Eurographics Symposium on Rendering 2017
P. Sander and M. Zwicker
(Guest Editors)

Volume 36 (2017), Number 4

Multiple Axis-Aligned Filters for Rendering of Combined Distribution Effects

Lifan Wu[1] Ling-Qi Yan[2] Alexandr Kuznetsov[1] Ravi Ramamoorthi[1]

[1]University of California, San Diego [2]University of California, Berkeley

Abstract

Distribution effects such as diffuse global illumination, soft shadows and depth of field, are most accurately rendered using Monte Carlo ray or path tracing. However, physically accurate algorithms can take hours to converge to a noise-free image. A recent body of work has begun to bridge this gap, showing that both individual and multiple effects can be achieved accurately and efficiently. These methods use sparse sampling, GPU raytracers, and adaptive filtering for reconstruction. They are based on a Fourier analysis, which models distribution effects as a wedge in the frequency domain. The wedge can be approximated as a single large axis-aligned filter, which is fast but retains a large area outside the wedge, and therefore requires a higher sampling rate; or a tighter sheared filter, which is slow to compute. The state-of-the-art fast sheared filtering method combines low sampling rate and efficient filtering, but has been demonstrated for individual distribution effects only, and is limited by high-dimensional data storage and processing.

We present a novel filter for efficient rendering of combined effects, involving soft shadows and depth of field, with global (diffuse indirect) illumination. We approximate the wedge spectrum with multiple axis-aligned filters*, marrying the speed of axis-aligned filtering with an even more accurate (compact and tighter) representation than sheared filtering. We demonstrate rendering of single effects at comparable sampling and frame-rates to fast sheared filtering. Our main practical contribution is in rendering multiple distribution effects, which have not even been demonstrated accurately with sheared filtering. For this case, we present an average speedup of* 6× *compared with previous axis-aligned filtering methods.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Distribution effects like diffuse global illumination, depth of field, and soft shadows, are crucial for physically-based interactive rendering. Simulating them accurately involves Monte Carlo ray or path-tracing, but this can require thousands of samples per pixel and hours of computation for convergence. In this paper, we develop one of the first practical methods for ray-traced near-interactive (at 1-2s per frame) rendering of multiple distribution effects (Fig. 2).

Background: We leverage a recent body of work on sparse sampling and reconstruction, which exploits the coherence between pixels, and in other dimensions [ZJL*15]. Specifically, [ETH*09] noted that distribution effects like motion blur lead to a wedge in the frequency domain, for which we can design adaptive 4D sheared filters; they later extended this to soft shadows and spherical harmonic occlusion [EDR11, EHDR11]. A sheared filter captures the wedge spectrum tightly, leading to a decrease in the required sampling rate by orders of magnitude. However, reconstruction with sheared filtering can be slow (minutes to hours) and memory intensive, requiring storage of all high-dimensional samples, and an irregular search at each pixel within the footprint of the 4D sheared filter. Thus, early papers demonstrated only offline usage.

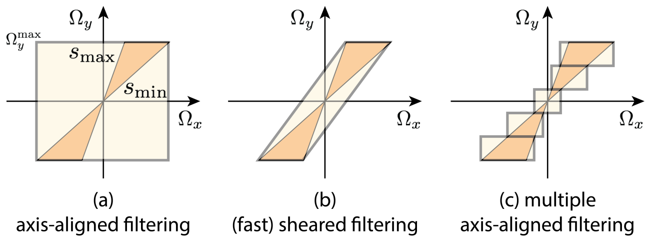
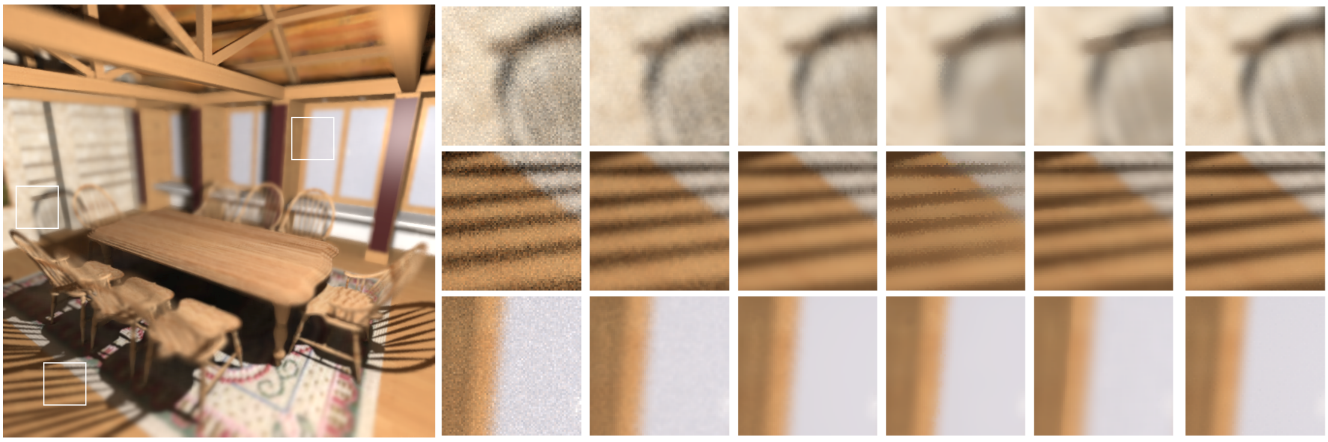


Figure 1: *2D filters for (a) axis-aligned filtering (AAF), (b) fast sheared filtering (FSF), and (c) our new multiple axis-aligned filtering (MAAF) with 5 filters. MAAF has the tightest fit (including in our practical implementation, which uses Gaussian filters), while enabling the simplicity and efficiency of single-pass axis-aligned filtering. In all cases, we seek to approximate the wedge Fourier spectrum bounded by slopes $s_{\min}$ and $s_{\max}$, with bandwidth $\Omega_y^{\max}$.*

Hence, an alternate body of work was developed for interactive rendering, based on axis-aligned filtering [MWR12, MWRD13].



CONFERENCE, MAAF, 32 spp, 2.34s | MC input 32 spp, 1.67s | AAF (ET) 45 spp, 2.35s | AAF (EQ) 158 spp, 17.1s | APR 32 spp, 88.0s | MAAF 32 spp, 2.34s | Ground Truth 6400 spp, 869s

Figure 2: *Conference room scene with defocus blur, soft shadows and global illumination, rendered with our Multiple Axis-Aligned Filters (MAAF) in 2.3s at 32 samples per pixel. Our method applies 6D MAAF, which consists of 25 component filters in total, to noisy Monte Carlo input from GPU raytracing in OptiX, removing most of the noise, and is considerably higher-quality than equal time axis-aligned filtering AAF (ET) [MYRD14]. (Readers are encouraged to view the images in the electronic PDF to observe the noise in insets in this and subsequent figures.) Equal quality axis-aligned filtering AAF (EQ) requires significantly more samples, and is slower for raytracing due to adaptive sampling and incoherent rays. Adaptive polynomial rendering (APR) involves significant additional offline computation but still has blurring artifacts and is only of comparable quality as our method. (Note that our APR implementation is on the CPU, but we scale the reported times to match GPU timings reported in the original paper). In this example, our MAAF method is more than 7× faster than [MYRD14] for equal quality, and is visually close to ground truth. Note that multiple distribution effects cannot even be accurately combined with fast sheared filtering [YMRD15].*

Here, axis-aligned refers to the higher dimensions rather than the image plane (light for soft shadows, lens for depth of field, or incident angle for global illumination). A *single* axis-aligned filter does not tightly bound the wedge-shape of the frequency spectrum, and includes a large region not in the wedge. While sampling rates are reduced compared to basic Monte Carlo sampling, they are much higher than sheared filtering. However, the simple nature of the axis aligned filter makes it very easy and efficient to implement. It is also naturally separable in pixel-light, pixel-lens space, requiring minimal storage and reducing to simple image filtering, thus enabling interactive rendering with a GPU raytracer (we use NVIDIA's OptiX) with minimal filtering overhead. Closest to our work, [MYRD14] use factored axis-aligned filters to combine distribution effects; in our paper, we demonstrate an average speedup of about 6× for equal quality (Fig. 2).

Most recently, [YMRD15] developed a fast sheared-filtering method, factoring the 4D sheared filter into four 1D filters, and dramatically reducing the computational complexity of sheared filtering. This enabled a significant reduction in sample count for individual distribution effects, compared to axis-aligned filtering, with a 4× speedup in rendering time. However, the method is complicated, requiring a multi-stage algorithm with significant high-dimensional storage overhead and processing, which can limit the technique to low sample counts in applications. It was also demonstrated in practice only for single effects (only soft shadows, or depth of field, or indirect illumination). [YMRD15] do propose an approximation for multiple effects in their appendix, but it essentially reduces to filtering each effect one by one, assuming the multiple effects are unrelated; this can cause errors and overblur.

Motivation: The process of sampling results in replicated spectra in the Fourier domain, as shown in Fig. 3. To avoid aliasing, our reconstruction filter must be able to isolate the central replica. Even if the replicas themselves don't overlap, a larger-than-necessary reconstruction filter may still lead to aliasing, and therefore to higher sampling rates to avoid aliasing. The ideal compact filter would therefore bound the double-wedge Fourier spectrum exactly. However, this does not lead to a simple filter in the spatial domain, and creates a highly irregular reconstruction problem. Therefore, current filters are tradeoffs between compactness and simplicity.

Axis-Aligned filtering (AAF) is aimed at simplicity, reducing to image-space filtering. However, as seen in Figs. 1a and 3a, there is more empty space outside the double wedge than within it. Sheared filtering is more compact/tighter (Figs. 1b and 3b). However, the double-wedge has two triangular-shaped regions, which cover only half the area of the sheared parallelogram filter. Moreover, the sheared 4D filter can be difficult to implement even with FSF, and involves high-dimensional storage and processing. Our goal is to design a filter that is (1) More compact in the Fourier domain than either AAF or FSF, and (2) Simple in the primal domain, to enable easy implementation and fast filtering.

Insights and Contributions: We develop a simple approach using *multiple axis-aligned filters (MAAF)*, shown for simplicity for a flatland or 2D wedge in Fig. 1 (our results use full 4D or higher-dimensional filters, obtained as products of these 2D filters). Our MAAF method in Fig. 1(c) covers the wedge accurately with a small number of axis-aligned filters (shown as boxes here; we use gaussians in practice). We obtain greater accuracy and compactness than axis-aligned filtering (AAF) or (fast) sheared filtering (FSF) (see analysis in Sec. 5). Moreover, each of the axis-aligned filters in

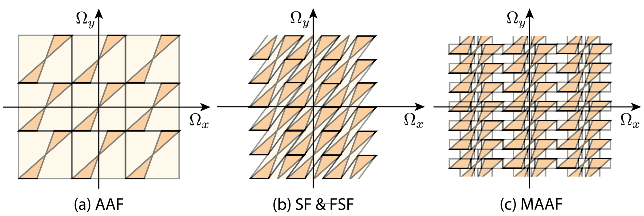


Figure 3: *Showing replicas from sampling, and how large empty space in axis-aligned filtering leads to a higher sampling rate. Sheared filtering and our multiple axis-aligned filter method lead to much tighter packing of replicas, and lower sampling rates.*

MAAF can be implemented very simply as image-space filtering, and the entire multiple filter algorithm is implemented in a single pass in graphics hardware. While overhead for MAAF grows with the number of filters, it is still less than the cost of GPU raytracing.

Since we bound the wedge filter tightly, our performance is comparable to fast sheared filtering for single effects like soft shadows. Crucially, we show how our method can be *extended to multiple distribution effects*. In this case, we demonstrate speedups of up to 7× over AAF [MYRD14]. Our main contributions are:

Multiple Axis-Aligned Filters (MAAF): Our key insight is the development of multiple axis-aligned filters (MAAF). Section 4 introduces the mathematical formulation of MAAF in the 2D Fourier domain, and shows how a pair of component filters can be written as two separable filters in the primal domain.

Analysis: In Sec. 5, we analyze MAAF for 2D wedge spectra, comparing to FSF and AAF. We consider both the *coverage*, or ratio in the area of the Fourier wedge spectra to the full filter, and *accuracy* of the filter within the wedge. We show that MAAF provides both better coverage and better accuracy.

Algorithms: In Sec. 6, we develop algorithms to use MAAF. We show how to multiply two 2D spectra together to get a full 4D MAAF. Finally, we show how to extend the algorithm to higher-dimensional 6D filters, as needed for multiple distribution effects.

Practical Results: Our results are presented in Fig. 2 and Sec. 7, demonstrating one of the first practical approaches for rendering accurate multiple distribution effects within a couple of seconds.

2. Previous Work

There is a long history of adaptive sampling, image filtering and denoising, going back to seminal work by [Mit91, Guo98]. [ZJL*15] classifies algorithms into *a-priori* methods [ETH*09] and *a-posteriori* techniques [HJW*08, ODR09]. Ours is an *a-priori* approach, relying on prior theoretical knowledge of the shape of the frequency spectrum, with parameters based on an initial sparse sampling pass. Our method leverages many seminal results on frequency domain and Fourier analysis for the wedge spectra of common distribution effects [CCST00, DHS*05], and also relates to algorithms derived from them [SSD*09, BSS*13]. While measured spectra shown in these papers are scene-dependent, the double wedge shape is a tight bound in all but the most contrived cases [EHDR11].

We build most directly on axis-aligned filtering [MYRD14] and fast sheared filtering [YMRD15]. Besides FSF, a few methods employ similar ideas for more limited effects like defocus blur [VMCS15], and combinations of defocus and motion blur [CM14, MVH*14]. Those methods are less general, assuming a partition of the scene into multiple layers, each of which is treated with a fixed filter.

A-posteriori methods make limited initial assumptions about the form of the image signals, filtering after the fact, and can handle general visual effects. However, they are intended for much slower offline rendering (recent approaches like [MGYM15] or [BEJM15] develop fast filtering methods, but are still not intended for near-interactive applications, or make approximations for multiple effects). Earlier work in the area includes random parameter filtering [SD12], statistical approaches like SURE [LWC12], non-local means [RKZ12], ray histograms [DMBM14], weighted local regression [MMMG16], and machine learning for denoising [KBS15]. These methods do not exploit the Fourier structure of the light field and typically require higher sampling rates, with a more complex offline reconstruction method.

Many approximate real-time techniques have been proposed for specific distribution effects. Soft shadow maps for area sources are a popular method [GBP07]. Other methods include [AM03, LAA*05]. All of these methods make tradeoffs in accuracy for speed [JHH*09]. For depth of field, simple post-processing algorithms have been known for decades [PC81]; some recent approaches are [YWY10, LH13]. A variety of real-time global illumination methods have been studied and used in interactive applications like games; a recent survey is [RDGK12]. These methods all make approximations, which can produce aliasing and artifacts, in exchange for high performance. In contrast, our method is based on Monte Carlo ray or path tracing, providing accurate physically-based results efficiently.

3. Background

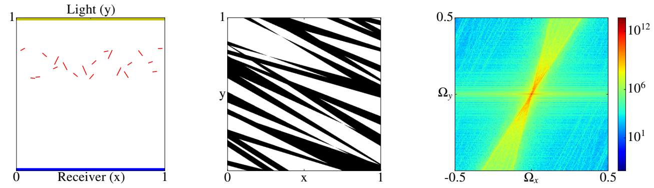
We first consider single distribution effects like soft shadows in flatland, and introduce the 2D wedge spectrum (illustrated in Fig. 4). Our notation is based on [MYRD14, YMRD15]. A simplified version of the rendering problem for soft shadows can be written as,

$$\bar{h}(x) = \int_{-L}^{L} f(x,y) I(y)\, dy, \tag{1}$$

where $\bar{h}(x)$ is the image intensity, $f(x,y)$ is the (usually binary) visibility function between pixel x and light source location y (Fig. 4b), and $I(y)$ is the intensity of the light. We use a bar on $\bar{h}(x)$ since this result will be noisy with a few samples, and we will eventually filter to obtain final intensity $h(x)$. [MWR12] assume a gaussian function for $I(y)$, with $L = 2\sigma_y$, where $2L$ is the length of the light source, and σ_y is the standard deviation of I. For simplicity, we have neglected BRDF effects and cosine terms, which are usually taken out of the integral in previous work.

Fourier Spectrum: For a planar (linear in flatland) occluder, the Fourier spectrum of f is a single line in 2D, with slope $s = d_1/d_2 - 1$ [EHDR11]. Here, d_1 is the distance from the light to the receiver and d_2 is the distance from light to occluder. For multiple occluders at different depths, most of the energy lies within a double wedge, bounded by slopes $s_{\min}$ and $s_{\max}$ (Fig. 4c). Note that this double-wedge spectrum is filtered by the light spectrum along the Ω_y axis, with a maximum bandwidth of $\Omega_y^{\max} = 2/\sigma_y = 4/L$. The resulting canonical double-wedge spectrum is shown in Fig. 1.

A sequence of papers has shown that a similar shape applies for most distribution effects, including motion blur, depth of field, and



(a) flatland scene (b) occlusion function (c) occluder spectrum

Figure 4: *Numerical simulation of a flatland scene for soft shadows (a) Scene setup, similar to Fig. 7 in [EHDR11]. (b) Graph of occlusion function. (c) Fourier spectrum. Notice higher energy in the center of the double wedge, and falloff further from the origin.*

diffuse indirect illumination. For depth of field, L in equation 1 is replaced by the lens aperture A, and the slope s is determined by the circle of confusion, and given formally by $s = V(F/z-1)/S$, where V is the half-width of the image in pixels, F is the focal distance, S is the size of the focal plane, and z is the depth of the object. For diffuse global illumination, [MWRD13] derives $\Omega_y^{\max} \approx 2.8$, with $s = z$ for a single reflector at depth z. We do not explicitly consider specularities in this paper, and show results only for diffuse objects. However, as in [MWRD13] slightly glossy reflectors and receivers can usually work in practice.

Axis-Aligned and Sheared Filters: Mathematically, the axis-aligned filter (Fig. 1a) corresponding to equation 1, is

$$W(\Omega_x,\Omega_y) = G\left(\Omega_x; \frac{\Omega_y^{\max}}{s_{\min}}\right) G(\Omega_y;\Omega_y^{\max}), \tag{2}$$

where $W(\Omega_x,\Omega_y)$ is the axis-aligned filter in the Fourier domain, written in terms of gaussians G along Ω_x and Ω_y respectively. The second argument to G is the bandwidth or standard deviation. Bandwidth $\Omega_y^{\max}$ is just the maximum light frequency, while the bandwidth along Ω_x is scaled to account for shearing by minimum slope $s_{\min}$. The dimensions can thus be treated separately, and the y dimension is just the standard integration against the light source. Therefore, we can simply define an image-space filter for spatially-varying convolution with final image $h(x)$ (note that β is proportional to the inverse width of Fourier filter, $s_{\min}/\Omega_y^{\max}$),

$$h(x) = \int_{x'} \bar{h}(x')w(x-x')dx' \qquad w(u) = g(u;\beta). \tag{3}$$

To distinguish notations, we use lowercase letters g and w for primal domain gaussians and the filter, respectively.

