Adaptive Wavelet Rendering

Ryan S. Overbeck* Columbia University
Craig Donner† Columbia University
Ravi Ramamoorthi‡ University of California, Berkeley

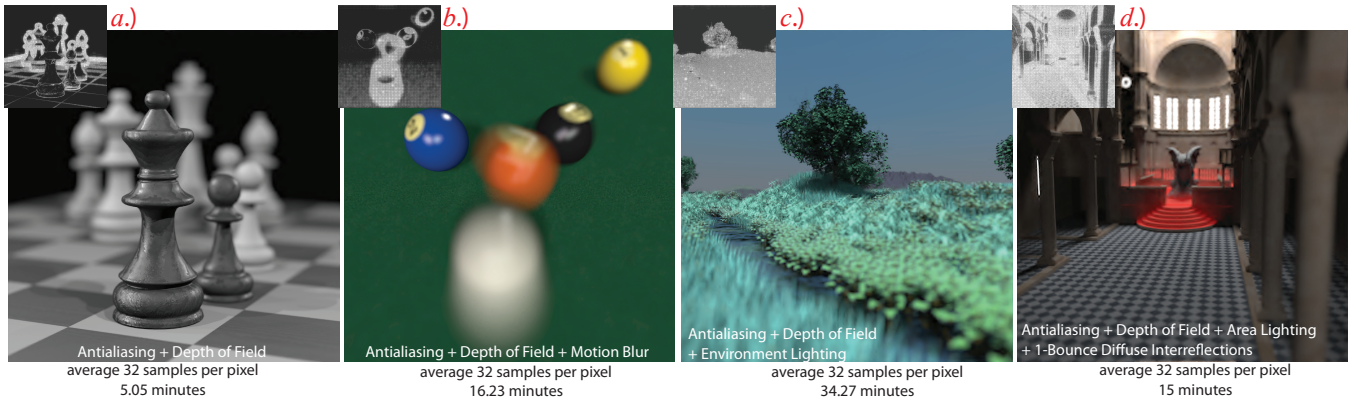

Figure 1: *All images were rendered at* 1024×1024 *on a single core of a 2.8GHz Core2 Extreme laptop. By adaptively sampling and reconstructing in a smooth wavelet basis, we get near-reference quality noise-free images with only 32 samples per pixel on average, with general high-dimensional combinations of rendering effects. The insets show the Monte Carlo sample distributions that generated the images.*

Abstract

Effects such as depth of field, area lighting, antialiasing and global illumination require evaluating a complex high-dimensional integral at each pixel of an image. We develop a new adaptive rendering algorithm that greatly reduces the number of samples needed for Monte Carlo integration. Our method renders directly into an image-space wavelet basis. First, we adaptively distribute Monte Carlo samples to reduce the variance of the wavelet basis' scale coefficients, while using the wavelet coefficients to find edges. Working in wavelets, rather than pixels, allows us to sample not only image-space edges but also other features that are smooth in the image plane but have high variance in other integral dimensions. In the second stage, we reconstruct the image from these samples by using a suitable wavelet approximation. We achieve this by subtracting an estimate of the error in each wavelet coefficient from its magnitude, effectively producing the smoothest image consistent with the rendering samples. Our algorithm renders scenes with significantly fewer samples than basic Monte Carlo or adaptive techniques. Moreover, the method introduces minimal overhead, and can be efficiently included in an optimized ray-tracing system.

1 Introduction

Rendering photorealistic images with effects such as depth of field, area lighting, motion blur and global illumination requires the evaluation of a complex high-dimensional integral at every pixel. Each effect adds one or more dimensions to the integral, and each dimension adds another potential source of variance. Monte Carlo integration is a robust approach for estimating this integral, but requires many samples to reduce variance to tolerable levels. Fortunately, natural images have smooth regions either in the image domain, over the other dimensions, or both. We should therefore adapt to this smoothness rather than performing an exhaustive sampling.

However, most adaptive algorithms sparsely sample *either* smoothly varying image regions, or slow variation in other dimensions, but not both. This leads to noise and artifacts at low sample counts, as seen in Figure 2. A recent adaptive multidimensional sampling method [Hachisuka et al. 2008] addresses these issues, but it scales poorly to general higher-dimensional integrals involving multiple effects. Even in low-dimensional situations, there can be computational and memory overheads in both its sampling and reconstruction stages that are particularly costly for recent optimized ray-tracers [Wald et al. 2001; Reshetov et al. 2005].

We propose *adaptive wavelet rendering* to directly estimate the image in the wavelet domain, and thus robustly handle all forms of variance. As opposed to pixels, wavelets present a multi-scale view of the image, and so provide a good representation for both image edges *and* smooth image features [Mallat 1999; Strang and Nguyen 1997]. This characteristic has made wavelets one of the most popular formats for image and video compression, and also for accelerating finite element methods such as radiosity [Gortler et al. 1993] and PRT [Ng et al. 2003]. Despite their proven benefits elsewhere, wavelets have rarely been used to speed up Monte Carlo sampling (with the notable exception of Bolin and Meyer [1998]), and we are inspired by recent work that shows their benefit for importance sampling [Clarberg et al. 2005].

Because the wavelet reconstruction of the image is hierarchical, coarse-scale wavelets are better at reconstructing large, smooth regions of the image, whereas finer-scale wavelets resolve small details, such as detailed texture and edges. We exploit this property in our algorithm to obtain an optimal hierarchical sample distribution.

As we describe in Section 4, our algorithm is composed of two simple stages: adaptive sampling and image reconstruction. In the first stage, we iteratively measure the variance of the wavelet basis'

*e-mail: roverbeck@gmail.com
†e-mail: craig.donner@gmail.com
‡e-mail:ravir@cs.berkeley.edu

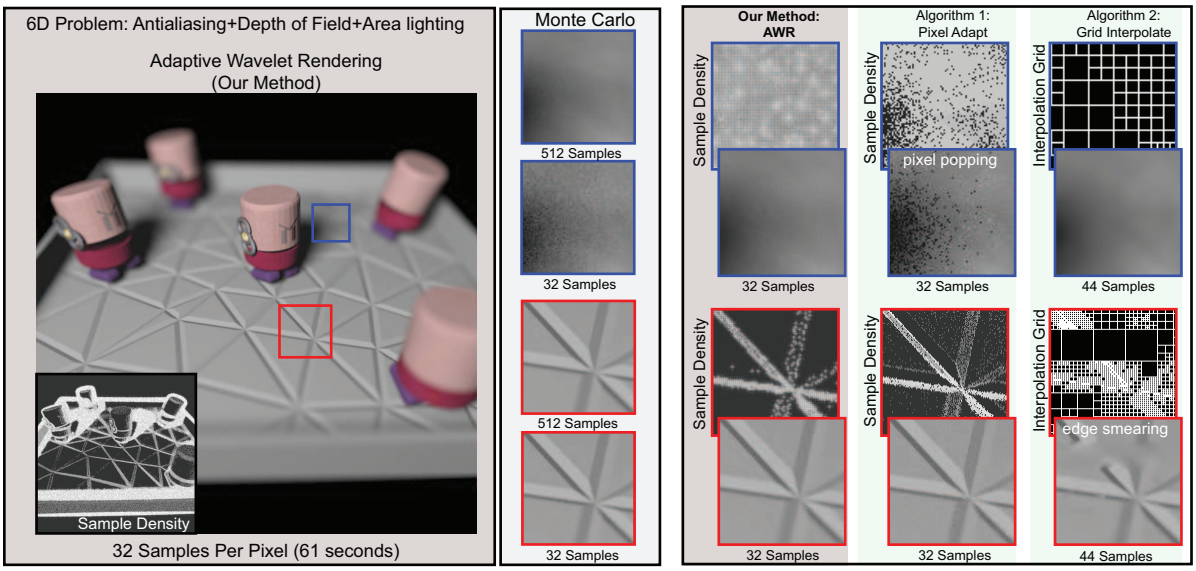

Figure 2: *Each pixel in this image is a 6 dimensional integral (2D image-space for antialiasing, 2D lens for depth of field, and 2D for an area light). Our method computes a reference-quality image using an average of only 32 samples per pixel. To the right are close-ups along with sample distributions generated using 4 different algorithms. The top row shows a smooth region of the image that has high variance from the other integral dimensions. The bottom row shows image-space edges that are smooth over the integral dimensions. Our method performs well in both regions. The pixel adaptive algorithm 1 has considerable noise in the smooth image areas on top, due to variance in the integral dimensions. The grid interpolation algorithm 2 has artifacts in the bottom row at edges, and it must exhaustively sample the integral dimensions. In the sample density images, note that adaptive wavelet rendering gives samples both to image-space edges and regions that are smooth in the image but have high variance in other integral dimensions. Samples also cluster at nodal points for wavelet interpolation.*

scale coefficients. Since high frequency details cause high amplitude wavelet coefficients, we use the wavelet magnitude to locate small-scale features, such as edges. The algorithm further samples those coarse-level scale coefficients that have high variance but do not have high wavelet magnitudes, i.e., do not have strong edges. These are image regions with high variance from the other dimensions. Finer scale coefficients receive samples to resolve the remaining high-frequency image features. New samples are drawn from the coefficients' scale function via importance sampling to reconstruct a smooth image in the reconstruction stage. Hence, our algorithm naturally adapts to both image-space edges as well as smooth regions with variance from the other integral dimensions.

