Energy Preserving Non-Linear Filters

Holly E. Rushmeier[1]
National Institute of Standards and Technology

Gregory J. Ward [2]
Lawrence Berkeley Laboratory

ABSTRACT

Monte Carlo techniques for image synthesis are simple and powerful, but they are prone to noise from inadequate sampling. This paper describes a class of non-linear filters that remove sampling noise in synthetic images without removing salient features. This is achieved by spreading real input sample values into the output image via variable-width filter kernels, rather than gathering samples into each output pixel via a constant-width kernel. The technique is non-linear because kernel widths are based on sample magnitudes, and this local redistribution of values cannot generally be mapped to a linear function. Nevertheless, the technique preserves energy because the kernels are normalized, and all input samples have the same average influence on the output. To demonstrate its effectiveness, the new filtering method is applied to two rendering techniques. The first is a Monte Carlo path tracing technique with the conflicting goals of keeping pixel variance below a specified limit and finishing in a finite amount of time; this application shows how the filter may be used to "clean up" areas where it is not practical to sample adequately. The second is a hybrid deterministic and Monte Carlo ray-tracing program; this application shows how the filter can be effective even when the pixel variance is not known.

CR Categories and Descriptors: I.3.3 [**Computer Graphics**]: Picture/image generation - *Display algorithms*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism - *shading*

General Terms: Algorithms

Additional Key Words and Phrases: Monte Carlo, Lighting Simulation, Noise Reduction.

1 Introduction

Over the past decade two approaches for computing global illumination for realistic images have emerged – Monte Carlo path tracing and finite element (radiosity) methods. An advantage of the Monte Carlo approach is that the computational work is concentrated in the visible image at image resolution – not distributed throughout the environment. A disadvantage is that Monte Carlo images generated in a fixed length of time appear noisy. In this paper we examine the origins of this noise and develop the idea of energy preserving non-linear filters to reduce noise in post-processing. The filters are energy preserving in that they spread "noisy" samples into small regions rather than throwing them out. The filters are non-linear because the local distribution of samples is based on their magnitudes, hence output pixels will not be tied to a uniformly weighted sum of the input values. (Although such a relation may hold over large portions of the image.) The formulation of the filters is unique in that they are constructed by specifying the influence of each input sample, rather than by specifying the support region for each output pixel.

We begin with a discussion of digital filter applications in computer graphics, and the need for a new filter for synthetic images. We then discuss the source of noise in Monte Carlo images by examining features of this solution to the rendering equation. We develop the overall design of an energy preserving non-linear filter and show implementations of these filters for two different rendering systems. Examples are presented to demonstrate how energy preserving non-linear filters can effectively improve image quality without requiring additional sampling.

2 Filters

Generating a synthetic image is a sampling problem. The extensive literature in signal processing provides many useful algorithms and insights for image generation. Principles from signal processing for sampling and reconstructing images have been explored by many researchers (e.g. [6],[9]). Because samples are computationally expensive for global illumination calculations, efficient techniques must be employed in selecting the original samples. However, as noted by Mitchell [10] the problem of sampling global illumination is more complicated than 2-D image sampling because it involves predicting noise in 2-D space resulting from sampling a higher dimensional space. As we will discuss in more detail in the following section, even with sophisticated sampling techniques, excessive supersampling is required to eliminate noise under many common circumstances.

Lee and Redner [8] have studied the problem of noise in

[1] Rm. B-146, Bldg. 225, NIST, Gaithersburg, MD 20899
[2] 1 Cyclotron Rd., 90-3111, Berkeley, CA 94720

stochastically sampled synthetic images. They note that the linear filters typically used in computer graphics are unsatisfactory for eliminating the spike noise encountered in images generated using stochastic sampling. Linear filters tend to blur image details that should be kept sharp, while failing to spread the spike noise adequately. To resolve this problem, Lee and Redner proposed using alpha trimmed filters. Alpha trimmed filters have proved successful for eliminating spike noise and preserving edges in image processing applications. Such filters throw out "outlier" sample values when computing filtered pixel values. Lee and Redner demonstrate that alpha trimmed filters produce synthetic images without noticeable noise artifacts. However, they do not address the effect of such filters on the accuracy of the resulting image, and some important features may be lost.

Alpha trimmed and similar non-linear filters (e.g. morphological filters [5]) produce good results for many types of physically recorded images. In physically recorded images, the noise in the image is frequently due to secondary inputs that corrupt the signal of interest. Examples of unwanted secondary inputs are thermal noise in a detector element, and bit errors introduced in image transmission. Throwing out samples from these extraneous sources is desirable. In synthetic images, there are no corrupting secondary inputs. All of the samples carry valid information about the signal, and their effect should be included in the final image.

Another difference between synthetic and recorded images is the confidence we have in the results for some pixels. In recorded images there is frequently a great deal of uncertainty throughout the image. The exact pixel values are often unimportant, since many applications only need to identify objects, rather than examine subtle lighting effects. For synthetic images we have a high level of confidence in the values of some pixels. A great deal of computational effort has been expended to obtain subtle effects. We don't want the values of these pixels altered by a post-process filter.

In effect, we want our rendering process to simulate a sort of "idealized camera" up to and possibly including the storage of our final image. In displaying the image, we may attempt to compensate for human visual response with a tone-mapping operator (as discussed in the following section), but if we do not have a valid physical result, we have nothing to offer as input to such an operator.

The differences between synthetic and recorded images introduce two constraints for a filter for synthetic images that are not usually imposed on filters for physical images – energy preservation and minimal disruption. Energy preservation comes from the consideration that we do not want to throw any samples out. If a sample is contributing to a noisy region, we want to reduce noise by spreading out the energy it carries – not by removing it from the image. Minimal disruption comes from the consideration that we do not want to alter pixels that we have a high level of confidence in. To the greatest extent possible, the radiance of these pixels should be the same in the filtered image as in the unfiltered image.

3 The Complete Rendering Equation

The generation of a synthetic image was originally characterized as the solution of a rendering equation by Kajiya [7]. A complete rendering equation gives the values to be set on an image display device as a function of the radiometric properties of the synthetic scene and the display device. For convenience in examining the source of image noise we will examine the rendering equation in three parts:

$$L(x,y,\theta,\phi) = L_e(x,y,\theta,\phi) + \int_i f_r(x,y,\theta_i,\phi_i,\theta,\phi) L_i(x,y,\theta_i,\phi_i) cos\theta_i d\omega_i \tag{1}$$

$$L_p = \int_{image\_plane} L(x,y,\theta,\phi) g(x,y) dx dy \tag{2}$$

$$N_p = T(L_p) \tag{3}$$

Equation 1 is the equation of transport for visible light in the synthetic scene. Equation 1 gives the radiance $L(x,y,\theta,\phi)$ at visible surface point (x,y) in the direction (θ,ϕ) that would reach the observer of a physical realization of the scene. $L_e(x,y,\theta,\phi)$ is the emitted radiance of the point (non-zero only for light sources) and $f_r(x,y,\theta_i,\phi_i,\theta,\phi)$ is the bidirectional reflection distribution function (BRDF) for the point. The integral on the right hand side accounts for all reflection of incident radiance $L_i(x,y,\theta_i,\phi_i)$ from solid angles $d\omega_i$. Radiance has dimensions of energy per unit time, area and solid angle.

Equation 2 expresses the radiance L_p of a discrete pixel from the function $L(x,y,\theta,\phi)$. The function $g(x,y)$ is the 2-D filter used to eliminate spatial aliasing errors. The function $g(x,y)$ has dimensions of 1/area and is normalized so that L_p has the same units and range as $L(x,y,\theta,\phi)$.

Equation 3 expresses the conversion of pixel radiance, which can take on any physically realizable value (i.e. from starlit to sunlit scenes), and converts it to a dimensionless setting for the display device N_p – usually to an integer between 0 and 255. The function $T()$ is the tone operator, which is constructed using properties of human perception and characteristics of the display device. Various forms for $T()$ are discussed in [4], [12], and [13]. Even simple tone operators are inherently non-linear because of quantization effects and clipping of out of range values.

Typical Monte Carlo renderers compute an estimate $\hat{L}_p$ by forming and averaging many sample values L'_p. As stated by Purgathofer [11], the number of samples M_t is determined by the number required to reduce the estimated deviation S_p of the average to less than a specified tolerance d with a specified confidence α using the percentage point of the t distribution $t_{1-\frac{\alpha}{2},M_t-1}$. That is:

$$\hat{L}_p = \sum_{q=1}^{M_t} L'_{p,q}/M_t \tag{4}$$

$$S_p \equiv \sqrt{\frac{1}{M_t(M_t-1)} \sum_{q=1}^{M_t} (L'_{p,q} - \hat{L}_p)^2} \tag{5}$$

$$S_p < d/t_{1-\frac{\alpha}{2},M_t-1} \tag{6}$$

The calculation of each value L'_p begins by selecting a random value of (x,y) in the right hand side of Eq. 2. At this location Eq. 1 is estimated by choosing a random direction for evaluating the integral on the right hand side. Since the value of $L_i(x,y,\theta_i,\phi_i)$ is unknown, and also governed by Eq. 1 , this estimation is done recursively, and a path of rays is generated [7]. After values of $\hat{L}_p$ are computed the tone operator in Eq. 3 is applied to display the image.

Using this naive approach, there is often high variance in the estimate of the integral on the right hand side of Eq. 1. This is because the integrand sample values can vary from the light source radiance to the radiance of dark, shaded objects in the room - with a dynamic range of 10^5 being common [4]. To make Monte Carlo solutions practical, the integral on the right hand side is rewritten as a sum of integrals [3].

$$\int_i f_r(x,y,\theta_i,\phi_i,\theta,\phi)L_i(x,y,\theta_i,\phi_i)cos\theta_i d\omega_i =$$

$$\int_s f_{r,d}(x,y,\theta_i,\phi_i,\theta,\phi)vis(s)L_e(x_s,y_s)cos\theta_i cos\theta_s dA_s/r_s^2$$
$$+\int_i f_{r,d}(x,y,\theta_i,\phi_i,\theta,\phi)L_{i,not\_s}(x,y,\theta_i,\phi_i)cos\theta_i d\omega_i$$
$$+\int_{spec\_lobe} f_{r,s}(x,y,\theta_i,\phi_i,\theta,\phi)L_i(x,y,\theta_i,\phi_i)cos\theta_i d\omega_i \quad (7)$$

In Eq. 7, $f_{r,d}$ is the diffuse-like component of the BRDF, and $f_{r,s}$ is the specular-like. The integral for the diffuse-like component is divided into parts. The first is the integral for direct illumination. The direct illumination is an integral over all light sources s in terms of the area of the sources A_s, visibility of the source $vis(s)$, emission from the source surface $L_e(x_s,y_s)$, angle from the source θ_s and distance to the source, r_s. The second is an integral for indirect illumination, i.e. over all incident light that does not come directly from the light source. The final integral on the right of Eq. 7 gives the specularly reflected light. Each of the integrals on the right of Eq. 7 generally has a lower variance than the integral on the left of Eq. 1, so the number of trials required is greatly reduced [1].

4 Designing a Filter for Synthetic Image Noise

One way to eliminating noise in Monte Carlo images is to increase sampling rates until the estimated error is less than the display device brightness resolution. However, in this section we show that in a typical image there will be regions in which the number of samples required to achieve this goal is impractical.

4.1 The Origin of Noisy Regions

A "worst case" estimate of the minimum number of trials M_t required for applying the test in Eq. 6 is given by Purgathofer [11]. Scaling the values so that samples range from 0 to 1, the number of trials required is:

$$M_t > \frac{\log(1-\alpha)}{\log(1-d)} \quad (8)$$

For an image anti-aliasing problem with samples that ranged in value from 0 to 255, Purgathofer found that useful results were obtained when an interval of $D = \pm 13$ (i.e. d = 13/255=.05 in Eq. 8) was allowed with a confidence of 80%. These values of d and α give a minimum sampling of 32 trials/pixel.

The sampling rates required when computing radiance in "real world" floating point values and subsequently mapping to the device with a tone operator are much higher. While the form of tone operators varies, a typical radiance value on the order of 10^{-3} times the light source radiance is mapped in Eq. 3 to a value N_p on the order of 100. An interval on the order of ± 10 in the final display then requires an interval d equal to 10^{-4} times the light source radiance. Scaling the problem so that the light source radiance is 1, a value of $d = 10^{-4}$ with a confidence of 80 % in Eq. 8 gives a minimum sampling rate of **16,094** trials per pixel!

The reorganization of the equation of transport given in Eq. 7 reduces the variance so that for most pixels in an image this worst case does not occur. However, there will be small regions in the image in which the "worst case" is encountered. These isolated regions will be noisy because they are undersampled. We summarize these high variance cases in Fig. 1.

Figure 1(a) illustrates the first type of high variance integration. Both light sources and non-light sources can be visible through some pixels. Sampling Eq. 2 for these pixels is essentially sampling a binomial distribution with several orders of magnitude in the two alternatives. Convergence is extremely slow in such cases.

Figure 1(b) illustrates a second high variance case - the integration of direct illumination for a diffuse-like surface (first integral on the right of Eq. 7). For a point which has a full view of the light source, a small number of trials are needed to estimate the cosine and distance terms in the integral. However, when the view of a light source is partially obscured a large number of trials may be required to estimate the visible area. The smaller the fraction of the source that is visible, the larger the number of trials needed. Since a light source has a high radiance, just one "hit" will result in a large sample standard deviation.

Figure 1(c) illustrates the integration of reflected light for a specular-like surface, the third integral on the right of Eq. 7. The BRDF is concentrated on a small lobe near the specular direction. A high variance in samples for this case occurs when a small portion of this lobe is subtended by a light source.

The fourth type of integration with high deviations is the case of "caustic paths", shown in Fig. 1(d). The integral diagrammed in Figure 1(d) is the second term on the right of Eq. 7. A "caustic" appears when a small portion of the incident hemisphere is subtended by the image of a light source in a specular-like reflection. When a sample ray hits this small image, a large deviation in the sample is introduced.

In any of the four cases, high deviations are expected in some region of the scene, not at one point. Light source edges, penumbrae, fuzzy specular reflections and caustics spread through regions. For Q pixels in one of these regions, only a small number of pixels q will have obtained samples hitting the high radiance portion of the domain of integration. The q pixels will *appear* to be adding noise to the region of $Q-q$ pixels that *appear* to be accurate. Actually, all Q pixels are equally valid. Rather than throwing out the q "noise" pixels, the true value that should have been calculated for the whole region should be an average of all Q values.

4.2 Filter Design

Our goal then is to design a filter that will spread the influence of q "noise" pixels into a larger region of Q pixels. We want to spread these pixels out without changing the total energy in the image, and without changing any pixel values unnecessarily.

First, to meet the goal of preserving energy, the filter must be applied to floating point sample values before the application of the tone operator. For example, take a simple tone operator that scales values by (128/.001), and clips values over 255. Consider a box filter in 1-D on the three

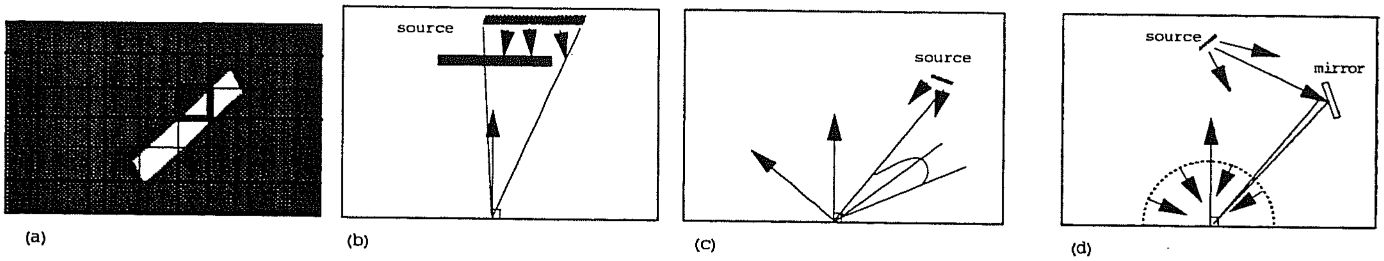

Figure 1: *High pixel variances result when: (a) a light source is partially visible through a pixel the variance in samples, (b) a diffuse-like surface views only a small fraction of a light source, (c) when a light source subtends a small portion of the specular lobe of a specular surface, or (d) the image of a light source is visible to a diffuse-like surface via specular reflections*

values .0001, .001, and 1. If used before the tone operator, the filter gives 0.3337 and the pixel is clipped to a value of 255. If the tone operator is used first, the values become 13, 128 and 255, and the filter erroneously produces a result of 132.

Because we also want minimal disruption of pixels which are not in noisy areas, we want to design a filter that spreads the influence of "noise" pixels *only*. Because we are working from the point of view of constructing rather than analyzing an image, we construct our filter from a different point of view than usual.

Figure 2 diagrams typical digital filters. A support region is defined for each output pixel. This region may be uniform, as shown in (a). This has the energy preserving property we seek, but has the disadvantages of blurring detail and not adequately distributing some values. A variable width region may be used to change the influence of some input samples [6]. However, as shown in the example in (b), this type of variable width kernel does not preserve all of the energy in the original sample set.

We propose constructing energy preserving filters based on the region of influence of each input sample, rather than defining a support region for each output sample. This is diagrammed in Fig. 3. A region is defined for each input sample. The weights assigned to the sample as it is distributed to the output image sum to one, and energy is preserved. By varying the region of influence for each input sample, large areas of the image can remain unaffected by the filter and the input image is disrupted a minimal amount.

The design of any input based energy preserving filter requires two rules: 1) a rule to identify "noise" inputs, and 2) a rule to determine the region into which the "noise" inputs will be distributed.

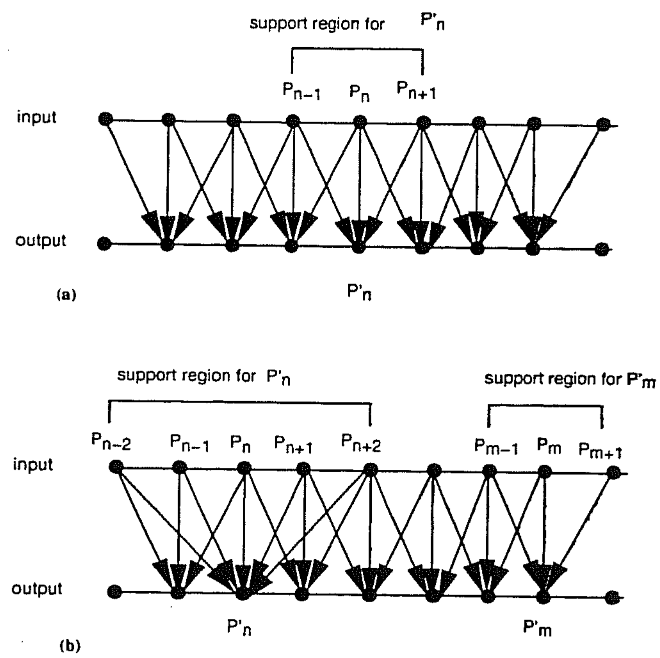

Figure 2: *(a) A constant width filter based on defining the support region for each pixel. The weights assigned to the samples in the support region sum to one, i.e. $P'_n = .25P_{n-1} + .5P_n + .25P_{n+1}$. Energy is preserved since the weight assigned to each input pixel sums to unity for the image as a whole. (b) A variable width filter based on defining the support region for each pixel. The weights assigned to the samples in the support region sum to one. The energy in the input image is not preserved, however. For example, P_n is weighted by .25 for P'_{n-1}, by .375 for P'_n and by .25 P'_{n+1}, so that 12.5% of the energy from P_n is lost.*

5 Example Applications

There are many ways that Monte Carlo solutions to the rendering equation can be constructed. Energy preserving filters can be used to reduce noise in any of these methods. We present the application of energy preserving filters to two different rendering methods.

5.1 Monte Carlo Path Tracing with a Radiosity Preprocess

The first example we consider is a Monte Carlo path tracing (MCPT) method with a radiosity preprocess to reduce variance. In this method the estimates $\hat{L}_p$ are made as described

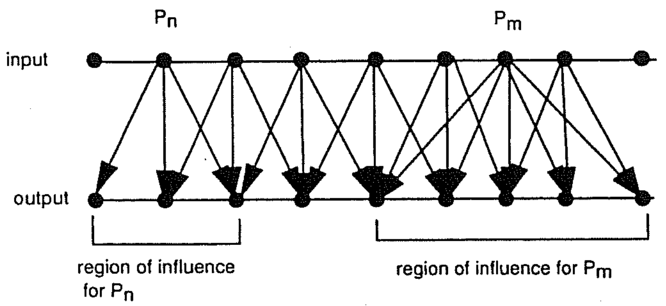

Figure 3: *A variable width filter based on defining the region of influence for each input sample. The sum of weights assigned to a sample as it is distributed to the output image is set to one to guarantee energy preservation. For example,* P_n *is weighted by .25 in computing* P'_{n-1}, *.5 for* P_n *and .25 for* P'_{n+1}. P_m *is weighted by .0625 for* P'_{m-2}, *.25 for* P'_{m-1}, *.375 for* P'_m, *.25 for* P'_{m+1} *and .0625 for* P'_{m+2}.

in Section 3, except that in estimating the incident radiance for diffuse-like surfaces in the indirect illumination integral, values from a radiosity preprocess are used. This is essentially the progressive multi-pass method described by Chen et al. [3], except that the "light ray" pass for explicit calculation of caustics is replaced by allowing caustic paths to be followed in the integration of indirect illumination. The "light ray" pass can be extremely costly when many light source/specular surface combinations need to be examined. By using an energy preserving filter, the noise that has been the drawback of including caustic paths in the Monte Carlo integration will be reduced.

For this method, "noise" pixels will be identified as pixels which have not converged to within a user specified interval $\pm D$. To be sure that most pixels have converged, we need a minimum value of M_t that will be valid for the majority of pixels. To predict M_t, we need to know typical values of S_p (defined in Eq. 5). We also need to estimate the floating point value visibility threshold L_{tvis} which will translate into a difference of one unit in N_p. The threshold depends on the form of T() in Eq. 3.

To predict S_p and L_{tvis} we take a small pilot sample (e.g. similar to the pilot sample suggested in [2]) by rendering a 100x100 pixel image with 16 trials per pixel. Since the deviation in our estimates decreases with the square root of the number of trials, the estimate of $S_{p\_M_t}$ for M_t samples will be the average sample deviation $\bar{S}_{p\_16}$ we calculate for the pilot sample times $\sqrt{16/M_t}$. For the purposes of estimating L_{tvis}, we assume a simple linear tone operator in which the average image value L_{ave} will be mapped to the middle of the display range, N_{MID}. That is:

$$L_{tvis} = L_{ave}/N_{MID} \tag{9}$$

Knowing L_{tvis} and an estimate of S_p, the number of trials required to obtain a result accurate to $\pm D$ is approximately:

$$M_t = \left(\frac{4\bar{S}_{p\_16}}{DL_{tvis}}\right)^2 \tag{10}$$

We construct our method to require a minimum of samples M_t as given by Eq. 10. (Note, because $t_{1-\frac{\alpha}{2},M_t-1}$ is near 1.0 for $\alpha = 80\%$ and $M_t > 16$ it does not appear in the estimate in Eq. 10.) Since the value of M_t using this technique is a typical, rather than maximum value, we multiply this number by heuristically determined value of 4 to establish $M_{t,ceiling}$ to insure that the majority of pixels will converge.

The simple linear tone operator is only assumed for the purpose of estimating the number of samples. The image will still be computed in floating point, and any tone operator can be applied to the result.

For the method just described, a pixel is considered a "noise" pixel to be treated by the filter if its sample variation is larger than the acceptable tolerance after $M_{t,ceiling}$ trials. The excess value at each unconverged pixel is defined as the difference between the pixel radiance and the average of its immediate neighbors. If the excess energy exceeds this average by more than the variation allowed in the original calculations (i.e. DL_{tvis}), it is spread into a region around the unconverged pixel.

Energy could be preserved by simply spreading the excess energy over the entire image. However, that would unnecessarily disrupt some pixels, and all definition of features such as caustics would be lost. To determine the size of a smaller region into which the energy should be spread, we use the criterion that we do not want to introduce non-physical high frequency artifacts into the image. The energy is spread so that the additional radiance at each pixel will be no more than L_{tvis}.

Any shape filter could be used. For a simple box shaped filter, the calculation is straightforward. The difference between the radiance of the unconverged pixel L_u and the average of its converged neighbors $\bar{L}_n$ is found. The amount that $L_u - \bar{L}_n$ exceeds the allowable error level DL_{tvis} is the excess value L_{excess} which needs to be distributed. To limit the effect on neighboring pixels, the number of pixels to which the excess is distributed is $ceiling(L_{excess}/L_{tvis})$.

Only converged pixels in the original solution are used to compute the average. The effect of the filter does not depend on the order in which pixels are traversed. In some instances an unconverged pixel will have no converged neighbors. In this case the filter is applied recursively, with pixels that have had excess energy spread out marked as converged in the next iteration.

The method just described doesn't account for the spectral variation of radiance. If the method were applied for each spectral sample individually, wavelengths for which a pixel had a high variance would be smoothed out, while values at other wavelengths would be left untouched. The result would be "noisy" areas tending to turn gray. To avoid this color shift, the luminance of each "noisy" pixel is compared to the average luminance of its neighbors, and excess luminance is distributed with the same spectral distribution as the original "noisy" pixel.

Figure 4 shows the results of applying the simple box filter to an image generated using MCPT with radiosity preprocess. Two small specular boxes, one blue and one red are located in a yellow-brown room. (The specular boxes do not appear shiny because they reflect the featureless walls of the room). An image rendered using Eq. 10 to determine M_t with $D = 5$ for a 0 to 255 display device is shown on the left. There are noisy regions in the image due to caustic paths in the indirect illumination calculation as diagrammed in Fig. 1(d). A noisy caustic due to the blue box can be seen on the ceiling near the light. More subtle noisy caustics due to the red box can be seen on the floor near the red box and on the ceiling above the red box. There is also a large penumbra region (as diagrammed in Fig. 1(a)) on the left wall due to the blue box.

The image after applying the filter is shown in the center.

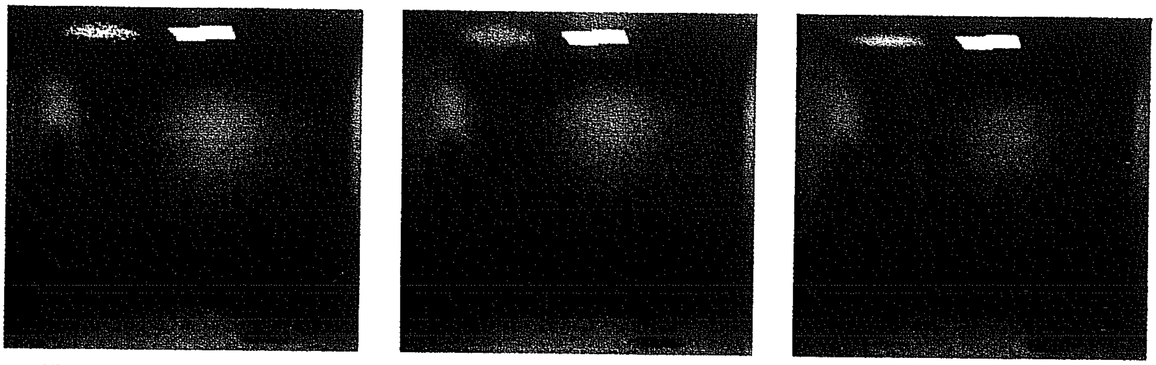

Figure 4: Application of energy preserving filter to images using Monte Carlo path tracing with radiosity preprocess. Left: Unfiltered image, 60–240 samples. Center: Left image after filtering. Right: Unfiltered image, 1500–6000 samples.

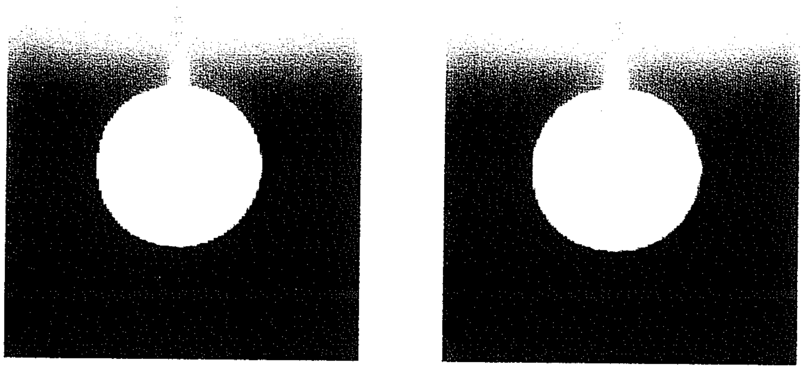

Figure 5: Low resolution images of a light source generated using *Radiance*. Pixels are enlarged to emphasize differences. Left: Uniform Gaussian filter. Right: Non–linear filter.

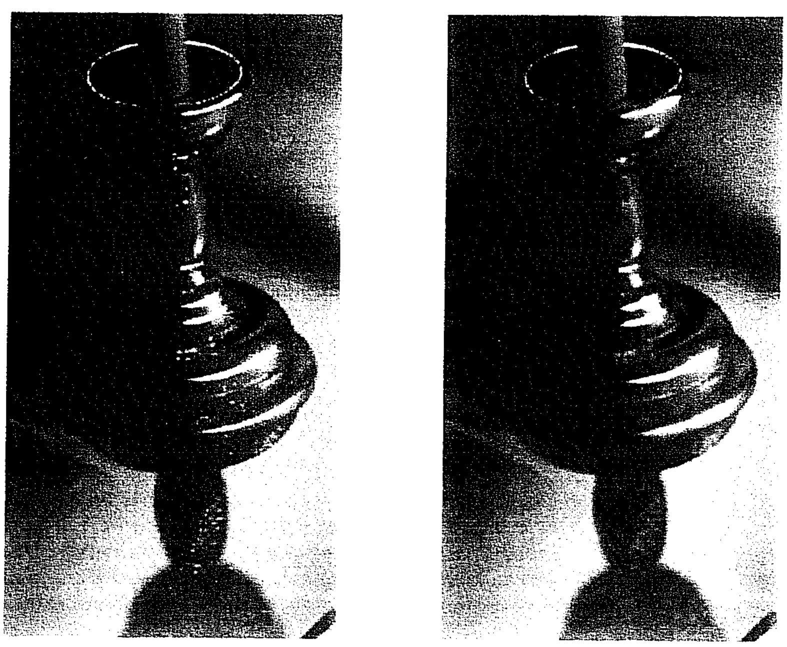

Figure 6: Images generated using *Radiance* showing noise in specular reflection. Left: Uniform Gaussian filter. Right: Non–linear filter.

A comparison of the unfiltered and filtered images shows that the filter correctly left features outside of the high noise areas undisrupted. Only a small number of pixels in the penumbra region are changed. The large caustic feature due to the blue box is retained and smoothed out, as are the smaller fainter red caustics.

The image in the lower right is the same scene rendered using a minimum sampling rate that is 25 times higher (i.e. M_t set for $D = 1$.) Not surprisingly, the filtered image appears somewhat noisier than the high sampling rate image which took **25** times longer to compute. This is because the filter is designed to affect the unconverged "high noise" pixels only, not to remove the ±5 units of variation allowed in the original low sampling rate calculation. However, a comparison of the filtered and high sampling rate images does show that the caustic features preserved by the filter are real.

5.2 RADIANCE

Radiance is a rendering system developed over many years at the Lawrence Berkeley Laboratory. It incorporates most kinds of light transport in a physically-based simulation of architectural (and other) environments, using a hybrid Monte Carlo and deterministic ray tracing approach that has been optimized to provide accurate results quickly in most cases [14]. In general, *Radiance* produces high quality results in much less time than the MCPT method just described.

"Noise" pixels are less common in *Radiance*, but they may still occur because of the Monte Carlo components of the calculation. However, because of the deterministic components of the calculation, an explicit measure of S_p is not available for each input sample. Alternative rules for identifying "noise" pixels and their region of influence are needed.

In *Radiance*, an initial floating-point picture is generated at the super-sample resolution (typically 3x3 times the final image resolution). Anti-aliasing and other filtering operations are carried out by a separate program. Since we do not have an estimate of the variance of each sample, we define "noisy" samples as pixel super-sample values that are very large (or very small) compared to their neighbors. We increase the radius of influence for these samples. The criterion for how much to spread a sample is simply stated:

> Any given super-sample is spread out sufficiently that its influence on any given output pixel is smaller than a specified tolerance.

A "noisy" super-sample is identified as having a greater influence on an output pixel than we can tolerate with the current filter kernel. The amount a given sample affects a weighted average of samples is derived very easily from the formula for weighted averages:

$$\bar{x} = \frac{\sum_i w_i x_i}{\sum_i w_i} \tag{11}$$

The average without sample x_k is simply:

$$\bar{x}_{k-} = \frac{\bar{x}(\sum_i w_i) - w_k x_k}{\sum_{i \neq k} w_i} \tag{12}$$

Taking the absolute difference between $\bar{x}$ and $\bar{x}_{k-}$ and dividing by $\bar{x}_{k-}$, we arrive at the absolute relative difference due to a super-sample's influence:

$$D_{ar} = \frac{|\frac{x_k}{\bar{x}} - 1|}{\frac{\sum_i w_i}{w_k} - \frac{x_k}{\bar{x}}} \tag{13}$$

$$\bar{x}, \bar{x}_{k-} > 0$$

Our goal is to find a kernel width that produces a D_{ar} less than or equal to the selected tolerance. In the context of a filter kernel whose weights sum to one, this translates to the following formula:

$$tolerance \geq \frac{|\frac{x_0}{\bar{x}} - 1|}{\frac{1}{w_0} - \frac{x_0}{\bar{x}}} \tag{14}$$

where:

x_0 is the super-sample value

w_0 is the super-sample weight at the central peak of the filter kernel

$\bar{x}$ is the kernel-weighted average centered on this super-sample

If the effect of a sample is above our tolerance level using the default kernel radius, the radius is expanded until the sample's effect is below tolerance. The tolerance given depends on the expected pixel variance, which is in turn related to the number of super-samples used for each pixel. From experience using a Gaussian kernel, a good tolerance value for 3x3 oversampling is 0.25, and a good tolerance for 4x4 oversampling is 0.15. These tolerance values are typically higher than the L_{tvis} value used for the previously described box filter because the influence of a Gaussian kernel always peaks near the closest output pixel, and drops off rapidly with distance.

The search for a kernel width to satisfy Relation 14 without going too far overboard could be expensive, so we have made a few optimizations. First, we work with pixel luminance values rather than colors, which reduces the number of operations and avoids the color shifting mentioned in the previous section. Second, we compute ring sums about the closest output pixel and use 1-dimensional vector multiplication to compute the different Gaussian-weighted averages, greatly simplifying the calculation of $\bar{x}$. Finally, we use numerical iteration to zero in on the appropriate kernel width more quickly. Treating 14 as an equation, we guess the next kernel width based on the D_{ar} computed for the current width. In most cases, this produces faster convergence than a simple binary search, but care must be taken to avoid infinite iteration on anomalous pixels. In our implementation of this filter, we have found it to take about three times as long as a standard Gaussian kernel, which is still insignificant compared with the overall rendering times.

Examples of applying a filter of this form are shown in Figs. 5 and 6. The examples demonstrate two types of high variance areas that can be encountered in *Radiance* renderings.

As discussed in Section 4.1, light source boundaries (diagrammed in Fig. 1(a)) may cause aliasing in the final image even when many samples are taken at a pixel, because just one sample landing on or off the light source makes a detectable contrast difference in the final result. As mentioned in Section 4.2 the usual solution of clamping before filtering produces incorrect results. It also destroys the physical units of the result. By applying an energy preserving nonlinear filter before mapping to the display device, extreme contrast boundaries are spread out and aliasing is reduced. The effect is a slight fuzziness to light source boundaries in proportion to their brightness, something that in appearance is quite natural because the eye loses acuity in these regions, anyway.

The image on the left of Fig. 5 shows a low-resolution closeup of a pendant fixture, filtered with a linear Gaussian kernel and 9 samples/pixel. Notice the jagged edges caused by inadequate sampling. The image on the right shows the same computation with an energy preserving non-linear filter applied during anti-aliasing. The source boundaries are now softer and smoother, as they would appear in real life. The results have not been changed, only dispersed slightly around the source edges. This is important for later analyses, which might need the absolute radiance values to evaluate glare or other visual quality metrics.

To avoid unpredictably long rendering times, *Radiance* uses a minimal number of shadow rays to light sources plus one specular ray per pixel super-sample per surface interaction, similar to Monte Carlo path tracing. The user chooses an initial sampling density that produces adequate convergence over most of the image, but in areas where the number of samples chosen is not enough, there will be noise. The most frequent source of objectionable sampling noise is rough specular reflection of light sources (diagrammed in Fig. 1(c).) The left image in Fig. 6 shows a rendering of a candle holder with a rough specular surface on a table using 16 samples per pixel and a linear Gaussian filter. In this case, a linear Gaussian filter compounds the sampling artifacts by spreading them out to neighboring pixels without sufficiently reducing their contribution. So, little bright spots become big bright spots. The right image demonstrates how a non-linear filter reduces image noise without compromising the results. The specular highlights that were present in the calculation are still present in the filtered image – only more evenly distributed. This can be compared with alpha trimmed filters that reduce noise simply by removing the offending samples, taking the very real highlight with them. The energy preserving quality of our filter ensures that we do not lose the information we have worked so hard to compute.

6 Conclusions

We have introduced a new class of non-linear filters that reduces sampling noise while preserving energy and important image features. In two example applications we have shown that such a filter may be used to clean up unconverged sections of a Monte Carlo image, or reduce artifacts in a hybrid deterministic and stochastic ray tracing system.

The new filtering technique has particular significance for physically-based rendering, where image accuracy is a key goal. Its energy preserving nature means none of the calculations are thrown away, and the filter's non-linear response is critical in a floating-point domain where sample values may differ by several orders of magnitude. Also, the characteristic of minimal disruption guarantees that converged pixels will not be adversely affected.

The key underlying theme in this paper is choosing between what is correct and what is acceptable in a physically-based rendering. The eye's relative sensitivity to high frequency noise tends to undermine the application of Monte Carlo techniques, since reducing variance in some parts of an image can be extremely expensive. Instead, we can recognize these areas as being inadequately sampled, and use a variable-width kernel to reconstruct them in a way that maintains overall accuracy without offending the eye. Our goal is to be as correct as possible and still be acceptable to the viewer. Since acceptability is such a subjective measure, it is difficult to say when and whether an optimal kernel has been found, but the general approach of scaling kernel width to target a specific variance seems to work quite well.

It is our hope that this new class of filters will help broaden the practical applications of Monte Carlo techniques in rendering by removing one of its principal drawbacks: image noise.

REFERENCES

[1] J. Arvo and D. Kirk. Particle Transport and Image Synthesis. Proc. of SIGGRAPH '90 (Dallas,TX, Aug. 6-10, 1990. *Computer Graphics*, 24(4):63–66, Aug. 1990.

[2] J. Arvo and D. Kirk. Unbiased Sampling Techniques for Image Synthesis. Proc. of SIGGRAPH '91 (Las Vegas,NV, Jul. 28- Aug. 2). *Computer Graphics*, 25(4):153–156, Jul. 1991.

[3] S. Chen, H. Rushmeier, G. Miller, and D. Turner. A Progressive Multi-Pass Method for Global Illumination. Proc. of SIGGRAPH '91 (Las Vegas,NV, Jug. 28- Aug. 2). *Computer Graphics*, 25(4):165–174, Jul. 1991.

[4] K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang, and K. Zimmerman. Spatially Non-Uniform Scaling Functions for High Contrast Images. In *Proc. of Graphics Interface 1993 (Toronto, May 19-21)*, pages 245–253.

[5] C.-H. Chu and E. Delp. Impulsive Noise Suppression and Background Normalization of Electrocardiogram Signals Using Morphological Operators. *IEEE Trans. on Biomedical Engineering*, pages 262–267, Feb. 1989.

[6] M. Dippé and E. Wold. Antialiasing Through Stochastic Sampling. Proc. of SIGGRAPH '85 (San Francisco,CA, Jul. 22- 26, 1991). *Computer Graphics*, 19(3):69–78, Jul. 1985.

[7] J. Kajiya. The Rendering Equation. Proc. of SIGGRAPH '86 (Dallas,TX, Aug. 18-22). *Computer Graphics*, 20(4):143–150, Aug. 1986.

[8] M. E. Lee and R. A. Redner. A Note on the Use of Nonlinear Filtering in Computer Graphics. *IEEE Computer Graphics and Applications*, pages 23–29, May 1990.

[9] D. Mitchell. Generating Antialiased Images at Low Sampling Densities. Proc. of SIGGRAPH '87 (Anaheim,CA, Jul. 27-31). *Computer Graphics*, 21(4):65–72, Jul. 1987.

[10] D. Mitchell. Spectrally Optimal Sampling for Distributed Ray Tracing. Proc. of SIGGRAPH '91 (Las Vegas,NV, Jul. 28- Aug. 2). *Computer Graphics*, 25(4):157–164, Jul. 1991.

[11] W. Purgathofer. A Statistical Method for Adaptive Sampling. *Computers & Graphics*, pages 157–162, 1987.

[12] J. Tumblin and H. Rushmeier. Tone Reproduction for Realistic Images. *IEEE Computer Graphics and Applications*, pages 42–48, Nov. 1993.

[13] G. