$. Equation 3.18 is essentially identical to Equation 3.8 for moving objects, if we define effective velocity $a = \gamma/\kappa'$, and $G(\Omega_x) = \frac{1}{|\kappa'|}(LR)(\Omega_x/\kappa')$. In both cases, the signal is a shear in both space-time and Fourier domains.

3.3.3 Visibility and Cast Shadows

We follow previous work [Soler and Sillion, 1998; Ramamoorthi *et al.*, 2004; Mahajan *et al.*, 2007], which shows that canonical shadow effects are often described by convolutions.

We first define the binary visibility function $v(x, \theta)$ as

$$v(x, \theta) = s(\mu(x) - \theta), \tag{3.19}$$

where s is the Heaviside step function, and $\mu(x)$ is an extremal angle that defines the boundary between occluded and unoccluded regions, as shown in Figure 3.5. For simplicity, we consider only a single visibility discontinuity for each x, but a linear combination of functions can be used for general visibility [Ramamoorthi *et al.*, 2007]. Consider relative motion β between the blocker and receiver,

$$s(\mu(x - x_0(t)) - \theta) = s(\mu(x - \beta t) - \theta). \tag{3.20}$$

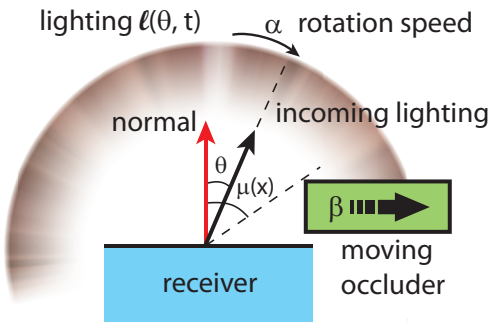

Figure 3.5: Schematic for analysis of motion-blurred shadows. The lighting can move with angular velocity α. The occluder can also move with speed β, leading to a change in the extremal angle $\mu(x)$ for visibility.

We now locally linearize $\mu(x) \approx vx$ [Ramamoorthi *et al.*, 2004]. In general, $|v| \sim \cos\mu/D$, where D is the distance to the blocker,

$$s(\mu(x-\beta t)-\theta) = s(v\cdot(x-\beta t)-\theta) = s(vx-\beta vt-\theta). \tag{3.21}$$

Finally, we define $l(\theta,t) = l(\theta-\alpha t)$ as in the BRDF case—effective values for angular velocity α can be computed for point and area lights, or environment maps. If we ignore the BRDF signal for the moment (cases with multiple surface, BRDF, and shadow signals are discussed later), we can write the reflection equation as

$$f(x,t) = \int l(\theta-\alpha t)s(vx-\beta vt-\theta)\,d\theta. \tag{3.22}$$

This has exactly the same form as Equation 3.14, only using s instead of the BRDF r, and v instead of the curvature κ'. If we similarly define $\gamma = \alpha+\beta v$, we obtain analogous to Equation 3.16,

$$f(x,t) = (l\otimes s)(vx-\gamma t). \tag{3.23}$$

This can be put in the same form as the specularity and motion case (e.g., Equation 3.17), with effective velocity $a = \gamma/v$.

Fourier Analysis: The Fourier formula in the shadow case is very similar to that for BRDF effects in Equation 3.18,

$$F(\Omega_x,\Omega_t) = \frac{1}{|v|}L\left(\frac{\Omega_x}{v}\right)S\left(\frac{\Omega_x}{v}\right)\delta\left(\Omega_x\frac{\gamma}{v}+\Omega_t\right), \tag{3.24}$$

which has an identical form to Equations 3.8 and 3.18 if we set the effective velocity $a = \gamma/\nu$ and $G(\Omega_x) = \frac{1}{|\nu|}(LS)(\Omega_x/\nu)$. One can also similarly define the Fourier transform of the motion-blurred signal H for specularity and shadows, as per Equation 3.9.

3.4 Spatial and Temporal BandLimits

We now study the spatial and temporal bandlimits. Since the mathematical form is very similar for all the visual effects in Secs. 3.3.1-3.3.3 (provided we define an effective velocity a), from now on we focus on Equations 3.8 and 3.9. Figure 3.6 illustrates the main ideas.

Time-Varying Signal $F(\Omega_x, \Omega_t)$: In general, the frequency spectrum is a wedge bounded by the minimum and maximum velocities/shears, as shown in Figure 3.6(a). From Equation 3.8, the spatial frequencies are bandlimited by $G(\Omega_x)$ so that $\Omega_x \in [-\Omega_x^{\max}, \Omega_x^{\max}]$, where $\Omega_x^{\max}$ is the highest spatial frequency in the signal g. Therefore, the temporal frequencies lie within $\Omega_t \in [-a_{\max}\Omega_x^{\max}, a_{\max}\Omega_x^{\max}]$, and the temporal frequency extent Ω_t^* is

$$\Omega_t^* = 2a_{\max}\Omega_x^{\max}. \tag{3.25}$$

According to the Nyquist theorem, we need to sample at this temporal rate to properly separate the Fourier domain replicas from sampling (Figure 3.6(c)). Otherwise, even after convolution with the low-pass camera shutter, the result would be inaccurate because of aliasing into low frequencies.[3]

Motion-Blurred Result $H(\Omega_x, \Omega_t)$: Finally, we convolve the time-varying signal with the camera shutter to obtain $h(x, t)$ and its associated Fourier transform per Equations 3.4 and 3.9. This leads to a low-pass filter along the vertical (time) axis as in Figure 3.6(b). Therefore, $\Omega_t \in [-\Omega_t^{\max}, \Omega_t^{\max}]$, where $\Omega_t^{\max}$ is the maximum frequency in the Fourier transform of the camera shutter $W(\Omega_t)$. Interestingly, the spatial frequencies are also bandlimited, since they must lie on the line $\Omega_x a + \Omega_t = 0$. Hence, it holds that:

$$\Omega_x^* = 2\frac{\Omega_t^{\max}}{a_{\min}} \tag{3.26}$$

[3] It is possible to pack the replicas slightly closer together, using a separation between Ω_t^* and $\Omega_t^*/2$. This leads to aliasing in F, but avoids aliasing in the final lower-frequency motion-blurred result H. For simplicity, we avoid that discussion here, which only corresponds to a factor of at most 2. The sheared filter in Sec. 3.5 focuses primarily on non-axis-aligned reconstruction, but does also exploit this small factor.

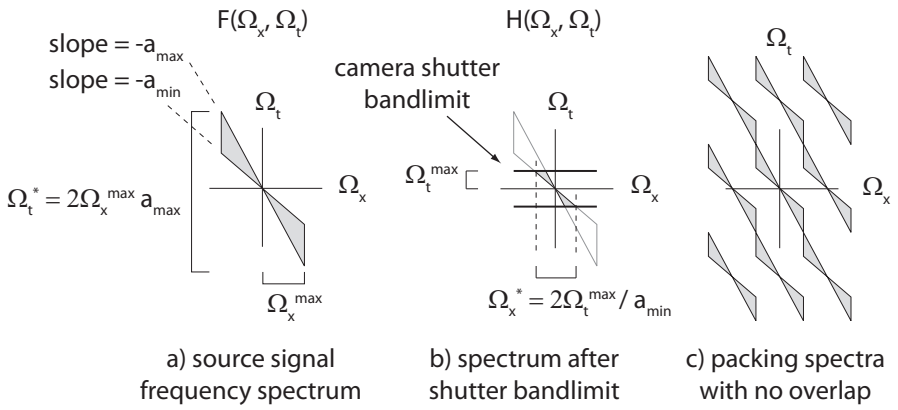

Figure 3.6: (a) Frequency spectrum of source signal $F(\Omega_x, \Omega_t)$ in space and time (Ω_x and Ω_t). We also mark the highest spatial frequency $\Omega_x^{\max}$, and the highest temporal frequency Ω_t^*, determined by the maximum velocity/shear $a_{\max}$. (b) The signal is bandlimited in time based on the camera shutter to temporal frequencies less than $\Omega_t^{\max}$. For images with medium to large amounts of motion blur, the spatial frequencies are also correspondingly filtered to Ω_x^*, depending on the minimum velocity $a_{\min}$. (c) Sampling introduces replicas of the base spectrum F. To achieve a low sampling rate we must bring the spectra as close as possible without aliasing.

Not surprisingly, the spatial frequency content is much lower due to motion blur.

The above result needs a small modification in the quasi-static case. If the velocity $a_{\min}$ is sufficiently small, the temporal frequencies Ω_t^* in Equation 3.25 is less than the filtering effect of the shutter response. Therefore, the motion blur filter has minimal impact on the signal (much as motion blur does not affect a static scene). In this case, we simply have $\Omega_x^* = 2\Omega_x^{\max}$. In general,

$$\Omega_x^* = 2 \min\left(\frac{\Omega_t^{\max}}{a_{\min}}, \Omega_x^{\max}\right). \tag{3.27}$$

Sampling Theorem: Sampling the time-varying signal f leads to replicas of $F(\Omega_x, \Omega_t)$ in the Fourier domain as shown in Figure 3.6(c). We must separate the replicas enough to avoid overlap or aliasing in reconstructing the motion-blurred signal H. Figure 3.7 shows this idea in both the space-time and frequency domains.

The exact separation of replicas needed depends on the reconstruction filter, and for now we consider a standard rectangular axis-aligned filter in the Fourier domain (Figures 3.7(A,B)). It is instructive to consider the product of spatial and temporal frequency ranges. By the Nyquist theo-

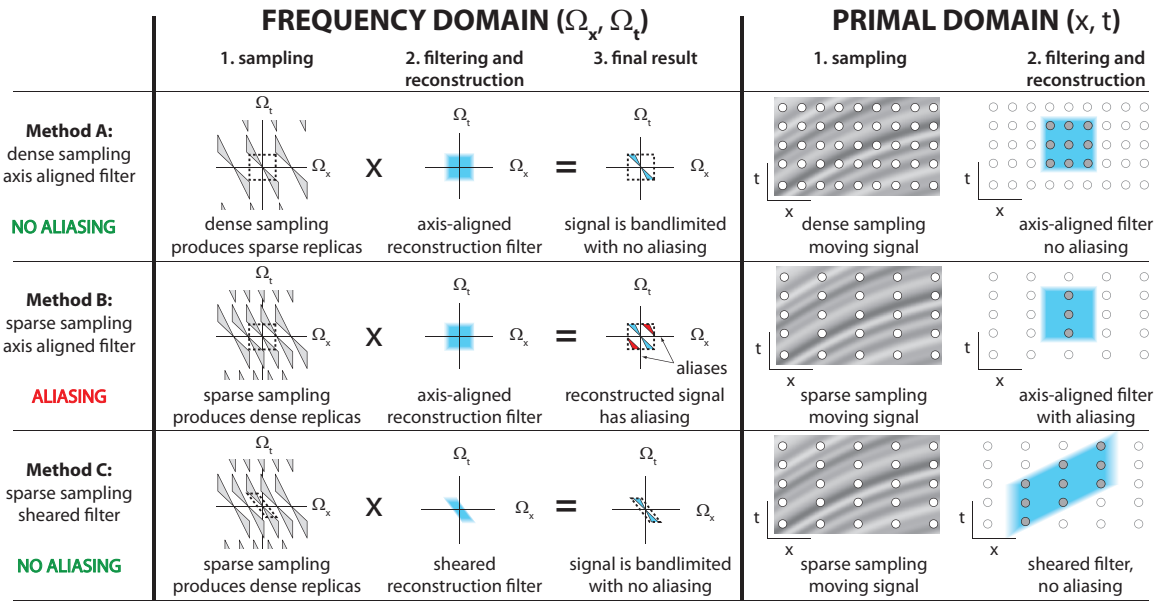

Figure 3.7: 1. Sampling in the primal domain creates replicas in the frequency domain. The denser the sampling the further apart the Fourier-domain replicas are spaced. 2. The Fourier transform of the spatial reconstruction filter bandlimits and reconstructs the signal for display. 3. Filtering the samples in the primal domain is equivalent to multiplying the Fourier transforms of steps 1 and 2. If replicas overlap with the Fourier domain reconstruction filter, the final result would contain spurious frequencies (aliasing). In Method/Row A, a relatively dense sampling is used in space and time to separate the Fourier domain replicas. Method/Row B shows that a sparser sampling rate with an axis-aligned filter leads to aliasing. Method/Row C shows that using a sheared reconstruction filter, we can reconstruct a correct image using a sparse sampling rate.

rem, the number of samples needed is also proportional to these bandlimits. For simplicity, we use Equations 3.25 and 3.26 (ignoring for now the special case in Equation 3.27),

$$\Omega^* = (\Omega_x^*)(\Omega_t^*) = 4\frac{a_{\max}}{a_{\min}}\Omega_x^{\max}\Omega_t^{\max}. \tag{3.28}$$

In the limit where we have a uniform velocity with $a_{\max} = a_{\min}$, the space-time sampling rate becomes $\Omega^* = 4\Omega_x^{\max}\Omega_t^{\max}$, *independent of the velocity a*. This indicates that as the motion a gets faster, the needed temporal sampling rate $\Omega_t^* = 2a\Omega_x^{\max}$ increases, but the spatial sampling rate needed $(2/a)\Omega_t^{\max}$ decreases correspondingly due to the spatial filtering or blurring of moving objects and texture.

3.5 Sheared Reconstruction Filter

We have taken a first step in finding spatial and temporal bandlimits. These bandlimits can directly be used to accelerate motion-blur rendering by adaptive sampling. We may sparsely sample in space and time according to Equations 3.25 and 3.27, then scale the standard one pixel wide axis-aligned reconstruction (spatial antialiasing and temporal shutter response) filter to reconstruct the sparse data.

However, Figure 3.6(c) and Figure 3.7(A) show the corresponding packing of replicas in Fourier space and illustrate that they still have a lot of free space between them. We seek to achieve sparse sampling, which means bringing the replicas tighter together. Packing replicas too tightly while using an axis-aligned filter will cause aliasing (Figure 3.7(B)). We now introduce a sheared filter that allows for much tighter packing of replicas and lower sampling densities (Sec. 3.5.1). It is based on two important observations: the shape of the spectrum is slanted and is best matched by a sheared filter, and we need to prevent overlap only in the central part of the wedge that is within the shutter bandwidth. Finally, we take a critical step towards a practical algorithm by deriving the sheared filter in the primal space-time domain (Sec. 3.5.2). This is done simply by appropriately transforming a standard axis-aligned filter (Figure 3.8(d)).

3.5.1 Sheared Filter and Sampling

As can be seen in Figure 3.8(a), we are really interested in the central wedge of frequencies for $H(\Omega_x, \Omega_t)$. Given the spectrum's wedge shape, it is best to separate the central spectrum from the replicas by using a non-axis-aligned parallelogram as the reconstruction filter, as shown in Figure 3.7(C) and Figure 3.8(a). Figures 3.8(b) and (c) show two ways of tightly packing the replicas, which we discuss next. Note that the frequency spectra for $F(\Omega_x, \Omega_t)$ do in fact alias in this reconstruction (shown in red). However, the final low-pass filtered form from motion blur $H(\Omega_x, \Omega_t)$ does not. The amount of free space in the Fourier domain is considerably reduced, compared to Figure 3.6(c), enabling lower sampling rates.

Intuitive Sampling Strategy 1—Pack Space Replicas First: The first sampling method we examine packs replicas tightly in Ω_x, then in Ω_t, as shown in Figure 3.8(b1-b2). This technique more closely follows Sec. 3.4 and is useful for developing our intuition for the benefits obtained

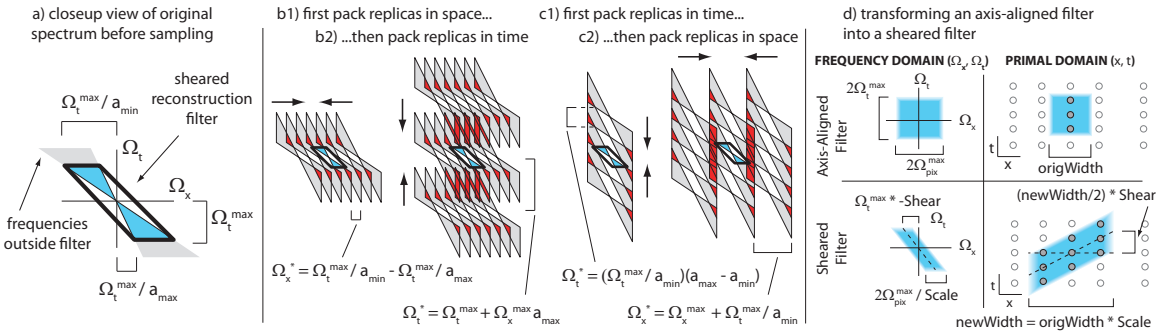

Figure 3.8: (a) Zoomed-in view of the frequency wedge and the sheared reconstruction filter. The distances between the Ω_t axis and the near and far points of the sheared filter are shown. Only the blue frequency content inside of the sheared reconstruction filter will be output for display. (b1) and (b2) show packing of replicas as tightly as possible first in space, then in time. (c1) and (c2) show packing of replicas as tightly as possible first in time, then in space. (d) Transforming an axis-aligned filter into a sheared filter. (d Top) We start with any standard axis-aligned filter in the frequency and primal domains. (d Bottom) We then consider the scale and shear in the frequency domain, applying the opposite scale and shear in the space-time domain (Equations 3.33 and 3.34).

from the sheared filter. Our practical algorithm uses the second sampling strategy, developed next, of packing the time replicas first.

First we compute the spatial sampling rate or bandlimit Ω_x^*. From simple trigonometry, Figure 3.8(b1), and Equation 3.26,

$$\Omega_x^* = \Omega_t^{\max}\left(\frac{1}{a_{\min}} - \frac{1}{a_{\max}}\right), \tag{3.29}$$

which is a significantly lower frequency (and hence sampling rate) than in Equation 3.26 when $a_{\min}$ is close to $a_{\max}$. Indeed, for nearly uniform velocity $a_{\min} \approx a_{\max}$, we obtain $\Omega_x^* \to 0$ (assuming the reconstruction filter extends infinitely far in space-time). As seen in Figure 3.8(b2), we must next pack the temporal replicas to determine Ω_t^*, and can then compute an overall bandlimit, $\Omega^* = \Omega_x^*\Omega_t^*$.

Practical Sampling Strategy 2—Pack Time Replicas First: It is also possible to proceed the other way, first packing along the temporal axis and then along the spatial axis (an illustration is in Figures 3.8(c1) and (c2)). This formulation gives essentially the same overall sampling rate Ω^* as the first sampling strategy above, and has advantages in practical applications where we usually

want the spatial samples denser than the temporal samples—with very few time samples required for high-quality motion blur. Having dense spatial sampling makes it easier to find high-frequency spatial discontinuities that can be caused by static occluders. Note that the sheared filter itself is the same in both cases.

From the geometry of Figures 3.8(c1) and (c2), we can derive

$$\Omega_t^* = \Omega_t^{\max}\left(\frac{a_{\max}}{a_{\min}} - 1\right) \qquad \Omega_x^* = \Omega_x^{\max} + \frac{\Omega_t^{\max}}{a_{\min}}, \tag{3.30}$$

with the product being given by

$$\Omega^* = \Omega_x^*\Omega_t^* = \left(\frac{a_{\max}}{a_{\min}} - 1\right)\Omega_x^{\max}\Omega_t^{\max} + \left(\frac{a_{\max}}{a_{\min}} - 1\right)\frac{(\Omega_t^{\max})^2}{a_{\min}}, \tag{3.31}$$

which is also proportional to the total number of samples needed.

With motion greater than 1 pixel per frame $a_{\min}\Omega_x^{\max} > \Omega_t^{\max}$, and the first term above will be dominant. Equation 3.31 is now

$$\boxed{\Omega^* \approx \left(\frac{a_{\max}}{a_{\min}} - 1\right)\Omega_x^{\max}\Omega_t^{\max}.} \tag{3.32}$$

The crucial benefit over Equation 3.28 is the use of $a_{\max}/a_{\min} - 1$ instead of $a_{\max}/a_{\min}$. If maximum and minimum velocities at a pixel for a given frame are similar, sheared reconstruction can be significantly more efficient. On the other hand, for pixels with significant occlusions or large velocity changes so $a_{\max}/a_{\min} \gg 1$, sheared filtering does not provide a large benefit over an optimally sized rectilinear filter.

3.5.2 Sheared Filter in Primal Domain

So far, we have considered frequency analysis, but practical rendering algorithms do not directly compute frequency spectra. Fortunately, we can create a sheared reconstruction filter directly in the space-time domain. We simply apply the corresponding transforms to any standard axis-aligned filter composed of a spatial antialiasing filter and the temporal shutter response (see Figure 3.8(d)).

Specifically, the original axis-aligned filter has some spatial bandlimit $\Omega_{\text{pix}}^{\max}$ (≈ 0.5 wavelengths per pixel) that we scaled (Figure 3.8(a)) to a diameter of $\Omega_t^{\max}(1/a_{\min} - 1/a_{\max})$. Based on Fourier theory, we must scale by the inverse in the primal domain:

$$\text{Scale} = \left(\frac{\Omega_t^{\max}}{2\Omega_{\text{pix}}^{\max}}\left(\frac{1}{a_{\min}} - \frac{1}{a_{\max}}\right)\right)^{-1}. \tag{3.33}$$

The shear of the filter in the Fourier domain is based on the filter intercepts $\Omega_t^{\max}/a_{\min}$ and $\Omega_t^{\max}/a_{\max}$ (Figure 3.8(a)). In the Fourier domain the shear in Ω_x per unit Ω_t is the average of $-1/a_{\min}$ and $-1/a_{\max}$. Again, based on Fourier theory, we need to apply the opposite shear in the primal domain (shearing in time per unit x):

$$\text{Shear} = \frac{1}{2}\left(\frac{1}{a_{\max}} + \frac{1}{a_{\min}}\right). \tag{3.34}$$

The shear corresponds to the direction of average motion in the space-time domain, with the filter "following the motion." The scale depends on the complexity of motion—the filter is larger (with a corresponding low sampling rate), the closer $a_{\min}$ and $a_{\max}$ are.

3.6 Algorithm and Results

We describe one approach for using these theoretical results—a simple practical method that uses sheared reconstruction filters to greatly reduce sample counts. While the analysis is in the Fourier domain, the actual practical algorithm need not explicitly compute spectra, and operates directly on space-time image samples.

Our method involves a three-stage process, shown in Figure 3.9. First, we do an initial sparse sampling to compute the effective velocities $[a_{\min}, a_{\max}]$ and frequency bounds $\Omega_x^{\max}$ (Sec. 3.6.1). Second, we determine a single sheared reconstruction filter for each pixel, along with spatial and temporal sampling densities Ω_x^* and Ω_t^* (Sec. 3.6.2). Our third stage involves a final round of sampling, and for each pixel we do a single application of the computed sheared filter to reconstruct the pixel's final color (Sec. 3.6.3). There are a few additional special cases and implementation details in Appendix A.

Our sheared reconstruction filter uses the sampling formulation in Figure 3.8(c). This method samples sparsely in time (packing the replicas tightly in the temporal frequency domain), but densely in space. For frequencies $\Omega_t^{\max}$ and $\Omega_x^{\max}$ in practical images, we sample every pixel of the frame at least once, but with many fewer samples than are required for the same quality output using standard Monte Carlo sampling. The source code for our program can be found at `http://www.cs.columbia.edu/cg/mb/`, and we include a Renderman shader that computes the relevant velocities and frequency bounds in our supplementary material.

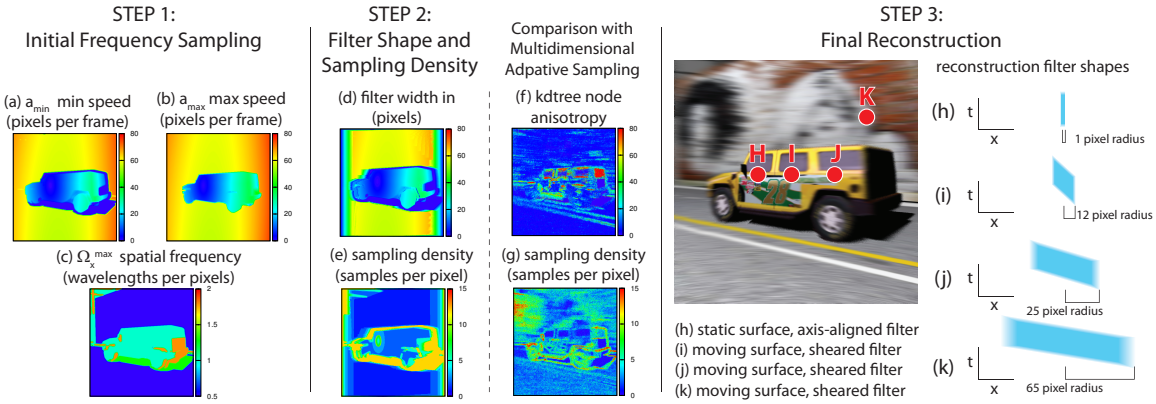

Figure 3.9: Illustration of our three-stage algorithm. The scene is shown in Figure 3.1. In Step 1, we do an initial sampling to compute velocities $[a_{\min}, a_{\max}]$ and maximum spatial frequencies $\Omega_x^{\max}$. These are visualized in (a,b) and (c) respectively. Note that areas of the image where $a_{\min} \approx a_{\max}$ will require very low sample counts as seen in (e). In Step 2, we determine the sheared filter shape, and the sampling densities at each pixel. The filter radius is shown in (d); note the very large filters for the background. (e) visualizes the total number of samples Ω^* at a pixel, and is seen to be high only in regions where the motion is non-uniform, such as the ends of the car (occlusion) and shadows moving over a textured surface (multiple signals). The edges of the image also require higher sampling. For areas of uniform motion a low sample count (often close to 1 per pixel) suffices. We also compare with MDAS, which adapts to the silhouette edges of the car, but has a more uniform sample distribution, and much less anisotropy in the filters. Finally, Step 3 shows the sheared filter shape for representative pixels (the image resolution for this scene is 512x512). In general (i,j,k), sheared filters of different widths are used, based on the speed of motion. For the special case of a static surface (sharp with minimal motion blur) like (h), our method gracefully reduces to a 1-pixel axis-aligned filter as required.

3.6.1 Stage 1: Velocity/Frequency Bounds

We start by sparsely computing local frequency information at each pixel. We sample the scene with N samples per pixel (our implementation uses $N = 2$). This cost is minimal, since it is less than what we would need to render a single antialiased image of a static scene. Moreover, only velocity information is required from the samples; all shading is computed in a separate pass in step 3. At each sample we compute the image space signal direction, velocity bounds $[a_{\min}, a_{\max}]$, and signal

bandlimit $\Omega_x^{\max}$.

Computing Effective Velocities and Bandlimits: We use surface shaders to compute the image-space velocities within the renderer for each of the three key signals: object motion, BRDF shading, and shadows.

For object motion, the effective velocity a is simply the instantaneous screen-space velocity for the surface (including projection and perspective effects). Assuming the surfaces use mip-mapping to bandlimit texture frequencies, we simply set $\Omega_x^{\max}$ to a maximum frequency of one wavelength per pixel.

The velocities and bandlimits for BRDF shading and shadows are based on Equations 3.18 and 3.24 respectively,

$$a_{shading} = \frac{\gamma}{\kappa'} = \frac{\alpha\kappa' + \beta}{\kappa'} \qquad \Omega_{x,shading}^{\max} = \min\left(L^{\max}\kappa', R^{\max}\kappa'\right) \tag{3.35}$$

$$a_{shadow} = \frac{\gamma}{\nu} = \frac{\alpha\nu + \beta}{\nu} \qquad \Omega_{x,shadow}^{\max} = \min\left(L^{\max}\nu, S^{\max}\nu\right), \tag{3.36}$$

where α is the angular velocity of the lighting, β is the linear velocity of the object or blocker, $\kappa' = 2\kappa$ is twice the screen-space curvature, and $|\ \nu\ | \sim \cos\mu/D$, where D is the distance to the blocker. These values can be calculated entirely inside the shader if the programmable shading language supports shader derivatives. For instance, occluder speed can be estimated by querying the velocity of any surface that blocks a shadow ray. Details on units and computing frequency bandlimits are in Appendix A.

Velocity and Frequency Bounds: After initial sampling, we compute a frequency bound for each pixel that captures frequency information across the entire frame. $[a_{\min}, a_{\max}]$ are simply the minimum and maximum velocities of all samples inside the pixel. Similarly, $\Omega_x^{\max}$ is simply the maximum frequency of all samples. Our implementation sparsely samples frequency information, so we use advection to gather nearby frequency samples that may overlap with the current pixel at a different moment in time. Values for the scene in Figure 3.1 are shown in Figure 3.9(a,b,c).