For sheared filtering, the filter is a parallelogram,

$$W(\Omega_x,\Omega_y) = G\left(\Omega_x - \frac{\Omega_y}{s_{\text{avg}}};\Omega_y^{\max}(s_{\min}^{-1}-s_{\max}^{-1})\right) G(\Omega_y;\Omega_y^{\max}), \tag{4}$$

where s_{avg} is the harmonic mean of slopes $s_{\min}$ and $s_{\max}$. The shearing couples Ω_x and Ω_y in the first gaussian. The spatial domain form is also a parallelogram, sheared in the orthogonal direction,

$$h(x) = \int\int f(x',y')w(x',y';x,y)\,dx'\,dy'$$
$$w(x',y';x,y) = g(x'-x;\beta_x)g(y'-\eta(x-x');\beta_y), \tag{5}$$

where the shear η and standard deviations β_x and β_y relate to the slopes $s_{\min}$ and $s_{\max}$, as well as the bandwidth $\Omega_y^{\max}$ [YMRD15], The precise relations are not critical for this paper.

Rendering: For interactive rendering, we use physically-accurate Monte Carlo raytracing, with NVIDIA's Optix raytracer on a GPU to determine the function $f(x,y)$ and compute the integral in equation 1. However, this may require hundreds or thousands of samples per pixel to converge. Instead, we use a very sparse set of rays (typically 4-9) at each pixel. This result is then convolved (filtered) with a spatially-varying filter, based on the frequency analysis of the function $f(x,y)$ discussed above. In practice, $s_{\min}$ and $s_{\max}$ are also esimated during ray-tracing, and filtering is done in the primal spatial (not Fourier) domain, as in previous work.

Axis-aligned filters (Fig. 1a) reduce to image filtering (spatially-varying convolution) of the noisy $\bar{h}(x)$, per equation 3. For sheared filters (Fig. 1b), the FSF algorithm [YMRD15] can be applied to equation 5. We show that our MAAF method (Fig. 1c) is more efficient than either approach. In summary, we differ from previous work in using a more accurate filter, that tightly bounds the Fourier spectrum of f, enabling a higher-quality, more efficient algorithm.

To handle a real three-dimensional scene instead of a flatland scene, we will extend the current 2D spectrum to 4D. The extension to 4D is a simple product of 2D filters; this, along with the multiple effect case, will be addressed later in the paper in Sec. 6.

4. Multiple Axis-Aligned Filtering

We introduce a mathematical formulation for 2D Multiple Axis-Aligned filters (MAAF); we will consider the extension to 4D filters as a product of 2D components in Sec. 6. Even though the Fourier filters are offset from the origin, we show how a separable filter can be derived in the primal domain. Thereafter, Sec. 5 analyzes the coverage and accuracy of MAAF, compared to AAF and FSF.

Insight—Covering the Spectrum: We cover the double wedge spectrum using a series of multiple axis-aligned filters (Figs. 1(c), 5(a)). For simplicity, one can think of MAAF as a number of boxes (we typically use 5); each "box" will in practice be an axis-aligned gaussian filter. The filters are arranged symmetrically; for each "box" except the central one, there is always another box/filter symmetric to it about the origin, denoted as a *pair*. The central filter is symmetric to itself, and considered a special pair. The technique of approximating anisotropic filters by several isotropic filters is similar to [MPFJ99] for anisotropic texture map filtering and [SBN15] for anisotropic spherical function decomposition.

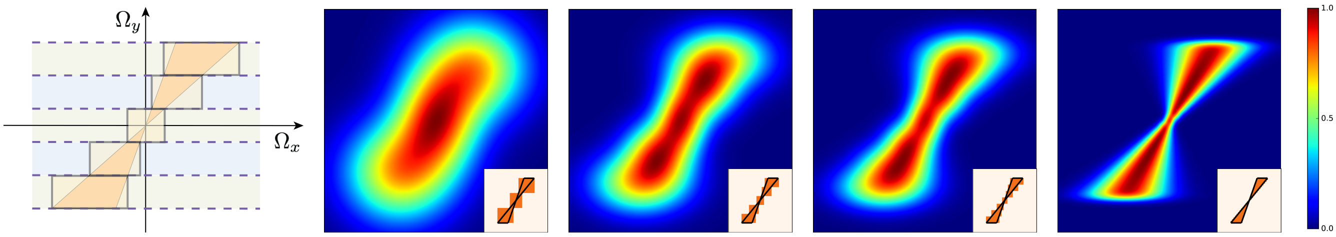
Numerically, MAAF accuracy (compactness) increases with the number of component filters, approaching the double wedge itself, as shown in Fig. 5. However, we have to consider each filter's contribution individually, so the filtering complexity is linear in the number of pairs in 2D, and even quadratic in 4D. To balance efficiency and compactness, we usually choose 3 to 7 filters for MAAF. If the number of pairs is N (including the central filter), there are $2N-1$ total filters, which partition the vertical extent of the double wedge $2\Omega_y^{\max}$ equally into $2N-1$ segments.

Fourier Domain Formula for MAAF: Starting at the origin (central filter), and moving outward, we label each component filter with numbers $p = 0, \pm 1, \pm 2, \ldots, \pm(N-1)$ and center (C_x^p, C_y^p),

$$C_y^p = \Omega_y^{\max}\left(\frac{2p}{2N-1}\right). \tag{6}$$

The extent in the Ω_x axis is chosen to bound the wedge as compactly as possible, as shown in Figs. 1(c) and 5(a). Specifically,

$$C_x^p = \frac{1}{2}\Omega_y^{\max}\operatorname{sgn}_p\left(\frac{1}{s_{\max}}\frac{2|p|-1}{2N-1} + \frac{1}{s_{\min}}\frac{2|p|+1}{2N-1}\right), \tag{7}$$



(a) (b) 3 components (c) 5 components (d) 7 components (e) 25 components

Figure 5: *Progressively more compact and accurate approximations of the double wedge with more component filters in MAAF. Insets show a schematic with a number of box filters; we actually use weighted gaussians for the filters. (a) is a schematic with 5 filters typically used.*

where sgn_p is $0,\pm 1$ depending on whether p is zero, positive or negative. Note the expected symmetry, $(C_x^{-p}, C_y^{-p}) = (-C_x^p, -C_y^p)$.

We also need the filter widths (σ_x^p, σ_y^p), which correspond to the extent of the filters in Figs. 1(c), 5,

$$\sigma_y^p = \frac{\Omega_y^{\max}}{2N-1} \qquad \sigma_x^p = \frac{1}{2}\Omega_y^{\max}\left(\frac{1}{s_{\min}}\frac{2\,|p|+1}{2N-1} - \frac{1}{s_{\max}}\frac{2\,|p|-1}{2N-1}\right). \quad (8)$$

Note that σ_x^0 needs to be modified for $p=0$, and is simply a standard axis-aligned filter, $\sigma_x^0 = \left(\Omega_y^{\max}/s_{\min}\right)/(2N-1)$.

Finally, we can write the expression for the Fourier domain MAAF as a sum over the component gaussian filters,

$$\boxed{W(\Omega_x,\Omega_y) = \sum_{p=-(N-1)}^{N-1} \mu_p\, G\left(\Omega_x - C_x^p; \sigma_x^p\right) G\left(\Omega_y - C_y^p; \sigma_y^p\right),} \quad (9)$$

where μ_p is a weight, which attenuates the component filters further away from the origin. This accounts for the usually lower contribution at the ends of the wedge spectrum. As shown in Fig. 4, there is usually higher energy at the center of the double wedge and less at the ends. In effect, μ_p gives lower importance to filters with high $|p|$. We use a gaussian over the entire double wedge,

$$\mu_p = G\left(C_x^p; \frac{\Omega_y^{\max}}{s_{\min}}\right) \cdot G(C_y^p; \Omega_y^{\max}). \quad (10)$$

Discussion: Equation 9 is similar to the axis-aligned filter in equation 2, except we sum over filters, with each filter offset, and with standard deviations set to tightly bound the wedge spectrum. There is also some implicit similarity to the sheared filter form in equation 4, since the centers (C_x^p, C_y^p) lie close to a sheared line with slope s_{avg} to match the wedge shape. We seek to combine benefits of axis-aligned and sheared filtering to provide an even more compact filter.

We have also tried other arrangement schemes of MAAF component filters, including optimizing for better placement and size of component filters, and empirically fitting placement and size using data-driven approaches. But, we did not observe significant improvement over our current simple approach. Moreover, these methods may potentially affect our interactive performance due to their overhead. Therefore, it is simplest to compute the filter parameters in the straight-forward, uniform way.

So far, we have considered only the Fourier domain. However, final filtering must be performed in the primal domain, as in prior work. While each component filter in equation 9 is separable along the Ω_x and Ω_y axes, the offsets make it unclear there is a correspondingly simple and separable form in the primal domain along x and y. We address this next, deriving a simple primal filter.

Pairwise Filtering in the Primal Domain: We seek to obtain the primal domain filter by inverse Fourier transforming equation 9, which can be done separately for each component filter. By the Fourier shift theorem, with $j = \sqrt{-1}$,

$$\mathcal{F}^{-1}\left[G(\Omega_x - C_x)G(\Omega_y - C_y)\right] = e^{jC_x x} e^{jC_y y} \mathcal{F}^{-1}\left[G(\Omega_x)G(\Omega_y)\right]. \quad (11)$$

For notational simplicity, we omit the standard deviations for now, as well as the superscript p, denoting the filter number. Note that the weight μ_p carries over directly to the primal domain.