In the second stage (image reconstruction), we use the wavelet basis to smooth away any remaining noise. We consider *all* of the wavelets (as opposed to standard truncation of small values), and simply subtract the measured variance from the wavelet coefficient magnitudes. This is conceptually similar to choosing the smoothest image that fits the measured statistics. In smooth regions, this allows the adaptive sampler to send more samples to the coarser scale functions, effectively sampling the image at a lower resolution. Thus, the adaptive sampling stage cooperates with the image reconstruction to efficiently compute a relatively noise-free image even with minimal sample budgets.

Adaptive wavelet rendering has the following key features:

Low Sample Counts: As seen in Figure 1, our results are relatively free of noise with an average of only 32 samples per pixel. In fact, because of the variance-reducing image reconstruction stage, visually consistent results are usually obtained even for 16 samples per pixel (see Figure 9).

Efficiency: Our algorithm has low computational and memory overheads. Moreover, it is conceptually simple and easy to implement. We have implemented it within a SIMD optimized packet ray tracer, and have found it to perform significantly faster than standard Monte Carlo path tracing [Kajiya 1986] and multidimensional adaptive sampling [Hachisuka et al. 2008]. All of the images in Figure 1 were rendered in a matter of minutes, and Figure 2 in only 61 seconds, both using a single core on a 2.8GHz Core2 Extreme processor. Images such as these often take several hours to generate using traditional methods.

Generality: The wavelet representation is only over the 2D image domain, and our algorithm directly considers only image-space values and variance. Thus, the method handles general combinations of effects, and does not suffer from the curse of dimensionality. Note that the scene in Figure 1*d* includes antialiasing, depth of field, area lighting, and diffuse global illumination for an 8D integral. Our method is most powerful when used to simulate effects that produce a smooth result, such as depth of field, which are particularly difficult for Monte Carlo algorithms.

2 Previous Work

2.1 Parametric Integration and Curse of Dimension

For solving a single integral, Bahvalov's theorems (see Haber [1969] which references Bahvalov [1959]) state that the best-case performance benefit of any numerical integration algorithm over standard Monte Carlo decreases exponentially with the number of dimensions. Fortunately, rendering is an instance of "parametric integration" with many correlated integrals. Based on this insight, Keller [2001] builds on the work of Heinrich and Sindambiwe [1999] to develop a multi-level Monte Carlo algorithm, with interpolation used to solve multiple integrals at once and make up for the curse of dimensionality. However, they note artifacts, such as smearing across discontinuities. We also choose to focus on image-space interpolation, rather than chase diminishing returns in the integral dimensions. Our work differs in that we include the image-space dimensions in both the parametric interpolation problem and the integral, and we use wavelets to distinguish image-space discontinuities from smooth variation.

Multidimensional Adaptive Sampling: The recent multidimensional adaptive algorithm in Hachisuka et al. [2008] also takes advantage of smoothness in the parametric image-space dimensions to produces high-quality images with very low sample densities. However, it is affected by the curse of dimensionality in two places. The adaptive sampling portion of the algorithm provides diminishing returns as the dimensionality increases (as predicted by Bahvalov's theorems) and the computational cost of the signal reconstruction stage is exponential in the dimensionality. We also find that this method introduces blocky artifacts for higher dimensional problems (see Figure 7o and t). As such, this solution is effective mostly for low dimensional problems with expensive shading costs ($d <= 4$).

2.2 Basic Adaptive Techniques

Most traditional adaptive sampling approaches fall into one of two categories. Some rely on a purely local measure of variance, usually within a single pixel, to adaptively determine the number of samples for the Monte Carlo integral [Whitted 1980; Mitchell 1987]. This works well for edges, but tends to provide uneven samples over smooth regions and so either generates artifacts or requires many more samples to reproduce a smooth result. Alternatively, algorithms in the second category exhaustively sample the integral at specific points, often the vertices of a grid. They then attempt to interpolate between these nodal points to reconstruct smooth features, while locating high-frequency image-plane regions to focus more samples upon. This approach can smear discontinuities or fail to locate small features. More recent advances [Guo 1998; Bala et al. 2003] better locate edges and other key image features but still must oversample the nodal points, and so do not take advantage of regions of low variance.

Figure 2 shows examples of these algorithms. Algorithm 1 is a simple example of the first category, while Algorithm 2 is a simple example of the second. The sample distributions depicted to the right are characteristic of such algorithms, and so are the corresponding artifacts. Our algorithm's distribution exhibits the best qualities of the two strategies. Similar to the first category, it spreads samples across the image, which is best for finding edges. More like the second category, the samples cluster at nodal points which are used to interpolate smooth results.

Veach and Guibas [1997] apply a variant of Metropolis Monte Carlo to simulating light transport. This algorithm is intended for rare event simulation to bring out highly focused local effects such as caustics or indirect light leaking through a small opening. It may be best to use our adaptive sampler for a baseline sampling, then Metropolis to capture the rare events, and lastly our wavelet reconstruction to remove the noise.

Perceptually Based Adaptive Sampling: Of particular note is the work of Bolin and Meyer [1998] who develop a sophisticated visual error metric and use it for adaptive sampling. Their work emphasizes that adapting to variance in a multiscale wavelet hierarchy corresponds more closely to the human visual system. However, their sampling algorithm still resorts to distributing samples to the leaves of the wavelet hierarchy and so is in a similar category as Algorithm 1 above. In smooth regions, our wavelet reconstruction removes the error at the finer wavelet levels, so we can send more samples to the coarse level scale coefficients, effectively sampling at a lower resolution while more accurately capturing smooth effects. Moreover, their sampling algorithm does not work for wavelets with overlapping support, and so can only use the Haar wavelet basis. The sampling stage of our algorithm is an efficient framework for working with arbitrary discrete wavelet bases, such as the smoother Daubechies 9/7 and LeGall 5/3. Note that Bolin and Meyer's advanced visual error metric can be used in place of the simple contrast metric in Section 4.1.

Multidimensional Lightcuts: A recent method is Multidimensional Lightcuts (MDLC) [Walter et al. 2006]. Their method is not an adaptive sampling approach in the same sense as our work and so is not directly comparable. For input, MDLC takes a constant number of primary shade points (gather points) and a constant number of light points. MDLC is an elegant approach for reducing the number of gather-point vs. light-point pairings. However, it does not adapt the number of gather-points or light-points, and so must start with an oversampling of both to guarantee high-quality convergence. Also, this method only considers gather-points within a pixel, and does not share information between neighboring pixels. As such, it may be best to combine their approach with ours. MDLC may help in situations where there are many spatially coherent light sources, while our algorithm can be used to adaptively introduce gather-points and interpolate the regions of smooth variation.

2.3 Adaptive Noise Removal

There has been significant work in adaptive post-production noise reduction filters. Wavelets are commonly used for noise reduction, using either hard thresholding or soft thresholding on the fine-scale wavelets [Strang and Nguyen 1997]. Hard thresholding simply clamps wavelets to a low value. Soft thresholding subtracts a constant value from the wavelet magnitudes. Our wavelet reconstruction improves on soft thresholding by subtracting a measure of the wavelet variance from the wavelet magnitude.

Besides wavelets, other bases may be used. The work of Meyer and Anderson [2006], for example, removes noise in animated sequences by projecting the image sequence onto a compressed PCA basis.

Two other directions of research derive from anisotropic diffusion introduced by Perona and Malik [1990] and bilateral filtering from Tomasi and Manduchi [1998]. Anisotropic diffusion is an iterative approach, and as such may be subject to instabilities. Moreover, it is often slower than either bilateral filtering or our reconstruction stage. Solutions based on bilateral filtering often suffer from objectionable ringing around image edges.

All of these methods focus on image reconstruction alone. Despite significant research dedicated to preserving image features [McCool 1999; Xu and Pattanaik 2005; Rushmeier and Ward 1994], it remains difficult for standalone post-processing noise removal algorithms to distinguish features from noise. This is because there is often simply not enough statistical information at the pixels to construct an accurate result.

Our work demonstrates the benefit of connecting adaptive sampling to reconstruction, and tailoring the adaptive sampling algorithm to the specific attributes of the reconstruction stage. By doing so, we are able to sample large smooth regions at an effectively lower resolution (see closeups in the top row of the right side of Figure 2) while also sending more focused samples to edges and other discontinuous image features (see bottom row of the right side of Figure 2).

2.4 Interactive Ray-Tracing

The works of Wald et al. [2001] and Reshetov et al. [2005] exploit coherence between ray samples to amortize ray-tracing and shading costs across packets of 16–256 rays and achieve interactive rates on simple scenes. Although the rays for multidimensional effects are incoherent (see Boulos et al. [2007] and Overbeck et al. [2008]), brute force rendering with an optimized ray-tracer is in some cases as effective as an expensive adaptive technique (see right two columns of Figure 7). In light of such benefits, we believe a valuable aspect of our algorithm is that it allows for use with a packet ray-tracer, since we adapt to image regions rather than only individual pixels.

In addition, our method has low overhead even when used with a highly optimized ray-tracer.

2.5 Frequency Analysis

Recent work has also studied light transport in the frequency domain [Durand et al. 2005; Egan et al. 2009; Soler et al. 2009], leading to simple image-space sampling heuristics based on local frequency content. Our work is in some ways the logical extreme of this approach, creating the image directly in the wavelet basis. Note however that general phenomena can be addressed, without needing to analyze or compute multidimensional space-angle Fourier spectra. Moreover, by using wavelets we can better localize both spatial (edges) and low-frequency (smooth) effects simultaneously.

3 Background: Wavelets and the DWT

Before describing the details of our algorithm, we provide a brief introduction to wavelets and the discrete wavelet transform (DWT).

A 1D wavelet basis is defined by the translates and dilates of a scale basis function ϕ, and a wavelet basis function ψ. Following the JPEG 2000 image compression standard [Skodras et al. 2001], we use Daubechies 9/7 wavelets [Cohen et al. 1992] (also referred to as Cohen-Daubechies-Feauveau wavelets) for all examples in this paper, and we also found LeGall 5/3 wavelets to be effective. Their precise forms are given in Appendix A, and profiles of their analysis scale functions are shown later in Figure 5.

A 1D wavelet basis is extended to 2D by taking the tensor products of the 1D basis functions. This creates 4 2D basis functions, 1 scale basis function and 3 wavelet basis functions:
$\Phi = \phi \otimes \phi^T$, $\Psi^0 = \phi \otimes \psi^T$, $\Psi^1 = \psi \otimes \phi^T$, and $\Psi^2 = \psi \otimes \psi^T$.

The entire 2D wavelet basis is defined as

$$\begin{aligned} \Phi_{k,ij}(x,y) &= \Phi(2^{-k}x - i, 2^{-k}y - j), \\ \Psi^{\alpha}_{k,ij}(x,y) &= \Psi^{\alpha}(2^{-k}x - i, 2^{-k}y - j), \end{aligned} \quad \text{with} \begin{cases} 0 \le \alpha < 3, \\ 1 \le k \le n, \\ 0 \le i < i_k, \\ 0 \le j < j_k. \end{cases}$$

In this expression, k indexes the "level" of the wavelet, with $k = n$ the coarsest or most dilated level, and $k = 1$ the finest level. We reserve $k = 0$ to refer to the original pixels. We use i and j for the translations, with $i_k = j_k = 2^{n-k}$ for square images of size 2^n.

The process of transforming a pixel image into a multi-scale wavelet basis is called *analysis*, and the inverse process back to pixels is called *synthesis*. The most general form of wavelet analysis computes the inner product between image $B(x, y)$ and wavelets:

$$S_{k,ij} = \langle B, \Phi_{k,ij} \rangle = \int\int B \cdot \Phi_{k,ij} dxdy, \tag{1}$$

$$W^{\alpha}_{k,ij} = \left\langle B, \Psi^{\alpha}_{k,ij} \right\rangle = \int\int B \cdot \Psi^{\alpha}_{k,ij} dxdy. \tag{2}$$

The $S_{k,ij}$ are referred to as *scale coefficients* and the $W^{\alpha}_{k,ij}$ are the *wavelet coefficients*.

In practice, we perform a discrete wavelet transform (DWT), for both analysis and synthesis, with cost linear in the number of pixels. For both analysis and synthesis, the DWT applies two filters, one low-pass and one high-pass, which together form a "filter bank". We use the so-called "non-standard" DWT, where we alternate between applying the 1D DWT on the image rows, and then on the columns. In our application we use the DWT for its efficiency, but for the rest of the paper, we will be using the inner product form in Equations 1 and 2 for notational convenience.

4 Wavelet Rendering Algorithm

Rendering a single pixel with anti-aliasing and high-dimensional effects requires the evaluation of an integral at every pixel:

$$B(x,y) = \int_y^{y+1} \int_x^{x+1} \int_{\mathbf{S}} F(u,v,\mathbf{s}) d\mathbf{s} du dv, \tag{3}$$

where $\mathbf{s}$ compactly denotes all the high-dimensional effects, such as depth of field (lens aperture), motion blur (time), and/or soft shadows (area light). The function F is evaluated by Monte Carlo sampling, and is treated as a black box by our method. The goal is to compute all $B(x, y)$ using as few samples, i.e., evaluations of F, as possible.

The adaptive sampling portion of our algorithm (Section 4.1) tightly cooperates with the reconstruction stage (Section 4.2) to produce a smooth result. Both stages use the wavelet basis to identify high variance regions of the integral. So rather than directly computing Equation 3, our algorithm keeps track of the scale and wavelet coefficients in the wavelet basis. To compute the scale coefficients, we estimate:

$$S_{k,ij} = \left\langle \int F d\mathbf{s}, \Phi_{k,ij} \right\rangle = \int\int\int F \cdot \Phi_{k,ij} d\mathbf{s} dx dy. \tag{4}$$

The adaptive sampler iteratively distributes samples to achieve a low variance estimate of Equation 4 for all scale coefficients. The reconstruction stage then subtracts the remaining estimated error from the wavelet coefficients to synthesize a smooth image.

4.1 Adaptive Wavelet Sampling

The key to the adaptive sampling process is determining which scale coefficient receives new samples at each iteration. Coarse-scale high variance image features (like the blur from an out-of-focus lens) will cause high variance at all levels in the wavelet hierarchy. However, if the final result is smooth, then the scale functions at the coarser levels may predict a more accurate result than the noisy wavelet functions at the finer levels. In these situations, we would like the adaptive sampler to compute an accurate result for the coarse scale coefficients, and rely on the reconstruction phase to remove the noisy wavelets. On the other hand, to handle more isolated fine-scale image features like edges, we prefer a more focused sample distribution.

Definitions: Before we detail our approach to meeting the above requirements, we must first define the variables we use to compute image variance and wavelet coefficients. We accumulate the Monte Carlo samples $F(u, v, \mathbf{s})$ at the image pixels, and maintain estimates of the pixel mean $\widetilde{B}$ and the variance of these samples, $\sigma^2(F)$. There are many methods to approximate the functional variance from a set of samples. We use the square of the contrast metric used by Mitchell [1987]:

$$\sigma^2(F) = (I_{\text{max}} - I_{\text{min}})^2 / (I_{\text{max}} + I_{\text{min}})^2, \tag{5}$$

where I_{max} and I_{min} are the maximum and minimum sample intensity respectively. The numerator provides an upper bound estimate of the pixel variance, and the denominator weights the adaptive sampler towards the darker image regions, where the human eye is more sensitive to error. Given these definitions, the estimator variance of the pixel mean is

$$\sigma^2(\widetilde{B}) = N^{-1}\sigma^2(F), \tag{6}$$

where $N(x, y)$ is the number of samples that land in pixel (x, y). We later show how to compute the variance of the scale coefficients using this information. The wavelet coefficients are computed simply by performing a DWT analysis on the pixel means:

$$\widetilde{W}^{\alpha}_{k,ij} = \left\langle \widetilde{B}, \Psi^{\alpha}_{k,ij} \right\rangle. \tag{7}$$

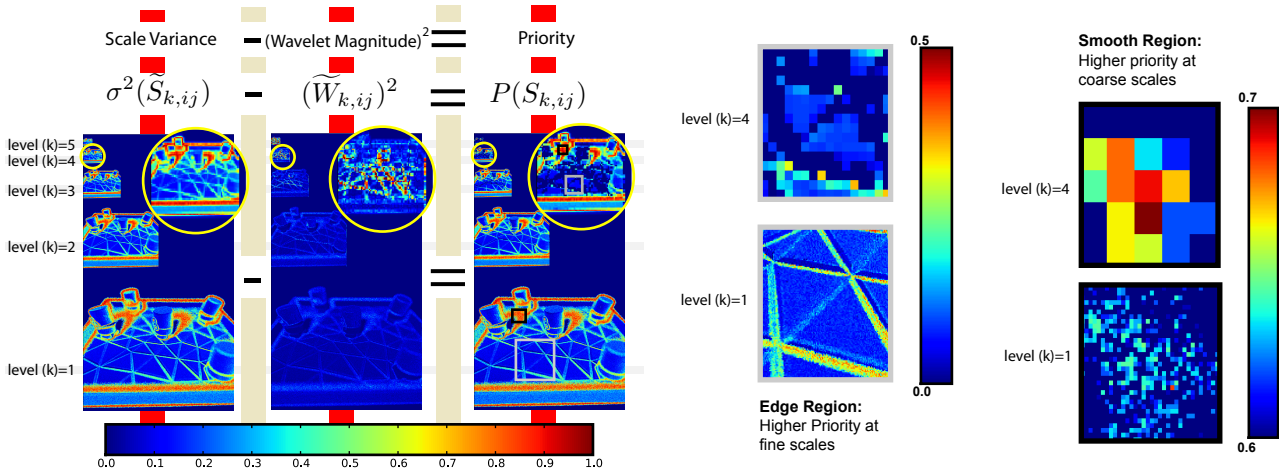

Figure 3: *The process of computing Equation 8, for the priority values of different scale coefficients. Smooth regions have higher priority at coarse scales. Edges have higher priority at finer scales.*

Step 1—Initialization: Coarsely sample the entire image, inserting scale coefficients at levels $0 \leq k \leq 5$ into a priority queue.

Step 2—Priority Computation: Update the priority of each scale coefficient in the queue.

Step 3—Sampling: Pop the next scale coefficient from the priority queue and importance sample it.

Step 4—Iterate: If samples remain, Goto 2, else finish.