Multiple Signals: Our theory focuses on the case where all signals (object motion, lighting, shadows) that affect a given pixel translate along a single direction. Therefore, in the special case that any of the frequency samples at a pixel has a direction vector that differs greatly in angle from

the others, we conservatively bound the frequencies by setting $a_{\min}$ for that pixel to 0 (the computations of $a_{\max}$ and $\Omega_x^{\max}$ are unaffected). When we are calculating a single frequency sample, a similar adjustment is occasionally required when multiple signals (more than one of surface texture, BRDF shading and shadows) have significant amplitude and frequency. Details for this case are discussed in Appendices A and D.

3.6.2 Stage 2: Sheared Filters and Sampling Rates

Based on the velocities and the frequency information from our initial sampling, we compute sheared reconstruction filters and sampling densities for each pixel. These are visualized in Figures 3.9(d) and (e) for an example scene. To derive properties for sheared filters in Sec. 3.5, we first determined the shape of the filter in Fourier space, and then determined how tightly we could pack replicas. Similarly, in our practical implementation, for each pixel, we first compute the widest possible reconstruction filter, and then determine the lowest possible sampling rate that avoids aliasing.

Computing the Shape of the Sheared Filter: To create an optimal sheared filter we use Equations 3.33 and 3.34 to scale and shear the user's preferred axis-aligned reconstruction filter. In image space, both the scale and shear operate strictly along the direction of motion, and the axis perpendicular to motion is unaffected.

To provide intuition, consider the case of nearly constant velocity where $a_{\min} \approx a_{\max} = a$. In this case, the space-time shear (Equation 3.34) is just $1/a$, as expected. The scale tends to infinity (since $1/a_{\min} \approx 1/a_{\max}$ in Equation 3.33)—we can use a very wide sheared filter in this case, since the velocity is constant. Indeed, very wide filters are used in Figure 3.9(d) for much of the car, and especially the background, which have nearly uniform velocity.

A special case arises for slow-moving signals (as indicated by Equation 3.27), and its handling is discussed in Appendix A. The final computation of filter widths is also complicated by the fact that once we select a filter size, we may include incompatible pixels inside the filter. For instance, in the example above, if $a_{\min} \approx a_{\max}$, the scale should be very large, but this wide filter may contain other pixels with a greater range of $[a_{\min}, a_{\max}]$. Computation of a final filter shape may require an iterative process where we eventually use a smaller filter size. Details are given in Appendix A.

Computing Sample Densities: In most cases, we can compute the sampling rates Ω_x^* and Ω_t^* directly from Equation 3.30. For scenes with moderate complexity, Ω_x^* and Ω_t^* usually require at least one sample per pixel per frame. To compute the sampling rate for a 2D image we must also include frequencies along the spatial axis perpendicular to motion. These frequencies should have little or no velocity, so we use the spatial bandlimit for static signals with an axis-aligned filter, $(2\Omega_x^{\max})$ (Equation 3.27):

$$\text{Pixel Samples} = \Omega^* = (\Omega_x^*)(\Omega_t^*)(2\Omega_x^{\max}). \tag{3.37}$$

In practice, we also cap the maximum number of samples for a pixel (usually to 4× the average number of samples per pixel).

The number of samples depends on both spatial complexity and motion complexity (how much it differs from uniform velocity). More samples will be given both where the motion varies ($a_{\max}/a_{\min}$ is large), and also where there are high spatial frequencies (complex textures or shadows/highlights with high $\Omega_x^{\max}$). Equation 3.37 provides a natural way to allocate samples to different visual effects.

3.6.3 Stage 3: Final Sampling and Reconstruction

The final sampling density for a pixel is simply the maximum density required by reconstruction filters that overlap that pixel. For sample placement across both space and time within a pixel, we use a 3D Halton sequence [Halton, 1960]. For low sampling densities (less than 8 samples per pixel), we do not jitter, and we mirror the Halton sequence at odd pixels. For higher sampling rates, we add a jittered offset. We limit our sample points to lie inside the shutter bounds to show compatibility with traditional rendering pipelines. Future implementations could find gains by sampling across time and sharing samples between frames of an animation.

We send the computed space-time sample locations to the renderer for processing, and read back the shaded results for each sample. Finally, we reconstruct the motion-blurred image using the sheared filters computed in step 2. Note that we do only one reconstruction per pixel, with a single application of the sheared filter for that pixel—this filter combines reconstruction, spatial antialiasing, and motion-blur integration over time. Figure 3.9(h,i,j,k) shows these filters for some representative image pixels. In most cases, these are sheared, with the size of the filter determined by the complexity of the motion (or getting clipped by the camera shutter bounds). In the special

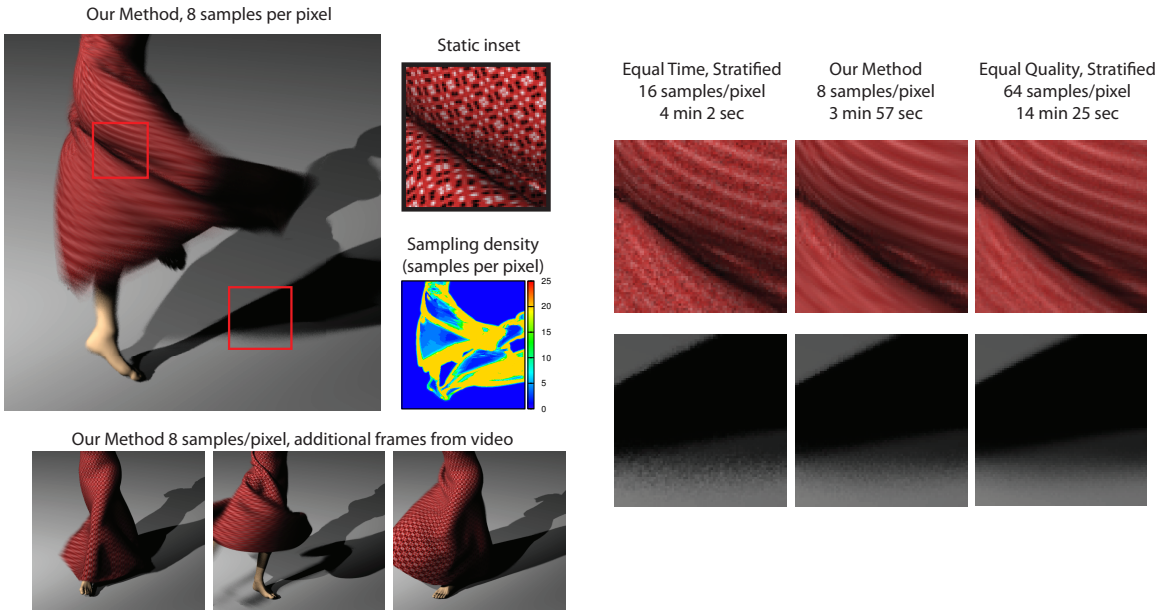

Figure 3.10: A scene of a ballerina with fast and varying motions. The dress is deforming as the dancer kicks, causing many non-uniform motions that stress the abilities of any motion-blur algorithm. Note that we focus samples on the most difficult areas: occlusion of the bottom of the dress, overlapping shadows, and areas of the dress that come out of shadow as the dancer rotates. It is clear from the insets that our method does not blur frequency content perpendicular to the direction of motion.

case of static regions (Figure 3.9(h)), the filter reduces to an axis-aligned filter of 1 pixel width, as it must.

3.6.4 Results

We modified the Pixie renderer [Arikan, 2009] to use our space-time sample placement algorithm for ray tracing color information in stage 3. We now describe the results obtained by our algorithm and compare to a stratified sampling Monte Carlo approach using jittering, and to the recent MDAS technique [Hachisuka *et al.*, 2008]. All images were rendered at a resolution of 512×512 with a single core on a 1.8GHz Core 2 Duo processor.

Scenes: Figures 3.1, 3.9, and 3.13 show a scene with a rotating camera, and a moving object (the car). Following a common photographic technique, the camera follows the car's motion to keep

it sharp (but not completely stationary), while the other areas have considerable motion blur. The lighting comes from a moderately distant point light source. Figure 3.10 is intended to be a stress test of our system, with a deforming dress and multiple non-rigid motions. The dress has a high-frequency texture, which leads to different patterns depending on the direction of motion. Moreover, we have mid-field lighting from two point light sources that cast overlapping moving shadows, and surfaces moving in and out of shadow as well as self-occluding.

Finally, Figure 3.11 shows an example with a motion-blurred glossy reflection of the moving background (the teapot is shiny with Phong exponent 100). This scene demonstrates that we can handle curved motion paths and global illumination effects. (Note that calculating motion-blurred global illumination effects requires that the shader can calculate the movement of indirect lighting.) The scene also has motion-blurred reflections of near sources, and sharp shadows on moving surfaces (shadow of the spout).

Evaluation and Comparison to Stratified Monte Carlo: In Figure 3.1b, stratified Monte Carlo sampling with 4 samples/pixel leads to considerable noise, especially in the motion-blurred areas of the background. In contrast, our method (Figures 3.1(a,d)) produces a high-quality result even at this very low sample count, with only minimal noise at difficult shadow boundaries. It would require at least an order of magnitude more samples to match it with direct Monte Carlo. Similar conclusions can be drawn from Figure 3.10. Our implementation properly computes motion blur and preserves high frequencies on the dress perpendicular to the direction of motion. Our method can also be used directly to produce motion-blurred sequences, as shown in the supplementary animation (stills in Figure 3.10). Note also that proper motion-blurred lighting and shadows are computed in Figures 3.1, 3.10 and 3.11.

Finally, the sheared filter can also be used by itself as a light-weight addition with standard Monte Carlo sampling and rendering (Figure 3.12). Applying sheared reconstruction to the standard (non-adaptive) stratified sampling pattern dramatically improves areas of uniform motion like the dress (Figure 3.12(b)). Of course, also using our adaptive sampling enables more samples at the dress silhouette, leading to a reduction in noise (Figure 3.12(c)).

Sampling Densities and Filter Widths: The filter widths and sampling densities for our method are shown in Figures 3.9(d,e) and in Figure 3.10. Interestingly, in the car scene, very few samples are

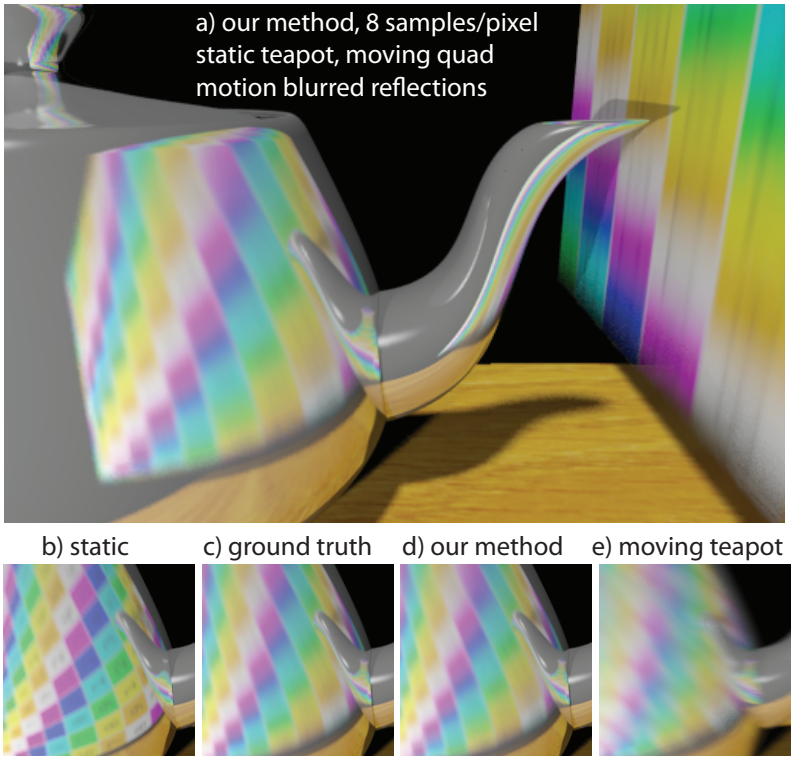

Figure 3.11: Shiny teapot (Phong exponent 100) with glossy reflections, rendered with an average of 8 samples/pixel. (a) Full image. The insets below show (b) static image, (c) ground truth, (d) our method, (e) our method where the teapot is also moving.

needed for much of the image, where motion although fast, is almost uniform. Note in Figure 3.9(e) that an average of close to only 1 sample per pixel suffices on much of the car body, background painting, and road. Samples are thus concentrated near the silhouette boundaries and edges of cast shadows, where the motions are very complex (also the case in Figure 3.10). The widths of the sheared reconstruction filter in Figure 3.9(d) clearly show how large our sheared filter can be in regions of nearly uniform motion such as the background, and much of the car body. Moreover, our method can gracefully fall back to axis-aligned reconstruction with 1 pixel-wide filters, in difficult or nearly static areas of the car (Figure 3.9h).

Comparison to Multi-Dimensional Adaptive Sampling: For comparisons on the car scene, we directly used MDAS software which is a plugin to the PBRT renderer [Pharr and Humphreys, 2004]. We did not do this comparison for the ballerina because the base PBRT renderer does not currently

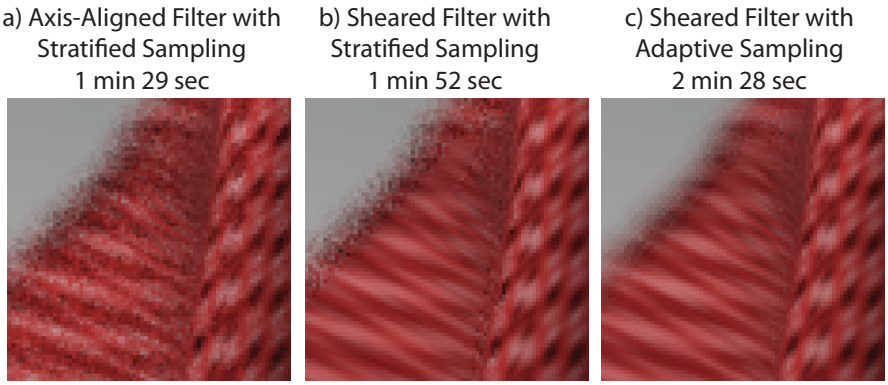

Figure 3.12: An inset from the left video frame in Figure 3.10. (a) Standard Monte Carlo stratified sampling exhibits fairly uniform noise. (b) Monte Carlo stratified sampling, combined with our sheared space-time filter. Noise has been reduced in areas of uniform motion. (c) Our adaptive method samples densely around difficult regions so that the noise in (b) at the silhouette of the dress is reduced.

support deforming meshes.

First, consider the sampling rates in Figure 3.9. MDAS (Figure 3.9g) has a much more uniform sampling density, with only a little more importance given to the edges of the car. Our technique is able to sample many motions more sparsely, allowing it to properly focus on difficult areas. Similarly, while MDAS takes anisotropy into account (Figure 3.9f), their kD-tree cells never shear and do not scale as much as our sheared filter does. In contrast, our reconstruction filter can rotate in any direction in image space and is sheared in time to better match the direction of motion.

Figure 3.13 shows what happens as we increase sample count from an average of 2 per pixel to 8 per pixel (4 samples/pixel is shown in Figure 3.1). At low sample counts (top row of Figure 3.13), we are already nearly perfect in the background and ground plane because of our wide sheared reconstruction filter, while MDAS is very noisy. On the other hand, MDAS is somewhat less noisy near shadow boundaries, since our method often needs to fall back to axis-aligned reconstruction for these complex motions. This is because MDAS discovers areas of coherence through numerical measurements, whereas we rely on conservative frequency bounds. At moderate sample counts (bottom row of Figure 3.13), both methods are close to converged, but MDAS still has a little noise on the background mural.

Timings and Overheads: At 8 samples per pixel, the car scene took our algorithm a total of 3 min, 8 sec, while the ballerina took 3 min, 57 sec. The time for reconstruction using sheared filters is about half the total for the car, and one-third for the ballerina. As one point of comparison, we are competitive with MDAS (a total of 3 min, 23 sec on the car, with the overhead for reconstruction being about one third of total running time). Moreover, MDAS has significant memory overheads. In our tests MDAS required 1GB of memory to render a 512×512 image with 8 samples per pixel. Our memory overhead is fairly low. To store all samples in memory for a 512×512 image requires 36MB during sparse sampling ($N = 2$), and 32MB during final sampling (using 8 samples per pixel).

In scenes with more complex shading, our method has much lower overhead (only 15% of the 25 minute rendering time for Figure 3.11). Even with very simple materials, such as Figure 3.10, we are only twice as slow as standard Monte Carlo (some of this is from overhead, and an equal amount from the fact that our ray-tracing phase focuses on difficult regions by design, which take more time). A visual equal quality comparison shows a net wall-clock speedup of more than 3.5× by our method in Figure 3.10.

Limitations: Our current implementation uses a line segment as the anisotropic filter shape. For highly-curved motion paths this may lead to over-blurring. Note that we do test all samples inside a filter to measure non-linearities in speed (divergence of $a_{\min}$ and $a_{\max}$) and direction. In particular, we set $a_{\min}$ to 0 if any direction differs significantly in angle from the others. These tests will cause our method to reduce filter sizes and increase sampling rates near areas of highly curved motion. With more complete motion information from the renderer, future implementations should be able to filter along curved paths.

Our method can resort to axis-aligned reconstruction and dense sampling in difficult cases, such as a shadow moving over a static textured surface. However, these areas are also difficult in most other motion-blur techniques. Moreover, our method can quickly converge on simpler parts of the scene, and then focus almost all of its sample budget on the difficult regions.

Similar to any practical rendering application, our system makes approximations during reconstruction that can lead to aliasing, (such as using a windowed gaussian filter instead of an infinitely wide sinc filter). Because wide filters overlap and share information, adjacent pixels employing

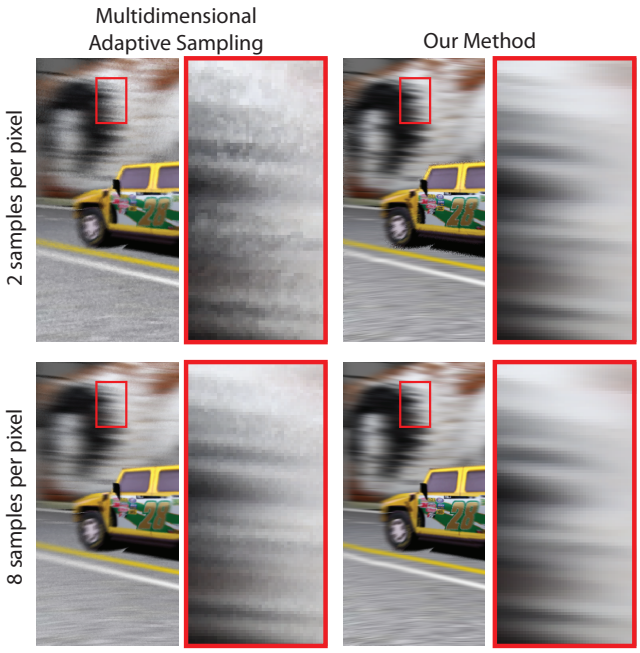

Figure 3.13: We show further results for our method and MDAS with 2 samples per pixel and 8 samples per pixel. For 2 samples per pixel our method does well for areas with uniform motion (in fact many areas use one sample per pixel and are in their final state), but does poorly for pixels where we detect a large difference between $a_{\min}$ and $a_{\max}$. At 8 samples per pixel our method devotes all of the new samples to the difficult areas, and many places such as the shadows improve dramatically. MDAS by comparison starts out with fairly uniform noise, and then improves evenly over all areas of the image. The insets show that in areas with fairly uniform motion our method computes high-quality results at extremely low sample counts.

wide filters will share aliasing data. The net visual effect is usually a small distortion in the image relative to ground truth. Note that these aliases are low frequency, and are visually hard to detect, nor do they cause temporal artifacts as seen in the video. Because of this, our method will often achieve an excellent visual match with ground truth, but a relatively high measure of mean squared error.

Like most adaptive techniques, our implementation does an initial sampling and can therefore miss information from very fast-moving or thin objects. In the future, we could try using space-time bounding boxes to more conservatively bound occlusions.

3.7 Discussion

We have presented a frequency-space analysis of time-varying signals and images, as needed for motion blur. We have shown that many motion-blur effects can be analyzed as shears in the space-time and Fourier domains. Our analysis gives precise guidelines for an intuitive observation: integrating over the shutter blurs a moving signal in the spatial dimension. This analysis in turn leads to a novel sheared reconstruction filter that again formalizes an intuitive notion: a moving sample should contribute not just to the

Chapter 4

Shadows from Planar Lights

4.1 Introduction

In the previous chapter we computed how to reduce the number of total shading points for motion blur calculations, but we still required that each shading point emit a large number of shadow rays. In this chapter we will greatly reduce the number of shadow rays required for computing soft shadows in a static scene.

Many algorithms have been used to generate soft shadows cast by area lights, but Monte Carlo sampling is the method of choice for production rendering due to its simplicity and widespread use for offline rendering. Unfortunately, when computing shadows from intricate geometry (see Figure 4.1), the (binary) visibility function on the light source is complex and high-frequency. While the *integral* of this function can still be relatively smooth, the Monte Carlo point samples (shadow rays) have high variance and considerable noise persists even for large sample counts (Figure 4.1), requiring the use of a prohibitive number of shadow rays. This is frustrating because the resulting shadows can be smooth and simple, despite the complex and costly calculation that went into them.

We propose to efficiently sample and filter the 4D shadow light field from a complex occluder, thanks to a new analysis of shadow sampling and reconstruction. We introduce a new 4D shadow light field cache that allows for integration and reuse across pixels. The sampling of our method is driven by a frequency analysis at the visible receivers, and a new sheared filter allows neighboring receiver points to share data and reduce sample count. Our specific contributions include:

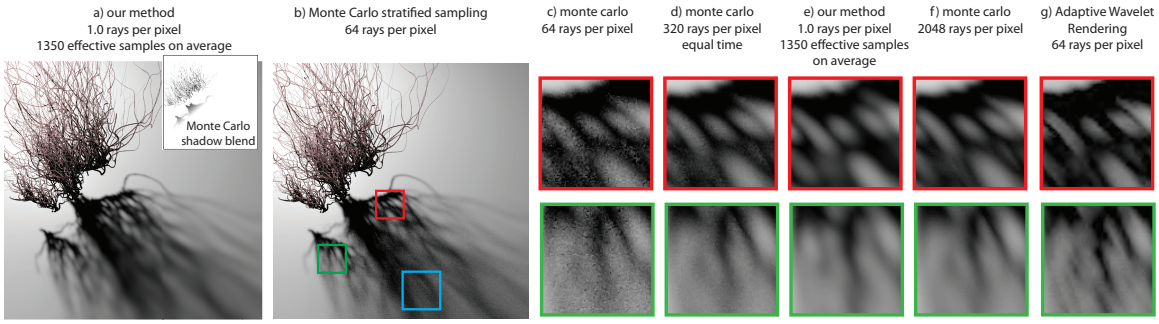

Figure 4.1: (a) Our method casting 1 shadow ray per pixel. Our wide filter gives an average effective sampling rate of 1350 samples for every pixel that is partially occluded. We use brute force Monte Carlo ray tracing for self-shadowing and near-field occlusion, and then blend into our results for mid- and far-field occlusion, as shown in the inset. (b) Monte Carlo stratified sampling with 64 samples has large amounts of noise due to the complex geometry, also shown in the insets in (c). (d) Even with 320 samples the shadow still has visible noise. (e) Our method using 1 ray per pixel. By sharing samples between neighboring receiver points, we obtain an effective sampling rate of 1350 samples per pixel. (f) Ground truth, generated using 2048 shadow rays per pixel. (g) Comparison to Adaptive Wavelet Rendering with 64 samples per pixel. Visible artifacts can be seen due to the high variance of the shadow samples. (blue box) Our method exhibits some overblurring in the area highlighted with the blue box. See §4.7 and Figure 4.15 for more details.

Frequency Analysis of Shadow Signal We first show that only a narrow wedge of the Fourier spectrum usually has significant amplitude if the depth range of the blockers is limited. Complex occluders with a bounded depth range are common in cases like dense foliage or irregular arrays of blockers. Our analysis subsumes and extends convolution soft shadows in parallel planes [Soler and Sillion, 1998].

Sheared Filters for Shadows We introduce a new reconstruction filter that is sheared in the receiver-light domain, and enables very sparse sampling since visibility samples can be shared among adjacent pixels. We generalize previous work on sheared filters in other contexts [Chai *et al.*, 2000] to irregular reconstruction problems—the depths of the receiver points may vary, which in turn causes the bundle of rays that we integrate over to have different shapes. We first design the sheared filter in the native coordinate system of the receiver point, and then transform to a parame-

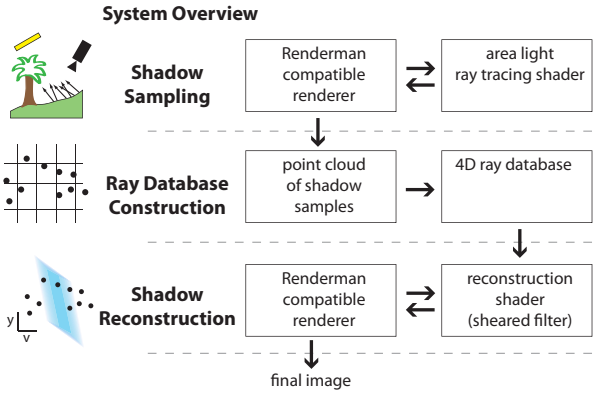

Figure 4.2: A flow chart showing the architecture and data flow in our system. (Shadow Sampling) Sparsely sample the light field using shadow rays and write out each ray result to disk. (Database Creation) Read in the ray samples and create a ray database. (Shadow Reconstruction) Query the ray database and use sheared filters to reconstruct the shadow.

terization that is agnostic to the receiver point.

Practical Algorithm An overview of our method can be seen in Figure 4.2. We first sparsely sample the occlusion light field by shooting a small number of shadow rays. We then store all ray samples in a ray database. Finally, at each receiver pixel, we use our frequency analysis to calculate the best filter shape for the receiver, and filter over the samples in our ray database. Our analysis shows that we can often use a wide filter across the shadow light field, effectively reusing rays cast from nearby receiver points.

4.2 Related Work

As a full review of shadow algorithms is beyond the scope of this thesis, we focus on approaches that produce accurate soft shadows. Readers are encouraged to read a survey of approximate real-time soft shadow techniques [Hasenfratz *et al.*, 2003], as well as a comparison of more recent methods [Johnson *et al.*, 2009].

Frequency Analysis and Reconstruction Methods Often high dimensional signals have long narrow spectra in the Fourier domain. In these cases adaptive sampling of the spectra can be

used [Soler *et al.*, 2009], as well as sheared filters that compactly capture the spectra in the Fourier domain and allow the use of sparser sampling rates [Shinya, 1993; Chai *et al.*, 2000; Zwicker *et al.*, 2007]. The shape of shadow spectra has been studied in the Fourier domain [Durand *et al.*, 2005; Ramamoorthi *et al.*, 2005; Lanman *et al.*, 2008]. We extend these analyses by showing that the frequency spectrum for practical scenes is most often a wedge based on the minimum and maximum depth of the occluder. We also draw attention to extreme cases where this assumption does not hold (§4.7). The use of first-order gradients to aid in reconstruction has been studied [Ramamoorthi *et al.*, 2007], and several new techniques for reconstructing general signals have also been developed [Hachisuka *et al.*, 2008; Overbeck *et al.*, 2009]. We use a sheared filter and extend previous work to solve the more general problem where the pixel integrands are not aligned to a regular grid. In §4.6 we compare to Adaptive Wavelet Rendering [Overbeck *et al.*, 2009], the state of the art in contrast-based adaptive reconstruction, and show that for low sample counts our sheared filter produces more accurate results.

Sheared Filters Our work is perhaps closest to the sheared filters developed for other problems like light fields [Chai *et al.*, 2000] and motion blur. Our theoretical analysis relates to these approaches, but we focus on shadows and show how our analysis reduces to convolution soft shadows in the special case of parallel planes [Soler and Sillion, 1998]. Moreover, previous methods assume a regular grid of cameras or that all pixels integrate over the same shutter interval. In contrast, we are integrating over a fixed plane (the light), but our sampling and filtering happen at points that are at many different depths. Thus, we must solve a more general irregular reconstruction problem. We therefore introduce an additional step, going from the actual receiver to a shadow light field that is independent of receiver depth.

Ray Traced Shadows Brute force ray tracing computes correct answers but is expensive [Cook *et al.*, 1984]. Photon mapping shoots shadow photons as an optimization to classify areas that are unoccluded, occluded, or partially occluded from direct lighting [Jensen and Christensen, 1995]. Our method focuses on areas with partial occlusion, whereas most photon mapping implementations fall back to Monte Carlo sampling in these areas rather than directly visualizing the shadow photon map. Multidimensional lightcuts uses a hierarchical tree graph for receiver points and point light sources, makes cuts through the receiver and light graphs at each pixel, and shoots shadow rays

for all pairs of nodes along the graph cuts [Walter *et al.*, 2006]. Our work is complementary to both photon mapping and multidimensional light cuts, since our sheared filter can be incorporated to select a large set of appropriate shadow rays to share for a given receiver point, further reducing shadow ray casts. Coherence across occluders and receivers has been used [Bala *et al.*, 1999; Hart *et al.*, 1999; Agrawala *et al.*, 2000; Ben-Artzi *et al.*, 2006], as well as separating near- and far-field occlusion [Arikan *et al.*, 2005]. Blurring sharp ray traced results in image space can also be used to approximate soft shadows and blurry reflections [Robison and Shirley, 2009]. Other methods have prefiltered partial occlusion at kd-tree cells, but darkening can occur when locally pre-filtered nodes are composited together [Lacewell *et al.*, 2008]. Our system enables sparser sampling than previous methods because we share samples and exploit coherence in the full 4D shadow light field.

Light Fields and Precomputed Radiance Transfer Many previous methods have used light fields for rendering [Gortler *et al.*, 1996; Levoy and Hanrahan, 1996; Isaksen *et al.*, 2000; Chen *et al.*, 2002; van der Linden, 2003; Stewart *et al.*, 2003]. The shape of occlusion light fields has been studied [Durand, 1999], as well as how to capture occlusion light fields [Lanman *et al.*, 2008]. These methods usually use image-based rendering where data is captured by photographs taken in a regular grid, whereas we sparsely sample only the areas of the light field that are used by the receivers of the image. Precomputed radiance transfer methods can also be used for relighting problems involving complex shadows [Ng *et al.*, 2003; Zhou *et al.*, 2005; Sun and Ramamoorthi, 2009], but most methods require dense sampling of an object or scene, and its light transport.

Shadow Maps There are a variety of area light source methods that use shadow maps [Neulander, 2008; Yang *et al.*, 2009], or a statistical description of occlusion [Annen *et al.*, 2008]. The main drawback to using shadow maps is that most area light source techniques either have a fixed resolution for the shadow map that can miss geometric detail, or they process occluders independently and use approximate methods to composite the result [Johnson *et al.*, 2009]. One exception to this rule is the Sample Based Visibility method that uses alias-free shadow maps and conservative triangle rasterization [Sintorn *et al.*, 2008]. The generation of soft shadow textures by [Soler and Sillion, 1998] shows that for parallel plane occluder-receiver pairs the resulting shadow is a convolution between the light source and the planar occluder, leading to a multiplication of light and occluder spectra in

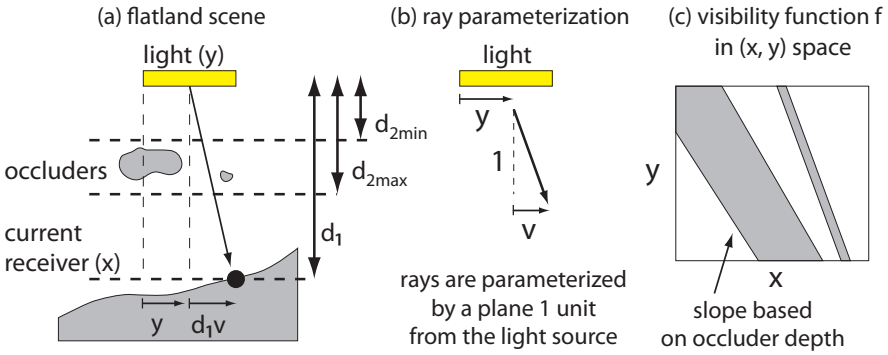

Figure 4.3: A simple illustration in flatland. (a) Note that we handle many occluders in a range of depths $[d_{2\,\min}, d_{2\,\max}]$, and that d_1 is the distance to the current receiver, but we do not assume all receivers are co-planar. The vertical line at the left side of the light serves as the origin of spatial coordinates for all planes. (b) We parameterize rays based on the ray origin and directional offset at a plane 1 unit away. (c) Occlusion in (x, y) space has coherent diagonal bands where occluders block the light source.

the frequency domain. We show that our analysis of non-planar occluders and receivers generalizes their approach (§4.3.2). Furthermore, our implementation samples across a 4D ray database and can handle receiver surfaces that smoothly vary from close to far away from the light source. In comparison, their method captures 2D information from a single point on the light source and can have discontinuities in areas that transition from one soft shadow texture to another.

Object Based Methods for Shadows Storing silhouette edges allows for efficient sampling of the light source [Laine *et al.*, 2005]. Penumbra wedges [Assarsson and Akenine-Möller, 2003] and beam tracing can also be used [Overbeck *et al.*, 2007]. These methods process triangle edges, which becomes a bottleneck for highly tessellated scenes or scenes with spiky geometry.

4.3 Shadow Signal and Light Field

We start our analysis of the shadow signal and light field with a simple scene in flatland, where the distance from the planar light to the current receiver point is d_1, and the occluding geometry is contained within a depth range of $[d_{2\,\min}, d_{2\,\max}]$ measured from the light (see Figure 4.3a). In our implementation, this analysis is applied to the local extent of a single pixel allowing our algorithm

to use a different value of d_1 per pixel and handle non-planar receiver surfaces. Parts of our analysis will examine frequencies of the receiver, and for these problems it is most natural to use a two-plane (x, y) parameterization, where y is the absolute distance along the planar light source, and x is the absolute distance along the plane parallel to the light source with distance d_1 (the receiver plane). However, because we want to share rays across many receivers, we store ray samples in a receiver-independent (v, y) parameterization, where y is still a distance along the light source, and v is measured as an offset from y at a plane one unit from the light (similar to [Durand *et al.*, 2005], see Figure 4.3b).

Our analysis in flatland is easy to extend to 3D where the light field has four dimensions (v_1, v_2, y_1, y_2). If we use orthogonal basis vectors to parameterize the area light source, the (v_1, y_1) subspace is linearly independent of the (v_2, y_2) subspace. Because of this, most of the computations can be broken down into two separate 2D problems.

We consider a single planar occluder parallel to the light source at distance d_2 away from the light source. The occluder is defined by its transparency function $g()$ in this plane, where $g()$ takes a 1D spatial parameter in flatland. Because a ray (v, y) intersects the occluder at spatial coordinate $(d_2 v + y)$, the visibility function $f(v, y)$ is defined by

$$f(v, y) = g(d_2 v + y), \tag{4.1}$$

where a value of one is fully visible and a value of zero is fully occluded. We will extend this to occluders with a range of depths later. As seen in Figure 4.3c, each occluder creates a diagonally shaped band in the x-y pixel-light space, and all bands are multiplied together to get the final visibility function. Our method efficiently exploits the coherence of these diagonal bands across the light field.

For shadow calculations we use the shadow light field $f(v, y)$ in conjunction with a single receiver point. The receiver is parameterized by a plane at a distance d_1 from the light, and an offset x along the plane. Note that we do not assume that all points are co-planar; we allow d_1 to vary with x. The incoming irradiance, with shadow $h(x)$, is:

$$h(x) = r(x) \int f\left(\frac{x - y}{d_1}, y\right) l(y)\, dy, \tag{4.2}$$

where $l(y)$ is the intensity of the light source, and $r(x)$ captures the geometric form factor from the receiver point to the area light (separating the form factor from the visibility is a common

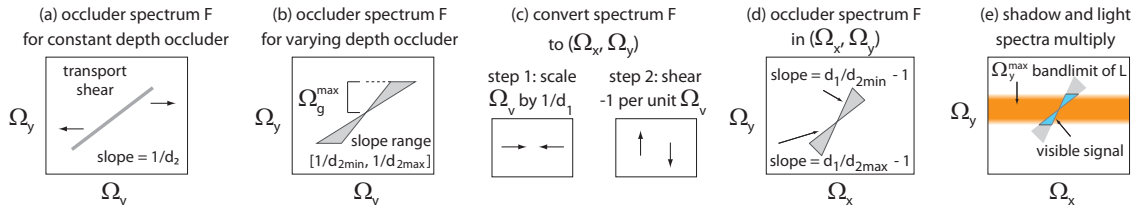

Figure 4.4: We design our filter in the Fourier domain, later reinterpreting these steps in the primal domain to obtain the filter used in our implementation. Based on Equation 4.3, the frequency content for the occluder in (v, y) space will be (a) a line for occluders with a constant depth, and (b) a wedge for occluders with a range of depths. (c) We scale and shear this picture to (d) the frequency space of the receiver, (Ω_x, Ω_y). Based on Equation 4.5, our filter must cover the overlap between the occluder spectrum and the light spectrum (e).

approximation [Soler and Sillion, 1998]). Since $r(x)$ is independent of shadows, we will omit it from later derivations. In our current implementation we consider diffuse BRDFs, and the reflected color will simply be the surface color multiplied by $h(x)$ (see §4.7 for a discussion of more general BRDFs). Note that in many applications, BRDFs are split into a diffuse component and a glossy component, with shadowing applied to the diffuse component and a different reflection technique employed for the glossy component.

4.3.1 Fourier Analysis

A Fourier analysis enables the design of a filter that is customized to the frequency content of shadows. Capital letters like F, G and H denote Fourier transforms. Figure 4.4 shows the process of mapping a given occluder spectrum into the receiver's local parameterization.

We first use the Fourier transform to compute the frequency spectrum of visibility $\mathcal{F}(f(v, y))$. Appendix B provides a detailed algebraic derivation—that also follows directly from Equation 4.1, and the Fourier linear transformation theorem [Bracewell *et al.*, 1993],

$$F(\Omega_v, \Omega_y) = G(\Omega_y)\delta(\Omega_v - d_2\Omega_y), \tag{4.3}$$

where $\delta(\cdot)$ is a delta function. For a constant depth, Equation 4.3 shows that the occluder spectrum lies along a line with slope $1/d_2$, as seen in Figure 4.4a. For practical scenes with a range of depths, the occluder spectrum has a wedge shape, shown in Figure 4.4b, with slopes bounded by

$[1/d_{2\,\min}, 1/d_{2\,\max}]$ [Chai *et al.*, 2000]. The approximated bandlimit for the occluder function $g()$ is $\Omega_g^{\max}$, and it bounds the $F(\Omega_v, \Omega_y)$ spectrum along Ω_y (Figure 4.4b).

Our next step is to consider the frequency spectrum of the shadow light field on the receiver, rather than in its canonical parameterization. This transformation is shown in Figure 4.4c, and involves both a scale and a shear. Formally, we compute $\mathcal{F}\left[f\left(\frac{x-y}{d_1}, y\right)\right]$ from Equation 4.2, using the Fourier linear transformation theorem or the detailed derivation in Appendix B,

$$\mathcal{F}\left[f\left(\frac{x-y}{d_1}, y\right)\right] = d_1\, F(d_1\Omega_x, \Omega_y + \Omega_x), \tag{4.4}$$

where we now use x and Ω_x along the receiver rather than v and Ω_v. Note the scaling $d_1\Omega_x$ in the first argument. The further the receiver point is from the light (large d_1), the more compressed the frequency spectrum is (shadows are smoother). On the other hand, for a receiver point close to the light (small d_1), the frequency spectrum is less compressed, with high-frequency effects near contact shadows.

We now take the Fourier transform of $h(x)$ to find shadow frequencies on the receiver. We use the fact that the integral in Equation 4.2 can be seen as convolving the product of f and l with a constant function of 1 to derive Equation 4.5. It follows that in the Fourier domain we only need the constant (zero frequency) of $(F \otimes L)$, where $\otimes$ represents a convolution:

$$H(\Omega_x) = d_1 \int F(d_1\Omega_x, \Omega_y + \Omega_x) L(-\Omega_y)\, d\Omega_y. \tag{4.5}$$

The calculation of $H(\Omega_x)$ is done by integrating over the product of the light frequencies L and the occlusion function F. In other words, to compute the shadow frequencies for H, we need to find all places where the non-zero amplitudes of L and F overlap, as shown in Figure 4.4e.

4.3.2 Relation to Parallel Plane Convolution

The above results generalize the seminal parallel plane convolution result of [Soler and Sillion, 1998]. In particular, if d_1 and d_2 are fixed, we simply substitute Equation 4.3 into Equation 4.4, so that the frequencies of the shadow light field are reparameterized for Ω_x at the receiver:

$$\mathcal{F}\left[f\left(\frac{x-y}{d_1}, y\right)\right] = d_1\, G(\Omega_y + \Omega_x)\delta\left(d_1\Omega_x - d_2(\Omega_y + \Omega_x)\right), \tag{4.6}$$

$$= \left(\frac{d_1}{d_2}\right) G\left(\frac{d_1}{d_2}\Omega_x\right)\delta\left(\left(\frac{d_1}{d_2} - 1\right)\Omega_x - \Omega_y\right). \tag{4.7}$$

In the last line, we bring the d_2 factor outside of the delta function and then use the delta function to set $\Omega_y = (d_1/d_2 - 1)\Omega_x$. Note that this implies that the occluder spectrum in the receiver coordinate space will have Fourier slope $d_1/d_2 - 1$. In our case, the spectrum is not simply a line, but a wedge with slopes ranging from $d_1/d_{2\,\min} - 1$ to $d_1/d_{2\,\max} - 1$, as shown in Figure 4.4d.

If we now substitute Equation 4.7 in Equation 4.5, the integral involves a delta function, and will therefore simply result in the integrand, in particular $L(-\Omega_y)$, being evaluated at $\Omega_y = (d_1/d_2 - 1)\Omega_x$,

$$H(\Omega_x) = \left(\frac{d_1}{d_2}\right) G\left(\frac{d_1}{d_2}\Omega_x\right) L\left(\left[1 - \frac{d_1}{d_2}\right]\Omega_x\right), \tag{4.8}$$

which is a simple multiplication in the frequency domain, and hence a (suitably reparameterized) convolution in the spatial domain.[1]

Our method generalizes this approach by keeping all needed frequencies of F for a range of depths (thus considering a frequency wedge for F rather than a simple line), and therefore allowing the receiver and blockers to be general (they need not be restricted to parallel planes). Note also that Figure 4.4e therefore involves an integration against the full wedge—when this wedge reduces to a line, the integration becomes a simple multiplication in frequency space, or a primal-space convolution as in [Soler and Sillion, 1998].

4.4 Sheared Filter

In this section, we present a new sheared filter that operates over shadow light fields. In the Fourier domain, we design our sheared filter to be as compact as possible to enable the tight packing of replicas in the Fourier domain and sparse sampling in the primal domain. Our filter must cover the overlap of the light L and occluder F spectra to reconstruct the shadow signal H accurately. In §4.5, we will use the shape of the sheared filter in the primal domain to enable sparse sampling across our 4D ray database. We begin by calculating the width and shear of the sheared filter in the Fourier domain. We then examine how to apply transformations to convert a simple axis-aligned filter into a sheared filter in the Fourier and primal domains. Because shadow receivers integrate

[1] Our notation differs slightly from [Soler and Sillion, 1998], with their d_1 corresponding to our $d_1 - d_2$, and their α corresponding to our $(d_1/d_2) - 1$. We also use x for receiver and y for light source, instead of vice-versa. Finally, they integrate over a two-dimensional light source, causing the outside factor in their convolution equation to be squared.

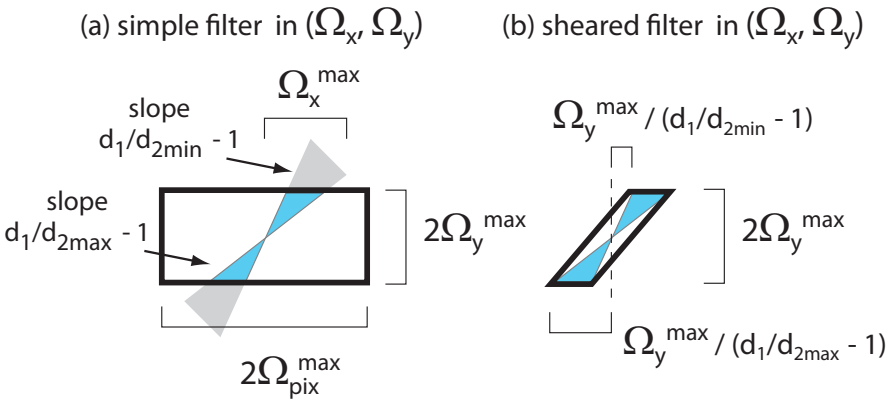

Figure 4.5: (a) A simple filter that captures all displayable frequencies in (Ω_x, Ω_y). The bandlimits for display are the pixel bandlimit $\Omega_{\text{pix}}^{\text{max}}$ and the light bandlimit Ω_y^{max}. (b) Our new sheared filter compactly covers the same non-zero frequencies that the simple filter does, but its compact shape enables much sparser sampling rates.

over irregular domains in the light field, our final step is to transform the primal filter from the (x, y) parameterization to the receiver-independent (v, y) parameterization used for our ray database.

Simple and Sheared Filter Shapes We first look at a simple filter in Fourier space that covers all displayable frequencies, as shown in Figure 4.5a. This simple filter is axis-aligned in (Ω_x, Ω_y) space and captures all frequencies within $\Omega_x \in [-\Omega_{\text{pix}}^{\text{max}}, \Omega_{\text{pix}}^{\text{max}}]$ and $\Omega_y \in [-\Omega_y^{\text{max}}, \Omega_y^{\text{max}}]$, where Ω_y^{max} is the bandlimit of the light intensity function $l(y)$, and $\Omega_{\text{pix}}^{\text{max}}$ is the maximum frequency in x that can be displayed in the output image. In the pixel domain the $\Omega_{\text{pix}}^{\text{max}}$ bandlimit is easy to define as 0.5 wavelengths per pixel. By measuring the projected x distance that a given pixel subtends, we can simply set $\Omega_{\text{pix}}^{\text{max}}$ to 0.5 wavelengths per subtended x pixel distance.

Our sheared filter, shown in Figure 4.5b, has the same spectral extent along Ω_y as the simple filter, but our filter is scaled and sheared to compactly bound the non-zero frequencies. Based on the distances from the $\Omega_x = 0$ axis, as shown in Figure 4.5b, we can see that the width of our filter in the Fourier domain is simply the difference of these two offsets $\Omega_y^{\text{max}}((d_1/d_{2\,\text{max}} - 1)^{-1} - (d_1/d_{2\,\text{min}} - 1)^{-1})$. Similarly, the shear is the ratio between the height of the filter and the average of the offsets $\frac{1}{2}((d_1/d_{2\,\text{max}} - 1)^{-1} + (d_1/d_{2\,\text{min}} - 1)^{-1})$.

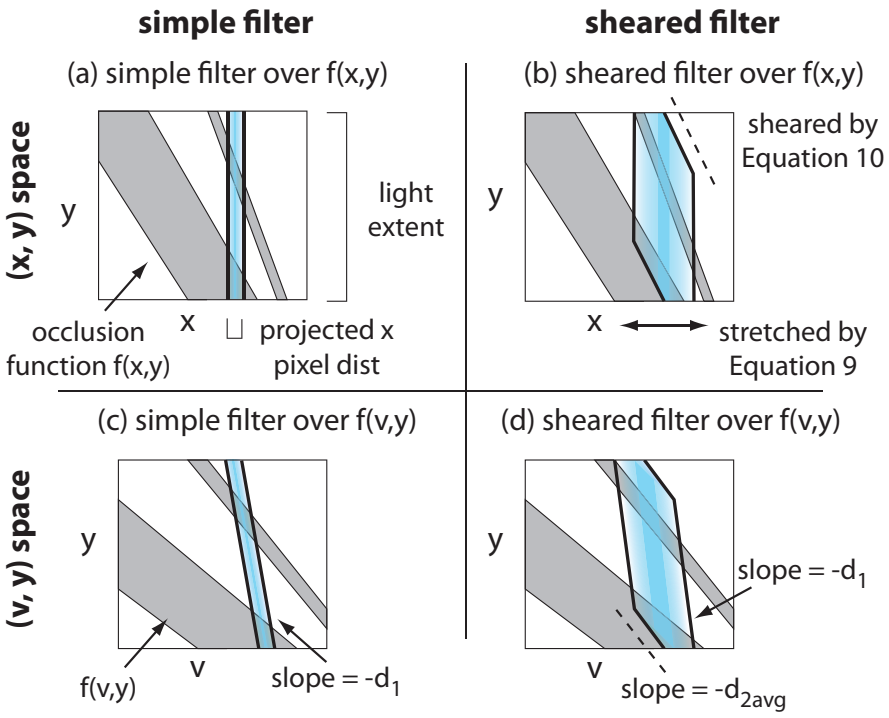

Figure 4.6: The occlusion signal f and the simple and sheared filters in the primal (x, y) and (v, y) domains. (a) The simple filter is axis-aligned in (x, y). (b) We create the sheared filter shape by taking the simple filter in (a) and applying the transformations in Equations 4.9 and 4.10. (c) The simple filter transformed to (v, y) using Equations 4.11 and 4.12. (d) The sheared filter transformed to (v, y) using Equations 4.11 and 4.12. The $d_{2\text{avg}}$ slope of the filter is between $d_{2\,\text{min}}$ and $d_{2\,\text{max}}$. Our ray database stores samples in (v, y) so this is the final shape of our filter in flatland. For our practical implementation in 3D there are two additional dimensions (v_2, y_2) that form an orthogonal subspace and independently undergo the same transformations.

Transformation to Sheared Filter in Primal Domain As with many previous analyses, the key insights come from Fourier theory, but our practical implementation operates directly on primal domain samples, and does not need explicit Fourier transforms. Now that we know the exact dimensions and slope of the sheared filter in the Fourier domain, we can derive the transformations necessary to convert a simple filter into a sheared filter in the Fourier domain. Knowing the Fourier domain transformations then makes it easy to compute the corresponding primal domain transformations.

In the Fourier domain, the first step is to scale along Ω_x by the sheared filter width divided by the simple filter width. Fourier theory dictates that for the primal domain we need to scale along x by the inverse amount:

$$\text{primalScale} = \frac{2\Omega_{\text{pix}}^{\max}}{\Omega_y^{\max}} \left[\left(\frac{d_1}{d_{2\max}} - 1 \right)^{-1} - \left(\frac{d_1}{d_{2\min}} - 1 \right)^{-1} \right]^{-1}. \tag{4.9}$$

The next step in the Fourier domain is to shear in Ω_y per unit Ω_x. In this case, Fourier theory tells us that we need to shear by the negated amount in y per unit x.

$$\text{primalShear} = -\frac{1}{2} \left[\left(\frac{d_1}{d_{2\max}} - 1 \right)^{-1} + \left(\frac{d_1}{d_{2\min}} - 1 \right)^{-1} \right]. \tag{4.10}$$

The original shape of the simple filter in the primal domain is axis-aligned, integrating over the projected x pixel distance and the light source y extent, as shown in Figure 4.6a. Using the transformations of Equations 4.9 and 4.10, this simple filter is transformed into a sheared filter, as shown in Figure 4.6b. Note that the shearing seeks to align the filter with the diagonal bands from occluders.

Sheared Filter in (v, y) In our implementation we store all samples in (v, y) space, so the last step is to transform the primal filter from (x, y) to to (v, y), as shown in Figures 4.6c and 4.6d. We know that $(v, y) = ((x - y)/d_1, y)$, and from this we can derive that we first need to shear -1 units in x per unit y, then scale by $1/d_1$ in x:

$$\text{vyShear} = -1, \tag{4.11}$$

$$\text{vyScale} = \frac{1}{d_1}. \tag{4.12}$$

Numerical Verification We verify our frequency analysis by plotting a complex scene in flatland, and examining the Fourier transform, as shown in Figure 4.7. Our flatland scene is composed of thin and round elements randomly placed in a depth range near the light source, with x and y both ranging between 0 and 1 (Figure 4.7a). In Figure 4.7b we graph the occlusion function in (x, y), using the object colors from Figure 4.7a as a means to visualize. We then take the Fourier transform of the occlusion function, as shown in Figure 4.7c. Note that the Fourier spectrum has our predicted wedge shape, and that amplitudes dissipate rapidly in areas farther from the constant zero frequency. We then zero out all frequencies that are outside of a sheared filter that covers a light bandlimit of

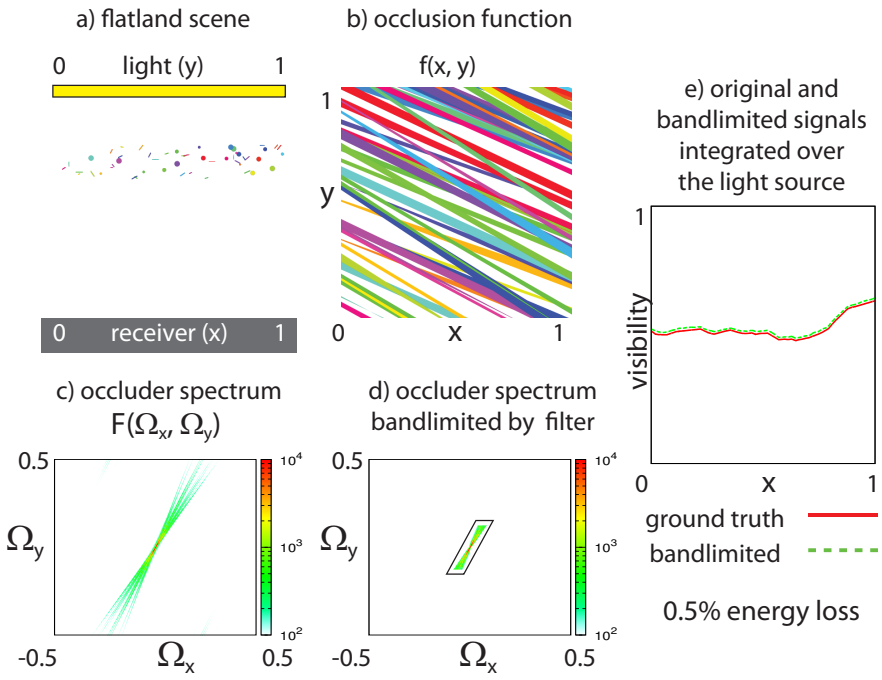

Figure 4.7: Numerical verification of our Fourier theory. (a) A set of occluders with random orientations and positions. (b) Graph of the occlusion function $f(x, y)$ for this scene. (c) We take the Fourier transform of $f(x, y)$ to get $F(\Omega_x, \Omega_y)$. (d) Our method captures frequencies inside the footprint of the sheared filter. All frequencies outside the filter are set to zero. (e) A graph of the original signal $f(x, y)$ and the corresponding bandlimited signal from (d). Each point x integrates over the light y to obtain the final visibility.

$\Omega_y^{\max}$ equal to $\frac{1}{8}$, as shown in Figure 4.7d (in this example 0.5% of the energy lies outside of the filter). Using both the original Fourier spectrum as well as the bandlimited spectrum, we convert back to the primal domain and integrate with the light source to compute the final receiver values (Figure 4.7e). This final graph of visibility shows that sheared filters can reconstruct shadow signals with minimal loss of fidelity, despite the high-frequency nature of the original signal.

4.5 Algorithm

Our rendering system provides a practical way to sample occlusion in the scene, produce a ray database of all samples, and reconstruct the shadows based on the analysis above. We use a very

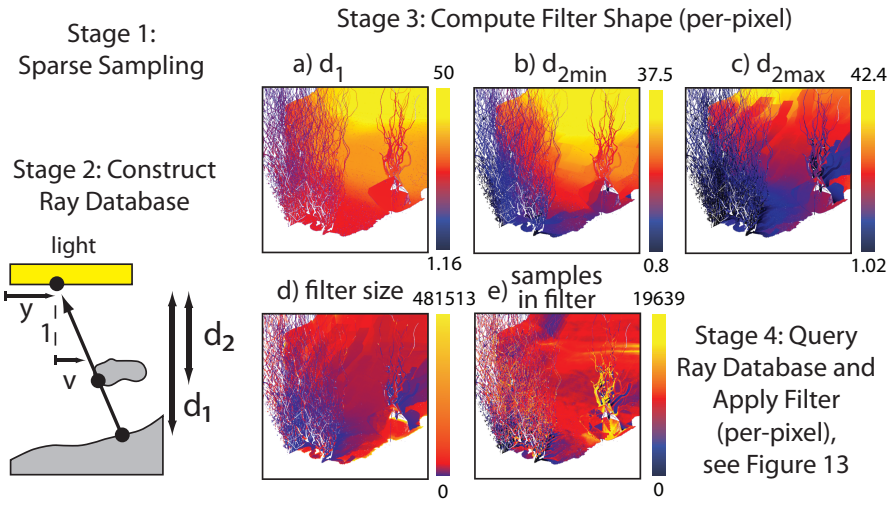

Figure 4.8: Individual steps of the algorithm illustrated using the scene from Figure 4.13. Stage 1 traces rays and writes samples to disk. Stage 2 converts the 3D information to the (v_1, v_2, y_1, y_2) 4D parameterization, along with distances d_1 and d_2. Stage 3 computes the shape of the filter using d_1, $d_{2\,\mathrm{min}}$, $d_{2\,\mathrm{max}}$. Stage 4 uses the filter to average together multiple samples and compute the shadow signal.

sparse sampling, often 1 shadow ray per pixel, and a suitably wide sheared reconstruction filter at each pixel. An overview of the algorithm is depicted in Figure 4.2 and detailed steps are shown in Figure 4.8.

Shadow Sampling The first stage is to sparsely sample occlusion in the 4D light field (Stage 1 in Figure 4.8). To generate samples we trace a small number of shadow rays from each pixel that contains a receiver surface. Our sampling is driven by the receivers that are visible in the actual image, unlike most other light field techniques that sample uniformly and densely in the 4D space of rays. Our implementation uses a simple programmable shader that traces rays from the receiver to the light source, and for each sample writes the receiver point, ray direction, and the occluder distance d_2 to a file (if the ray is unoccluded we set d_2 to -1).

In principle, we could compute sampling rates at each pixel, directly from the Fourier analysis, as described in Appendix B. In practice however, we have found that areas that receive soft shadows need very sparse sample counts, on the order of 1 to 8 samples per pixel. We have developed a program to do adaptive sampling, but the quality of the final image is usually easier to control by

simply setting a uniformly low sample density.

When we shade a receiver point that lies inside the shadow light field's $[d_{2\,\min}, d_{2\,\max}]$ depth bounds, our theory can no longer provide tight bounds on the spectrum, and we cannot safely apply a shear or scale to the primal filter (this can happen with self-shadowing and other cases of near-field occlusion). In this case we simply revert back to using stratified Monte Carlo sampling.

Ray Database Construction The second stage reads samples from disk, computes the 4D parameterization of each ray, and stores the sample into a 4D ray database (Stage 2 in Figure 4.8). Our current implementation uses a simple 2D grid as an acceleration structure, indexing across direction parameters v_1 and v_2. We have experimented with a 4D grid and other bounding volumes, but have found so far that they delivered little speedup when queried with the highly anisotropic shapes and varied orientations of sheared filters generated by a practical scene. The depth range of the light field, $[d_{2\,\min}, d_{2\,\max}]$, is also calculated at this stage.

The memory requirements for our method are small, consisting only of loading the ray database into memory. Each sample in the ray database consists of (v_1, v_2, y_1, y_2) 32 bit floating point coordinates, with an additional distance d_2, that stores the distance to the closest occluder or indicates an unoccluded ray. For the scene in Figure 4.1 the final ray database was 17 MB.

Shadow Reconstruction To reconstruct shadows, another rendering pass uses a programmable shader that accesses the ray database. For each receiver point, the shader computes the shape of the appropriate sheared filter, queries the ray database with the filter shape, and weights all samples inside the filter's 4D footprint (Stage 3 in Figure 4.8).

The first step is to compute d_1, $d_{2\,\min}$, and $d_{2\,\max}$ for the current receiver as shown in Figure 4.8a-c (see Optimizations below for more details). The next step is to compute the shape of the sheared filter. Although the shape of a sheared filter in 4D may be hard to visualize, it is simple to compute: We require that the planar area light is parameterized with orthogonal basis vectors, guaranteeing that (v_1, y_1) and (v_2, y_2) span orthogonal 2D subspaces of the 4D light field. Consequently, we treat the sheared filter as the product of two 2D sheared filters in (v_1, y_1) and (v_2, y_2). For each 2D subspace, we first determine the basis vectors that define the light and pixel filter extent of a simple filter in (x, y) (Figure 4.6a). We then transform the basis vectors using Equations 4.9, 4.10, 4.11, and 4.12 such that the basis vectors now represent the centerline and "shear axis" of the sheared filter

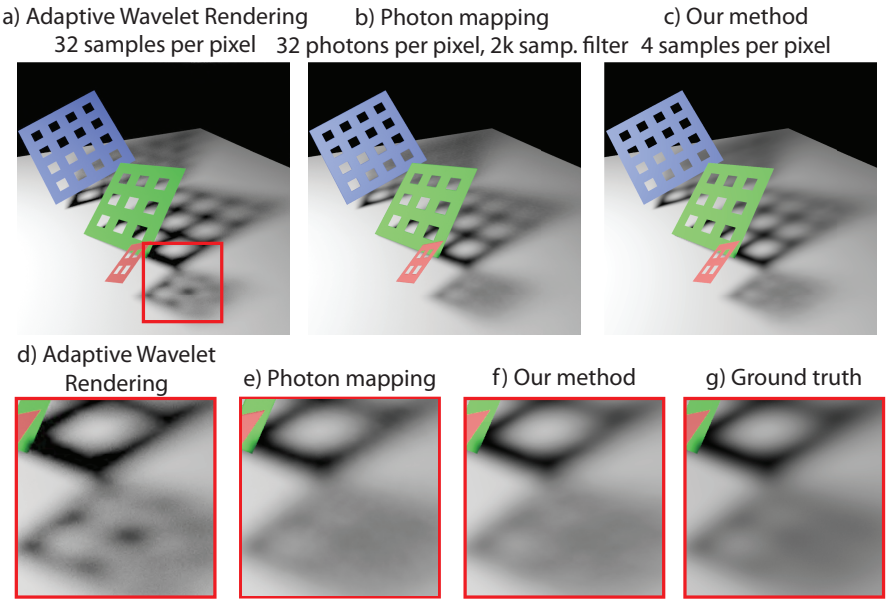

Figure 4.9: (a) Adaptive Wavelet Rendering with 32 samples per pixel. (b) Direct visualization of photon map to compute soft shadows with 32 photons per pixel (photon map requires 1.8GB of memory). The soft shadows near the top of the image still have a fair amount of noise. (c) Our method using 4 samples per pixel. (d) Adaptive Wavelet Rendering computes a soft image, but the shape of the soft shadow is slightly off in this case. (e) Direct visualization of the photon map has converged in this area, but other areas have noise. (f) Our method accurately reconstructs the soft shadow signal. (g) Ground truth using 2048 rays with Monte Carlo.

in (v, y) (Figure 4.6d). The range of occluder depths $(d_{2\,\max} - d_{2\,\min})/d_{2\,\max}$ is shown in Figure 4.8d, as well as the total number of samples inside the filter in Figure 4.8e.

The next step is to determine where the filter is centered. Focusing on the (v_1, y_1) dimensions, and given the positioning of the receiver point and the light in 3D space, we can compute a v_1 value (ray direction) for any given y_1 value (light position). It is convenient to compute v_1 for $y_1 = 0$ since this is always defined to be one edge of the light in our implementation. Similary we compute v_2 for $y_2 = 0$, completely anchoring the centerline of our 4D sheared filter.

We have now defined the placement and shape of our filter. We now process every sample in every grid cell that lies inside the filter's v_1 and v_2 extents. We then calculate the sample's coordinates relative to the transformed light extent and pixel extent basis vectors. These coordinates

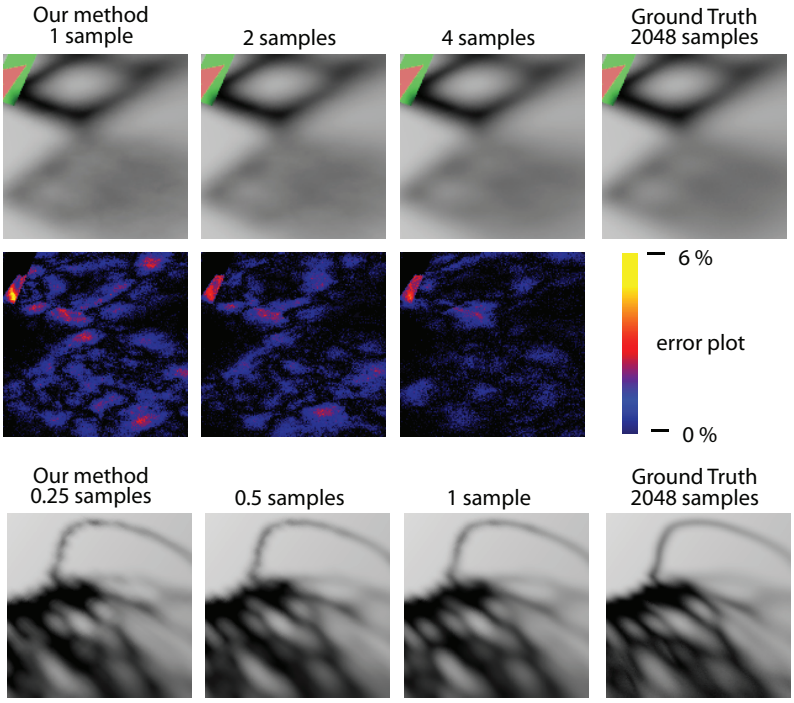

Figure 4.10: Analysis of different quality levels for different sample counts. The first and second rows show a closeup from the scene in Figure 4.9. As the quality increases to four samples we can see in the error plot that we are converging. The third row shows a new closeup from the scene in Figure 4.1, near the upper part of the shadow. We see here the spotting artifacts that can occur in areas of undersampling.

can then be interpreted in the original (x, y) space, where one coordinate determines the pixel filter response and the other determines the light intensity (Stage 4 in Figure 4.8).

Optimizations Computation of the depth bounds $d_{2\,\min}$ and $d_{2\,\max}$ can often be done by simply evaluating the global range of d_2 occluder values contained in the ray database (this was done for Figure 4.1). For more complicated scenes with many interacting occluders and receivers, it becomes necessary to compute $d_{2\,\min}$ and $d_{2\,\max}$ per receiver (this was done for Figures 4.12 and 4.13). To compute $d_{2\,\min}$ and $d_{2\,\max}$ per receiver, we precompute a 3D hierarchical sphere tree with all occluder positions to supplement the ray database. For each receiver we cull out points that are outside of the receiver-light frustum, and then compute the $[d_{2\,\min}, d_{2\,\max}]$ bounds on the remaining points. This computation is inexpensive relative to filtering, but it can at times lead to discontinuities in $d_{2\,\min}$ or

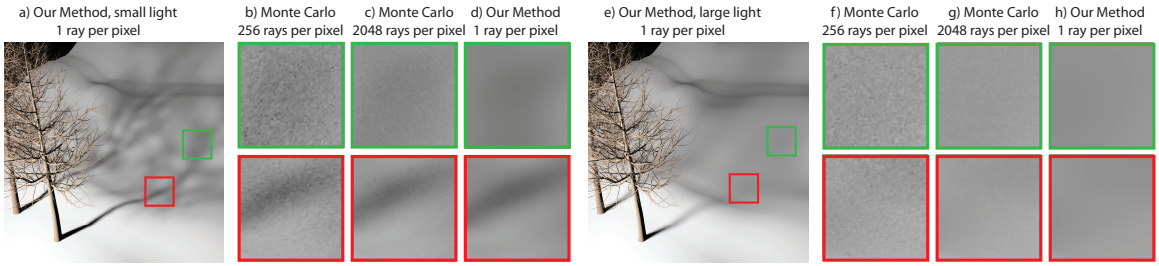

Figure 4.11: Comparisons between our method and stratified Monte Carlo sampling. We show two scenes: a smaller area light in a) through d), and a larger area light in e) through h). The timings show that as rays become more incoherent (in this case due to a larger light source), brute force ray tracing becomes very expensive. This is primarily due to the cost of updating the cached geometry for ray tracing. All timings are for rendering the right half of the image where shadow computation dominates the overall time of execution (the one exception is our sparse sampling pass, which processes the entire image).

$d_{2\,\max}$ across receiver points.

4.6 Results

We demonstrate our results with five scenes, which showcase a variety of challenging situations in Figures 4.1, 4.9, 4.11, 4.12, and 4.13. We also show comparisons and timings with respect to stratified Monte Carlo sampling, the current method of choice, as well as optimizations like photon mapping and the recent development of Adaptive Wavelet Rendering [Overbeck *et al.*, 2009].

All examples were run using Pixar's Renderman Pro Server 15.0 on a dual quad-core Xeon 2.33 GHz processor with 4 GB of memory. Due to our modular plug-in architecture, our code trivially runs in parallel for any number of threads. All scenes in this chapter use a planar area light with a circular Gaussian falloff that captures two standard deviations within the light radius.

4.6.1 Canonical "Grid" Scene

We start with the canonical scene in Figure 4.9, which shows grid occluders with shadows that smoothly go from sharp to wide. The very regular and smooth growth of the penumbra makes small artifacts easier to spot, but our method produces high-quality results.

Figure 4.9 also compares to alternative rendering approaches, like photon mapping [Jensen and Christensen, 1995] (Figures 4.9b,e use similar parameters to those in our method). We see that direct visualization of the photon map has converged in some areas, while other areas still have a fair amount of noise, even when using 32 photons per pixel. In addition, storing the 33M photons takes 1.8GB of memory in Renderman's implementation. For these reasons, the photon map is usually used for caustic or global illumination effects, and rarely visualized directly for soft shadows.

Therefore, in the remainder of the chapter, we focus on comparing to stratified Monte Carlo sampling, and to the state of the art adaptive reconstruction method, Adaptive Wavelet Rendering (AWR) [Overbeck *et al.*, 2009]. The AWR comparisons in Figure 4.9 (and 4.1) directly use the original AWR software, with the same scene setup and light source location and falloff as our method. We see in Figure 4.9 that our method better captures the widening blur of some areas of the shadow signal. Adaptive Wavelet Rendering with 32 samples per pixel has not converged due to the high variance of the shadow signal.

We analyze the effect of increasing sample counts in our method in the first two rows of Figure 4.10 (we will analyze the bottom row in §4.6.2). We first note that even when using only one sample or shadow ray per pixel, our method is quite accurate, with the maximum pixel error less than 6%. We are able to use such low sample counts because our sheared reconstruction filter effectively shares samples between many neighboring pixels. However, some difficult regions can be noisy, and these rapidly become more accurate with a moderate increase in sample count. Indeed, 4 samples per pixel reduces error almost to 0 everywhere in the image.

4.6.2 Detailed Occluding Geometry

In Figure 4.1, we show a detailed model (1.3M triangles) with many complex sillhouettes and thin features casting a shadow on a flat receiver. Our method works well in this case because it can handle complex occluders, and the scene has predominantly mid- and far-field occlusion, which lets our method use vastly fewer samples than other algorithms. For this scene we traced 1 ray per pixel during the initial sampling phase and created a ray database with 637,000 samples. In this scene we used Monte Carlo ray tracing with 4 rays for self-shadowing within the occluder and 64 rays for near-field occlusion on the receiver (a higher number of rays were necessary because of the extremely thin features of the occluder). We used the Monte Carlo solution for receiver positions

with $d_1 \leq d_{2\,\max}$, and did a smooth blend between the Monte Carlo solution and our solution up to a user-specified distance of $(1.1)d_{2\,\max}$ (as shown in the Figure 4.1a inset).

Figure 4.11 shows another difficult example with two complex tree occluders, this time shadowing a curved receiver. The tree trunk and branches are modeled with subdivision surfaces. In Figures 4.11a-d, a medium sized light source is used, and ray tracing shadows is fairly coherent. In Figures 4.11e-h, a larger light source is used, causing incoherence among rays and a much more expensive cost per ray. Our method is most beneficial when the cost per ray is high, which can be seen in more detail in the timings section below.

Stratified Monte Carlo sampling is still usually the method of choice for high-end rendering. However, for complex occluders in scenes like Figures 4.1 and 4.11, stratification has minimal benefit, since every shadow ray has very high variance—there is almost no coherence across the occlusion signal. Therefore, stratified Monte Carlo sampling requires a very large number of samples before the variance of the shadow is not visually noticeable (approximately 2048 samples in our case). While Adaptive Wavelet Rendering in Figure 4.1g is beginning to converge with 64 samples, the high variance of the signal leads to some low amplitude aliasing in the wavelet basis. In contrast, the third row of Figure 4.10 shows that our method can get decent results even when using 0.25 samples per pixel during the sparse sampling stage. In this case most of the spotting artifacts are removed by going up to just 1 ray per pixel.

Timings We report wall clock running times for Figures 4.1 and 4.11. For both images we measure the cost of rendering the right half of the image, since the shadows are mostly concentrated there (the costs on the left side of the image are dominated by scan conversion of the occluding geometry, which is not relevant to our or other algorithms). In our tests, we noticed that timing results can be fairly non-linear with the number of rays traced per pixel, which we believe is primarily dependent on how ray trace queries interact with Renderman's geometry caching algorithm. We therefore report numbers for canonical numbers of samples, which allow for equal time and quality comparisons with stratified Monte Carlo.

In Figure 4.1a, our method (with 1 ray per pixel) took 1 min 17 sec for the sparse sampling phase and 4 min 1 sec to reconstruct the shadows using the ray database, for a total time of 5 min 18 sec. In Figure 4.1b, Monte Carlo sampling with 256 samples per pixel took 5 min 6 sec, and in

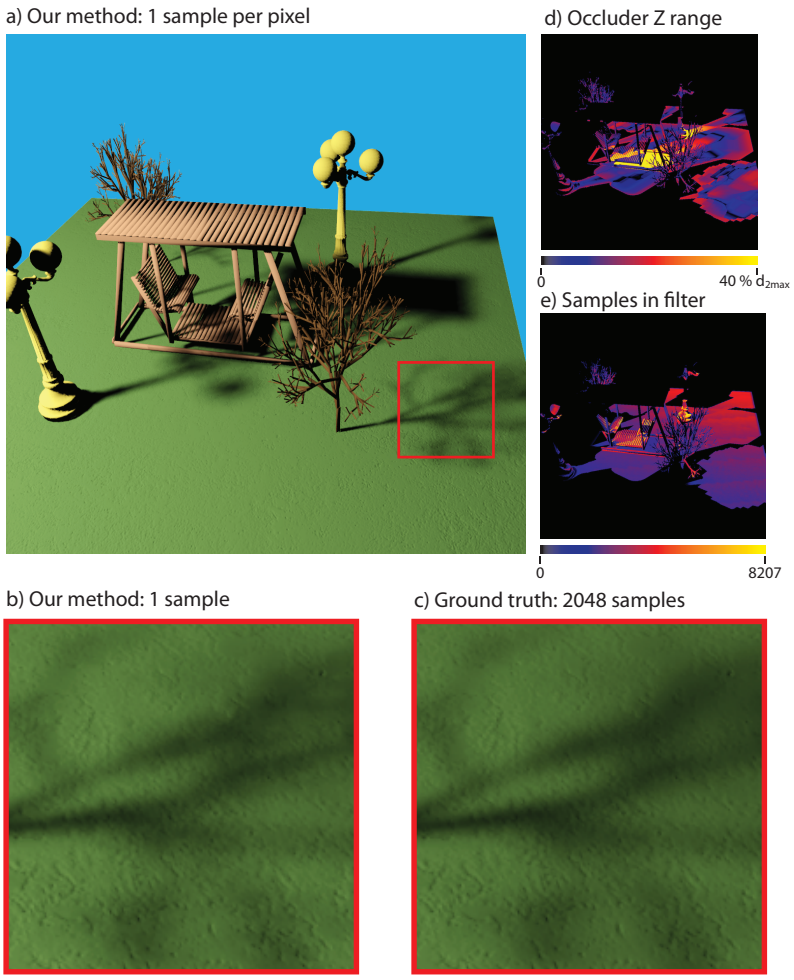

Figure 4.12: Scene with a wide array of occluders and receivers, as well as a curved ground surface with high frequency displacement.

Figure 4.1e, Monte Carlo sampling with 2048 samples took 23 min 3 sec.

In Figure 4.11a, our method (with 1 ray per pixel) took 5 min 25 sec for sparse sampling, 1 min 25 sec for reconstruction, and a total of 6 min 50 sec. Monte Carlo with 256 samples in Figure 4.11b took 6 min 9 sec, and Monte Carlo with 2048 samples in Figure 4.11c took 16 min 19 sec. The wider area light source in Figure 4.11e produces more significant speedups because of the incoherent shadow rays. Our method took 6 min 32 sec for sparse sampling, 6 min 30 sec for reconstruction, and 13 min 2 sec total. Monte Carlo with 256 samples in Figure 4.11f took 1 hr 15 min, and Monte Carlo with 2048 samples in Figure 4.11g took 3 hr 34 min, for a net speedup for our method of more than an order of magnitude.

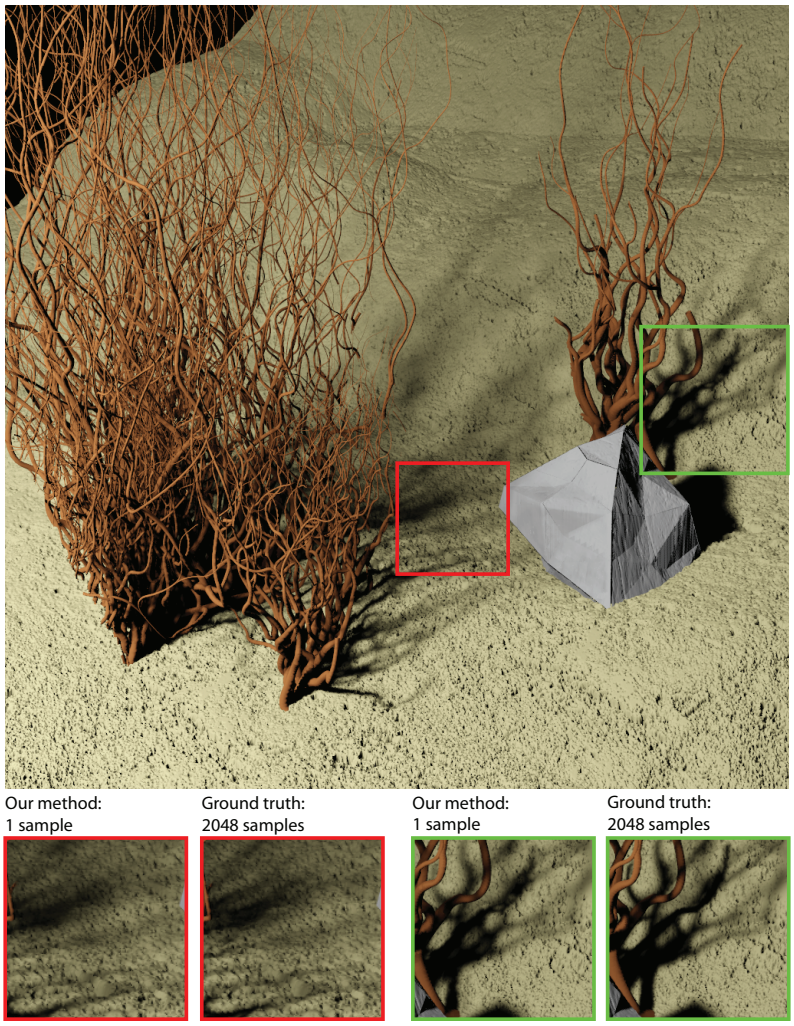

Figure 4.13: Complex tentacles scene with multiple occluders and receivers.

The AWR implementation uses an optimized packet ray tracer and a more stripped down shading system for speed, making comparisons to our Renderman plug-in difficult. They report taking 34 seconds to reconstruct their wavelet basis using 32 samples at image resolutions of 1024×1024. In our test scenes it appears that any noise in the wavelet basis is not visually noticeable after 256 samples. Because our method uses drastically fewer samples, our method will be preferrable whenever ray tracing is expensive, such as for highly tessellated models that may not fit into main memory. Note that for the scenes in this chapter we showed significant performance gains relative to the highly optimized Renderman ray tracer.

In general we note that our system drastically reduces ray tracing computation, making a trade

for increased filter computation. We have already shown that this is beneficial even for moderately complex scenes. A commonly noted trend in production rendering is that when computational power increases, artists will immediately increase the complexity of their scenes rather than enjoy faster render times. As long as this trend continues, geometric complexity for rendered scenes will increase, and our substitution of ray casts for filter computation will become more and more valuable.

4.6.3 Robustness: Complex Occluders and Receivers

Figure 4.12 shows a scene with many interacting occluders and receivers. In Figure 4.12b, we show that the range of occluder depths $[d_{2\,\min}, d_{2\,\max}]$ computed at each pixel can vary by large amounts. In Figure 4.12c we show the number of pixels contained in each pixel's custom reconstruction filter. Note that our method produces smooth results because of the high number of samples processed by each pixel. Finally, we show a scene with the foliage model from Figure 4.1, as well as a complex displacement-mapped receiver surface, and a number of other objects. This scene showcases a variety of intricate shadowing effects, such as complex objects casting and receving shadows. In both scenes we used 1 sample per pixel for the sparse sampling stage. In Figure 4.13 we used 4 Monte Carlo samples per pixel to compute near-field occlusion, while in Figure 4.12 we did not use any Monte Carlo sampling. Figures 4.12 and 4.13 show the robustness of our method for dealing with a range of complex occluder and scene configurations.

4.6.4 Animation

We have focused on still images, but it is also interesting to examine whether our method can produce stable animations. We show that our method can indeed produce high-quality animations, but may require a higher sampling rate to eliminate temporal aliasing.

In our supplementary video we animate the grids and trees scenes (stills from the video are shown in Figure 4.14). In the grids animation, the grids descend towards the ground plane, and we show that with 0.3 rays per pixel there are noticeable artifacts, but these artifacts go away using 3.0 rays per pixel. We rotate both trees in the tree scene (Figure 4.11a) to provide a stress test of many thin occluders moving relative to each other. In this case with 3.0 rays per pixel, the still images are often visually acceptable, but flickering can be seen as the tree rotates during animation. Small

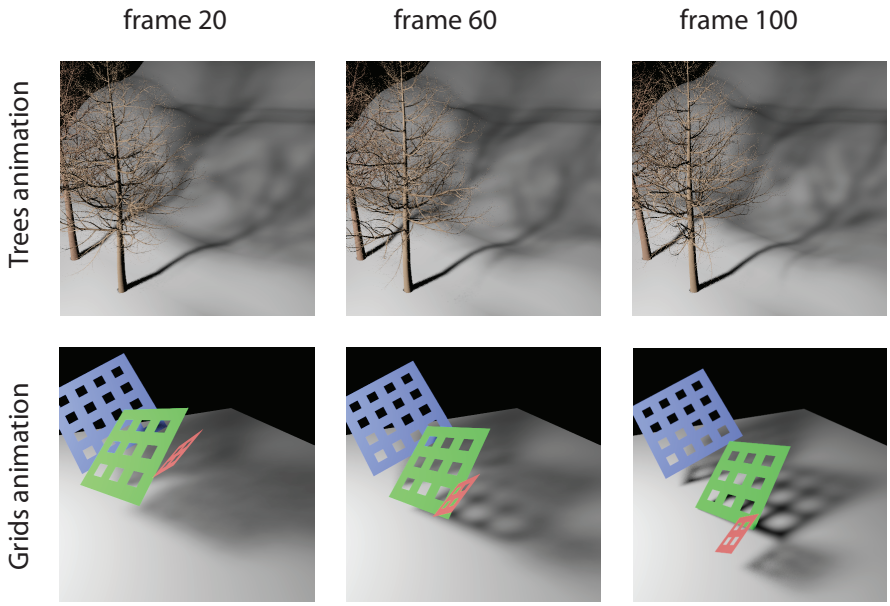

Figure 4.14: Still frames from an animation. The grids animate relative to the ground plane (3.0 rays per pixel). The trees each rotate relative to the ground plane (10.0 rays per pixel).

amounts of undersampling may cause medium to low frequency error relative to ground truth, but these errors are often visually imperceptible for still images. However, during animation these small errors can flicker, which is much more noticeable. When we increase the sampling rate to 10.0 rays per pixel the animating shadows become more stable and the flickering artifacts disappear. With 10.0 rays per pixel the sparse sampling pass took 5 min 46 sec, 16 min 16 sec for reconstruction (right half of 1k image), and a total of 22 min 2 sec.

Our supplementary video also compares our results to Monte Carlo integration during animation. Even with 2048 samples per pixel a small amount of noise is still visible in the animation using Monte Carlo. Our method with 10.0 rays per pixel delivers a smoother result with no visual flickering.

4.7 Artifacts and Convergence

We discuss the limitations and possible artifacts that come from our method. We first look at the artifacts that occur from undersampling occlusion and how the light bandlimit $\Omega_y^{\max}$ affects rendering. We also discuss how these two factors affect the convergence of our method. We then look at

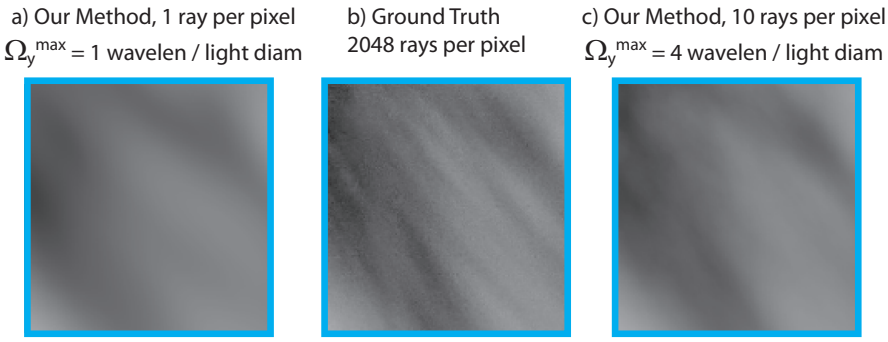

Figure 4.15: An inset from Figure 4.1 with contrast increased by 4x. With 1 ray per pixel and a low value for the $\Omega_y^{\max}$ light bandlimit our results are overblurred. By increasing the number of samples to 10 rays per pixel and increasing the $\Omega_y^{\max}$ light bandlimit our method can capture more shadow frequencies. However, even with these higher quality settings our method is missing some detail.

how undersampling can also manifest itself in the use of the occluder depth bounds $d_2^{\min}$ and $d_2^{\max}$. Finally we discuss two extreme cases where precise occluder configurations break our assumptions.

Undersampling and Light Bandlimit If the sparse sampling pass does not adequately sample the occlusion signal, our results will have mid to low frequency artifacts, as seen in Figure 4.10. Our method can also overblur if we set the $\Omega_y^{\max}$ light bandlimit too low, as seen in Figure 4.15. From Equation 4.9 we can see that the higher the $\Omega_y^{\max}$ light bandlimit is set, the smaller the scale of the reconstruction filter. Using a smaller filter subsequently requires a higher sampling rate to avoid spotty artifacts. If $\Omega_y^{\max}$ is set too low then the reconstruction filter sizes will be large and overblurring may occur.

Undersampling can be more visually noticeable during animation as it can lead to flickering artifacts. This is shown in the supplementary video and discussed in § 4.6.4.

In Figure 4.16 we can see how the sampling rate and $\Omega_y^{\max}$ light bandlimit interact with each other. As we increase $\Omega_y^{\max}$ we see more and more details in the shadow. If we use a large value for $\Omega_y^{\max}$ but keep a low sampling rate, we can start to see noise (upper left image in Figure 4.16). If $\Omega_y^{\max}$ is too low then the shadows will stay blurry even as we increase samples (bottom row of Figure 4.16). For low sampling rates it is best to keep $\Omega_y^{\max}$ lower (lower left image in Figure 4.16), and for high-quality renders that use a high sampling rate it is best to use a higher value of $\Omega_y^{\max}$ (upper right image in Figure 4.16).

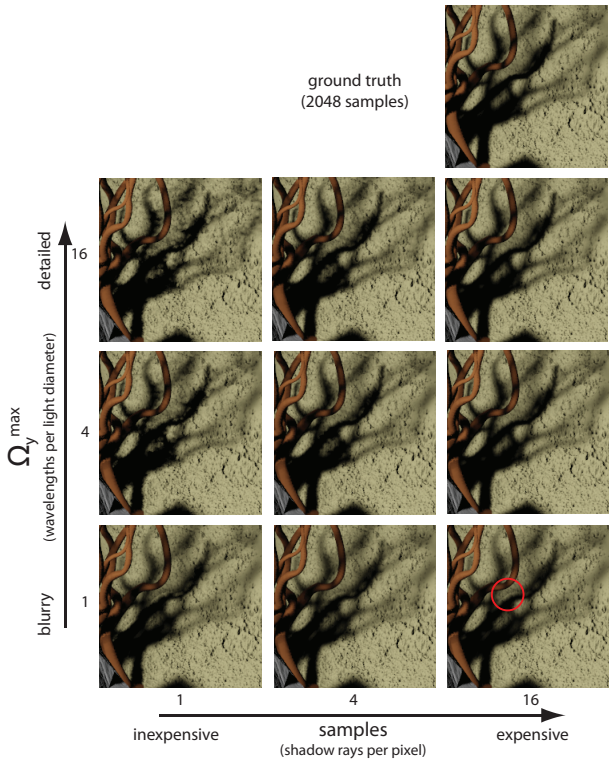

Figure 4.16: An inset from Figure 4.13 rendered with different sampling rates and $\Omega_y^{\max}$ light bandlimit. In general the best quality per cost is shown along the diagonal from lower left to upper right (upper right being the highest quality and the most expensive). The upper left image is inexpensive and noisy, and the lower right image is expensive and overblurred. The red circle highlights a ringing artifact that can occur when depth bounds change suddenly.

Depth bounds $d_2^{\min}$ and $d_2^{\max}$ When computing the $d_2^{\min}$ and $d_2^{\max}$ depth bounds per receiver, sudden changes in these bounds can sometimes get ringing artifacts (this can be seen in the bottom row of Figure 4.16 and in Figure 4.17(ii)). This is due to one pixel using a filter that is much wider than the neighboring pixel's (blurring the $d_2^{\min}$ and $d_2^{\max}$ bounds as is done in Section 3.6.1 would help to alleviate this problem). The depth bounds can also be inaccurate when nearby occluder hit points are culled by the receiver-light frustum, which in turn leads to improper filtering, as seen in Figure 4.17(i). We believe that this is due to our current implementation using a receiver-light frustum that converges to a point instead of properly covering the entire extent of the receiver pixel.

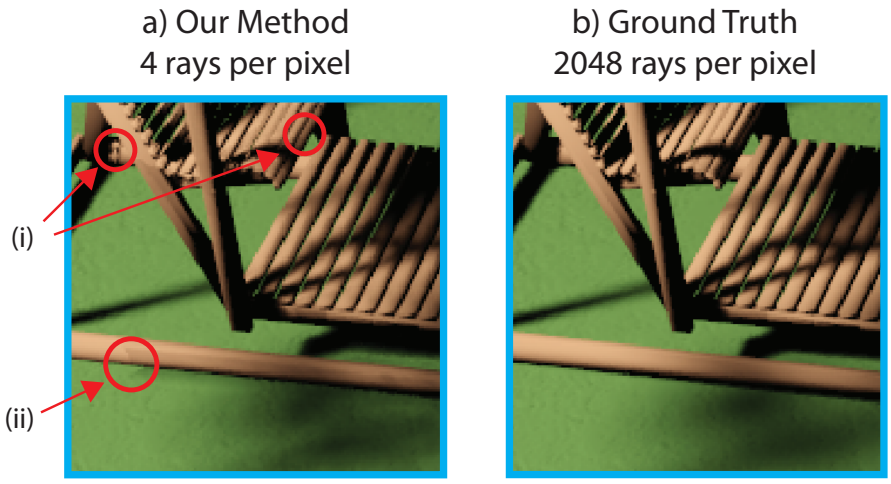

Figure 4.17: Our method with 4 samples produces results which are very close to the ground truth (scene is an inset from Figure 4.12). But there are issues, which can produce subtle differences: (*i*) Shadow regions can be inaccurate if the $d_2^{\min}/d_2^{\max}$ calculation misses occluder samples. (*ii*) In this case the $d_2^{\min}/d_2^{\max}$ calculation is not smooth, which leads to jumps in the filter size.

General BRDFs Our current implementation only handles diffuse BRDFs. While other reflection techniques are often more appropriate for glossy BRDFs, for future work we would like to extend our method to handle any general BRDF. This can be achieved by replacing the lighting response $l(y)$ with the product of lighting and the BRDF response of the current receiver point $\rho(y)$. If we replace L with $(P \otimes L)$ in our analysis, we see that for specular BRDFs containing high frequency content we will have less savings, as it becomes more and more difficult to share rays between receivers. Any second-order terms from surface curvature should be minimal, since our analysis is local to the receiver surface subtended by a single pixel.

Theoretical Limitations For all practical scenes that we have tested, the shape of the occluder spectrum has been a good fit with the wedge shape shown in Figure 4.4b. However, there are extreme cases that break the wedged-shaped spectrum assumption used by our method and previous work [Chai *et al.*, 2000]. We depict the first case in Figure 4.18a. Using an array of planar occluders with length and separation proportional to the distance to the receiving plane, the final shadow is roughly a triangle wave (the signal will be an exact triangle wave for infinitely wide area lights). Using this setup, we can create arbitrarily high shadow frequencies with no change in amplitude,

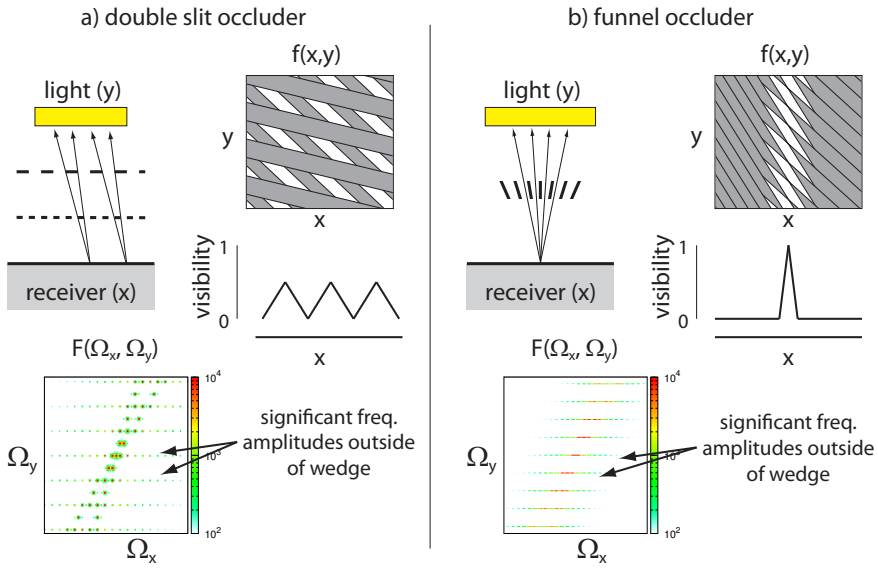

Figure 4.18: Failure cases for our method. (a) A double slit configuration that causes the visibility of the receiver to change roughly as a triangle wave. (b) A funnel configuration that shadows all areas except for one small area that transitions to full visibility. The slanted line segments of the funnel have a wedge shaped occlusion signal in (x, y). In both failure cases the precise shapes of the occluders create regular patterns aligned along the y-axis, creating sharp changes in visibility across the receiver x-axis. In the Fourier domain there is significant energy in $F(\Omega_x, \Omega_y)$ that exists outside of the modeled wedge.

for any two depths, by scaling the length and gaps between occluders closer and closer to zero.

It is also possible to create a funnel-shaped occluder that provides compete visibility to an arbitrarily small area, which then quickly fades to no visibility outside of the area, as shown in Figure 4.18b. By squeezing the funnel edges closer and closer together, we can achieve a sharper and sharper "spike" in visibility. The visibility in (x, y) space for both of these cases is also shown in Figure 4.18b. Both cases apply directly to previous work in light field rendering [Chai *et al.*, 2000], and both cases can be applied to motion blur by replacing the receiver plane with a camera that moves across different x positions over time (Chapter 3). In most practical scenes some high frequencies may exist due to correlation of occluders, but the amplitude of these high frequencies will usually be very low compared to the overall signal.

4.8 Discussion

We have presented a new frequency analysis of complex occluders, and a rendering algorithm that leverages sparsity in the Fourier domain of the 4D light field. We have shown large speedups for a range of complex occluders and scene configurations. Furthermore, our results show that our method excels when dealing with very soft shadows, which is precisely where other methods have the most difficulty.

Our method delivers the biggest performance gains for soft shadows cast by mid to far occluders. However, we could extend our current system to gracefully handle more general cases of self shadowing by subdividing occlusion data into multiple light fields. This could provide a large improvement to our method's performance when occluders are visible at many different depths.

Looking forward, we expect that our generalization of sheared filtering to irregular integrands, as well as the use of more sophisticated filtering techniques, will spur further advances for rendering and other areas.

Chapter 5

Shadows from Distant Lighting

5.1 Introduction

In the previous chapter we greatly reduced the number of shadow rays required for planar light sources. However, this previous analysis does not allow us to model the large angular domain from distant lighting, and only focuses on the diffuse component of the BRDF when doing the frequency analysis. In this chapter we will introduce a new filter that is well suited for large angular light sources, and we will also do a Fourier analysis of shadows using general BRDFs.

Modern production rendering algorithms often compute low frequency hemispherical occlusion, where the surrounding environment is approximated to either be a solid white dome (ambient occlusion), or a series of low frequency spherical harmonics. Two different bodies of work related to ambient occlusion were given scientific Academy Awards in 2010 [AcademyAwards, 2010], and the movie Avatar used ray-traced ambient and spherical harmonic occlusion for lighting and final rendering [Pantaleoni *et al.*, 2010]. While fully sampling the surrounding illumination at each receiver is the completely accurate way to compute global illumination, these approximations of distant lighting work well in practice. Another advantage is that the ambient occlusion and spherical harmonic calculations are independent of the final lighting environment and can be reused throughout the lighting process.

Ray-traced occlusion is often very expensive to compute due to the large number of incoherent ray casts. The authors of the PantaRay system state that they typically shoot 512 or 1024 rays per shading point to compute occlusion [Pantaleoni *et al.*, 2010]. One of their test frames from

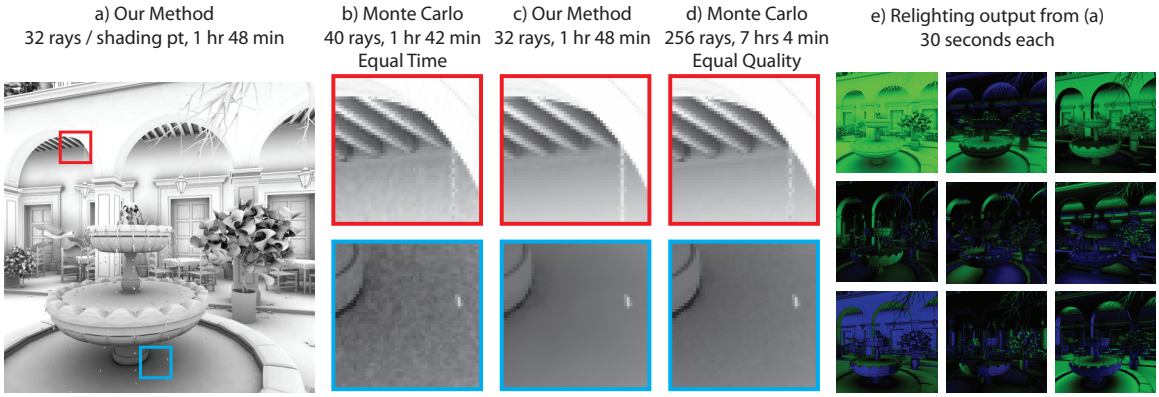

Figure 5.1: (a) A visualization of ambient occlusion produced by our method. This scene used 32 samples per shading point, 13 rays in the sparse sampling pass (41 min) and 19 rays in the second pass in areas with contact shadows (1 hr 7 min). Total running time for both passes was 1 hr 48 min. (b) Closeups of Monte Carlo using equal time (40 samples, 1 hr 42 min), noise can be seen. (c) Closeups of our method. (d) Closeups of Monte Carlo with equal quality (256 samples, 7 hrs 4 min). (e) At little extra cost our method can also compute spherical harmonic occlusion for low frequency lighting. While computing (a) our method also outputs directional occlusion information for 9 spherical harmonic coefficients (green is positive, blue is negative).

the movie Avatar took over 16 hours and 520 billion rays to compute an occlusion pass. While our scenes do not have the complexity of production environments, our method shows substantial performance benefits with scenes of moderate complexity. As scenes become more computationally bound by ray tracing, the benefits of our method increase.

We propose a method that speeds up occlusion calculations by shooting fewer rays and sharing data across shading points (shown in Figure 5.2). We present a new frequency analysis of occlusion from an omni-directional distant light source that also includes normal mapping and a general BRDF at the receiver point. Using this analysis, we develop a method to share rays across the full hemisphere of directions, vastly cutting down on the number of expensive incoherent ray casts. Our method makes a number of important contributions:

Frequency Analysis of Distant Lighting: We present a frequency analysis of distant lighting from all possible incoming directions over the hemisphere in Section 5.3. We first derive new equations to handle distant lighting by splitting up the spherical domain into linear subdomains

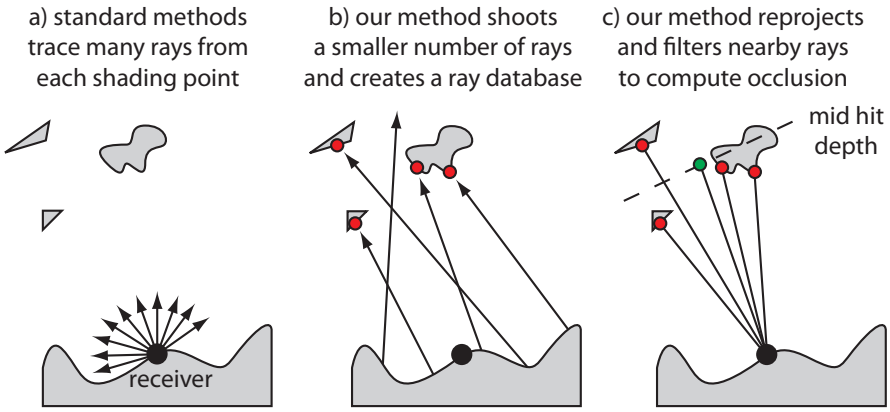

Figure 5.2: (a) We show a simple scene in flatland. Standard methods for computing ray-traced ambient occlusion shoot many rays. (b) Our method shoots a sparse set of rays in a first pass and saves them to a point cloud. Red dots are ray hits. (c) In a second pass we use our theory to reproject and weight nearby rays to compute directional or ambient occlusion. The green dot represents the target point of the unoccluded ray (the intersection between the orginal ray and the mid depth plane).

(such as cube map faces). We then derive the appropriate bandlimits and filter sizes for each linear sub-domain. This theory is used as the basis for our new rotationally-invariant filter.

General BRDFs and Normal Mapping: We also show how the occlusion signal interacts with general BRDFs. We show that the sum of the lighting and BRDF bandlimit determines the cutoff frequencies for occlusion. Furthermore, as long as the surface BRDF is bandlimited, high frequency changes in the normal do not affect our occlusion calculations. Our method takes advantage of this by sparsely sampling occlusion, which greatly reduces the the number of expensive ray casts. We show results with sparse occlusion sampling, glossy BRDFs and low frequency environment lighting (Figure 5.8).

Rotationally-Invariant Filter: We present a filter that uses the above theory, modified such that it is rotationally-invariant (Section 5.4). This property of our filter allows us to handle large angular domains without needing to stitch together linear sub-domains. The filter is intuitive, easy to implement, and constructed using the frequency analysis in Section 5.3. Results with our filter are shown in Figures 5.1, 5.7, 5.8, and 5.9.

5.2 Related Work

Shadows and Ambient Occlusion: Ambient occlusion is an approximation of global illumination that is simply the aggregate visibility for a solid white dome [Zhukov *et al.*, 1998; Landis, 2008]. Spherical harmonic occlusion improves on this, computing the aggregate visibility for a number of low order spherical harmonics. We focus on the methods most closely related to our paper, and we refer the reader to a survey of recent ambient occlusion techniques [Méndez-Feliu and Sbert, 2009].

The PantaRay system uses GPU ray tracing, various spatial hierarchies, and geometric LOD to compute spherical harmonic occlusion [Pantaleoni *et al.*, 2010]. Our method is complementary because we focus on reducing the number of rays cast, whereas PantaRay focuses on reducing the cost per ray. Another recent method reduces cost per ray by using similar sample patterns across many receivers so that they can process multiple receivers in parallel on the GPU [Laine and Karras, 2010]. Pixar's RenderMan represents distant occluding geometry using an octree that contains point sampled data as well as spherical harmonics at parent nodes, and rasterizes this data onto a coarse grid at each receiver [Christensen, 2008]. We compare our results to point based occlusion, and discuss the relevant tradeoffs with both methods in Section 5.6.

Interactive techniques for approximating ambient occlusion are also used. Ambient Occlusion Volumes compute analytic occlusion per polygon [McGuire, 2010]. Because some polygons may be double counted, the method approximates the aggregate occlusion using a compensation map, whereas our method samples occlusion using ray tracing and does not suffer from double counting. Screen space methods can be very efficient, but may miss geometry that is not directly visible to the camera [Bavoil and Sainz, 2009]. Since we focus on ray tracing our method accounts for all relevant occluders.

Frequency Analysis and Reconstruction: Our method builds upon recent sheared filtering techniques [Chai *et al.*, 2000]. Other methods have also examined occlusion in the Fourier domain [Soler and Sillion, 1998; Durand *et al.*, 2005; Ramamoorthi *et al.*, 2005; Lanman *et al.*, 2008]. We extend the theory to include distant lighting, a general BRDF, and high frequency normal maps. We also introduce a rotationally invariant filter that uses the theory for sheared filtering but is able to orient itself in any direction across a large angular domain.

A number of other techniques have also shared data between neighboring receiver points to

cut down on computation. Irradiance caching is used to filter sparse samples of low frequency indirect lighting [Ward *et al.*, 1988]. Irradiance caching aggregates angular information, and then shares it spatially, whereas our method is able to share samples across both space and angle. Radiance caching [Křivánek *et al.*, 2005] shares recorded radiance values and gradients between surface points while assuming that the visibility of shared radiance samples does not change between receivers. Our method determines which samples are appropriate to share by setting the filter radius based on the minimum and maximum occluder depths. Irradiance decomposition uses low frequency radiance caching for far-field components and switches to a heuristic for occluders closer than a fixed depth chosen by the user [Arikan *et al.*, 2005]. Recently Lehtinen et al. [2011] developed a method that locally reconstructs multiple points within a pixel using the GPU. Unlike these methods, the signal processing framework used by our method smoothly scales our filter and tells us how much information to share.

Sparse Transport Computation: Precomputed Radiance Transport is the foundation for current relighting methods [Sloan *et al.*, 2002; Ng *et al.*, 2003]. However, the precomputation time required for these methods is often prohibitive when using standard Monte Carlo sampling. A number of methods have used various techniques to sparsely sample light transport. Row-column sampling uses shadow maps to sparsely compute light transport for a single surface point to all lights (one row), or a single light to all surface points (one column) [Hašan *et al.*, 2007]. The big advantage of row-column sampling is the batch visibility computation achieved by using shadow maps, whereas our method uses ray tracing to avoid shadow map artifacts. Huang et al. sparsely precompute the light transport matrix by only densely sampling the angular domain at selected "dense" vertices [Huang and Ramamoorthi, 2010]. Our method extends this by more intelligently sharing rays across space and angle. Our method also computes filter widths based on the range of depths of the occluders.

5.3 Theory

We first discuss the basic reflection equation and derive equations for occlusion from distant lighting in the primal domain. We then look at preliminary Fourier derivations taken from previous work, before deriving new equations that give insight into occlusion and reflection under distant lighting.

Our final reconstruction uses a filter that is based on our theory but uses geometric measures that can rotate to any given sample in the hemispherical domain.

5.3.1 Occlusion from Distant Lighting

5.3.1.1 Preliminaries

The reflected radiance $h(x, \omega_o)$ at a surface point x in direction ω_o can be written as follows:

$$h(x, \omega_o) = \int r(x, \omega_i, \omega_o) l(\omega_i) f(x, \omega_i) \, d\omega_i \quad .$$

The reflected radiance is computed by integrating over all incoming light directions ω_i. Inside the integral is a product of the spatially varying BRDF $r(x, \omega_i, \omega_o)$ (which includes the damped cosine term for compactness), the distant lighting $l(\omega_i)$, and occlusion $f(x, \omega_i)$.

Our method focuses on direct lighting with a fixed camera such that the viewing angle ω_o is fixed for a given spatial location x,

$$h(x) = \int r(x, \omega_i) l(\omega_i) f(x, \omega_i) \, d\omega_i \quad . \tag{5.1}$$

To compute ambient occlusion, we simply set $l(\omega_i)$ to a constant value of 1, and $r(x, \omega_i)$ to a clamped cosine $\max(n(x) \cdot \omega_i, 0)$ in Equation 5.1 (where $n(x)$ is the surface normal). For more accurate low frequency relighting, we replace the lighting $l(\omega_i)$ term with a set of low order spherical harmonic basis functions $[s_0(\omega_i), ..., s_n(\omega_i)]$, and a corresponding intermediate response function $[h'_0(x), ..., h'_n(x)]$ (visualized in Figure 5.1e). Using these intermediate values, we can then easily relight the scene by projecting any distant lighting $l(\omega_i)$ into the orthonormal spherical harmonic basis, taking the resulting coefficients $[l_0, ..., l_n]$, and performing a dot product:

$$h(x) = \sum_{i=0}^{n} h'_i(x) l_i \quad . \tag{5.2}$$

Directional and ambient occlusion are primarily used to aid in the realistic approximation of slightly glossy or matte components of a BRDF. If a BRDF has sharper specular reflections this is usually calculated more directly using a different reflection algorithm.

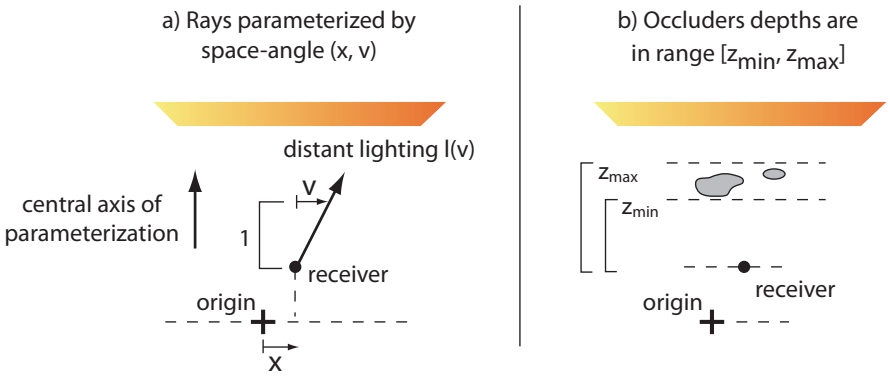

Figure 5.3: For our theoretical derivation we split up the environment into distinct faces and use a linearized angle. We parameterize the receiver based on the spatial offset x from the origin. The linearized angle v is the offset from the central direction of the cube map at a plane one unit away from the receiver. The occluders are bounded by a range of distances $[z_{\min}, z_{\max}]$ from the receiver.

5.3.1.2 Distant Lighting in Primal Domain

The above angular parameterization using ω is not easy to analyze in a Fourier context. We could expand the lighting into spherical harmonics, but proceeding to analyze the spatial-angular occlusion of $f(x, \omega)$ and subsequent transport becomes intractable. Thus, like Durand et al. [2005], we use the Fourier basis with a linearized measure of angle.

We first reparameterize the circle of angles in flatland to a square map (the analog of a cube map in 3D). For each map face we define a linearized angle v measured against that map's central axis of projection. The v measures a direction vector's offset from the axis of parameterization at a plane 1 unit away (shown in Figure 5.3), similar to previous methods [Soler and Sillion, 1998; Durand *et al.*, 2005]. This parameteriztion allows us to analyze how the relevant signals (distant lighting, occlusion and BRDF) interact, as opposed to using spherical harmonics. We analyze the contribution of each lighting "face" separately, and the final answer is the sum of all face contributions. Our final filter uses the following derivations, but with a simpler geometric measure that does not require reparameterization to cube maps.

We define visibility along a ray as $f(x, v)$ where x is a spatial measure perpendicular to the central axis of projection (Figure 5.3). We first look at a single planar occluder defined by a binary visibility function $g(x)$ at constant distance z from the receiver. We will extend this to occluders

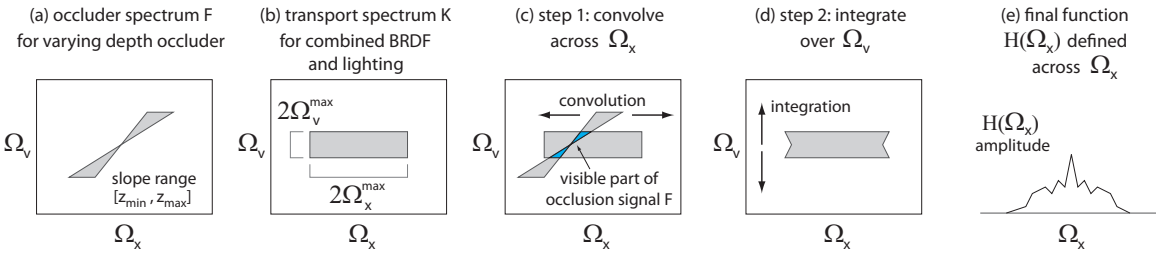

Figure 5.4: (a) The occluder spectrum F in the Fourier domain. (b) The combined lighting and BRDF response has small angular extent for low frequency transport ($\Omega_v^{\max}$), but may have large spatial extent if the normal varies rapidly ($\Omega_x^{\max}$). (c) The inner integral of Equation 5.7 is a convolution across Ω_x. Note that F effectively becomes bandlimited by the $\Omega_v^{\max}$ bandlimit of K. (d) The outer integral of Equation 5.7 integrates over Ω_v. (e) The final result is $H(\Omega_x)$, the Fourier spectrum of the spatial occlusion function $h(x)$.

with a range of depths later.

$$f(x, v) = g(x + zv) \quad . \tag{5.3}$$

We now have a re-parameterized BRDF $r(x, v)$ and distant lighting function $l(v)$. With this change of variables we have to adjust for the Jacobian $|\partial\omega_i/\partial v|$, which we incorporate into $l(v)$ (note that Jacobian calculations will not be necessary for the final rotationally invariant version of our filter). Now the reflection equation for $h(x)$ is:

$$\begin{aligned} h(x) &= \int r(x, v) l(v) f(x, v)\, dv \\ &= \int r(x, v) l(v) g(x + zv)\, dv \quad . \end{aligned} \tag{5.4}$$

We now combine the BRDF and lighting $r(x, v)l(v)$ into a new combined response function $k(x, v)$,

$$h(x) = \int k(x, v) g(x + zv)\, dv. \tag{5.5}$$

5.3.2 Fourier Analysis

5.3.2.1 Preliminaries

When we can express a two-dimensional function f in terms of a one-dimensional function g in the form given in Equation 5.3, previous work has shown that the Fourier transform of $f()$, which we name $F()$, lies along a line segment in the Fourier domain [Shinya, 1993; Chai *et al.*, 2000; Durand *et al.*, 2005]:

$$F(\Omega_x, \Omega_y) = G(\Omega_x)\delta(\Omega_y - z\Omega_x) \quad . \tag{5.6}$$

where $G()$ is the 1D Fourier transform of $g()$, and δ is the delta function. Intuitively, if you have a 1D function embedded in a 2D domain, it makes sense that the frequency spectrum is also 1D.

If all occluders are planar and lie along a single depth z then the occlusion spectrum corresponds exactly to Equation 5.6. In practical scenes with a range of depths $[z_{\min}, z_{\max}]$ the occlusion spectrum F is a double wedge determined by the distance between the occluder and the receiver [Chai *et al.*, 2000]. The double wedge can be thought of as a spectrum that is swept out by many line segments that correspond to z values between $[z_{\min}, z_{\max}]$. This is shown in Figure 5.4a.

5.3.2.2 Fourier Spectrum for Distant Lighting

We now derive the occlusion spectrum for distant lighting, as well as the interaction between the spectra considering complex occluders and a surface with a general BRDF and normal maps. Both of these are novel contributions of our paper.

Response Function in the Fourier Domain: The response function spectrum $K(\Omega_x, \Omega_v)$ is shown in Figure 5.4b. Because r and l are multiplied the spectrum K is the convolution $R \otimes L$. Because L has no spatial dimension we can conclude that the spatial bandlimit of K, $\Omega_x^{\max}$, is simply the spatial bandlimit of R. The angular bandlimit of K, $\Omega_v^{\max}$, is the sum of the angular bandlimits for R and L.

Normal Mapping in the Fourier Domain: Rapid variation in the normal rotates the BRDF and causes the spectrum K's spatial bandlimit, $\Omega_x^{\max}$, to be large. However, rapid rotation in the normal does not affect the angular bandlimit. To see that this is true we can split up the 2D Fourier transform into two 1D transforms, and first take the Fourier transform in v for a given fixed value of x. After

transforming in v each slice of the angular transform along $\Omega_v = v_0$ is zero where $v_0 > \Omega_v^{\max}$, and therefore the spatial Fourier transform of this slice is zero. Therefore if the BRDF at every x location is bandlimited by $\Omega_v^{\max}$, then the final spectrum K will be bandlimited by $\Omega_v^{\max}$ as well. We call the the portion of the F spectrum that is non-zero after after bandlimiting by $\Omega_v^{\max}$ the "visible frequencies" of F, as seen in Figure 5.5.

Lighting and Surface Reflection in the Fourier Domain: Taking the Fourier transform of $h(x)$ in Equation 5.5 is not trivial because the response function $k(x, v)$ has two dimensions but one of the dimensions is integrated out (see Appendix C for the derivation). In the end we get the following:

$$H(\Omega_x) = \int \left(\int F(\Omega_x - s, -t) K(s, t)\, ds \right) dt, \tag{5.7}$$

where s is a temporary variable used to compute the inner 1D convolution across Ω_x (shown in Figure 5.4c). The t variable is used to compute the outer integral across the Ω_v dimension of the resultant spectrum (as seen in Figure 5.4d). Finally we are left with $H(\Omega_x)$, the spectrum of occlusion across the spatial axis (Figure 5.4e).

Discussion: The above analysis includes a number of important results. First, we have derived bandlimits for the occlusion spectrum from distant lighting. We split the angular domain of directions into sub-domains, and then reparameterized each sub-domain separately using a linearized angle formulation.

Second, our frequency analysis of occlusion can handle surfaces with a general BRDF and high frequency normal maps. We have shown that the transport spectrum K may have high frequencies in the *spatial domain* due to high frequency changes in the normal. However, an important result is that the visible portion of the occlusion spectrum F is still low frequency in the *angular domain* and is bandlimited by $\Omega_v^{\max}$ (Figure 5.4c).

Our method directly reconstructs visibility $f(x, v)$ using sparse ray casting. We can use sparse sampling because the compact shape of the sheared filter in the Fourier domain lets us pack Fourier replicas closer together. Our method densely samples the combined lighting-BRDF term $k(x, v)$ which is much cheaper to sample and may include high frequency normal maps. Attempting to

share the integrated product $h(x)$ directly, as in irradiance caching, has less benefit because of the possible high spatial frequencies in $H(\Omega_x)$.

5.3.3 Sheared Filtering Over Linear Sub-Domains

Now that we have defined the sparse shape of the visible occluder spectrum, we can design a sheared filter that compactly captures the frequency content of the signal we care about. We can then transform the filter back to the primal domain where it is used to reconstruct our final answer, while allowing for sparse sampling and the sharing of information across pixels [Chai *et al.*, 2000]. The shape of the sheared filter will guide our design of our rotationally-invariant filter in Section 5.4.

The visible parts of the occlusion spectrum F that we need to capture are shown in Figure 5.5. The Fourier footprint for a standard reconstruction filter is shown in Figure 5.5a, and a sheared filter that tightly bounds the visible parts of F is shown in Figure 5.5b. By applying a scale and shear we can transform the filter shown in Figure 5.5a to the one shown in Figure 5.5b.

We can see from the measurements in Figure 5.5 that, in the Fourier domain, our filter is scaled along the Ω_x axis and sheared in Ω_x per unit Ω_v. In the primal domain we need to scale along the x-axis by the inverse amount, and shear in v per unit x:

$$\text{primalScale} = \frac{2\Omega_{\text{pix}}^{\max}}{\Omega_v^{\max}} \left[\left(\frac{1}{z_{\min}} \right) - \left(\frac{1}{z_{\max}} \right) \right]^{-1}, \tag{5.8}$$

$$\text{primalShear} = -\frac{1}{2} \left[\left(\frac{1}{z_{\min}} \right) + \left(\frac{1}{z_{\max}} \right) \right]. \tag{5.9}$$

where $\Omega_{\text{pix}}^{\max}$ represents the smallest wavelength that can be displayed. We set $\Omega_{\text{pix}}^{\max}$ to be $(0.5/\mathit{shadingDiameter})$, meaning that the highest frequency that can be captured is half a wavelength per output diameter. We set $\Omega_v^{\max}$ to be 2.0, which approximates that the BRDF * lighting function can be captured with two wavelengths per unit of linearized angle (45 degrees).

In areas of sharp contact shadows the visible portion of F may extend beyond the $\Omega_{\text{pix}}^{\max}$ bandlimit of the standard filter. In this case we must make sure that our filter does not capture any portion of F outside the $\Omega_{\text{pix}}^{\max}$ bandlimit. We check this by testing if $\Omega_v^{\max}/z_{\min} > \Omega_{\text{pix}}^{\max}$, and if so we revert to brute force Monte Carlo. Our implementation stores separate $z_{\min}$ values for different portions of the hemisphere, see Section 5.5 for details.

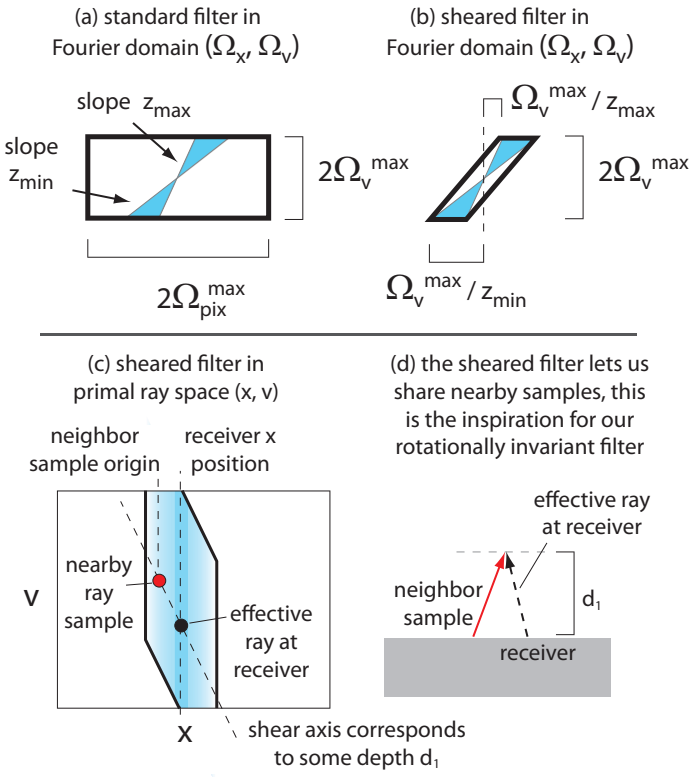

Figure 5.5: (a) The visible spectrum of F is shown, as well as the Fourier transform of a standard filter that is axis-aligned in x and v. (b) A sheared filter that tightly bounds the visible parts of F. (c) The theoretical sheared filter graphed in (x, v) space. The sheared filter effectively reprojects rays along the shear axis which is based on harmonic average of depth bounds $[z_{\min}, z_{\max}]$. However, the sheared filter uses a linearized angle v that depends on a fixed axis of projection. (d) A visualization of how a sheared filter shares samples in the primal domain. We use this as inspiration for the rotationally invariant filter presented in Section 5.4 that does not require a fixed axis of projection.

We can now use Equations 5.8 and 5.9 to construct a filter that operates over a linearized sub-domain of the sphere (as shown in Figures 5.5c). Intuitively this takes a nearby sample, and warps it to create an effective ray that originates at the current receiver point, pictured in Figure 5.5d. However, splitting the sphere into multiple sub-domains requires stitching together the results from each filtering operation. We solve this problem by introducing our rotationally-invariant filter.

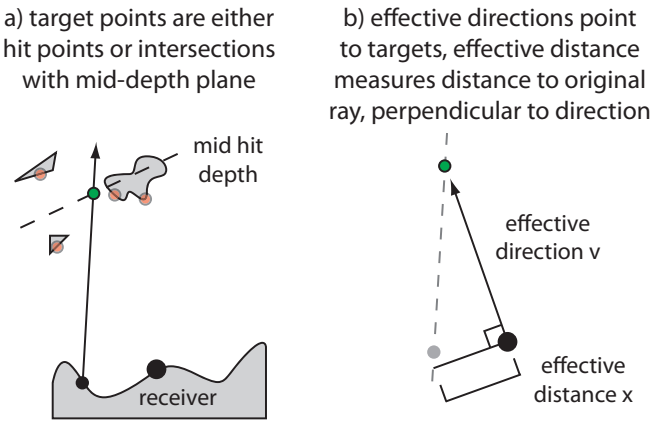

Figure 5.6: (a) To use our rotationally invariant filter filter we first compute target positions for the current receiver. Unoccluded rays use an intersection with a plane that goes through the harmonic average of $[z_{\min}, z_{\max}]$ for the current receiver. (b) The effective directions for the samples simply connect the current receiver to the target positions. To calculate effective distance we compute the intersection of the original ray and a plane that goes through the current receiver and is perpendicular to the effective direction.

5.4 Rotationally-Invariant Filter

We now develop our rotationally-invariant filter that allows us to easily filter over large angular domains. The sheared filter from the previous section essentially has two outputs: what filter weight to apply to a given sample, and also what is the effective ray direction when we warp a nearby sample to the receiver point. The directional information is useful when we bucket effective ray directions to account for non-uniform sampling densities across the receiver.

Implementing a set of finite linear subdomains has a number of drawbacks. There may be discontinuities where the linear subdomains meet, and we have to account for the Jacobian of the linearization in the transfer function $k(x, v)$. One possible strategy would be to subdivide the sphere into even more subdomains. Of course this may reduce possible artifacts but we would still have to worry about discontinuities and Jacobians. Our approach is to take the limit of subdivision where each sample that we consider is defined to be in its own infinitesimal subdomain. This leads to a rotationally-invariant filter that uses the theory from Section 5.3 and is easy to implement.

Figure 5.5d and Equation 5.9 together show that warping a sample ray to an effective ray origi-

nating from the current receiver is based on the shear, and the shear in turn is based on the harmonic mean of $z_{\min}$ and $z_{\max}$. If we consider each sample to be in its own infinitesimally small subdomain, then for occluded rays we can say that the occluder hit point is the only occlusion distance z that we care about. In this case the effective ray direction is simply the vector from the receiver to the sample hit point. If the ray is unoccluded, we make a ray "target" point where the ray intersects a mid depth plane based on the harmonic mean of occluders in a surrounding region. The calculation of the depth plane will be explained in more detail in Section 5.5.

To compute a filter weight we must compute the x coordinate for the sample. Using an infinitesimal subdomain our axis of projection is defined to be the same as our effective ray direction. The x measure needs to be tangent to our axis of projection, and we compute the intersection point between the sample ray and plane that is perpendicular to the effective ray direction and goes through the receiver point. The x measure is simply the offset vector from the receiver to this intersection point. Our rotationally-invariant filter with the computed target point for a given sample is shown in Figure 5.6a, and the filter effective direction and spatial x value are shown in Figure 5.6b.

Given a ray target point we compute an effective direction that originates from the origin, and then compute the tangent x coordinate for the ray sample. These computations are tightly bound with the derivations in Section 5.3. We used Equation 5.9 and the intuitive notion of the filter shear (Figure 5.5d) to guide setting the effective direction for a sample. The x coordinate and $primalScale$ from Equation 5.8 are used as the input and scale respectively for computing filter weights.

Discussion: Our rotationally-invariant filter operates on an infinite number of of small subdomains, and it is fair to ask how this affects the earlier Fourier derivations. In Equations 5.8 and 5.9 the only Fourier bandlimit that is affected is the transfer function $\Omega_v^{\max}$ angular bandlimit. Instead of computing $\Omega_v^{\max}$ for a fixed set of linear subdomains, we can instead precompute the largest $\Omega_v^{\max}$ over a range of different localized areas and orientations both for each BRDF and for the distant lighting. As stated before, the $\Omega_v^{\max}$ value for the transfer spectrum K is the sum of angular bandlimits for the BRDF and lighting.

5.5 Implementation

Our implementation takes a two pass approach: sampling followed by filtering. Our first pass shoots a sparse set of rays (4 - 32 rays per shading point) and writes the rays to a point cloud. This is similar to common Monte Carlo implementation, except for the low number of rays used (Figure 5.2b). The second pass reads in the point cloud and filters over a large number of nearby ray samples at each shading point to compute a smooth and accurate result (Figure 5.2c).

We implemented our algorithm by writing shaders and a C++ plugin for a RenderMan compliant renderer [Pixar, 2005]. The core filtering algorithm runs inside our plugin for each shading point and can be seen in Algorithm 1. At a high level we compute depth bounds for different sections of the hemisphere (lines 1 to 1 in Algorithm 1), then use these depth bounds to filter neighboring samples (lines 2 to 2).

Computing Bounds for the Hemisphere The first thing we do is compute the $[z_{\min}, z_{\max}]$ depth bounds for the current receiver (lines 1 to 1). To compute tighter depth bounds we divide the hemisphere of visible directions into cells of equal projected area and compute depth bounds per cell (our implementation subdivides the disk into 8x8 cells). We also filter results per cell, which helps to compensate for possible non-uniform sample densities.

In our implementation, a user-specified screen space radius determines the set of potential neighbor samples (for our results we used a radius of 8-16 pixels). For each neighbor sample that is occluded and whose hit point is in the correct hemisphere, we compute the effective sample distance x and effective direction v (line 1 as described in section 5.4). We then use the direction v to lookup the appropriate cell (line 1) and update the cell's depth bounds (line 1).

After all samples are processed, we compute the spatial radius and mid depth for each cell (lines 1 and 1). We compute the spatial radius by multiplying the current micro polygon diameter by `primalScale` (from Equation 5.8). In Equation 5.9 we can see that the shear value is simply the harmonic average of the minimum and maximum depths. Therefore we store the harmonic average of $z_{\min}$ and $z_{\max}$ for the cell's mid depth. If a cell has zero samples, or if $\Omega_v^{\max}/z_{\min} > \Omega_{\text{pix}}^{\max}$ (see Section 5.3), the `ComputeSpatialRadius()` function marks the cell as requiring brute force computation.

```
   // Filtering algorithm for one shading point
   // 1.  Calculate hemisphere cell info
1  foreach s in SampleImageCache do
2      if s.isOccluded then
3          targetPoint = s.hitPoint;
4          if InCorrectHemi(targetPoint) then
5              (x, v) = ComputeFilterCoords(s, targetPoint);
6              cell = GetCell(v);
7              UpdateCellMinMaxDepth(cell, s);
8          end
9      end
10 end
11 foreach cell in HemisphereCellArray do
12     cell.spatialRadius = ComputeSpatialRadius(cell);
13     cell.midDepth = ComputeMidDepth(cell);
14 end
   // Listing continued in Part 2...
```

Filtering Samples Now that we have $[z_{\min}, z_{\max}]$ depth bounds at each cell we can filter over neighboring samples (lines 2 to 2 in Algorithm 1). We first compute the target point of each sample (lines 2 to 2). For occluded samples, the target point is simply the hit point of the occluded ray (line 2). For non-occluded samples, we use a two step process to compute the target point. We first lookup an initial cell based purely on the ray direction (line 2). We construct a plane that is perpendicular to the central direction of the cell and whose distance to the current receiver is the same as the mid depth of the cell. We set the target point to be the intersection of the sample ray with this plane (line 2).

Once we have the target point we can compute the final sample distance x and effective direction v (line 2). Using the direction v we can lookup the final cell for the sample (line 2). Using the sample distance x and the cell's spatial radius we can compute the filter weight (line 2). With the effective incoming light direction v we can also compute the BRDF and lighting response (line 2).

```
// ...  continued from Part 1
// 2.  Filter over neighboring samples
cellResults = InitializeArray(numCells);
foreach s in SampleImageCache do
    if s.isOccluded then
        focusPoint = s.hitPoint;
    else
        initCell = GetInitialCell(s);
        targetPoint = ComputeTarget(s, initCell);
    end
    if InCorrectHemi(targetPoint) == FALSE then continue;
    (x, v) = ComputeFilterCoords(s, targetPoint);
    cell = GetCell(v);
    if IsMarkedForBruteForce(cell) then continue;
    weight = ComputerFilterWeight(x / cell.spatialRadius);
    (BRDF,light) = ComputeBRDFandLighting(v);
    AddWeightedSample(cellResults[cell.index],
            weight, BRDF, light, s.visibility);
end
NormalizeWeights(cellResults);
return cellResults
```

Algorithm 1: *Our algorithm for filtering results at each shading point (this is implemented in our C++ RenderMan plugin). For each shading point we return an array of values, one per hemisphere cell. Each value either represents a filtered BRDF * lighting value, or a flag that tells the calling RenderMan shader to brute force the corresponding hemispherical cell.*

Many implementations will let the user fade out occlusion beyond a certain distance, so that distant occluders are not counted [McGuire, 2010]. This is especially necessary for indoor scenes. We have incorporated this in our method by testing the distance from the receiver to the sample target point and reducing visibility accordingly.

After we have processed all of the samples, we normalize the total contribution per cell based on the weight in each cell. Any cells with too little weight (we use a weight that corresponds to approximately 2 samples) are marked as requiring brute force computation. This is the end of Algorithm 1, and where our C++ plugin hands control back to the RenderMan shader. The shader then goes through each cell, and uses Monte Carlo ray tracing to compute answers for any cells that needed to be brute forced (for our results we shot a single ray to estimate these cells).

5.6 Results

5.6.1 Setup

All results are 512x512 and were generated on a dual quad-core Xeon 2.33 GHz processor with 4 GB of memory using Pixar's RenderMan Pro Server 15.2. Our plug-in is thread safe and is designed to run in parallel (we used 8 threads for our results).

We used the `trace()` call in RenderMan which finds the nearest hit point regardless of occluder surface. Other functions such as the `occlusion()` function have options to cap the maximum distance of a ray, which could reduce the working set of ray-traced geometry and reduce paging to disk. Of course, with more dense geometry, the problem of paging to disk will easily reoccur. The San Miguel scene (Figure 5.1) is the only scene that was computationally bound by paging to disk, and the Sponza scene (Figure 5.7) was limited by computation (ray tracing and displacement).

5.6.2 San Miguel

In the San Miguel scene (Figure 5.1a) we show a complex scene that contains both large smooth areas, and areas of very high complexity. The smooth areas are challenging because the human visual system is drawn to any small errors or oscillations. Accurately calculating occlusion for areas of high geometric detail and/or contact shadows is difficult because these areas can easily be missed or undersampled.

The scene is 5.2 million triangle faces. Because of the high geometric complexity of the scene RenderMan generated and shaded an average of 5 micro polygons per pixel. Because of the complexity of the scene and the incoherent nature of the rays, geometry was constantly paged in and out of memory. While it is always possible to increase memory on a machine, artists will in turn keep

producing larger models. RenderMan reported using 5.5GB of virtual memory (with 2GB devoted to ray-traced geometry) and 3.5GB of physical memory. This scene is well suited for our method because the cost per ray is very high relative to the cost of filtering over nearby ray samples.

In Figure 5.1, we show equal time and equal quality comparisons with stratified Monte Carlo sampling. Our method took 1 hr and 48 min, spending about 40% of the time in the first pass casting sparse ray samples (13 rays per shading point), and the remaining time in the second pass filtering and casting rays for areas with very close occluders (19 rays per shading point). Our method gave a 4x speed up over Monte Carlo with 256 samples (7 hrs 4 min) as well as an order of magnitude reduction in the number of ray casts. We also show significantly less noise versus Monte Carlo with 40 samples using equal time. Our method is smoother than Monte Carlo with 256 samples in many areas, although our method does have some areas with noise (inside the lip of the fountain) and overblurring (contact shadows with the leaves on the ground). See section 5.6.6 for more discussion on limitations and artifacts.

The filtering operation in our method is more expensive than simple Monte Carlo, which accounts for the discrepancy between our reduction in rays and speed up. One reason for the increased filtering cost is the increased algorithmic complexity of our filtering (shown in Algorithm 1). Another reason is that our filter often needs more samples to achieve a smooth result due to samples being unstratified when they are warped onto the hemisphere of the current receiver.

In Figure 5.1e we also show that our method can output spherical harmonic occlusion. Our RenderMan shader takes the cell reflection values returned from Algorithm 1, calculated per hemispherical cell, and uses environment maps that represent low order spherical harmonic basis functions $[s_0(\omega_i), ..., s_n(\omega_i)]$ as the distant lighting $l(\omega_i)$ for each cell (Equation 5.2). As of version 15.2 RenderMan does not support an efficient method for producing spherical harmonic occlusion using its point based algorithm.

5.6.3 Bumpy Sponza

In the bumpy Sponza scene we apply a displacement shader to show that our method can handle complex occluders as well as high frequency changes in receiver surface and normal orientation. The scene only has 300,000 triangles (before displacement) and fit inside memory during our renders. In Figure 5.7 we compare the quality of our method with point based occlusion. We use the

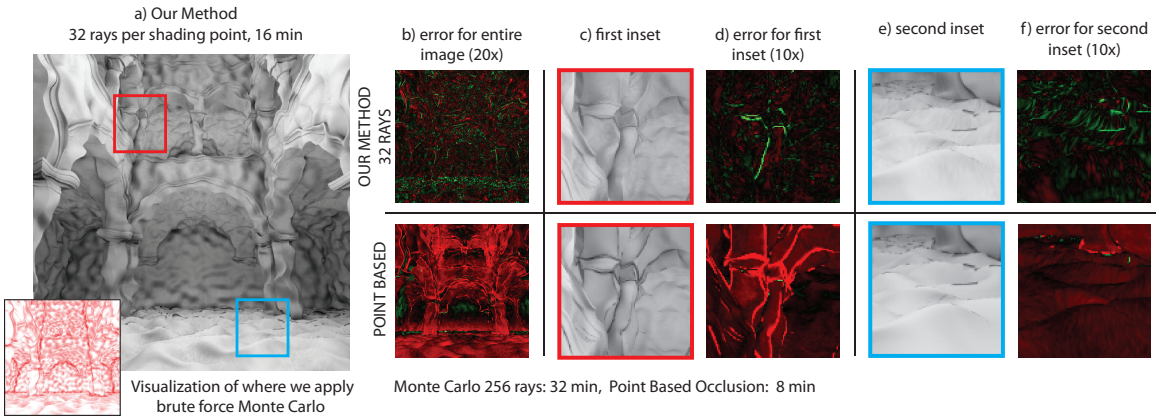

Figure 5.7: This scene shows complex occluders and receivers with displacement. (a) Our method with 32 rays per shading point (9 rays in the first pass, 23 rays in the second pass). As an inset we also show a visualization of where our method reverted back to brute force Monte Carlo. We decide whether to filter over neighboring samples or use Monte Carlo at each hemisphere cell, so many shading points use a mix of both methods. In (c) and (d) the error in our method comes from overblurring and missing occlusion within some creases. Point based occlusion has larger error and over darkens the creases. Green error values are areas that are too bright, red error values are areas that are too dark. In (e) and (f) our method can be seen to have some noise (due to some cells of the hemisphere requiring Monte Carlo integration). The point based result is consistently too dark, but smoother.

RenderMan point based occlusion implementation with high quality settings (6x increase in micro polygon density for initial geometry pass, clamping turned on, rasterresolution set to 32, max solid angle set to 0.01). For this scene our method used 32 rays per shading point and took 16 minutes, while Monte Carlo used 256 rays and took 32 minutes. Our method reduces the number of ray casts by an order of magnitude, and is still 2x faster than Monte Carlo with 256 samples even when the scene fits in memory and the cost per ray is low.

Point based occlusion is a popular solution because it is fast and the results are generally smooth (in this example point based occlusion took 8 minutes). However, it can also have a number of disadvantages. In Figure 5.7d, we visualize the image error for our method and point based occlusion (red is used for areas that are too dark, and green for areas that are too bright). We can see that in some of the crease areas point based occlusion can produce results that are too dark. Our method is

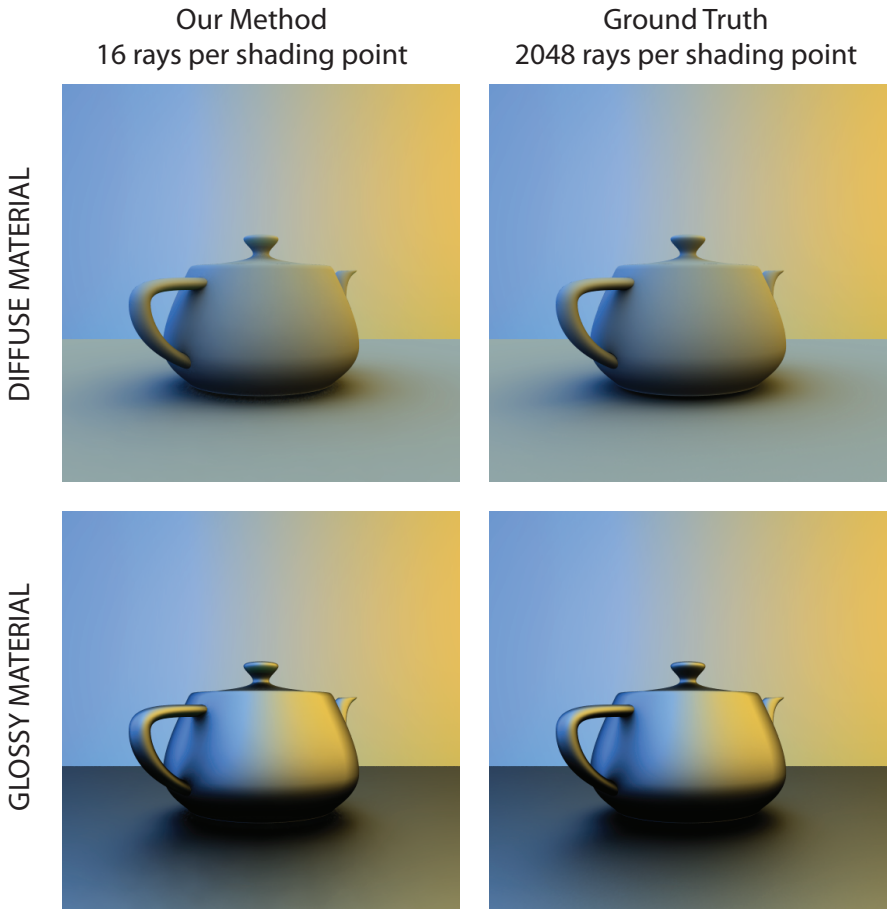

Figure 5.8: Here we show an example of our method with both matte and glossy BRDFs with environment lighting.

slightly too bright due to overblurring and missing some details, but the amplitude of our errors is much less.

In Figures 5.7e and 5.7f our method has noise in some areas. Because of the high frequency displacement many surfaces have nearby occluders which can trigger our method to fallback to brute force Monte Carlo. A visualization of where and to what degree our method used Monte Carlo sampling can be seen in Figure 5.7a. In this scene 74% of shading points used Monte Carlo for less than half of their hemispherical cells. The higher the $\Omega_v^{\max}$ is set, the more often our method falls back to Monte Carlo. Point based occlusion is consistently slightly too dark in this area, but it does produce smooth results.

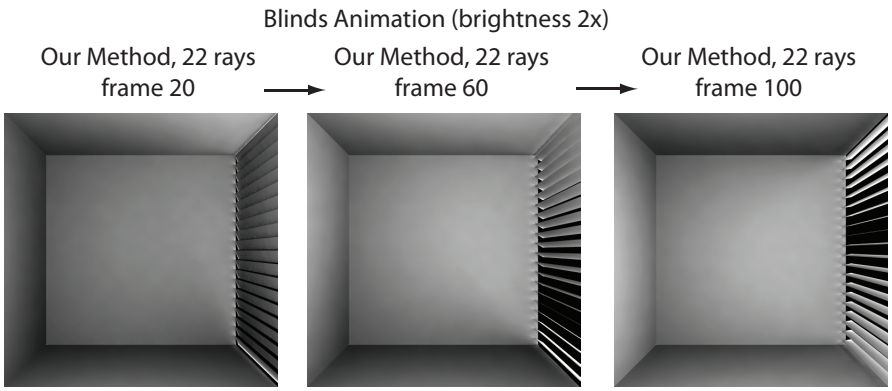

Figure 5.9: Frames from our supplementary video (brightness multiplied by 2). Our method accurately captures the angular content of the small slats.

5.6.4 Glossy

In Figure 5.8, we show a glossy teapot scene demonstrating our method's ability to handle diffuse and glossy surfaces with spherical harmonic lighting. The different colored shadows to the left and right of the teapot show that we are capturing directional occlusion. The shape of the shadow on the ground plane also changes as the material goes from diffuse to glossy. Our method very closely matches ground truth with either a diffuse or glossy BRDF and environment lighting.

5.6.5 Blinds Animation

We also provide a supplemental animation (frames shown in Figure 5.9) that shows a set of blinds rotating together. The angular content of the occluders is very important in this example, and we show that our method still handles this well. Our method has a low amplitude of error overall (Figures 5.10a and 5.10b). At object boundaries the non-uniform distribution of samples can lead to some small errors (in this case the edges of slats are too bright). The point based solution is smooth, but the results are consistently too dark (Figures 5.10d and 5.10e).

5.6.6 Limitations and Artifacts

We now examine the limitations and possible artifacts that can occur in our method. In Figures 5.10c and 5.10f we show the artifacts that can occur when we don't use enough samples in the first pass of our algorithm (we use 4 and 1 rays per shading point for the first pass in Figures 5.10c and

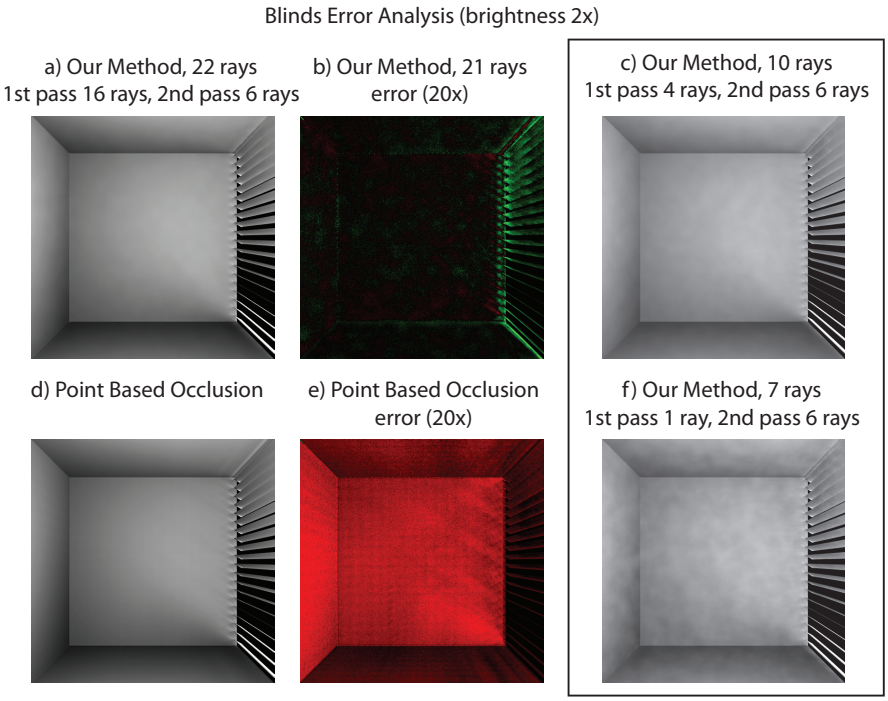

Figure 5.10: We show the errors with point based occlusion and our method for the blinds scene frame 60 (ambient occlusion brightness is multiplied by 2x, green error values are areas that are too bright, red error values are areas that are too dark). (a) (b) Our method produces a scene with overall low amplitude error. (d) (e) Point based occlusion produces an image that is consistently too dark. (c) (f) Output from our method using a reduced ray count. The medium frequency noise is due to too few rays stored in the first pass (4 and 1 rays in (c) and (f) respectively).

5.10f respectively). Because of our wide filtering, any undersampling in the first pass shows up as low amplitude medium frequency error instead of high frequency noise. While the amplitude of these errors is often low, spurious changes in the derivative can be visually noticeable, especially in areas of constant or linearly changing occlusion. Raising the sample count in the first pass of our algorithm reduces, and at a certain point, eliminates this problem.

In areas where we revert to brute force computation, our method can show noise, such as in Figure 5.1a near the lip of the fountain. We can again reduce noise in these areas by increasing sampling density, but this will in turn reduce performance. Our method can also smooth out some areas of detail. In Figure 5.1a some of the contact shadows under the leaves and around the detailed geometry of the door are slightly washed out as compared to ground truth.

5.7 Discussion

We have presented a new frequency analysis for occlusion that incorporates distant lighting, general BRDFs, and high frequency normal maps for complex receivers and occluders. In addition, we have also provided a new rotationally-invariant filter that is parameterized according to our analysis, and that is capable of sharing samples across a large angular domain. Our results show that our method can substantially reduce the number of rays cast, and can lead to large speed up in scenes that are computationally bound by ray tracing costs.

For future work we want to investigate methods for stratifying samples in such a way that the results will be stratified across multiple receivers. We also want to investigate alternative ways to integrate over sample point sets. If we could improve our accuracy when dealing with smaller point sets of non-uniform density, we could reduce our filter radius and speed up the filtering process substantially.

In summary, directional occlusion is of increasing importance in Monte Carlo rendering. We have taken an important step towards fully exploiting the space-angle coherence. We expect many further developments in this direction, based on a deeper analysis of the characteristics of the occlusion function.

Chapter 6

Conclusion

This thesis has presented new fourier analysis and filtering methods for many of the most expensive tasks in rendering. In each case we examined the relevant signals in the Fourier domain and detailed how these signals combined to form the resulting image. Based on these observations we proposed new filtering techniques that customized the filtering at each shading point to take advantage of the underlying signals being processed.

We first looked at space-time signals for computing motion blurred images. We showed that the spectrum is often contained within a double wedge based on the minimum and maximum velocities. We then proposed a new filter that is sheared to match the underlying velocity of the signal (Chapter 3).

We then extended the technique to irregular integration problems by looking at complex shadows cast on complex receivers. We created a 4D ray database independent of receiver position, and transformed each shading point's sheared filter into this parameterization (Chapter 4).

We then looked at shadows from distant lighting with large angular extent. We also extend our theory to take into account general BRDFs at the receiver surface. We propose a new rotationally invariant filter that easily handles integration over the hemispherical domain (Chapter 5).

6.1 Future Work

For future work we would like to analyze a larger class of indirect lighting effects. We would also like to generalize our insights to other problems involving sheared signals.

Currently the implementations for our different techniques are separate. Combining these techniques into one solution would be a very useful area of future work. Ideally we would like a solution that handled many concurrent effects, but as we have shown in Appendix D, handling three concurrent motion blur effects already leads to a fairly unwieldy result, where the resulting Fourier spectrum is difficult to analyze.

We would also like to look at hierarchical integration methods to speed up filtering. For instance, cells in the ray database could store approximate values based on averaging all sample values in the cell. However, in our experiments we have found that replacing multiple samples with a single aggregate value often leads to artifacts. The sheared filters used to query the ray database are thin (relative to the overall size of the database), and sheared at many different angles. Having too few samples inside of the filter can lead to a less smooth result with visually noticeable errors.

Bibliography

[AcademyAwards, 2010] AcademyAwards. Scientific and Technical Achievements to be Honored with Academy Awards, 2010. http://www.oscars.org/press/pressreleases/2010/20100107.html.

[Agrawala *et al.*, 2000] M. Agrawala, R. Ramamoorthi, A. Heirich, and L. Moll. Efficient image-based methods for rendering soft shadows. In *SIGGRAPH 2000*, pages 375–384, 2000.

[Akenine-Möller *et al.*, 2007] Tomas Akenine-Möller, Jacob Munkberg, and Jon Hasselgren. Stochastic Rasterization using Time-Continuous Triangles. In *Graphics Hardware*, pages 7–16, 2007.

[Annen *et al.*, 2008] Thomas Annen, Zhao Dong, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph.*, 27(3):1–8, 2008.

[Arikan *et al.*, 2005] Okan Arikan, David A. Forsyth, and James F. O'Brien. Fast and Detailed Approximate Global Illumination by Irradiance Decomposition. *ACM Trans. on Graph. (SIGGRAPH)*, 24:1108–1114, 2005.

[Arikan, 2009] Okan Arikan. Pixie - Open Source RenderMan. http://www.renderpixie.com, 2009.

[Assarsson and Akenine-Möller, 2003] Ulf Assarsson and Tomas Akenine-Möller. A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Trans. Graph.*, 22(3):511–520, 2003.

[Bala *et al.*, 1999] Kavita Bala, Julie Dorsey, and Seth Teller. Radiance interpolants for accelerated bounded-error ray tracing. *ACM Trans. Graph.*, 18(3):213–256, 1999.

[Bavoil and Sainz, 2009] Louis Bavoil and Miguel Sainz. *ShaderX7 - Advanced Rendering Techniques*, chapter Image-Space Horizon-Based Ambient Occlusion. 2009.

[Ben-Artzi *et al.*, 2006] Aner Ben-Artzi, Ravi Ramamoorthi, and Maneesh Agrawala. Efficient Shadows from Sampled Environment Maps. *Journal of Graphics Tools*, 11(1):13–36, 2006.

[Bracewell *et al.*, 1993] R. Bracewell, K. Chang, A. Jha, and Y. Wang. Affine theorem for two-dimensional fourier transform. *Electronics Letters*, 29:304, 1993.

[Cammarano and Jensen, 2002] Mike Cammarano and Henrik Wann Jensen. Time Dependent Photon Mapping. In *EG Symposium on Rendering*, pages 135–144, 2002.

[Catmull, 1984] E. Catmull. An Analytic Visible Surface Algorithm for Independent Pixel Processing. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, volume 18, pages 109–115. ACM, 1984.

[Chai *et al.*, 2000] J. Chai, X. Tong, S. Chan, and H. Shum. Plenoptic Sampling. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, ACM, pages 307–318. ACM Press / ACM SIGGRAPH, 2000.

[Chen *et al.*, 2002] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk. Light field mapping: efficient representation and hardware rendering of surface light fields. *ACM Trans. Graph.*, 21(3):447–456, 2002.

[Christensen, 2008] Per H. Christensen. Point-Based Approximate Color Bleeding. Technical Report 08–01, Pixar Animation Studios, 2008.

[Christmas, 1998] William J. Christmas. Spatial Filtering Requirements for Gradient-Based Optical Flow Measurement. In *British Machine Vision Conference*, pages 185–194, 1998.

[Cook *et al.*, 1984] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed Ray Tracing. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, volume 18, pages 137–145. ACM, 1984.

[Cook *et al.*, 1987] Robert L. Cook, Loren Carpenter, and Edwin Catmull. The Reyes Image Rendering Architecture. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, volume 21, pages 95–102. ACM, 1987.

[Durand *et al.*, 2005] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X. Sillion. A Frequency Analysis of Light Transport. *ACM Transactions on Graphics (SIGGRAPH 05)*, 24(3):1115–1126, 2005.

[Durand, 1999] Frédo Durand. *3D Visibility: analytical study and applications*. PhD thesis, Université Joseph Fourier, Grenoble I, July 1999.

[Gortler *et al.*, 1996] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *SIGGRAPH 96*, pages 43–54, 1996.

[Hachisuka *et al.*, 2008] T. Hachisuka, W. Jarosz, R. Weistroffer, K. Dale, G. Humphreys, M. Zwicker, and H. Jensen. Multidimensional Adaptive Sampling and Reconstruction for Ray Tracing. *ACM Transactions on Graphics (SIGGRAPH)*, 27(3):33:1–33:10, 2008.

[Haeberli and Akeley, 1990] Paul Haeberli and Kurt Akeley. The Accumulation Buffer: Hardware Support for High-Quality Rendering. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, volume 24, pages 309–318. ACM, 1990.

[Halton, 1960] J. H. Halton. On the Efficiency of Certain Quasi-Random Sequences of Points in Evaluating Multi-Dimensional Integrals. *Numerische Mathematik*, 2(1):84–90, 1960.

[Hart *et al.*, 1999] David Hart, Philip Dutré, and Donald P. Greenberg. Direct illumination with lazy visibility evaluation. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 147–154, 1999.

[Hasenfratz *et al.*, 2003] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A survey of real-time soft shadows algorithms. *Computer Graphics Forum*, 22(4):753–774, 2003.

[Hašan *et al.*, 2007] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Matrix row-column sampling for the many-light problem. *ACM Trans. Graph.*, 26(3):26:1–26:10, 2007.

[Huang and Ramamoorthi, 2010] Fu-Chung Huang and Ravi Ramamoorthi. Sparsely Precomputing the Light Transport Matrix for Real-Time Rendering. *Computer Graphics Forum (EGSR 10)*, 29(4), 2010.

[Isaksen *et al.*, 2000] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically Reparameterized Light Fields. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, ACM, pages 297–306. ACM Press / ACM SIGGRAPH, 2000.

[Jensen and Christensen, 1995] Henrik Wann Jensen and Niels Jørgen Christensen. Efficiently rendering shadows using the photon map. In *Compugraphics '95*, pages 285–291. Publications, 1995.

[Johnson *et al.*, 2009] Gregory S. Johnson, Warren A. Hunt, Allen Hux, William R. Mark, Christopher A. Burns, and Stephen Junkins. Soft irregular shadow mapping: fast, high-quality, and robust soft shadows. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 57–66, 2009.

[Kajiya, 1986] J. Kajiya. The rendering equation. In *SIGGRAPH 86*, 1986.

[Korein and Badler, 1983] Jonathan Korein and Norman Badler. Temporal Anti-Aliasing in Computer Generated Animation. In *Computer Graphics (Proceedings of SIGGRAPH 83)*, volume 17, pages 377–388. ACM, 1983.

[Křivánek *et al.*, 2005] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Trans. on Visualization and Computer Graphics*, 11(5), 2005.

[Lacewell *et al.*, 2008] Dylan Lacewell, Brent Burley, Solomon Boulos, and Peter Shirley. Raytracing prefiltered occlusion for aggregate geometry. In *IEEE Symposium on Interactive Raytracing 2008*, 2008.

[Laine and Karras, 2010] Samuli Laine and Tero Karras. Two methods for fast ray-cast ambient occlusion. *Computer Graphics Forum (EGSR 10)*, 29(4), 2010.

[Laine *et al.*, 2005] Samuli Laine, Timo Aila, Ulf Assarsson, Jaakko Lehtinen, and Tomas Akenine-Möller. Soft shadow volumes for ray tracing. *ACM Trans. Graph.*, 24(3):1156–1165, 2005.

[Landis, 2008] Hayden Landis. Production ready global illumination. In *ACM SIGGRAPH Course Notes: RenderMan in Production*, pages 87–102, 2008.

[Lanman *et al.*, 2008] Douglas Lanman, Ramesh Raskar, Amit Agrawal, and Gabriel Taubin. Shield Fields: Modeling and Capturing 3D Occluders. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 27(5), 2008.

[Lehtinen *et al.*, 2011] Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. Temporal Light Field Reconstruction for Rendering Distribution effects. *ACM Trans. Graph.*, 30(4), 2011.

[Levin *et al.*, 2008] Anat Levin, Peter Sand, Taeg Sang Cho, Frédo Durand, and William T. Freeman. Motion-Invariant Photography. *ACM Transactions on Graphics (SIGGRAPH)*, 27(3):71:1–71:9, 2008.

[Levoy and Hanrahan, 1996] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH 96*, pages 31–42, 1996.

[Loviscach, 2005] J. Loviscach. Motion Blur for Textures by Means of Anisotropic Filtering. In *EG Symposium on Rendering*, pages 105–110, 2005.

[Mahajan *et al.*, 2007] Dhruv Mahajan, Ira Kemelmacher Shlizerman, Ravi Ramamoorthi, and Peter Belhumeur. A Theory of Locally Low Dimensional Light Transport. *ACM Transactions on Graphics (SIGGRAPH)*, 27(3):62:1–62:10, 2007.

[Max and Lerner, 1985] Nelson L. Max and Douglas M. Lerner. A Two-and-a-Half-D Motion-Blur Algorithm. In *Computer Graphics (Proceedings of SIGGRAPH 85)*, volume 19, pages 85–93. ACM, 1985.

[McGuire, 2010] Morgan McGuire. Ambient Occlusion Volumes. In *Proceedings of High Performance Graphics 2010*, pages 47–56, June 2010.

[Méndez-Feliu and Sbert, 2009] Alex Méndez-Feliu and Mateu Sbert. From Obscurances to Ambient Occlusion: A Survey. *Vis. Comput.*, 25(2):181–196, 2009.

[Mitchell, 1991] D. Mitchell. Spectrally Optimal Sampling for Distribution Ray Tracing. In *Computer Graphics (Proceedings of SIGGRAPH 91)*, volume 25, pages 157–164. ACM, 1991.

[Neulander, 2007] Ivan Neulander. Pixmotor: A Pixel Motion Integrator. In *SIGGRAPH 2007: Sketches*, 2007.

[Neulander, 2008] Ivan Neulander. Pismo: Parallax-Interpolated Shadow Map Occlusion. In *SIGGRAPH 2008: Talks*, 2008.

[Ng *et al.*, 2003] R. Ng, R. Ramamoorthi, and P. Hanrahan. All-Frequency Shadows Using Non-Linear Wavelet Lighting Approximation. *ACM Trans. on Graph. (SIGGRAPH 03)*, 22(3):376–381, 2003.

[Nicodemus *et al.*, 1977] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. *Geometric Considerations and Nomenclature for Reflectance*. National Bureau of Standards (US), 1977.

[Overbeck *et al.*, 2007] Ryan Overbeck, Ravi Ramamoorthi, and William R. Mark. A real-time beam tracer with application to exact soft shadows. In *EuroGraphics Symposium on Rendering*, June 2007.

[Overbeck *et al.*, 2009] Ryan S. Overbeck, Craig Donner, and Ravi Ramamoorthi. Adaptive Wavelet Rendering. *ACM Transactions on Graphics (SIGGRAPH Asia 09)*, 28(5):1–12, 2009.

[Pantaleoni *et al.*, 2010] Jacopo Pantaleoni, Luca Fascione, Martin Hill, and Timo Aila. PantaRay: Fast Ray-Traced Occlusion Caching of Massive Scenes. *ACM Trans. on Graph. (SIGGRAPH 10)*, 29(4):37:1–37:10, 2010.

[Pharr and Humphreys, 2004] M. Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2004.

[Pixar, 2005] Pixar. The RenderMan Interface, version 3.2.1. https://renderman.pixar.com/products/rispec/rispec_pdf/RISpec3_2.pdf, 2005.

[Potmesil and Chakravarty, 1983] Michael Potmesil and Indranil Chakravarty. Modeling Motion Blur in Computer-Generated Images. In *Computer Graphics (Proceedings of SIGGRAPH 83)*, volume 17, pages 389–399. ACM, 1983.

[Ramamoorthi *et al.*, 2004] R. Ramamoorthi, M. Koudelka, and P. Belhumeur. A Fourier Theory for Cast Shadows. In *European Conference on Computer Vision 2004*, pages I–146–I–162, 2004.

[Ramamoorthi *et al.*, 2005] Ravi Ramamoorthi, Melissa Koudelka, and Peter Belhumeur. A fourier theory for cast shadows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(2):288–295, 2005.

[Ramamoorthi *et al.*, 2007] Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. A First-Order Analysis of Lighting, Shading, and Shadows. *ACM Transactions on Graphics*, 26(1):2:1–2:21, 2007.

[Robison and Shirley, 2009] Austin Robison and Peter Shirley. Image Space Gathering. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009*, pages 91–98, New York, NY, USA, 2009. ACM.

[Sen and Darabi, 2011] Pradeep Sen and Soheil Darabi. On Filtering the Noise from the Random Parameters in Monte Carlo Rendering. *ACM Transactions on Graphics (TOG)*, to appear, 2011.

[Shinya, 1993] Mikio Shinya. Spatial anti-aliasing for animation sequences with spatio-temporal filtering. In *SIGGRAPH '93*, pages 289–296, 1993.

[Sintorn *et al.*, 2008] Erik Sintorn, Elmar Eisemann, and Ulf Assarsson. Sample-Based Visibility for Soft Shadows Using Alias-Free Shadow Maps. *EGSR*, 27(4):1285–1292, June 2008.

[Sloan *et al.*, 2002] P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM TOG (SIGGRAPH 02)*, 21(3), 2002.

[Soler and Sillion, 1998] C. Soler and F. Sillion. Fast Calculation of Soft Shadow Textures Using Convolution. In Michael Cohen, editor, *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, ACM, pages 321–332. ACM Press / ACM SIGGRAPH, 1998.

[Soler *et al.*, 2009] Cyril Soler, Kartic Subr, Frédo Durand, Nicolas Holzschuch, and François Sillion. Fourier Depth of Field. *ACM Transactions on Graphics*, 28(2):18:1–18:18, 2009.

[Stewart *et al.*, 2003] J. Stewart, J. Yu, S. J. Gortler, and L. McMillan. A new reconstruction filter for undersampled light fields. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 150–156, 2003.

[Sun and Ramamoorthi, 2009] Bo Sun and Ravi Ramamoorthi. Affine Double and Triple Product Wavelet Integrals for Rendering. *ACM Transactions on Graphics*, 28(2):1–17, Apr 2009.

[Sung *et al.*, 2002] K. Sung, A. Pearce, and C. Wang. Spatial-Temporal Antialiasing. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):144–153, 2002.

[van der Linden, 2003] Jarno van der Linden. Multiple light field rendering. In *GRAPHITE '03*, pages 197–ff, 2003.

[Veach, 1997] E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.

[Walter *et al.*, 2006] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. Multidimensional Lightcuts. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):1081–1088, 2006.

[Ward *et al.*, 1988] G. Ward, F. Rubinstein, and R. Clear. A Ray Tracing Solution for Diffuse Interreflection. In *SIGGRAPH 88*, pages 85–92, 1988.

[Yang *et al.*, 2009] Baoguang Yang, Jieqing Feng, Gaël Guennebaud, and Xinguo Liu. Packet-based Hierarchal Soft Shadow Mapping. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering 2009)*, 28(4):1121–1130, 2009.

[Zhou *et al.*, 2005] Kun Zhou, Yaohua Hu, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Precomputed shadow fields for dynamic scenes. In *SIGGRAPH 05*, pages 1196–1201, 2005.

[Zhukov *et al.*, 1998] Sergey Zhukov, Andrei Iones, and Grigorij Kronin. An ambient light illumination model. In *Rendering Techniques (Eurographics 98)*, pages 45–56, 1998.

[Zwicker *et al.*, 2007] Matthias Zwicker, Sehoon Yea, Anthony Vetro, Clifton Forlines, Wojciech Matusik, and Hanspeter Pfister. Display pre-filtering for multi-view video compression. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 1046–1053, 2007.

Appendix A

Motion Blur Implementation Details and Special Cases

Computing Velocities and Bandlimits (Sec. 3.6.1): Any angular/spatial units can be used as long as $\kappa, \alpha, L^{\max}, R^{\max}, S^{\max}, \nu$ and the corresponding trigonometry functions all use the same units. Corresponding maximum frequencies or bandlimits are expressed in inverse pixel and time units. Analogously, all velocities are calculated by the shader in pixel distances per unit time, and account for projection effects. Calculations of frequency bandlimits are specific to the shading model being used and can be done either before or during rendering. For example, an implementation may dynamically relate the specularity and frequency of a BRDF using an analytic function, but precompute $L^{\max}$ for an environment map by running a Fourier transform. For surface points with occlusion, $S^{\max}$ will usually have infinite frequencies so we can simply set $\Omega^{\max}_{x,shadow} = L^{\max}\nu$.

Finally, note that our implementation sparsely samples frequency information, so we use advection to gather nearby frequency samples that may overlap with the current pixel at a different moment in time.

Multiple Signals (Sec. 3.6.1): The final color for a single sample is obtained by multiplying the surface texture, BRDF and shadow signals, so we need to bound the frequencies of that product. Occasionally, samples may have more than one signal (texture, BRDF or shadow) each with significant amplitude and frequency. In Fourier space, the product of the signals corresponds to a convolution of spectra. If all signals have similar velocities, the frequencies will lie along the same Fourier line,

as will the final spectrum—we can simply add the individual signal bandlimits to obtain $\Omega_x^{\max}$.

However, when two different signals have different effective velocities, we can obtain a spectrum unlike the wedges in Figures 3.2(i) and 3.2(j). One example is a textured surface moving vertically and a shadow moving horizontally. We can bound this convolved spectrum by setting $a_{\min} = 0$, take the maximum value of $a_{\max}$, and sum all relevant values of $\Omega_x^{\max}$.

For a full derivation of the general case see Appendix D.

Low Velocities and Axis-Aligned Filters (Sec. 3.6.2): From Equation 3.34, we know that the applied shear grows large as $a_{\min}$ decreases. For slow-moving signals there is a crossing point where using a standard axis-aligned filter is preferable. In Equation 3.27, we saw that the spatial bandlimit Ω_x^* can be less than $\Omega_t^{\max}/a_{\min}$ when $a_{\min}$ is small. For this reason, we fall back to the standard axis-aligned filter when (FreqSpacing + $\Omega_t^{\max}/a_{\min}$) > $\Omega_x^{\max}$ (see below for details).

Filter Width (Sec. 3.6.2): After computing $a_{\max}$, $a_{\min}$, and $\Omega_x^{\max}$ for the samples inside of the current pixel, we can compute a filter shape using Equations 3.33 and 3.34. However, if this new filter overlaps with other pixels we must recompute $a_{\max}$, $a_{\min}$, and $\Omega_x^{\max}$ for all pixels inside the filter. The more samples inside the filter, the greater $a_{\max}$, $a_{\min}$, and $\Omega_x^{\max}$ will diverge. For this reason the widest possible filter width may be smaller than the width originally computed using samples only inside the current pixel (this is common when a filter is close to an occlusion discontinuity). We do a binary search to find the widest possible filter width, searching between a scale of 1.0 on the low end, and the scale predicted initially by samples inside the current pixel on the high end.

In cases where the final filter radius (ActualPrimalRadius) is not as wide as the ideal size (IdealPrimalRadius), the shear is unchanged (Equation 3.34), but the scale is changed (Equation 3.33). We must adjust our sampling rates (Equation 3.30 and Figure 3.8(c)) to account for the fact that we have effectively added FreqSpacing to the radius of our reconstruction filter along the Ω_x axis in the Fourier domain:

$$\text{RadiusRatio} = \text{IdealPrimalRadius}/\text{ActualPrimalRadius}$$

$$\text{FreqSpacing} = (\text{RadiusRatio} - 1)\Omega_{\text{pix}}^{\max}/\text{Scale}.$$

When (FreqSpacing + $\Omega_t^{\max}/a_{\min}$) > $\Omega_x^{\max}$ one corner of the filter has passed beyond $\Omega_x^{\max}$ and we switch to using an axis-aligned filter (along with Equations 3.25, 3.27, and 3.37). If we are using a

sheared filter we have

$$\Omega_t^* = \left(\text{FreqSpacing} + \frac{\Omega_t^{\max}}{a_{\min}}\right) a_{\max} - \Omega_t^{\max} \tag{A.1}$$

$$\Omega_x^* = \Omega_x^{\max} + \frac{\Omega_t^{\max}}{a_{\min}} + \text{FreqSpacing}. \tag{A.2}$$

Appendix B

Shadows from Area Lights

Fourier Derivations

To derive Equation 4.3 we have:

$$\mathcal{F}\left[f(v,y)\right] = \int\int g(d_2 v + y)\exp(-i2\pi(v\Omega_v + y\Omega_y))dvdy \tag{B.1}$$

$$\begin{aligned} u = d_2 v + y \qquad y = u - d_2 v \qquad dy = du \\ &= \int\int g(u)\exp(-i2\pi(v\Omega_v + (u - d_2 v)\Omega_y))dvdu \\ &= \int\int g(u)\exp(-i2\pi(v(\Omega_v - d_2\Omega_y) + u\Omega_y))dvdu \\ &= \int\left[\int g(u)\exp(-i2\pi u\Omega_y)du\right]\exp(-i2\pi v(\Omega_v - d_2\Omega_y))dv \\ &= G(\Omega_y)\int\exp(-i2\pi v(\Omega_v - d_2\Omega_y))dv \end{aligned}$$

$$\mathcal{F}\left[f(v,y)\right] = G(\Omega_y)\delta(\Omega_v - d_2\Omega_y). \tag{B.2}$$

To derive Equation 4.4 we have:

$$\mathcal{F}\left[f\left(\frac{x-y}{d_1}, y\right)\right] = \int\int f\left(\frac{x-y}{d_1}, y\right)\exp(-i2\pi(x\Omega_x + y\Omega_y))dxdy \tag{B.3}$$

$$u = \frac{x-y}{d_1} \qquad x = ud_1 + y \qquad dx = (d_1)du$$

$$= \int\int f(u,y)\exp(-i2\pi((ud_1 + y)\Omega_x + y\Omega_y))(d_1)dudy$$

$$= d_1\int\int f(u,y)\exp(-i2\pi(ud_1\Omega_x + y(\Omega_x + \Omega_y)))dudy$$

$$\mathcal{F}\left[f\left(\frac{x-y}{d_1}, y\right)\right] = d_1 F(d_1\Omega_x, \Omega_y + \Omega_x). \tag{B.4}$$

Sampling Rates

Sampling in the primal domain creates replicas in the Fourier domain, and the sparser the sampling rate the closer together the replicas are packed. We want to compute the lowest possible sampling rate such that we prevent the replicas from overlapping the footprint of our filter. The compact shape of our sheared filter allows for much tighter packing of replicas, which allows for much lower sampling rates, which in turn leads to faster render times. We can use a derivation similar to Section 3.5.1. to compute the minimal sampling rates for our sheared filter shape:

$$\Omega_x^* = \Omega_x^{\max} + \Omega_y^{\max}\left(\frac{d_1}{d_{2\max}} - 1\right)^{-1}, \tag{B.5}$$

$$\Omega_y^* = \Omega_y^{\max}\left(\frac{d_1}{d_{2\max}} - 1\right)\left[\left(\frac{d_1}{d_{2\max}} - 1\right)^{-1} - \left(\frac{d_1}{d_{2\min}} - 1\right)^{-1}\right]. \tag{B.6}$$

In the above equations, Ω_x^* and Ω_y^* are the required sampling rates in the x and y dimensions respectively. These values are derived by measuring the distance between replicas along Ω_x and Ω_y (the exact derivation is omitted for brevity). To compute the number of samples requested by a receiver point we calculate the 4D product $(\Omega_x^*)^2(\Omega_y^*)^2$ and divide by the 4D volume of the sheared filter in (x_1, x_2, y_1, y_2) (because the subspaces are orthogonal this is simply the product of the filter areas in (x_1, y_1) and (x_2, y_2)). The Ω_x^{max} bandlimit is the extent of the occluder wedge along Ω_x (see Figure 4.5a). Looking at the shape of the original occluder spectrum, $F(\Omega_v, \Omega_y)$ (see Figure 4.4b), we find that the transformation from (Ω_x, Ω_y) to (Ω_v, Ω_y) results in a $\Omega_x^{\max}$ being equal to $\Omega_g^{\max}\frac{d_{2\max}}{d_1}$.

Appendix C

Ambient Occlusion Derivations

We temporarily define h and H to be 2D to make the derivation more concise (this allows us to use a 2D convolution operator). The second angular dimension for both of these functions will not be important, and our final step will be to reduce H to 1D. We define $h(x, v)$ to be constant across v, such that $\forall v \in \mathbb{R},\ h(x, v) = h(x, 0)$.

$$m(x, v) = \delta(x) \tag{C.1}$$

$$h(x, v) = \int f(x, t)k(x, t)dt \tag{C.2}$$

$$h(x, v) = \int \int (f(s, t)k(s, t))\, m(x - s, v - t)dsdt \tag{C.3}$$

$$h(x, v) = (fk) \otimes m \tag{C.4}$$

Because $h(x)$ is constant across v it is not surprising that spectrum H will only have frequencies along the $\Omega_v = 0$ line.

$$M(\Omega_x, \Omega_v) = \delta(\Omega_v) \tag{C.5}$$

$$H(\Omega_x, \Omega_v) = (F \otimes K)M \tag{C.6}$$

$$H(\Omega_x, \Omega_v) = \int \int F(\Omega_x - s, \Omega_v - t)K(t, s)\delta(\Omega_v)\, ds\, dt \tag{C.7}$$

To reduce a 2D spectrum to 1D we must integrate across the dimension we want to remove. We integrate out the Ω_v dimension to measure the spatial frequencies of H along the Ω_x axis:

$$H(\Omega_x) = \int \int F(\Omega_x - s, -t)K(s, t)\, ds\, dt \tag{C.8}$$

Appendix D

Derivation of Motion Blur General Case

D.1 General Case

Using the angular form of the reflection equation in Chapter 3 we can derive what happens when all three effects are present (moving texture, moving reflection, moving shadow). We also make the derivation more general by adding the outgoing angle θ_o as a parameter to the reflection equation. We will see that the general case is fairly complex, but with some simplifying assumptions we can return to the simple forms seen in chapter 3. While more compact derivations are surely possible, we give a detailed derivation so that each step is easy to verify.

This assumes a fixed camera (the x measurement is relative to the camera), with surface motion represented by β, occluder motion represented by τ, and the center of the camera shutter defined by t_0. We also add a ζ offset parameter to the linearization of $s()$ so to avoid any restrictions on the x parameter. Otherwise this derivation matches Equations 3.4, 3.14 and 3.22. We start with with the reflection function $h()$.

$$h(x, t_0, \theta_o) = \int w(t_0 - t) g(x, t) \int l(\theta_i, t) \tag{D.1}$$
$$s(x, \theta_i, t) r(x, t, \theta_i, \theta_o) d\theta_i dt$$
$$= \int w(t_0 - t) g(x - \beta t) \int l(\theta_i - \alpha t) \tag{D.2}$$
$$s(\mu(x - \tau t) - \theta_i) r(\theta_o + 2(n(x, t) - \theta_o) - \theta_i) d\theta_i dt$$
$$= \int w(t_0 - t) g(x - \beta t) \int l(\theta_i - \alpha t) \tag{D.3}$$
$$s(\nu x - \tau \nu t + \zeta - \theta_i) r(\kappa' x - \beta \kappa' t + \eta' - \theta_o - \theta_i) d\theta_i dt$$
$$\tag{D.4}$$

Now we plug in spectral versions of functions into $h()$ using the inverse Fourier transform.

$$w(t) = \int W(m) e^{i2\pi m t} dm \tag{D.5}$$
$$g(x) = \int G(n) e^{i2\pi n x} dn \tag{D.6}$$
$$l(\theta) = \int L(q) e^{i2\pi q \theta} dq \tag{D.7}$$
$$s(\theta) = \int S(y) e^{i2\pi y \theta} dy \tag{D.8}$$
$$r(\theta) = \int R(z) e^{i2\pi z \theta} dz \tag{D.9}$$
$$\tag{D.10}$$

$$
\begin{aligned}
h(x, t_0, \theta_o) = & \int \left(\int W(m) e^{i2\pi m(t_0 - t)} dm \right) \left(\int G(n) e^{i2\pi n(x - \beta t)} dn \right) \\
& \int \left(\int L(q) e^{i2\pi q(\theta_i - \alpha t)} dq \right) \left(\int S(y) e^{i2\pi y(\nu x - \tau \nu t + \zeta - \theta_i)} dy \right) \\
& \left(\int R(z) e^{i2\pi z(\kappa' x - \beta \kappa' t + \eta' - \theta_o - \theta_i)} dz \right) d\theta_i dt
\end{aligned} \tag{D.11}
$$

$$
\begin{aligned}
h(x, t_0, \theta_o) = & \int \int \int \int \int \int \int W(m) G(n) L(q) S(y) R(z) \\
& e^{i2\pi m(t_0 - t)} e^{i2\pi n(x - \beta t)} e^{i2\pi q(\theta_i - \alpha t)} e^{i2\pi y(\nu x - \tau \nu t + \zeta - \theta_i)} \\
& e^{i2\pi z(\kappa' x - \beta \kappa' t + \eta' - \theta_o - \theta_i)} dm dn dq dy dz d\theta_i dt
\end{aligned} \tag{D.12}
$$

$$
\begin{aligned}
h(x, t_0, \theta_o) = & \int \int \int \int \int \int \int W(m) G(n) L(q) S(y) R(z) \\
& e^{i2\pi x(\kappa' z + \nu y + n)} e^{i2\pi t_0 m} e^{i2\pi \theta_o(-z)} \\
& e^{i2\pi \theta_i(q - y - z)} e^{i2\pi t(-m - \beta n - \alpha q - \tau \nu y - \beta \kappa' z)} \\
& e^{i2\pi y \zeta} e^{i2\pi z \eta'} dm dn dq dy dz d\theta_i dt
\end{aligned} \tag{D.13}
$$

Now take the Fourier transform of h to compute $H()$

$$
\begin{aligned}
H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = & \int \int \int \int \int \int \int \int \int \int W(m) G(n) L(q) S(y) R(z) \\
& e^{-i2\pi x \Omega_x} e^{i2\pi x(\kappa' z + \nu y + n)} \\
& e^{-i2\pi t_0 \Omega_{t0}} e^{i2\pi t_0 m} \\
& e^{-i2\pi \theta_o \Omega_{\theta o}} e^{i2\pi \theta_o(-z)} \\
& e^{i2\pi \theta_i(q - y - z)} e^{i2\pi t(-m - \beta n - \alpha q - \tau \nu y - \beta \kappa' z)} \\
& e^{i2\pi y \zeta} e^{i2\pi z \eta'} dx dt_0 d\theta_o dm dn dq dy dz d\theta_i dt
\end{aligned} \tag{D.14}
$$

$$
\begin{aligned}
H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = & \int \int \int \int \int \int \int \int \int \int W(m) G(n) L(q) S(y) R(z) \\
& e^{-i2\pi x(\Omega_x - \kappa' z - \nu y - n)} e^{-i2\pi t_0(\Omega_{t0} - m)} \\
& e^{-i2\pi \theta_o(\Omega_{\theta o} + z)} e^{-i2\pi \theta_i(-q + y + z)} \\
& e^{-i2\pi t(m + \beta n + \alpha q + \tau \nu y + \beta \kappa' z)} e^{-i2\pi(-y\zeta)} e^{-i2\pi(-z\eta')} \\
& dx dt_0 d\theta_o dm dn dq dy dz d\theta_i dt
\end{aligned} \tag{D.15}
$$

Now we simplify for θ_i and q.

$$
\begin{aligned}
H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int\int\int\int\int\int W(m)G(n)L(q)S(y)R(z) \\
\left(\int e^{-i2\pi\theta_i(-q+y+z)} d\theta_i\right) \\
e^{-i2\pi x(\Omega_x-\kappa' z-\nu y-n)} e^{-i2\pi t_0(\Omega_{t0}-m)} e^{-i2\pi\theta_o(\Omega_{\theta o}+z)} \\
e^{-i2\pi t(m+\beta n+\alpha q+\tau\nu y+\beta\kappa' z)} e^{-i2\pi(-y\zeta)} e^{-i2\pi(-z\eta')} \\
dx dt_0 d\theta_o dm dn dq dy dz dt
\end{aligned} \tag{D.16}
$$

$$
\begin{aligned}
H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int\int\int\int\int W(m)G(n)S(y)R(z) \\
\left(\int L(q) e^{-i2\pi t(m+\beta n+\alpha q+\tau\nu y+\beta\kappa' z)} \delta(-q+y+z) dq\right) \\
e^{-i2\pi x(\Omega_x-\kappa' z-\nu y-n)} e^{-i2\pi t_0(\Omega_{t0}-m)} \\
e^{-i2\pi\theta_o(\Omega_{\theta o}+z)} e^{-i2\pi y(-\zeta)} e^{-i2\pi z(-\eta')} \\
dx dt_0 d\theta_o dm dn dy dz dt
\end{aligned} \tag{D.17}
$$

$$
-q+y+z=0 \qquad q=y+z \tag{D.18}
$$

$$
\begin{aligned}
H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int\int\int\int\int W(m)G(n)L(y+z)S(y)R(z) \\
e^{-i2\pi x(\Omega_x-\kappa' z-\nu y-n)} e^{-i2\pi t_0(\Omega_{t0}-m)} e^{-i2\pi\theta_o(\Omega_{\theta o}+z)} \\
e^{-i2\pi t(m+\beta n+\alpha(y+z)+\tau\nu y+\beta\kappa' z)} e^{-i2\pi y(-\zeta)} e^{-i2\pi z(-\eta')} \\
dx dt_0 d\theta_o dm dn dy dz dt
\end{aligned} \tag{D.19}
$$

Now we simplify for t and m.

$$
\begin{aligned}
H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int\int\int\int & W(m)G(n)L(y+z)S(y)R(z) \\
& \left(\int e^{-i2\pi t(m+\beta n+(\tau\nu+\alpha)y+(\beta\kappa'+\alpha)z)}dt\right) \\
& e^{-i2\pi x(\Omega_x-\kappa' z-\nu y-n)}e^{-i2\pi t_0(\Omega_{t0}-m)}e^{-i2\pi\theta_o(\Omega_{\theta o}+z)} \\
& e^{-i2\pi y(-\zeta)}e^{-i2\pi z(-\eta')}dxdt_0d\theta_o dmdndydz
\end{aligned}
\tag{D.20}
$$

$$
\begin{aligned}
H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int\int\int & G(n)L(y+z)S(y)R(z) \\
& \left(\int W(m)e^{-i2\pi t_0(\Omega_{t0}-m)}\right. \\
& \left.\delta(m+\beta n+(\tau\nu+\alpha)y+(\beta\kappa'+\alpha)z)dm\right) \\
& e^{-i2\pi x(\Omega_x-\kappa' z-\nu y-n)}e^{-i2\pi\theta_o(\Omega_{\theta o}+z)} \\
& e^{-i2\pi y(-\zeta)}e^{-i2\pi z(-\eta')}dxdt_0d\theta_o dndydz
\end{aligned}
\tag{D.21}
$$

$$
\begin{aligned}
m+\beta n+(\tau\nu+\alpha)y+(\beta\kappa'+\alpha)z &= 0 \\
m &= -\beta n-(\tau\nu+\alpha)y-(\beta\kappa'+\alpha)z
\end{aligned}
\tag{D.22}
$$

$$
\begin{aligned}
H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int\int\int & \\
& W(-\beta n-(\tau\nu+\alpha)y-(\beta\kappa'+\alpha)z)G(n)L(y+z)S(y)R(z) \\
& e^{-i2\pi x(\Omega_x-\kappa' z-\nu y-n)}e^{-i2\pi\theta_o(\Omega_{\theta o}+z)} \\
& e^{-i2\pi t_0(\Omega_{t0}+\beta n+(\tau\nu+\alpha)y+(\beta\kappa'+\alpha)z)} \\
& e^{-i2\pi y(-\zeta)}e^{-i2\pi z(-\eta')}dxdt_0d\theta_o dndydz
\end{aligned}
\tag{D.23}
$$

Equation D.23 defines the spectrum using all three concurrent effects (moving textures, reflections and shadows). However, the result is difficult to analyze. We now show that this general equation simplifies down to previously discussed cases with the addition of a few simplifying assumptions.

D.2 Moving Texture

Now we can derive the case where we have diffuse surfaces and no shadowing (the moving texture example in Section 3.3.1). In this case we assume that $s()$ and $r()$ are the constant 1 function, and

$S()$ and $R()$ are the $\delta()$ functional.

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int\Big(\int\int W(-\beta n - (\tau\nu + \alpha)y - (\beta\kappa' + \alpha)z)G(n)L(y+z) e^{-i2\pi x(\Omega_x - \kappa' z - \nu y - n)} e^{-i2\pi\theta_o(\Omega_{\theta o} + z)} e^{-i2\pi t_0(\Omega_{t0} + \beta n + (\tau\nu+\alpha)y + (\beta\kappa'+\alpha)z)} e^{-i2\pi y(-\zeta)} e^{-i2\pi z(-\eta')} \delta(y)\delta(z)dydz\Big)dxdt_0 d\theta_o dn \tag{D.24}$$

$$y = 0 \qquad z = 0 \tag{D.25}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int W(-\beta n)G(n)L(0) e^{-i2\pi x(\Omega_x - n)} e^{-i2\pi\theta_o(\Omega_{\theta o})} e^{-i2\pi t_0(\Omega_{t0} + \beta n)} dxdt_0 d\theta_o dn \tag{D.26}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = L(0)\left(\int e^{-i2\pi\theta_o(\Omega_{\theta o})} d\theta_o\right)\int W(-\beta n)G(n) \left(\int e^{-i2\pi t_0(\Omega_{t0} + \beta n)} dt_0\right)\left(\int e^{-i2\pi x(\Omega_x - n)} dx\right) dn \tag{D.27}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = L(0)\delta(\Omega_{\theta o})\int W(-\beta n)G(n) \delta(\Omega_{t0} + \beta n)\delta(\Omega_x - n)dn \tag{D.28}$$

$$\Omega_x - n = 0 \qquad \Omega_x = n \tag{D.29}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = L(0)\delta(\Omega_{\theta o})W(-\beta\Omega_x)G(\Omega_x)\delta(\Omega_{t0} + \beta\Omega_x) \tag{D.30}$$

For any non-zero value, $\Omega_{t0} = -\beta\Omega_x$.

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = L(0)\delta(\Omega_{\theta o})W(\Omega_{t0})G(\Omega_x)\delta(\Omega_{t0} + \beta\Omega_x) \tag{D.31}$$

The $\delta(\Omega_{\theta o})$ term shows that the non-zero frequencies only exist along the $\Omega_{\theta o} = 0$ line because there is no angular variation. The $L(0)$ term shows that the brightness (zero component of spectrum) for the constant environment has a linear effect on the final outcome. Equation D.31 provides a more general version of Equation 3.9.

D.3 Moving Reflection

Now we will derive the case where we have moving reflections with no texture and no shadows (the moving texture example in Section 3.3.2). In this case we assume that $s()$ and $g()$ are the constant 1 function, and $S()$ and $G()$ are the $\delta()$ functional.

$$\begin{aligned} H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int\Big(\int\int \\ & W(-\beta n - (\tau\nu + \alpha)y - (\beta\kappa' + \alpha)z)L(y+z)R(z) \\ & e^{-i2\pi x(\Omega_x - \kappa' z - \nu y - n)} e^{-i2\pi\theta_o(\Omega_{\theta o}+z)} \\ & e^{-i2\pi t_0(\Omega_{t0} + \beta n + (\tau\nu+\alpha)y + (\beta\kappa'+\alpha)z)} e^{-i2\pi y(-\zeta)} e^{-i2\pi z(-\eta')} \\ & \delta(n)\delta(y)dndy\Big)dxdt_0 d\theta_o dz \end{aligned} \tag{D.32}$$

$$n = 0 \qquad y = 0 \tag{D.33}$$

$$\begin{aligned} H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int \\ & W(-(\beta\kappa' + \alpha)z)L(z)R(z) \\ & e^{-i2\pi x(\Omega_x - \kappa' z)} e^{-i2\pi\theta_o(\Omega_{\theta o}+z)} \\ & e^{-i2\pi t_0(\Omega_{t0} + (\beta\kappa'+\alpha)z)} e^{-i2\pi z(-\eta')} \\ & dxdt_0 d\theta_o dz \end{aligned} \tag{D.34}$$

$$\begin{aligned} H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = & \int W(-(\beta\kappa' + \alpha)z)L(z)R(z) \\ & \left(\int e^{-i2\pi x(\Omega_x - \kappa' z)} dx\right)\left(\int e^{-i2\pi\theta_o(\Omega_{\theta o}+z)} d\theta_o\right) \\ & \left(\int e^{-i2\pi t_0(\Omega_{t0} + (\beta\kappa'+\alpha)z)} dt_0\right) e^{-i2\pi z(-\eta')} dz \end{aligned} \tag{D.35}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int W(-(\beta\kappa' + \alpha)z)L(z)R(z) \\ \delta(\Omega_x - \kappa' z)\delta(\Omega_{\theta o} + z) \\ \delta(\Omega_{t0} + (\beta\kappa' + \alpha)z)e^{-i2\pi z(-\eta')}dz \tag{D.36}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int W(-(\beta\kappa' + \alpha)z)L(z)R(z) \\ \left(\frac{1}{|\kappa'|}\delta(\frac{\Omega_x}{\kappa'} - z)\right)\delta(\Omega_{\theta o} + z) \\ \delta(\Omega_{t0} + (\beta\kappa' + \alpha)z)e^{-i2\pi z(-\eta')}dz \tag{D.37}$$

$$\frac{\Omega_x}{\kappa'} - z = 0 \qquad z = \frac{\Omega_x}{\kappa'} \tag{D.38}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \frac{1}{|\kappa'|}W(\frac{-\beta\kappa' + \alpha}{\kappa'}\Omega_x)L(\frac{\Omega_x}{\kappa'})R(\frac{\Omega_x}{\kappa'}) \\ \delta(\Omega_{\theta o} + \frac{\Omega_x}{\kappa'})\delta(\Omega_{t0} + \frac{(\beta\kappa' + \alpha)}{\kappa'}\Omega_x)e^{i2\pi(\eta'/\kappa')\Omega_x} \tag{D.39}$$

For non-zero values $\Omega_{t0} = \frac{-\beta\kappa' + \alpha}{\kappa'}\Omega_x$.

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \frac{1}{|\kappa'|}W(\Omega_{t0})L(\frac{\Omega_x}{\kappa'})R(\frac{\Omega_x}{\kappa'}) \\ \delta(\Omega_{\theta o} + \frac{\Omega_x}{\kappa'})\delta(\Omega_{t0} + \frac{(\beta\kappa' + \alpha)}{\kappa'}\Omega_x)e^{i2\pi(\eta'/\kappa')\Omega_x} \tag{D.40}$$

Equation D.40 provides a more general version of Equation 3.18.

D.4 Moving Shadow

Now we will derive the case where we have moving shadows with no texture and a diffuse BRDF (the moving shadow example in Section 3.3.3). In this case we assume that $r()$ and $g()$ are the constant 1 function, and $R()$ and $G()$ are the $\delta()$ functional.

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int\Big(\int\int W(-\beta n - (\tau\nu + \alpha)y - (\beta\kappa' + \alpha)z)L(y+z)S(y) e^{-i2\pi x(\Omega_x - \kappa' z - \nu y - n)} e^{-i2\pi\theta_o(\Omega_{\theta o} + z)} e^{-i2\pi t_0(\Omega_{t0} + \beta n + (\tau\nu+\alpha)y + (\beta\kappa'+\alpha)z)} e^{-i2\pi y(-\zeta)} e^{-i2\pi z(-\eta')} \delta(n)\delta(z)dndz\Big)dxdt_0 d\theta_o dy \tag{D.41}$$

$$n = 0 \qquad z = 0 \tag{D.42}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int\int\int\int W(-(\tau\nu + \alpha)y)L(y)S(y) e^{-i2\pi x(\Omega_x - \nu y)} e^{-i2\pi t_0(\Omega_{t0} + (\tau\nu+\alpha)y)} e^{-i2\pi y(-\zeta)} dxdt_0 d\theta_o dy \tag{D.43}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int W(-(\tau\nu + \alpha)y)L(y)S(y) \left(\int e^{-i2\pi x(\Omega_x - \nu y)} dx\right)\left(\int e^{-i2\pi\theta_o(\Omega_{\theta o})} d\theta_o\right) \left(\int e^{-i2\pi t_0(\Omega_{t0} + (\tau\nu+\alpha)y)} dt_0\right) e^{-i2\pi y(-\zeta)} dy \tag{D.44}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int W(-(\tau\nu + \alpha)y)L(y)S(y) \delta(\Omega_x - \nu y)\delta(\Omega_{\theta o}) \delta(\Omega_{t0} + (\tau\nu + \alpha)y) e^{-i2\pi y(-\zeta)} dy \tag{D.45}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \int W(-(\tau\nu + \alpha)y)L(y)S(y) \left(\frac{1}{|\nu|}\delta(\frac{\Omega_x}{\nu} - y)\right)\delta(\Omega_{\theta o}) \delta(\Omega_{t0} + (\tau\nu + \alpha)y) e^{-i2\pi y(-\zeta)} dy \tag{D.46}$$

$$\frac{\Omega_x}{\nu} - y = 0 \qquad y = \frac{\Omega_x}{\nu} \tag{D.47}$$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \frac{1}{|\nu|}\int W(-\frac{(\tau\nu + \alpha)}{\nu}\Omega_x)L(\frac{\Omega_x}{\nu})S(\frac{\Omega_x}{\nu}) \delta(\Omega_{\theta o})\delta(\Omega_{t0} + \frac{\tau\nu + \alpha}{\nu}\Omega_x) e^{i2\pi(\zeta/\nu)\Omega_x} \tag{D.48}$$

For non-zero values $\Omega_{t0} = -\frac{\tau\nu+\alpha}{\nu}\Omega_x$

$$H(\Omega_x, \Omega_{t0}, \Omega_{\theta o}) = \frac{1}{|\nu|}\int W(\Omega_{t0})L(\frac{\Omega_x}{\nu})S(\frac{\Omega_x}{\nu}) \\ \delta(\Omega_{\theta o})\delta(\Omega_{t0} + \frac{\tau\nu+\alpha}{\nu}\Omega_x)e^{i2\pi(\zeta/\nu)\Omega_x} \tag{D.49}$$

Equation D.49 provides a more general version of Equation 3.24.