Equation 11 is not practical to apply directly, since the exponential terms introduce imaginary parts. (In fact, it can only be applied to the central filter where $C_x^0 = C_y^0 = 0$, which is a standard axis-aligned filter.) However, when a pair of symmetric filters is transformed together, the imaginary parts cancel as follows,

$$\begin{aligned} &\mathcal{F}^{-1}\left[G(\Omega_x - C_x)G(\Omega_y - C_y)\right] + \mathcal{F}^{-1}\left[G(\Omega_x + C_x)G(\Omega_y + C_y)\right] \\ &= \left(e^{j(C_x x + C_y y)} + e^{-j(C_x x + C_y y)}\right)\mathcal{F}^{-1}\left[G(\Omega_x)G(\Omega_y)\right] \\ &= 2\cos(C_x x + C_y y)g(x)g(y). \end{aligned} \quad (12)$$

Note that the primal domain gaussians have widths proportional to the inverse of the Fourier domain versions, i.e., σ_x^{-1} and σ_y^{-1}.

The above expression is not yet separable, as required for primal domain axis-aligned filtering. To achieve this, we further expand the cosine term, collecting x and y terms separately as

$$\begin{aligned} &2\cos(C_x x + C_y y)g(x)g(y) \\ &= 2\cos(C_x x)\cos(C_y y)g(x)g(y) - 2\sin(C_x x)\sin(C_y y)g(x)g(y) \\ &= 2\left[\cos(C_x x)g(x)\right] \cdot \left[\cos(C_y y)g(y)\right] - \\ &\quad 2\left[\sin(C_x x)g(x)\right] \cdot \left[\sin(C_y y)g(y)\right]. \end{aligned} \quad (13)$$

Equation 13 indicates that a pair of filters in the Fourier domain transforms into two *separable* filters in the primal domain. This enables filtering much like standard AAF, and storage is significantly lower than FSF. Given a pixel in x, for each sample in y, we just need to pre-accumulate its corresponding $\cos(C_y y)g(y)$ and $\sin(C_y y)g(y)$ terms during the sampling process, instead of storing them individually in FSF. The only difference from standard AAF are the multiplicative sine and cosine weights. Subsequent filtering in x works directly in the image domain, like AAF.

In general, we consider N pairs, for a total of $2N-1$ filters, which are summed up in the primal domain. While the computational complexity of filtering does grow with N, all corresponding filters in the primal domain can be summed together in a single pass, without the need for additional storage. Given the very low overhead in axis-aligned filtering, MAAF also involves acceptable overheads.

5. Analysis

We now compare MAAF with axis-aligned and sheared filters, as well as to the theoretically optimal double-wedge spectrum bound. While we use the labels AAF, FSF as before, note that these results are independent of any algorithm, looking at the intrinsic properties of how well the various filters fit the 2D double-wedge spectrum in the Fourier domain. As such, this is also a valuable baseline analysis for any future filter development techniques.

Filter Metrics: We assume the function to be approximated is the (completely filled) double wedge, denoted as T, with support $\mathrm{supp}(T)$. The most compact filter is simply the double wedge, 1 over $\mathrm{supp}(T)$ and 0 outside. We evaluate each of our filters $W(\Omega_x,\Omega_y)$ in terms of standard signal-processing notions of *accuracy* α and *coverage* γ [Pra07]. We define *accuracy* simply by subtracting normalized error, i.e., the standard L_2 difference between the double wedge and the filter within its support, normalized by the area of the double wedge,

$$\alpha = 1 - \sqrt{\frac{\iint_{\mathrm{supp(T)}} (W(\Omega_x,\Omega_y)-1)^2\, d\Omega_x\, d\Omega_y}{\iint_{\mathrm{supp(T)}} d\Omega_x\, d\Omega_y}}, \tag{14}$$

where we set $T(\Omega_x,\Omega_y)=1$ within its support, and the square root coresponds to considering the RMS error. Also note that if we define filters W using box functions instead of gaussians, they fully cover the support of the wedge for AAF, FSF and MAAF, so $\alpha=1$. However, gaussians introduce some error, which we measure.

We define the coverage as the ratio between the area of the filter within the wedge spectrum, and the total area of the filter. Coverage of 1.0 (100%) is ideal with lower values indicating wasted space in the filter outside the double wedge,

$$\gamma = \frac{\iint_{\mathrm{supp(T)}} \parallel W(\Omega_x,\Omega_y)\parallel\, d\Omega_x\, d\Omega_y}{\iint \parallel W(\Omega_x,\Omega_y)\parallel\, d\Omega_x\, d\Omega_y}. \tag{15}$$

Analytic coverage for box/parallelogram filters: Analytic forms for α,γ are easier to derive when the filters are (unweighted) boxes or parallelograms, rather than gaussians. In these cases, $\alpha=1$ by construction, while the coverage varies. From simple geometry, the area of the double wedge is simply $\left(\Omega_y^{\max}\right)^2\left(s_{\min}^{-1}-s_{\max}^{-1}\right)$. The area of the simple axis-aligned filter is $4\left(\Omega_y^{\max}/s_{\min}\right)^2$. Similarly, a sheared filter is simply a tight parallelogram instead of a wedge (triangle) with net area $2\left(\Omega_y^{\max}\right)^2\left(s_{\min}^{-1}-s_{\max}^{-1}\right)$ and coverage is

$$\gamma_{AAF} = \frac{s_{\min}}{4}\left(1-\frac{s_{\min}}{s_{\max}}\right) \qquad \gamma_{FSF} = \frac{1}{2}. \tag{16}$$

It can readily be seen that AAF has poor coverage, with significant wasted space. In fact, γ reduces to 0 if $s_{\min}=s_{\max}$, in which case the wedge itself degenerates to a single line, but a larger axis-aligned box must still be used. Note however, that FSF also has coverage of only 50%, and is twice as large as necessary.

For MAAF, we need to compute the area of the filter, or sum of box filters. The area of filter p (assumed a rectangular box) is given by $4\sigma_x^p\sigma_y^p$, multiplying the widths in equation 8, and the total MAAF area sums over all boxes. After some algebraic rearrangements,

$$\sum_{p=-(N-1)}^{N-1} 4\sigma_x^p\sigma_y^p = 4\left(\frac{\Omega_y^{\max}}{2N-1}\right)^2 \times \left(\frac{1}{(s_{\min})^2}+2\sum_{p=1}^{N-1}\left[p\left(\frac{1}{s_{\min}}-\frac{1}{s_{\max}}\right)+\frac{1}{s_{avg}}\right]\right). \tag{17}$$

After some simplifications, the coverage is then derived as,

$$\gamma_{MAAF} = \left[\frac{N(N-1)}{\left(N-\frac{1}{2}\right)^2} + \frac{\left(\frac{1}{(s_{\min})^2}+\frac{2(N-1)}{s_{avg}}\right)}{\left(\frac{1}{s_{\min}}-\frac{1}{s_{\max}}\right)\left(N-\frac{1}{2}\right)^2}\right]^{-1}. \tag{18}$$

It is helpful to consider the limits of the expression above. When $N=1$, we have only the central filter, and this reduces to axis-aligned filtering. Indeed, γ_{MAAF} simplifies to γ_{AAF} for $N=1$. On the other hand, for large N, the first term dominates and we have that $\gamma_{MAAF}\to 1$, approximating the double wedge accurately.

In Table 1, we compare coverages of AAF, FSF, and MAAF with increasing numbers of component filters. The results are averaged over many different slopes of $s_{\min}$ and $s_{\max}$. Conceptually, we are simulating many different instances of Fig. 4. Specifically, for all the simulations in this section, we randomly generate 50 sets of double wedge spectra with $0<s_{\min}<s_{\max}<10$. This range is similar to those in real scenes. We compute the results for each set, and average over all 50 sets. From Table 1, coverage for MAAF with a small number of filters is comparable to FSF. However, as the number of filters increases, MAAF coverage approaches 1.

Average Coverage Rate (#components)			
AAF	FSF	MAAF (3)	MAAF (5)
0.097	0.5	0.197	0.275
MAAF (7)	MAAF (25)	MAAF (75)	MAAF (125)
0.401	0.677	0.876	0.943

Table 1: *Coverage computation for AAF, FSF, and MAAF. This table shows box/parallelogram filters for simplicity. Note that coverage converges to* 1 *with more filters in MAAF.*

Shape		Coverage Rate (#components)				
$s_{\min}$	$s_{\max}$	AAF	FSF	MAAF (3)	MAAF (5)	MAAF (7)
Average		0.088	0.199	0.173	**0.256**	0.316
0.5	0.8	0.079	0.168	0.122	**0.174**	0.218
1.5	2.0	0.047	0.134	0.090	**0.156**	0.210
0.7	2.2	0.172	0.216	0.245	**0.315**	0.383

Shape		Accuracy (#components)				
$s_{\min}$	$s_{\max}$	AAF	FSF	MAAF (3)	MAAF (5)	MAAF (7)
Average		0.662	0.731	0.748	**0.799**	0.827
0.5	0.8	0.585	0.826	0.833	**0.903**	0.933
1.5	2.0	0.639	0.686	0.761	**0.811**	0.842
0.7	2.2	0.616	0.742	0.734	**0.793**	0.824

Table 2: *Accuracy and coverage for different filtering methods, with gaussian filters as used in practical implementations. With a small number of filters, MAAF outperforms AAF and FSF.*

Numerical evaluation for Gaussian filters: We now consider the weighted gaussian filters we actually use, per equation 9, and numerically evaluate accuracy and coverage. Table 2 provides results for AAF, FSF and MAAF (all using gaussian filters as in practical

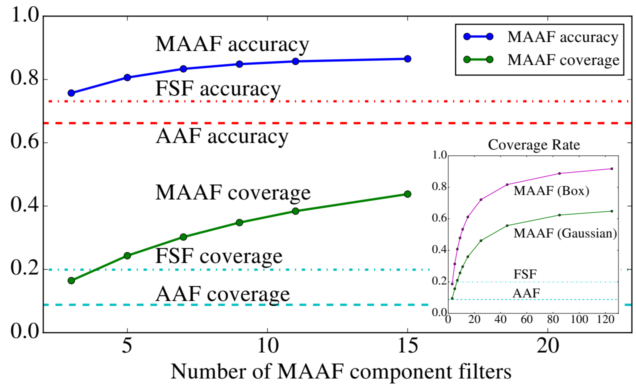


Figure 6: *Graph showing increasing coverage and accuracy of MAAF as the number of component filters is increased.*

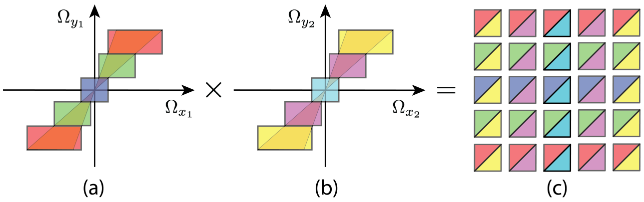


Figure 7: *Constructing 4D MAAF filters from 2D filters along orthogonal dimensions, using different colors to distinguish different components. (a) 5 components in $(\Omega_{x_1}, \Omega_{y_1})$. (b) 5 components in $(\Omega_{x_2}, \Omega_{y_2})$. (c) When multiplied in the frequency domain to cover the 4D spectrum, these result in 25 filters in 4D.*

implementations). We consider both the average over 50 randomly generated wedges as above, as well as results for 3 representative values of $s_{\min}$ and $s_{\max}$. We show MAAF with 3,5,7 filters (we usually use 5 filters, shown in bold). MAAF with only 5 gaussian filters is more accurate than either FSF or AAF, and has higher coverage than either. Note that the coverage rates for both FSF and MAAF are lower than for the box case, since we are using gaussians. Finally, Fig. 6 shows how MAAF accuracy and coverage increase with increasing numbers of filters. Note that the MAAF coverage rate and accuracy do not converge to 1 in this case, owing to the tails of the gaussians. However, MAAF still performs significantly better than FSF or AAF. As seen in Figs. 2, 6, a small number of 5-7 filters suffice for superior coverage/accuracy.

6. Algorithms

We now describe our algorithm for rendering with MAAF. In Sec. 6.1, we consider single distribution effects, explaining the method in flatland with 2D spectra, then discussing the extension to 3D and 4D spectra. In Sec. 6.2, we discuss the extension to combining multiple distribution effects, which is our main practical contribution. Results with multiple effects are presented in Sec. 7.

6.1. Single Effects

Basic 2D Algorithm: As in previous work, we first use a real-time GPU raytracer (NVIDIA's OptiX 3.9) to compute a sparse sampling of $f(x,y)$ in equation 1 for each pixel x. Typically we use 4-9 stratified samples per pixel to enable interactive performance. At this time, we also determine the parameters of the wedge spectrum $s_{\min}$ and $s_{\max}$. The maximum filter bandwidth $\Omega_y^{\max}$ is known from the size of the light for soft shadows, or aperture for depth of field, and uses a fixed value for diffuse/glossy receivers for indirect global illumination. (Similar ideas apply to motion blur, but OptiX does not easily support ray-tracing motion-blur). Our goal is to compute the final image $h(x) = \iint f(x',y)w(x',y;x)\,dx'\,dy$ similar to equation 5, where

$$\begin{aligned} w(x',y;x) &= g(x-x';\frac{1}{\sigma_x^0})g(y;\frac{1}{\sigma_y^0}) \\ &+ 2\sum_{p=1}^{N-1}\cos(C_x^p(x-x')+C_y^p y)g(x-x';\frac{1}{\sigma_x^p})g(y;\frac{1}{\sigma_y^p}) \end{aligned} \quad (19)$$

is our MAAF filtering in primal domain.

We now set parameters for the MAAF, determining the Fourier domain centers (C_x^p, C_y^p) and bandwidths (σ_x^p, σ_y^p) of the component filters as per equations 7 and 8. We then use equation 13 to decompose each cosine term and accumulate the values at each pixel in the primal (spatial) domain,

$$\begin{aligned} h_c^p(x) &= \int f(x,y)\cos\left(C_y^p y\right) g\left(y;\frac{1}{\sigma_y^p}\right) dy \\ h_s^p(x) &= \int f(x,y)\sin\left(C_y^p y\right) g\left(y;\frac{1}{\sigma_y^p}\right) dy, \end{aligned} \quad (20)$$

where the gaussian g uses the inverse Fourier-domain bandwidth, and the integrals are done by standard Monte Carlo summation over the sparse samples in y, only including an additional sine or cosine weight. Note that since we consider filter pairs, we only need to use $p \geq 0$. No new ray-tracing needs to be done, beyond computing $f(x,y)$. The final image can be computed by filtering as per equation 3, summing over the component filters,

$$\begin{aligned} h(x) &= \int_{x'} h_c^0(x')g\left(x-x';\frac{1}{\sigma_x^0}\right)dx' \\ &+ 2\sum_{p=1}^{N-1}\int_{x'} h_c^p(x')\cos\left(C_x^p(x-x')\right)g\left(x-x';\frac{1}{\sigma_x^p}\right)dx' \\ &- 2\sum_{p=1}^{N-1}\int_{x'} h_s^p(x')\sin\left(C_x^p(x-x')\right)g\left(x-x';\frac{1}{\sigma_x^p}\right)dx'. \end{aligned} \quad (21)$$

We are still doing spatially-varying convolutions, only weighting

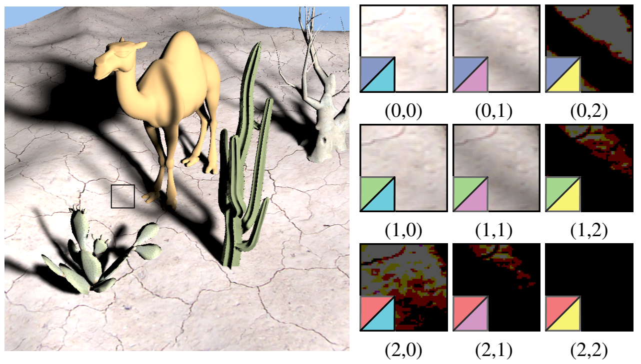
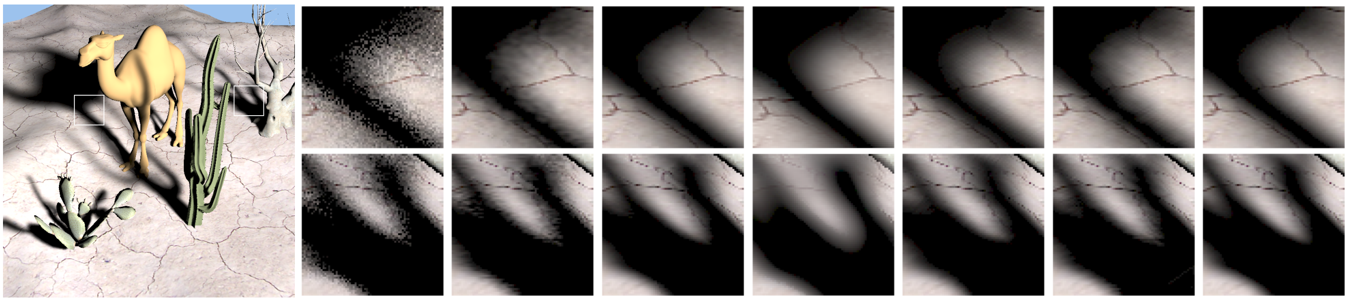


Figure 8: *Schematic of accumulating contributions for each component filter. (i, j) means $h_c^i + h_s^i$ for (x_1, y_1) and $h_c^j + h_s^j$ for (x_2, y_2); we also color-code based on the schematic in Fig. 7. Note that the center $(0,0)$ produces a highly overblurred result, and the outer filters contribute high frequency details. The outermost filters contribute less due to lower importance.*



Soft Shadows, MAAF 9 spp, 0.10s	MC Input 9 spp, 0.07s	AAF (ET) 12 spp, 0.10s	AAF (EQ) 35 spp, 0.33s	APR 13 spp, 26.0s	MAAF 9 spp, 0.10s	FSF 9 spp, 0.10s	Ground Truth 1K spp, 10.9s

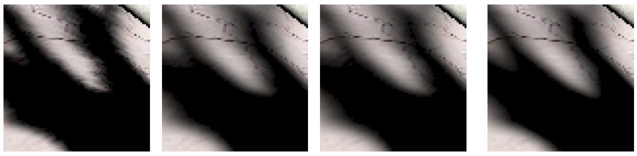
Figure 9: *Soft shadows rendered with 25 total 4D filters for MAAF (5 filters for each 2D subspace) at 9 samples per pixel, also compared to AAF and FSF. Our method is about $3\times$ faster than AAF, and comparable to FSF for single distribution effects.*

the gaussian filter with appropriate sines and cosines. This can still be accumulated in a single pass in graphics hardware, with time complexity proportional to N. Note that each component filter involves cost only comparable to standard axis-aligned filtering.

Extension to 3D with 4D Filters: The extension to 3D is straightforward, and we handle the full 4D spectrum ($f(x,y)$ now becomes $f(x_1,x_2,y_1,y_2)$ since pixels and lights are both 2D quantities). As in previous work, we can consider the 4D filtering as a product of two orthogonal 2D subspaces, in each of which we can use our 2D filters. The product of these 2D subspaces forms the 4D space. Hence, 4D MAAF filters are constructed by combining (taking the product of) 2D filters along orthogonal dimensions, as shown in Fig. 7. The number of 4D filters does grow as the square of the number of 2D filters, but overhead for filtering is still low. Figure 8 visualizes the contribution of each h_c^p and h_s^p.

Example: Figure 9 shows a simple example with soft shadows, also comparing the performance of MAAF, AAF and FSF (similar results hold for depth of field and global illumination). We also compare with the offline APR (Adaptive Polynomial Rendering) algorithm [MMMG16]. We achieve interactive performance of about 10fps in this case with only 9 samples per pixel, a factor of about $4\times$ fewer than what is needed for equal quality AAF. While we need to consider $5^2 = 25$ filters for the 4D spectrum, the filtering overhead is still typically only $5-8\%$ of the total time, which is dominated by GPU raytracing cost. For single distribution effects, our performance is similar in practice to fast sheared filtering (FSF). Both methods use low sample counts (9spp in this example), and overhead is only a fraction of total cost, so their GPU raytracing and overall costs are almost identical. We do have some theoretical benefits, even with only a small number of filters, as shown in Fig. 6. Moreover, we do not need to store or process high-dimensional data, unlike FSF. Our major practical contribution is in rendering multiple distribution effects, discussed in Sec. 6.2, which has not been demonstrated accurately with FSF.

We evaluate the benefits of increasing the number of component MAAF filters in Fig. 10. Using only 3 filters in 2D (9 total filters for 4D MAAF) leads to shadows being too hard, due to insufficient sampling rate (since coverage/accuracy is less for 3 filters). Increasing the number of MAAF filters does help, but 5-7 filters in 2D (25 or 49 4D MAAF filters) is already very close to ground truth. Therefore, we use 5 MAAF filters in 2D (and a total of 25) in all of our practical results.



(a) MAAF (3) (b) MAAF (5) (c) MAAF (7) (d) Ground Truth

Figure 10: *Soft shadows (Fig. 9) rendered with 3, 5 and 7 filters for each 2D subspace (9, 25, 49 total 4D filters for MAAF). Using 3 component filters is inaccurate (shadows too hard) due to the insufficient sampling rate, while 5 filters is already accurate.*

6.2. Multiple Distribution Effects

We now extend our method to handle multiple distribution effects at the same time. Specifically, we handle soft shadows, diffuse global illumination and depth of field effects together. We follow the notations in [MYRD14], using x for pixel positions, u for positions on the lens, and y for soft shadows or global illumination (as in previous work, we handle direct and indirect effects separately). Similar to equation 19, our goal is to compute the final image $h(x) = \iiint f(x',y,u)w(x',y,u;x)\,dx'\,dy\,du$, where

$$
\begin{aligned}
w(x',y,u;x) &= g(x-x';\frac{1}{\sigma_x^0})g(y;\frac{1}{\sigma_y^0})g(u;\frac{1}{\sigma_u^0}) \qquad (22)\\
&+2\sum_{p=1}^{N-1}\cos(C_x^p(x-x')+C_y^p y+C_u^p u)g(x-x';\frac{1}{\sigma_x^p})g(y;\frac{1}{\sigma_y^p})g(u;\frac{1}{\sigma_u^p})
\end{aligned}
$$

is our 3D MAAF filtering in primal domain.

It is shown in [MYRD14] that the spectrum of $f(x,y,u)$ is a 3D wedge when x, y and u are in flatland, and 6D in the three-dimensional real world. Following the development in the previous subsection, we first consider the flatland case, deriving an extension of 2D MAAF to 3D MAAF, using the same number of 3D "boxes" instead of 2D boxes. We then construct the full 6D spectrum as combinations (products) of these 3D boxes from two orthogonal 3D subspaces. This is just like the single effect case, with the same number of total filters and complexity as in Sec. 6.1. Therefore, we only discuss the flatland case with a 3D wedge spectrum below.

The 3D wedge projects to a 2D wedge on both (Ω_x,Ω_y) and (Ω_x,Ω_u) planes, as shown in Fig. 11. This allows us to design multiple axis-aligned 3D filters to tightly pack the 3D spectrum, as described below. Similar to 2D wedge spectra, we typically use 5 "boxes", which in practice are now 3D gaussian filters in $(\Omega_x,\Omega_y,\Omega_u)$. The practical implementation only requires a simple

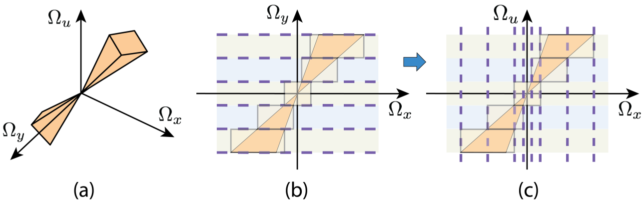


Figure 11: *Constructing MAAF for multiple distribution effects. (a) Illustration of 3D spectrum derived in flatland. (b) Partition along Ω_y first, getting the corresponding Ω_x partition. (c) Use Ω_x partition to get Ω_u partition.*

modification to equation 20 to also include u,

$$h_c^p(x) = \iint f(x,y,u)\cos\left(C_y^p y + C_u^p u\right) g\left(y; \frac{1}{\sigma_y^p}\right) g\left(u; \frac{1}{\sigma_u^p}\right) dy du$$
$$h_s^p(x) = \iint f(x,y,u)\sin\left(C_y^p y + C_u^p u\right) g\left(y; \frac{1}{\sigma_y^p}\right) g\left(u; \frac{1}{\sigma_u^p}\right) dy du. \quad (23)$$

Note that we now also integrate over u with a gaussian filter in u as well. However, this poses no further issues, since the GPU raytracer simply samples u and y for each sample. We still only need to accumulate the same $h_c^p(x)$ and $h_s^p(x)$ terms as before, and do not need to store any additional information. Only the arguments to the sines and cosines need to be modified to also consider u. Finally, the actual image-space filtering in equation 21 remains unchanged.

The only remaining question is to determine the centers and bandwidths for the 3D MAAF filters. As shown in Fig. 11(b), we first uniformly partition along the Ω_y axis within the range $[-\Omega_y^{\max}, \Omega_y^{\max}]$ based on the number of component filters. This is the same as introduced in Sec. 4. Therefore, C_y^p and σ_y^p (as well as C_x^p and σ_x^p for equation 21) are the same as in Secs. 4, 6.1.

Based on the projected 2D wedge on the (Ω_x, Ω_y) plane, we immediately know the x range each component covers (Fig. 11(c)). Then, we keep the x ranges, and use them to partition u accordingly, so that the components cover both u and y tightly. The resulting center and bandwidth for Ω_u are (analogous to equations 7 and 8),

$$C_u^p = \frac{1}{2}\Omega_y^{\max}\mathrm{sgn}_p\left(\frac{s_{\min}^u}{s_{\max}^u}\frac{2|p|-1}{2N-1} + \frac{s_{\max}^u}{s_{\min}^y}\frac{2|p|+1}{2N-1}\right)$$
$$\sigma_u^p = \frac{1}{2}\Omega_y^{\max}\left(\frac{s_{\max}^u}{s_{\min}^y}\frac{2|p|+1}{2N-1} - \frac{s_{\min}^u}{s_{\max}^y}\frac{2|p|-1}{2N-1}\right). \quad (24)$$

Superscripts u and y for s^u and s^y denote slopes on those wedges. Without loss of generality, we can assume $s_{\max}^u > s_{\min}^u > 0$.

We have also explored a simple optimization. Since both 2D spectra over (Ω_x, Ω_y) and (Ω_x, Ω_u) planes give a maximum bandwidth $\Omega_x^{\max}$ over x, we choose the smaller one. So, before we start partitioning, we first update the corresponding $\Omega_y^{\max}$ or $\Omega_u^{\max}$ using $\Omega_x^{\max}$. This allows us to pack the spectrum even more compactly.

Discussion: As noted above, the extension from single to multiple distribution effects is straightforward in MAAF, since it shares many of the characteristics of axis-aligned filters. Moreover, we can consider the product of orthogonal 3D MAAF filters to obtain the full 6D MAAF, just as we can compose 2D MAAF filters to determine a 4D spectrum for single effects. In fact, no additional storage is required, and the image-space filtering algorithm is unchanged. In contrast, sheared filtering requires operating in a higher-dimensional 6D space. It is unclear how the FSF algorithm would even extend to 6D, and it would require storage of and processing on a 6D sample set, which could be prohibitive.

Implementation Details: Our algorithm is implemented using NVIDIA OptiX 3.9 and CUDA 7.5. Since MAAF derives from axis-aligned filters, our algorithm is relatively easy to implement, with only a few modifications from an existing AAF implementation. We will make the source code available online upon publication. The algorithm consists of the following stages.

Sampling: We first trace 16 path samples per pixel. Similar to [MYRD14], a path sample consists of a primary lens ray, a shadow ray and a one-bounce indirect illumination sample. At each pixel, we collect direct slopes, indirect slopes $(s_{\min}^y, s_{\max}^y)$, defocus slopes $(s_{\min}^u, s_{\max}^u)$, world locations, normals and the pixel's projected area in world-space. Note that for a pixel near the focal plane, we will have $s_{\max}^u \cdot s_{\min}^u < 0$, which means the double wedge shape effectively degenerates to a box, and this pixel needs more samples. We adaptively trace up to 100 more samples for these pixels. The number of additional samples is estimated conservatively using the sampling rate equations in Sec. 7 of [MYRD14].

Pre-filtering: To avoid discontinuity artifacts, we average the double wedge slopes, world locations and normals over a 5×5 image window (similar to most previous work), to ensure accurate parameter estimates. Given double wedge slopes, MAAF parameters are computed as discussed above. We use equation 23 to pre-filter (sum over) y and u, then store only the accumulated values in each pixel. Our storage is thus proportional to the image size only as in AAF, and we do not need to store the full 6D data, unlike FSF.

MAAF Filtering: The final filtering pass described in equation 21 is performed in image-space. The direct illumination (soft shadows) and the indirect illumination are filtered seperately. To avoid artifacts, we reject pixels whose normals deviate more than 20 degrees.

7. Results

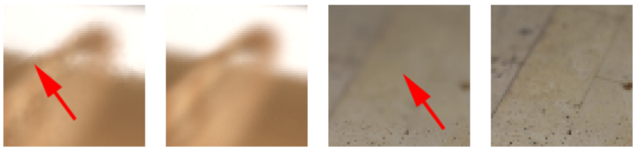
We now present the results of four scenes rendered with MAAF. Each scene includes defocus, soft shadows, and diffuse indirect illumination. MAAF is applied separately for direct and indirect components, handling soft shadows and global illumination respectively. Both components consider depth of field effects. The storage of MAAF scales quadratically with the number of component filters N; this is larger than for AAF, but much less than FSF since there is no need to store the high dimensional light field.

We compare with AAF [MYRD14] for equal time and equal quality (using the optimal sampling rates in Sec. 7 of [MYRD14]). We do not compare to FSF, since an accurate version has not been demonstrated for multiple effects. Note that our scenes include larger area lights, leading to more complex shadowing with longer raytracing times, as compared to [MYRD14].

We also compare with the state-of-the-art a-posteriori offline denoising method, adaptive polynomial rendering (APR) [MMMG16]. We implemented APR on the CPU, but this is significantly (250x) slower than a GPU implementation, as in the original paper, in terms of Gflops. For a fair comparison, we scale all reported APR running times by the Gflop ratio, which enables closely matching the timings in [MMMG16]. Even after this scaling, APR is slower than our method by nearly two orders

		MAAF (5)							AAF (Equal Quality)			
Scene	Tris	spp	Sampling	Pre filt.	Filter	Overhead	Total	Speedup	spp	Sampling	Filter	Total
CONFERENCE	163 K	32	1.67s	0.54s	0.13s	28.6%	**2.34s**	**7**.3x	158	17.07s	0.03s	17.10s
STILL LIFE	233 K	17	1.32s	0.26s	0.10s	21.4%	**1.68s**	**5**.3x	96	8.85s	0.04s	8.89s
SPONZA	262 K	16	1.29s	0.33s	0.17s	27.9%	**1.79s**	**6**.3x	107	11.25s	0.05s	11.30s
TOASTERS	2.5 K	21	0.86s	0.14s	0.12s	23.2%	**1.12s**	**3**.2x	97	3.52s	0.04s	3.56s

Figure 12: *Timings for MAAF. We see that the overhead is less than 30%, and the method requires significantly fewer samples than AAF, being significantly faster. Sampling time includes both initial sampling (path/ray tracing) and further adaptive sampling. Pre-filtering time includes slope smoothing and accumulation over y and u. Filtering is only the final spatially-varying convolution in the image domain.*



(a) MAAF 32 spp, 2.34s (b) Ground Truth 6400 spp, 869s (c) MAAF 16 spp, 1.79s (d) Ground Truth 6400 spp, 1074s

Figure 13: *Image insets from CONFERENCE (a)(b) and SPONZA (c)(d). Our method produces results close to the ground truth generally. But there are some subtle artifacts: (a). Discontinuity around shadow and focal plane boundaries, where the double wedge shape changes a lot in nearby pixels. (c). Oversmoothed region.*

of magnitude, and still has some noticeable artifacts, e.g., the over and under-smoothed shadows in Fig. 2.

We use MAAF with 5 filters in 3D (25 total filters in 6D). The results were all run at a resolution of 720×720 on an NVIDIA GTX 980 GPU. Timings are shown in Fig. 12, and result images are in Figs. 2 and 14. The accompanying video shows that MAAF also achieves temporal coherence in videos.

Timings: Figure 12 breaks down the performance of the different stages. The most expensive part of our algorithm is the pre-filtering and accumulation, which averages 0.3 seconds on these scenes. This cost is mostly from writing N^2 buffers at a time. Moreover, our experience is that the OptiX program used for prefiltering runs slower than CUDA kernels for later stages of our algorithm. The actual MAAF filtering only requires a tenth of a second. In all, the overhead is only about 25% of the total running time, which is still dominated by GPU raytracing cost. Thus, our MAAF algorithm, while introducing larger overheads than AAF, is still fast enough to be used with high-performance raytracing, especially given the considerable reduction in sample count. Total wall clock time for producing the results in Fig. 14 averages less than 2 seconds, a factor of about 6× faster on average than axis-aligned filtering (Fig. 2 has a greater than 7× speedup). This is an important step towards interactive physically-based rendering of multiple distribution effects on commodity graphics hardware.

Rendering Results: Figure 2 shows the conference room scene with 163K triangles. We are able to achieve high-quality results comparable to ground truth. AAF for the same time is noisy, and AAF equal quality requires almost 5 times as many samples. The raytracing time for AAF is 17s, nearly an order of magnitude more than our method; adaptive sampling in [MYRD14] leads to high sample counts in difficult regions with incoherent raytracing, and degrades performance. Our total running time is only 2.3s, which is more than 7× faster than equal quality AAF.

The top row in Fig. 14 shows a still life scene with 233K triangles, including strong defocus effects, as well as soft shadows and global illumination. We require only minimal additional adaptive sampling, using a total of only 17 path samples per pixel, as against the initial 16. Again, we achieve high quality results, more than 5× faster than AAF. The middle row of Fig. 14 shows the Sponza Atrium with 262K triangles, including soft shadows, defocus and indirect illumination. In this example, 16 samples per pixel suffice with no need for further adaptive sampling. We render this scene in under 2 seconds, a speedup of more than 6× over AAF, which requires 107 samples per pixel for equal quality. Finally, the bottom row of Fig. 14 shows the toasters scene, where we achieve a 3× speedup compared to AAF even on this relatively simple example. In this case, rays remain coherent, and our higher overhead leads to a smaller speedup than the 5× sample count reduction.

Limitations and Artifacts: While we achieve high accuracy, there are limitations of our method, which may lead to some oversmoothing or minor artifacts at focal plane boundaries. For instance, in the top inset in Fig. 2, the thin structures on the chair are somewhat oversmoothed, which is also the case for AAF and even APR. There is also some blurring of the ground in Sponza (middle inset). Figure 13 shows more examples. There is often a discontinuity around the region where the double wedge shape changes a lot. In this case, the filters in the neighboring pixels could be very different, violating the implicit assumption that the filters change smoothly. Another limitation is that using sparse initial samples could lead to inaccurate double wedge estimation. Our method shares this common limitation with previous work, e.g., AAF and FSF. We use a comparable number (16) of initial samples for estimating wedge slopes, and prefilter the slopes to alleviate this issue. Similar issues also arise with AAF and previous work. However, the overall quality of our results is close to ground truth visually, and our renderings only required from 1.1 to 2.3s for these examples.

8. Conclusions and Future Work

This paper takes an important step towards efficient, near-interactive rendering of multiple distribution effects. While many approximate solutions have been proposed in the past, we believe that ours is one of the first practical methods based on physically-accurate GPU raytracing. We extend sparse sampling and reconstruction filtering in a significant way, developing a multiple axis-aligned filter approximation of the common wedge spectrum. This representation is more compact than sheared filtering, while preserving the ease-of-use of axis-aligned filtering, with relatively small overheads. No expensive storage or multi-stage precomputation is required, and multiple distribution effects can be treated together, unlike in fast sheared filtering. For analysis of the filter, we introduce new mathematical ideas of accuracy and coverage, which can be important baselines for future efforts. In future work, we would like to explore the limits of the MAAF idea, seeing if

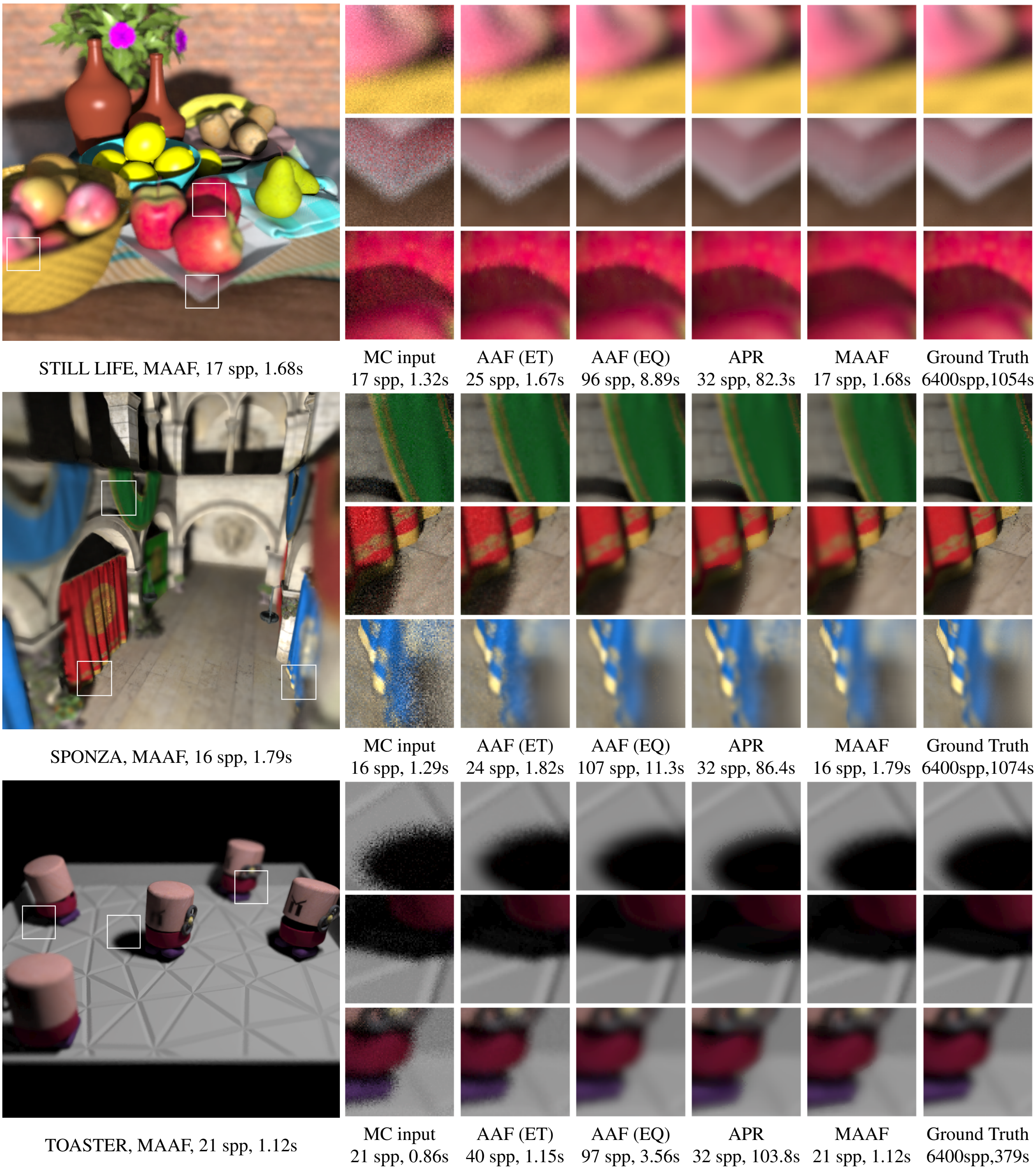


Figure 14: *Scenes with multiple distribution effects together: defocus, soft shadows, and indirect illumination. We compare MAAF results to those obtained by AAF equal time and equal quality, as well as the state of the art offline denoising method, adaptive polynomial rendering (APR). MAAF usually obtains a speedup of about* $6\times$ *with high quality results.*

one can approximate the double wedge exactly with a simple filter, or develop triangular filters that can be implemented in a fashion comparable to axis-aligned filters. Moreover, we seek to bring these ideas back to filter design in the signal-processing literature.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions. This work was supported in part by NSF grant 1451830 and the UC San Diego Center for Visual Computing.

References

[AM03] ASSARSSON U., MÖLLER T.: A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics (SIGGRAPH 03) 22*, 3 (2003), 511–520. 3

[BEJM15] BAUSZAT P., EISEMANN M., JOHN S., MAGNOR M.: Sample-based manifold filtering for interactive global illumination and depth of field. *Computer Graphics Forum 34*, 1 (2015), 265–276. 3

[BSS*13] BELCOUR L., SOLER C., SUBR K., HOLZSCHUCH N., DURAND F.: 5D covariance tracing for efficient defocus and motion blur. *ACM Transactions on Graphics 32*, 3 (2013). 3

[CCST00] CHAI J., CHAN S., SHUM H., TONG X.: Plenoptic Sampling. In *SIGGRAPH 00* (2000), pp. 307–318. 3

[CM14] CLARBERG P., MUNKBERG J.: Deep shading buffers on commodity GPUs. *ACM Transactions on Graphics (SIGGRAPH Asia 14) 33*, 6 (2014). 3

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

[DMBM14] DELBRACIO M., MUSE P., BUADES A., MOREL J.: Boosting monte caro rendering by ray histogram fusion. *ACM Transactions on Graphics 33*, 1 (2014). 3

[EDR11] EGAN K., DURAND F., RAMAMOORTHI R.: Practical filtering for efficient ray-traced directional occlusion. *ACM Transactions on Graphics (SIGGRAPH Asia 11) 30*, 6 (2011). 1

[EHDR11] EGAN K., HECHT F., DURAND F., RAMAMOORTHI R.: Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Transactions on Graphics 30*, 2 (2011). 1, 3, 4

[ETH*09] EGAN K., TSENG Y., HOLZSCHUCH N., DURAND F., RAMAMOORTHI R.: Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Transactions on Graphics 28*, 3 (2009). 1, 3

[GBP07] GUENNEBAUD G., BARTHE L., PAULIN M.: High-quality adaptive soft shadow mapping. *Computer Graphics Forum 26*, 3 (2007), 525–533. 3

[Guo98] GUO B.: Progressive radiance evaluation using directional coherence maps. In *SIGGRAPH 98* (1998), pp. 255–266. 3

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

[JHH*09] JOHNSON G., HUNT W., HUX A., MARK W., BURNS C., JUNKINS S.: Soft irregular shadow mapping: fast, high-quality, and robust soft shadows. In *I3D 2009* (2009), pp. 57–66. 3

[KBS15] KALANTARI N., BAKO S., SEN P.: A machine learning approach for filtering monte carlo noise. *ACM Transactions on Graphics (SIGGRAPH 15) 34*, 4 (2015). 3

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

[LH13] LEI K., HUGHES J.: Approximate depth of field effects using few samples per pixel. In *I3D 13* (2013), pp. 119–128. 3

[LWC12] LI T., WU Y., CHUANG Y.: SURE-based optimization for adaptive sampling and reconstruction. *ACM Transactions on Graphics (SIGGRAPH Asia 2012) 31*, 6 (2012). 3

[MGYM15] MOON B., GUITIAN J., YOON S., MITCHELL K.: Adaptive rendering with linear predictions. *ACM Transactions on Graphics (SIGGRAPH 15) 34*, 4 (2015). 3

[Mit91] MITCHELL D.: Spectrally Optimal Sampling for Distribution Ray Tracing. In *SIGGRAPH 91* (1991), pp. 157–164. 3

[MMMG16] MOON B., MCDONAGH S., MITCHELL K., GROSS M.: Adaptive polynomial rendering. *ACM Transactions on Graphics (SIGGRAPH 16) 35*, 4 (2016). 3, 8, 9

[MPFJ99] MCCORMACK J., PERRY R., FARKAS K. I., JOUPPI N. P.: Feline: Fast elliptical lines for anisotropic texture mapping. SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 243–250. 4

[MVH*14] MUNKBERG J., VAIDYANATHAN K., HASSELGREN J., CLARBERG P., MOLLER T.: Layered reconstruction for defocus and motion blur. *Computer Graphics Forum (EGSR 14) 33*, 4 (2014), 81–92. 3

[MWR12] MEHTA S., WANG B., RAMAMOORTHI R.: Axis-aligned filtering for interactive sampled soft shadows. *ACM Transactions on Graphics (SIGGRAPH Asia 12) 31*, 6 (2012). 1, 3

[MWRD13] MEHTA S., WANG B., RAMAMOORTHI R., DURAND F.: Axis-aligned filtering for interactive physically-based diffuse indirect lighting. *ACM Transactions on Graphics (SIGGRAPH 13) 32*, 4 (2013). 1, 4

[MYRD14] MEHTA S., YAO J., RAMAMOORTHI R., DURAND F.: Factored axis-aligned filtering for rendering multiple distribution effects. *ACM Transactions on Graphics (SIGGRAPH 14) 33*, 4 (2014). 2, 3, 8, 9, 10

[ODR09] OVERBECK R., DONNER C., RAMAMOORTHI R.: Adaptive Wavelet Rendering. *ACM Transactions on Graphics (SIGGRAPH ASIA 09) 28*, 5 (2009). 3

[PC81] POTMESIL M., CHAKRAVARTY I.: A lens and aperture camera model for synthetic image generation. In *SIGGRAPH 81* (1981), pp. 297–305. 3

[Pra07] PRATT W. K.: *Digital Image Processing: PIKS Scientific Inside*. John Wiley & Sons, Inc., New York, NY, USA, 2007. 6

[RDGK12] RITSCHEL T., DACHSBACHER C., GROSCH T., KAUTZ J.: The state of the art in interactive global illumination. *Computer Graphics Forum 31*, 1 (2012), 160–188. 3

[RKZ12] ROUSELLE F., KNAUS C., ZWICKER M.: Adaptive rendering with non-local means filtering. *ACM Transactions on Graphics (SIGGRAPH Asia 2012) 31*, 6 (2012). 3

[SBN15] SOLER C., BAGHER M. M., NOWROUZEZAHRAI D.: Efficient and accurate spherical kernel integrals using isotropic decomposition. *ACM Transactions on Graphics (TOG) 34*, 5 (2015), 161. 4

[SD12] SEN P., DARABI S.: On filtering the noise from the random parameters in monte carlo rendering. *ACM Transactions on Graphics 31*, 3 (2012). 3

[SSD*09] SOLER C., SUBR K., DURAND F., HOLZSCHUCH N., SILLION F.: Fourier depth of field. *ACM Transactions on Graphics 28*, 2 (2009). 3

[VMCS15] VAIDYANATHAN K., MUNKBERG J., CLARBERG P., SALVI M.: Layered Light Field Reconstruction for Defocus Blur. *ACM Transactions on Graphics 34*, 2 (2015). 3

[YMRD15] YAN L., MEHTA S., RAMAMOORTHI R., DURAND F.: Fast 4D sheared filtering for interactive rendering of distribution effects. *ACM Transactions on Graphics 35*, 1 (2015). 2, 3, 4

[YWY10] YU X., WANG R., YU J.: Real-time depth of field rendering via dynamic light field generation and filtering. *Computer Graphics Forum 29*, 7 (2010), 2099–2107. 3

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.: Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *Computer Graphics Forum (EUROGRAPHICS STAR 2015) 34*, 2 (2015), 667–681. 1, 3