Figure 4: *Steps of the adaptive sampling stage. Our algorithm starts with a fixed sample budget, and iteratively distributes samples to high variance scale coefficients until the samples are depleted.*

The Algorithm: Now, given a fixed budget of samples, the adaptive sampling stage proceeds according to the steps in Figure 4.

Step 1—Initialization: We generally start by coarsely sampling the image with 4 samples per pixel. We now insert scale coefficients from levels $0 \leq k \leq 5$ (including the pixels, which we may recall are the finest-level scale functions) into the priority queue.

Step 2—Priority Computation: The priority function or oracle $P(S_{k,ij})$ determines which coefficients require more samples. A number of oracles have been proposed in other contexts, for instance for refinement of links in wavelet radiosity [Gortler et al. 1993]. Our heuristic is designed to send more samples to coarse scale coefficients in smooth regions of high variance, and so it prioritizes coefficients that have a large functional variance over their support. However, when a high-frequency and non-oscillating image feature like an edge exists, it is better to allocate samples to finer scales, where the edge is resolved clearly.

Since the wavelet coefficients are (by definition) equal to the error due to image edges and other high-frequency features, simply subtracting the wavelet magnitudes from the scale variance results in the desired heuristic:

$$\boxed{P(S_{k,ij}) = \sigma^2(\widetilde{S}_{k,ij}) - (\widetilde{W}_{k,ij})^2} \tag{8}$$

We use the average of the 3 wavelets squared for the wavelet magnitude: $(\widetilde{W})^2 = \frac{1}{3}\sum_\alpha (\widetilde{W}^\alpha)^2$. To estimate $\sigma^2(\widetilde{S}_{k,ij})$, we perform a wavelet analysis on the per-pixel variance, using the square of the scale function:

$$\sigma^2(\widetilde{S}_{k,ij}) \approx \left\langle \sigma^2(\widetilde{B}), \Phi^2_{k,ij} \right\rangle . \tag{9}$$

We derive this equation in Appendix B. The inner product in Equation 9 is an unbiased estimator (as long as the pixel samples are uncorrelated). Since the DWT is more efficient to compute, we perform a DWT analysis on the per-pixel variances ($\sigma^2(\widetilde{B})$) using the squares of the filter bank coefficients. This biases our estimate of $\sigma^2(\widetilde{S}_{k,ij})$ when using filters with overlapping support, but it works well in practice. Note that this bias only affects the priority values in Equation 8 and not the final result. For efficiency, we only use the low-pass portion of the filter bank, since we only require the resulting scale coefficients.

Discussion: The heuristic in Equation 8 distinguishes between two types of image-space variance: 1) smooth regions of high variance (like the blur from an out-of-focus lens or the penumbra of a soft shadow), and 2) edges and other nonsmooth image features. For the smooth regions, we should send more samples to the coarse scale functions, and our wavelet reconstruction will interpolate the result across the smooth region. For edge regions, we need more focused samples to resolve the feature, so samples should go to the finer scale functions. In the far left image in Figure 3, observe that for smooth regions (like the penumbra from one of the toasters' shadows), the scale variance increases from fine scale to coarse scale. Thus, the scale variance correctly prioritizes these regions. For edge regions (like the edges on the floor), the variance is about the same across levels, and so will not necessarily target the finer scale functions. However, observe that the values of the squared wavelet coefficient magnitudes (the second image from the left in Figure 3) tend to grow from fine to coarse for edge regions, and stay the same across levels for smooth regions. Thus by simply subtracting the squared wavelet magnitudes from the scale variance, as in Equation 8, we achieve a heuristic which correctly handles both regions.

The images on the right of Figure 3 are close-ups of an edge region and a smooth region from the priority image. We have rescaled the color maps to highlight the different priority values between levels. Note that the priorities for the smooth region are higher at the coarser level 4 than level 1. Alternatively, the priorities for the edges are higher at the finer level 1 than level 4.

Implementation Details: The estimator in Equation 9 can be tuned to target samples toward coarser or finer levels by renormalizing the $\Phi^2_{k,ij}$ filters to sum to a value > 1. We found empirically that renormalizing to 1.05 works well in most situations and tends to target levels $0 \leq k \leq 4$.

Computing Equation 8 requires 2 DWT analyses: one standard analysis for the wavelet coefficients $\widetilde{W}^\alpha_{k,ij}$, and one with the squared

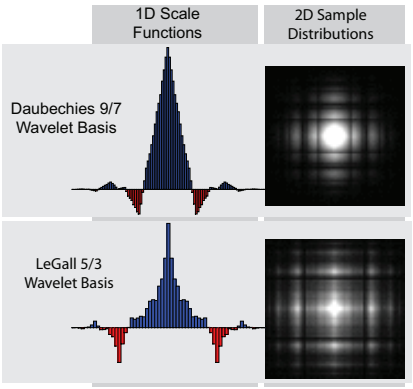

Figure 5: *The tables used to compute the CDFs for Step 3 and the resulting sample distributions for LeGall 5/3 and Daubechies 9/7 scale functions. On the left, blue values are positive and red are negative. These are scale functions from level $k = 4$.*

low-pass filter for $\sigma^2(\widetilde{S}_{k,ij})$. To make these analyses efficient, we perform them locally over an affected image area as new samples are added. This requires that we keep track of some amount of information that is usually discarded during a wavelet transform. Specifically, we must keep all scale coefficients, and the intermediate wavelet coefficients that result from applying the non-standard 2D DWT in one dimension before applying it in the other dimension. Despite this extra memory requirement, the performance benefit from updating only the affected coefficients greatly outweighs this minor expense.

Step 3—Sampling: In Step 3, the highest priority scale coefficient is taken from the priority queue. These scale coefficients cover regions rather than individual pixels, so we allocate multiple samples at a time to amortize the overheads introduced in Step 2. For the examples in this paper, we allocate 64×2^k samples at each iteration, but tuning this parameter offers a trade between speed and quality of adaptation. This also allows us to amortize the costs of ray casting and shading by using SIMD accelerated ray packets [Wald et al. 2001]. We use the partition traversal algorithm in [Overbeck et al. 2008] throughout our system to operate on large groups of (possibly incoherent) rays and shade samples.

The sample locations in the image plane are determined by importance sampling the shape of the scaling function, while samples in other dimensions are chosen via random sampling. The right images in Figure 5 show the sample distributions for LeGall 5/3 and Daubechies 9/7 wavelets. The importance sampling requires precomputing a cumulative distribution function (CDF) once for each scale basis function. Note that we only need the CDF for the 1D basis function ϕ, since we importance sample each dimension independently. The plots on the left of Figure 5 show the 1D basis functions that we use to compute the CDF for each dimension. After the coefficient is sampled, it is re-inserted in the priority queue, so that it can receive more samples in later iterations if needed.

Step 4—Iterate: If there are more samples in the budget, the algorithm returns to Step 2. There, the priorities $P(S_{k,ij})$ will be updated (in general, the priority for the scale coefficient just chosen will decrease since its variance is reduced—other scale coefficients that overlap its support will also be affected). Then, the algorithm moves to Step 3, sampling the new highest-priority scale function.

In its current design, our algorithm operates with a fixed sample budget, and is finished when this budget is depleted. It would also be possible to use a quality metric as a stopping criteria, such as stopping when there is no measured variance greater than some epsilon. However, since any measure of variance is only approximate, it may be better for the user to visually inspect the results and incrementally add more samples until visual convergence is achieved.

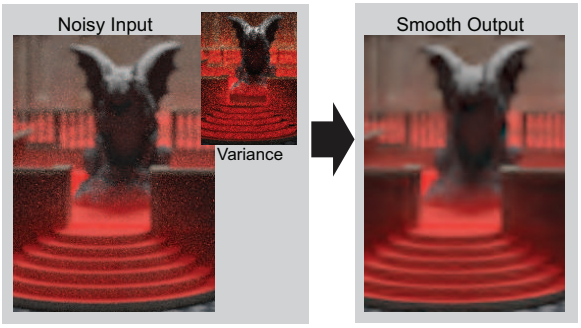

Figure 6: *Wavelet image reconstruction takes a noisy image and a variance image ($\sigma^2(\widetilde{B})$ in Equation 10) as input, and produces a smooth image as output. Here we show close-ups of the out-of-focus gargoyle in Figure 1.*

4.2 Adaptive Wavelet Reconstruction

At the conclusion of the sampling phase, we have reduced the variance for the coarse scale coefficients in smooth regions of high variance, as well as the fine scale coefficients near edges. The variance that remains is noise, and should be removed by our wavelet reconstruction.

Noise from undersampled Monte Carlo integration appears as fine-grained jump discontinuities in the image, and so should be captured by the fine-scale wavelet coefficients. Therefore, to reduce noise, we can simply reduce the magnitude of the fine-scale wavelet coefficients. One simple approach would be to just ignore coefficients with low magnitudes, as in standard image compression, or threshold the wavelet magnitudes [Strang and Nguyen 1997]. The sampling stage, however, provides more information on the expected reconstruction error in the form of the variance in each coefficient:

$$\Delta^{\alpha}_{k,ij} = \sqrt{\left\langle \sigma^2(\widetilde{B}), \left(\Psi^{\alpha}_{k,ij}\right)^2 \right\rangle}. \qquad (10)$$

The square root above is necessary to convert a variance measure to a standard deviation error.

The inner product $\left\langle \sigma^2(\widetilde{B}), \left(\Psi^{\alpha}_{k,ij}\right)^2 \right\rangle$ is computed similarly to $\left\langle \sigma^2(\widetilde{B}), \Phi^2_{k,ij} \right\rangle$ in Step 2 (Equation 9) of the adaptive sampling stage, with two exceptions. First, we perform a full DWT using the squares of both high-pass and low-pass filters. Second, the functional variance is computed as the squared difference of the maximum and minimum sample intensities: $\sigma^2(F) = (I_{\max} - I_{\min})^2$. This is used instead of the squared Mitchell contrast in Equation 5. While the sampler should be weighted towards darker regions to reduce error there, the reconstruction should smooth both dark and bright regions equally.

The standard deviation in Equation 10 provides a range of statistically valid values for the wavelet coefficients. Whereas standard Monte Carlo integration uses the middle of this range, or the average of the samples, we instead take the value of smallest magnitude. This is equivalent to choosing the smoothest image which fits the chosen rendering samples.

Subtracting the standard deviation from the magnitude of the wavelet coefficients gives this result:

$$W^{\alpha}_{k,ij} = \text{sign}(\widetilde{W}^{\alpha}_{k,ij}) \cdot \max\left(0, |\widetilde{W}^{\alpha}_{k,ij}| - c_s \cdot \Delta^{\alpha}_{k,ij}\right), \qquad (11)$$

where $\widetilde{W}$ are the wavelet coefficients from the pixel means, and c_s (the smoothing constant) is a user-supplied constant that provides a trade-off between noise and wavelet artifacts. Larger values of c_s make smoother images but may introduce ringing around edges or produce a blocky reconstruction. The specific form of the wavelet artifacts depends on the particular wavelet basis used. To avoid coarse scale artifacts, we damp out the smoothing by renormalizing the $\left(\Psi^{\alpha}_{k,ij}\right)^2$ filters to sum to a value < 1. This is analogous to how wavelet compression methods allocate more bits to coarser scale coefficients. For all of the examples in this paper, we renormalize to $2^{-1/2}$, and we set $c_s = 1$, but it may be possible to tune these values for smoother or sharper results.

As shown in Figure 6, our wavelet reconstruction simply requires the noisy results from the adaptive sampling stage and a per-pixel and per-color-channel variance image as input, and successfully removes almost all noise. Note that the entire image reconstruction phase requires at most 2 DWTs (for Equation 10 and synthesis from Equation 11), and is therefore very efficient.

5 Results

All results in this paper were generated on a laptop with a 2.8 GHz Core2 Extreme processor and 3 GB of RAM using one thread, and all images were rendered at 1024×1024.

The five scenes we use are intended to represent a broad range of applications. The "toasters" scene in Figures 2 and 7 is a simple scene with only 11k triangles, a single rectangular area light, and Phong shading to reduce ray-tracing and shading costs and highlight our system's low overhead. The "chess" scene in the top row of Figure 7 has 50k triangles and 9 point lights. The black queen and pawn in the foreground use more complex shaders with bump-mapping, gloss-mapping, mip-mapping, and PBRT's "substrate" material. The "pool" scene in the second row of Figure 7 has 57k triangles, 9 point lights, complex shaders, and includes time-varying motion blur effects. The "chess" scene and the "pool" scene (without depth of field) were originally used for the multidimensional adaptive sampling paper [Hachisuka et al. 2008] using PBRT [Pharr and Humphreys 2004]. The "plants" scene in the third row of Figure 7 is from the PBRT distribution and has very high geometric complexity. With over 12k instanced plants and trees, it effectively has over 19 million triangles, many of which are smaller than a pixel. Our packet ray tracer is slower on this scene due to the incoherence of the rays relative to the complex geometry. Finally, we added a gargoyle statue to the "sibenik" scene in order to create Figures $1d$ and 6. This scene has a total of 251k triangles with 1 point light and 1 area light. It uses purely Lambertian materials to demonstrate diffuse interreflections. We added a small red ambient component to the red rug to exaggerate the red color-bleed onto the walls and the gargoyle. We did not include this scene in Figure 7 because MDAS is not built to run on this higher dimensional scene. While our paper is focused on still image rendering, we also include a video of the "chess" scene with animated camera view in the supplementary material which shows the temporal coherence of our method.

5.1 Comparisons to Monte Carlo, LDS, Mitchell, MDAS

We have already compared to basic adaptive sampling algorithms in Figure 2. In Figure 7, we compare our adaptive algorithm to Monte Carlo, low discrepancy sampling (LDS), Mitchell's adaptive sampler [Mitchell 1987], and multidimensional adaptive sampling (MDAS) [Hachisuka et al. 2008]. The implementations we have for these algorithms are in the PBRT system which focuses more on generality than ray casting speed, while we use a speed optimized packet ray tracer for our system and Monte Carlo. Therefore the ray casting times for these systems should not be directly compared to ours. The scenes increase in dimensionality from the top row to the bottom. The chess scene in the top row antialiases the image dimensions and simulates camera depth of field for a 4-dimensional rendering problem. Refer to [Hachisuka et al. 2008] for comparisons to the Mitchell adaptive sampling algorithm using this scene. The middle row is a 5D problem with antialiasing, depth of field, and motion blur. The scenes in the bottom two rows are 6D with both scenes using antialiasing and depth of field. The "toasters" scene uses a rectangular area light, and the "plants" scene uses environment map lighting.

Out Method: With an average of only 32 samples per pixel (Figure $7a$,c,f,h,k,m,p, and r), our method faithfully reproduces the out-of-focus areas that require considerable integration over the camera aperture (and over the time dimension for Figure $7f$ and h, and over the light source for Figure $7k$,m,p, and r). In these smooth but high variance areas, our adaptive sampling delivers more samples to the coarser scale coefficients. With a low variance estimate at the coarse scale coefficients, the wavelet reconstruction synthesizes a noise-free result.

Monte Carlo: Our basic Monte Carlo renderer, without adaptivity, requires at least 512 samples in Figures $7d$, i, n, and s to achieve comparable results. We use 56 samples in Figure $7b$ to provide a comparable time comparison, and 32 samples in Figure $7q$ to compare at the same sample count. At these low sample densities, Monte Carlo introduces significant noise. For the "chess" scene, our algorithm's overhead is low enough that it is as fast with 32 samples per pixel as Monte Carlo using 56 samples, even though we use SIMD optimized packet ray-tracers.

Low Discrepancy Sampling: LDS generates noisy results in Figure $7l$ with 64 samples per pixel. As predicted by Bahvalov's theorems, LDS's benefits over basic Monte Carlo are significantly reduced for this higher dimensional example. Note that we use LDS from the PBRT system which focuses more on generality and accuracy than speed. For the "toasters" scene, we expect a speed optimized version may achieve comparable time to our system at about 64 samples.

Mitchell: Mitchell's adaptive sampler [Mitchell 1987] is a popular image-space adaptive technique. It produces noisy results for the motion blurred region in Figure $7g$. Similar to Algorithm 1 in Figure 2, it adapts to only fine-scale per-pixel variance, and may therefore sample unevenly in regions that should be smooth. Also, it only uses two passes of adaptivity, and will not iteratively continue to provide samples to this high variance region.

Multidimensional Adaptive Sampling: MDAS is the state-of-the-art adaptive sampling method, and produces a high quality result with only 16 samples per pixel in Figure $7e$. It uses a sophisticated reconstruction algorithm, and therefore requires fewer samples than our method for this example. However, it introduces significant overhead and takes about $4.7\times$ longer than our system at 32 samples per pixel. 849 seconds are spent in MDAS's sampling stage, and 568 seconds in image reconstruction, for a total of 1414 seconds. This is only $2\times$ faster than our Monte Carlo renderer with 512 samples per pixel (2835s vs. 1414s). This disparity is partially due to the fact that we use a speed optimized ray-tracing and shading system, and MDAS uses PBRT, which is optimized for generality rather than speed. However, this only accounts for some of the time spent in MDAS's sampling stage, and our method's full rendering pipeline is almost $2\times$ faster than MDAS's reconstruction stage alone. Thus, our method requires significantly less overhead independent of the particular rendering architecture used.

Moreover, MDAS's results degrade as the problem dimensionality increases. Blocky artifacts begin to appear for the 5D image in

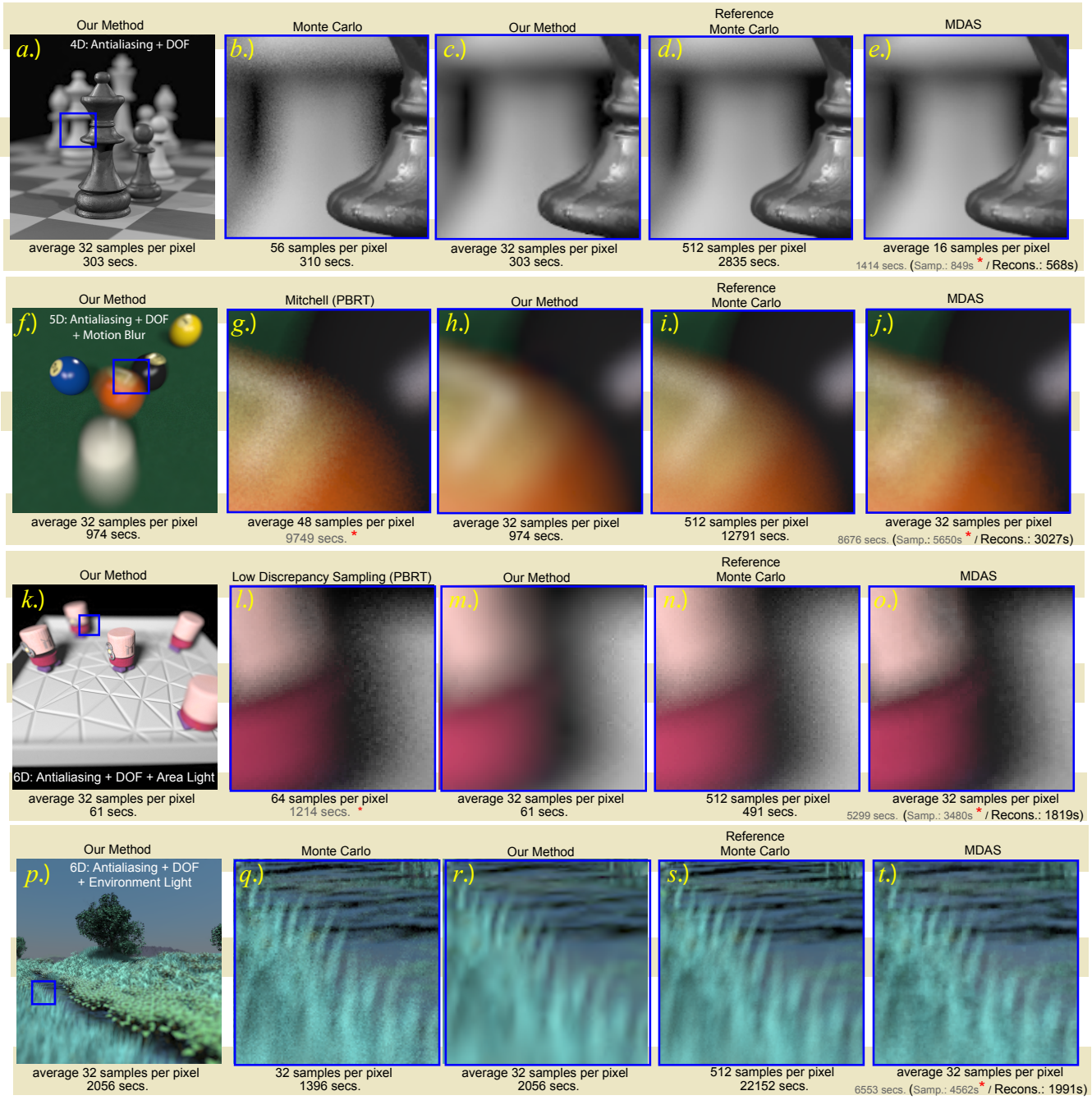

Figure 7: *All images were rendered at 1024x1024. The problem dimensionality increases from 4D in the top row to 6D in the bottom row. Top row is a 4D scene with antialiasing and depth of field. The middle row is a 5D scene with antialiasing, depth of field, and motion blur, and the bottom row is a 6D scene with antialiasing, depth of field, and area lighting. Our method consistently achieves near reference quality images with only 32 samples per pixel for all of these examples. MDAS's results degrade as the problem dimensionality increases, and other methods generate noisy results at low sample counts.*

Figure 7j, and are significantly more apparent in the 6D image in Figure 7o. It is difficult to use MDAS with more samples per pixel, because MDAS requires 400 bytes per sample (about 12GB for a 1024×1024 image at 32 samples per pixel). Our system does not need to store the samples, and so scales better to higher sample densities. We tiled the image to 8×8 in order to render Figure 7o with 32 samples per pixel. MDAS employs best-candidate sampling for better sample distributions and performs a per-sample k-nearest neighbors search for high quality reconstruction. These approaches work well for 4D scenes like Figure 7e, but they both run in time exponential in the number of dimensions while their benefit in accuracy decreases exponentially. This explains why MDAS takes $4\times$ longer to render the 6D scene in Figure 7o at only $2\times$ the number of samples than in Figure 7e.

5.2 Generality, Efficiency, and Visual Consistency

Generality: Due to our algorithm's insensitivity to the problem dimensionality, our algorithm is $5\times$ faster at rendering the 6D scene in Figure 7k than the 4D scene in Figure 7a because of the simpler

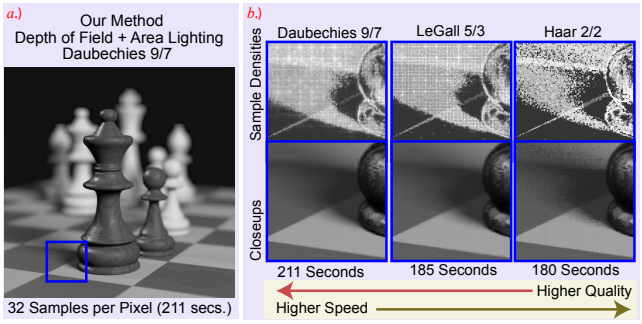

Figure 8: *A 6D render with depth of field and area lights rendered with an average of 32 samples per pixel. Different wavelet bases offer different speed vs. quality. Haar is the fastest but produces blocky artifacts. LeGall is also fast, and offers reasonable quality. The Daubechies 9/7 filter bank generates the smoothest images.*

geometry and shaders in the "toasters" scene. To further emphasize the point we removed all but one of the point lights in Figure 7a, added an area light, and rendered Figure 8a. With fewer lights overall, it takes less time for our system to render this image than Figure 7a (211s vs. 303s).

The "plants" scene in the bottom row of Figure 7 and in Figure 1c, and the "sibenik" scene in Figure 1d present a stress test for our system by introducing large amounts of variance throughout the scene. For the "plants" scene, variance is introduced by environment lighting and sub-pixel geometry. In Figure 7r, our algorithm captures the out-of-focus wisps of grass, and produces smooth reflections in the water which are noisy even for Monte Carlo with 512 samples per pixel in Figure 7s. The "sibenik" scene uses Monte Carlo path tracing for one-bounce diffuse interreflections. Note the red color-bleed onto and around the gargoyle statue in the middle as well as the region under the archway on the right that is lit only by indirect illumination.

Variance is high everywhere in the "plants" and "sibenik" scenes, so the adaptive sampler cannot identify local regions to adapt to. Nonetheless, our algorithm still seeks out the regions of smooth variance, and samples these at a lower resolution. Note the multi-scale grid patterns in the sample distributions in the insets of Figure 1c and d. So we still achieve high-quality results with an average of only 32 samples per pixel. Standard Monte Carlo path tracing using our optimized ray tracer requires at least 512 samples per pixel to generate a comparable quality image. This takes over 2 hours for the sibenik scene and over 7 hours for the "plants" scene. At 512 samples, Monte Carlo is still noticeably noisier, but our result with 32 samples has some perceptible wavelet artifacts.

Efficiency: Table 1 breaks down the overheads of our algorithm for rendering Figure 8 with the Daubechies 9/7 filter bank. Ray-tracing (including the initial sparse sampling and shading) takes 176.5s and our algorithm takes only 34.5s of the total 211s. Thus, our overhead is low, even though we use an optimized packet ray-tracer and shading system. All overhead costs can be adjusted by increasing the amortization (sending more rays for each coefficient in Step 3) or changing the wavelet basis to a simpler basis such as LeGall. The wavelet reconstruction takes less than 1 second, and is dependent on the size of the image alone. We also include all significant memory overheads required by our system. In today's scenes, with 100s of megabytes worth of texture data, 88MB is relatively minor. This size is only dependent on the size of the rendered image so will not change with increased sample densities, scene complexity, or problem dimensionality.

Comparing Wavelet Bases: The close-ups in Figure 8b compare results using 3 popular wavelet bases. The Haar 2/2, LeGall 5/3,

Algorithm Component	**Time (Seconds)**
Sampling: Step 1. Sparse RT and Shading	14.5s
Sampling: Step 2. Update Wavelets	12.2s
Sampling: Step 2. Update Variance	10.6s
Sampling: Step 2. Update Priorities	10.7s
Sampling: Step 3. Adaptive RT and Shading	162s
Reconstruction	$<$ 1s
Total	**211 s**

Memory Overheads	**Memory (MBytes)**
Intensity Images for Wavelet Updates (4 float)	16 MB
Intensity Images for Variance Updates (2 float)	8 MB
Image for per-pixel mean (1 RGB)	12 MB
Min/Max Images for per-pixel contrast (2 RGB)	24 MB
Image for per-pixel sample count (1 int)	4 MB
Images for priority queue (2 float, 4 int)	24 MB
Total	**88 MB**

Table 1: *A breakdown of the timings and memory overheads for generating Figure 8 with Daubechies 9/7 wavelets. The time is mostly dominated by ray-tracing and shading costs, and our memory overheads are low.*

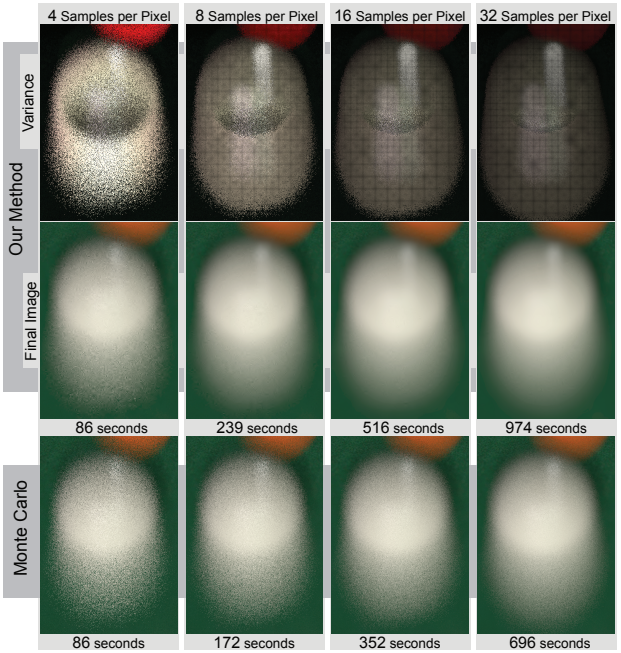

Figure 9: *Close-ups of a difficult region in Figure 7f. The cue ball is motion blurred and out-of-focus. At an average of 16 samples, our method produces a mostly smooth result with only minor wavelet artifacts. At 32 samples, our result is essentially converged. Monte Carlo results are noisy by comparison. In the top row, we include scaled up images of variance after the adaptive sampling stage to illustrate the coarse sampling strategy used in smooth regions.*

and Daubechies 9/7 filter banks offer different levels of speed and quality. Note in the sample count images in Figures 5 and 8b that the Daubechies filter bank has a more circular and smoother 2D projection as compared to LeGall and Haar. Thus Daubechies provides the smoothest results, but is also slower due to its wide filters. In general, smooth symmetric wavelet bases appear to work best.

Visual Consistency: Figure 9 contains close-ups of the cue ball in Figures 1b and 7f, and demonstrates the progression from initial sampling to convergence. After our initial sampling of the image at

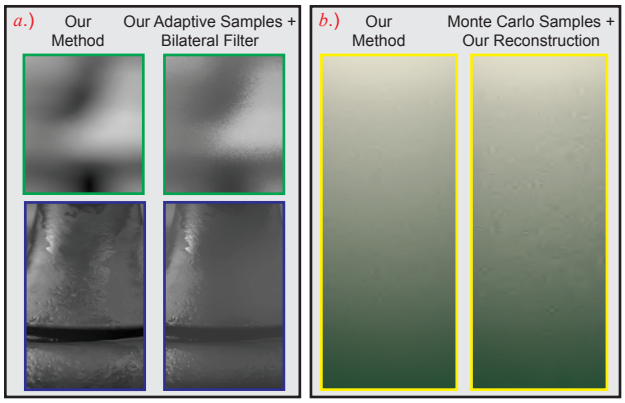

Figure 10: *Our wavelet sampler and reconstruction may be used independently, but are more powerful together. In a.), we use a bilateral filter on the output from our adaptive sampling stage on the "chess" scene. Our adaptive samples reduce the variance significantly, but the bilateral filter is unable to remove the noise in the top images, and loses important detail in the bottom images. In b.), we use our wavelet reconstruction on the output from pure Monte Carlo on the "pool" scene. The wavelet reconstruction removes almost all noise, but without our adaptive samples, it introduces some subtle artifacts.*

4 samples per pixel, we have a high variance estimate, as shown in the variance image in the top left. Even at this low sample count, our result looks quite reasonable, with only moderate noise and some wavelet artifacts. At an average of 8 samples per pixel, our algorithm has iteratively distributed samples, and the grid structure of a coarse wavelet level appears in the variance image. At this point, we have a relatively low variance estimate at this coarse level, and so after removing the variance at the finer levels, we achieve a mostly smooth result. At 16 samples per pixel, the grid of a finer wavelet level appears around the structure of the cue ball, and the highlight at the top, and most of the artifacts are gone. Finally, at an average of 32 samples per pixel, our result is essentially converged even though we only have an accurate estimate at the low-resolution wavelet grid. Our adaptive sampler leaves most of the pixels with high variance, which is subtracted by our wavelet image reconstruction. Monte Carlo, by comparison, is noisy even with 32 samples.

Sampling vs. Reconstruction It is also possible to use our adaptive sampler and our wavelet reconstruction independently. While each stage has many benefits, in its own right, they work best together. Figure 10*a* shows results of our adaptive sampler used with a bilateral filter, and Figure 10*b* uses standard Monte Carlo with our wavelet reconstruction. Our adaptive sampler significantly reduces the variance of the difficult out-of-focus regions in Figure 10*a*, and the bilateral filter can smooth some of the remaining noise. However, some noise remains because the filter cannot take advantage of the fact that our sampler provides more precise results around coarser wavelet nodal points. Also, this bilateral filter does not differentiate between noise from variance and oscillating texture detail which can resemble noise, so some surface detail is washed away.

Our variance-based wavelet reconstruction can be a powerful tool when used with non-adaptive Monte Carlo samples. It removes most of the noise in Figure 10*b*. However, the coarse wavelet coefficients are not as precise as if they were computed with our adaptive sampler, so some low-frequency noise remains. This low-frequency noise appears as wavelet artifacts in the smooth regions.

5.3 Discussion

Quality/Speed Settings: There are several key points of our algorithm that allow tuning of quality and speed parameters. However,

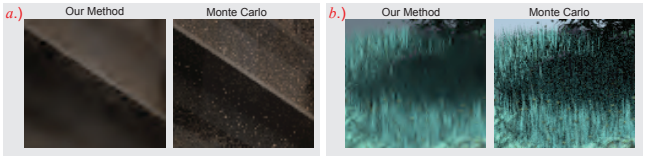

Figure 11: *Without enough samples (32 samples per pixel in this figure), difficult high variance regions may have artifacts. a.) an edge in the middle of a high variance region. Our result has some ringing at the edge. b.) sub-pixel geometry under environment lighting. Our method is unable to differentiate variance from visibility and variance from lighting, so our result becomes overly smoothed.*

we should emphasize that we didn't change any of these settings for the results in this paper. To reiterate, we use the Daubechies 9/7 filter bank, we set number of rays cast per adaptive iteration to 64×2^k, we set the reconstruction smoothing constant $c_s = 1$, and we renormalize the sampling variance filters to sum to 1.05 and the reconstruction variance filters to sum to $2^{-1/2}$. It may be possible to tune for better results in specific situations.

Limitations: Our method uses the wavelet basis to adapt to different sources of variance, including edges and smooth image features, and also to reconstruct a smooth image even in regions of high variance. When used for image compression, different wavelet bases can exhibit different forms of artifacts, including blockiness and ringing around edges. At low sample densities with too few samples to provide a precise result, we assume a smooth result, and our reconstruction algorithm effectively performs a wavelet compression on the results. For piece-wise smooth natural images, this is generally a reasonable assumption. However, when our assumption is incorrect, some wavelet artifacts may appear. Most of the results in this paper are essentially converged and so do not exhibit significant artifacts. Figure 11 show two examples of wavelet artifacts in the two most difficult scenes, the "plants" scene and the "sibenik" scene. Figure 11a is an edge in the middle of a smooth high variance region, and our wavelet reconstruction produces some ringing along this edge. The grass under the shadow of the tree in Figure 11b is somewhat overly smoothed. These cases are difficult for Monte Carlo approaches, and require more samples to model accurately. Nonetheless, these subtle artifacts affect only a small region of the image, are largely imperceptible when looking at the full picture, and the benefits with respect to Monte Carlo are shown clearly in Figure 7. Moreover, these artifacts are less distracting than the systemic noise introduced by an equally undersampled Monte Carlo simulation in Figure 11. It may also be possible to use depth and normal information in an approach similar to the coherence maps used in McCool [1999] to get better statistical data for these regions.

6 Conclusions and Future Work

We have presented adaptive wavelet rendering, a new method to adapt to both edges and smooth regions of high-dimensional variance. Our method is general, and renders complex effects like depth of field, area lighting, motion blur, and global illumination. The technique is simple to implement and efficient, with timings often an order of magnitude faster than previous adaptive algorithms, or optimized Monte Carlo ray-tracers.

Our algorithm currently makes no effort to sample optimally in any dimension other than the image plane. By doing so, we avoid the curse of dimension. However, even scenes with high-dimensional effects are often locally low-dimensional. For example, a stationary object doesn't need motion blur. While our algorithm handles this example well, and will not spend a lot of samples for this event, it

may be worthwhile to use an approach that can take better advantage of these locations of low-dimensionality.

In this paper, we focus on rendering still images, and reserve animated scenes for future work. Preliminary results demonstrate that our method produces smooth animations without temporal artifacts (see video in our supplementary material). Moreover, the time dimension offers extra opportunities for further adaptive sampling and reconstruction. One simple approach would be to expand to a 3D wavelet basis for faster rendering of animated sequences.

We believe our algorithm may be relevant to applications beyond rendering, given the very successful use of wavelets in other domains. It should be emphasized that the only part of our method that may be specific to graphics is the assumption that the image signal is locally smooth. However, many signals studied in other applied sciences have similar characteristics, and our algorithm should be viewed as a general method for high quality parametric integration.

Acknowledgements

Thanks to the anonymous reviewers for their insightful comments. This work was supported in part by the National Science Foundation (NSF grants 05-41259, 07-01775 and 09-24968), the Office of Naval Research (N00014-07-1-0900), a Sloan Research Fellowship, an Intel graduate fellowship, and generous gifts and equipment from NVIDIA, Adobe and Pixar. The toasters scene is provided by Andrew Kensler via the Utah 3D Animation Repository, chess scene by Wojciech Jarosz, Sibenik cathedral by Marko Dabrovic, gargoyle by Tudor Mint via AIM@SHAPE, and plants from PBRT [Pharr and Humphreys 2004].

References

BAHVALOV, N. S. 1959. On approximate calculation of multiple integrals. Tech. rep., Vestnik Moscow Univ.

BALA, K., WALTER, B., AND GREENBERG, D. P. 2003. Combining edges and points for interactive high-quality rendering. *ACM TOG (SIGGRAPH 03) 22*, 3, 631–640.

BOLIN, M. R., AND MEYER, G. W. 1998. A perceptually based adaptive sampling algorithm. In *ACM SIGGRAPH 98*, 299–310.

BOULOS, S., EDWARDS, D., LACEWELL, J. D., KNISS, J., KAUTZ, J., SHIRLEY, P., AND WALD, I. 2007. Packet-based Whitted and Distribution Ray Tracing. In *Proc. Graphics Interface*, 177–184.

CLARBERG, P., JAROSZ, W., AKENINE-MÖLLER, T., AND JENSEN, H. W. 2005. Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions. *ACM TOG (SIGGRAPH 05) 24*, 3, 1166–1175.

COHEN, A., DAUBECHIES, I., AND FEAUVEAU, J.-C. 1992. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics 45*, 5, 485–560.

DURAND, F., HOLZSCHUCH, N., SOLER, C., CHAN, E., AND SILLION, F. 2005. A frequency analysis of light transport. *ACM TOG (SIGGRAPH 05) 24*, 3, 1115–1126.

EGAN, K., TSENG, Y.-T., HOLZSCHUCH, N., DURAND, F., AND RAMAMOORTHI, R. 2009. Frequency analysis and sheared reconstruction for rendering motion blur. *ACM TOG (SIGGRAPH 09) 28*, 3, 93.

GORTLER, S. J., SCHRÖDER, P., COHEN, M. F., AND HANRAHAN, P. 1993. Wavelet radiosity. In *ACM SIGGRAPH 93*, 221–230.

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

HABER, S. 1969. Stochastic quadrature formulas. *Mathematics of Computation 23*, 108, 751–764.

HACHISUKA, T., JAROSZ, W., WEISTROFFER, R. P., DALE, K., HUMPHREYS, G., ZWICKER, M., AND JENSEN, H. W. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM TOG (SIGGRAPH 08) 27*, 3, 33.

HEINRICH, S., AND SINDAMBIWE, E. 1999. Monte Carlo complexity of parametric integration. *J. Complex. 15*, 3, 317–341.

KAJIYA, J. T. 1986. The rendering equation. In *ACM SIGGRAPH 86*, 143–150.

KELLER, A. 2001. Hierarchical Monte Carlo image synthesis. *Mathematics and Computers in Simulation 55*, 1–3, 79–92.

MALLAT, S. 1999. *A Wavelet Tour of Signal Processing, Second Edition (Wavelet Analysis & Its Applications)*. Academic Press.

MCCOOL, M. D. 1999. Anisotropic diffusion for Monte Carlo noise reduction. *ACM TOG 18*, 2, 171–194.

MEYER, M., AND ANDERSON, J. 2006. Statistical acceleration for animated global illumination. *ACM TOG (SIGGRAPH 06) 25*, 3, 1075–1080.

MITCHELL, D. P. 1987. Generating antialiased images at low sampling densities. In *ACM SIGGRAPH 87*, 65–72.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2003. All-frequency shadows using non-linear wavelet lighting approximation. *ACM TOG (SIGGRAPH 03) 22*, 3, 376–381.

OVERBECK, R., RAMAMOORTHI, R., AND MARK, W. R. 2008. Large ray packets for real-time Whitted ray tracing. In *IEEE/EG Symp. on Interactive Ray Tracing*, 41–48.

PERONA, P., AND MALIK, J. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell. 12*, 7, 629–639.

PHARR, M., AND HUMPHREYS, G. 2004. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc.

RESHETOV, A., SOUPIKOV, A., AND HURLEY, J. 2005. Multi-level ray tracing algorithm. *ACM TOG (SIGGRAPH 05) 24*, 3, 1176–1185.

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

SKODRAS, A., CHRISTOPOULOS, C., AND EBRAHIMI, T. 2001. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine 18*, 5, 36–58.

SOLER, C., SUBR, K., DURAND, F., HOLZSCHUCH, N., AND SILLION, F. 2009. Fourier depth of field. *ACM TOG 28*, 2, 18.

STRANG, G., AND NGUYEN, T. 1997. *Wavelets and Filter Banks*. Wellesley-Cambridge Press.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *ICCV 98*, 839.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *ACM SIGGRAPH 97*, 65–76.

WALD, I., SLUSALLEK, P., BENTHIN, C., AND WAGNER, M. 2001. Interactive rendering with coherent ray tracing. *Computer Graphics Forum (EUROGRAPHICS 01) 20*, 3, 153–164.

WALTER, B., ARBREE, A., BALA, K., AND GREENBERG, D. P. 2006. Multidimensional lightcuts. *ACM TOG (SIGGRAPH 06) 25*, 3, 1081–1088.

WHITTED, T. 1980. An improved illumination model for shaded display. *Commun. ACM 23*, 6, 343–349.

XU, R., AND PATTANAIK, S. N. 2005. A novel Monte Carlo noise reduction operator. *IEEE Comput. Graph. Appl. 25*, 2, 31–35.

A Daubechies 9/7 and LeGall 5/3 Filter Banks

We found the Daubechies 9/7 and LeGall 5/3 wavelets to work better for our algorithm than the simpler Haar wavelet basis or various other filter banks that we tested. They are both symmetric biorthogonal filter banks. The quality of a wavelet is often measured by the number of vanishing moments: the lowest degree polynomial for which the results of applying the high-pass filter are non-zero. The Daubechies 9/7 filter bank has 4 vanishing moments, so can exactly fit polynomials of cubic degree or lower. LeGall 5/3 has 2 vanishing moments. These are the highest theoretically possible for their support widths. The Daubechies 9/7 wavelet is used for high quality lossy encoding in JPEG 2000 because it is very smooth. LeGall 5/3 is an instance of a binlet, a filter bank that can be implemented as all integer values, so it is used for efficient lossless encoding in JPEG 2000.

Since they are both symmetric, we list only half of the coefficients. The rest can be obtained by reflecting about the first coefficient.

Daubechies 9/7
Analysis Low-Pass:
$(\sqrt{2})$ $\times$\{0.602949, 0.266864, -0.078223, -0.016864, 0.026749\}
Analysis High-Pass:
$(1/\sqrt{2})$ $\times$\{1.115087, -0.591272, -0.057544, 0.091272\}
Synthesis Low-Pass:
$(1/\sqrt{2})$ $\times$\{1.115087, 0.591272, -0.057544, -0.091272\}
Synthesis High-Pass:
$(\sqrt{2})$ $\times$\{0.602949, -0.266864, -0.078223, 0.016864, 0.026749\}

LeGall 5/3
Analysis Low-Pass: $(\sqrt{2})$ $\times(1/8)\times$\{6, 2, -1\}
Analysis High-Pass: $(1/\sqrt{2})$ $\times(1/2)\times$\{-2, 1\}
Synthesis Low-Pass: $(1/\sqrt{2})$ $\times(1/2)\times$\{2, 1\}
Synthesis High-Pass: $(\sqrt{2})$ $\times(1/8)\times$\{-6, 2, 1\}

B Derivation of Equation 9

Equation 9 accumulates the approximate variance at the scale coefficients from an estimate of variance at the pixels. Equation 4 states that the scale coefficients are equal to the inner product of the pixel values with the scale function. For a discrete wavelet basis, this inner product is a weighted sum:

$$S = \left\langle \widetilde{B}, \Phi \right\rangle = \sum_i \widetilde{B}_i \Phi_i, \tag{12}$$

where the Φ_i are the discrete scale function's filter coefficients, and the $\widetilde{B}_i$ are the pixel means. Note that we have dropped the wavelet translation and dilation subscripts from S and Φ for simplicity. We seek the variance of the scale coefficient: $\sigma^2(S) = \sigma^2\left(\sum_i \widetilde{B}_i \Phi_i\right)$.

This can be computed using the identity:

$$\sigma^2\left(\sum_i c_i x_i\right) = \sum_i c_i^2 \sigma^2(x_i) + \sum_i \sum_{j>i} 2 c_i c_j \mathrm{cov}(x_i, x_j), \tag{13}$$

which holds for any set of constants c_i and random variables x_i. If the x_i are uncorrelated random variables, then the covariance term tends to zero, and we have:

$$\sigma^2\left(\sum_i c_i x_i\right) \approx \sum_i c_i^2 \sigma^2(x_i). \tag{14}$$

Finally, this gives us:

$$\sigma^2\left(\sum_i \widetilde{B}_i \Phi_i\right) \approx \sum_i \Phi_i^2 \sigma^2\left(\widetilde{B}_i\right). \tag{15}$$

The right side of Equation 15 is equivalent to the right side of Equation 9.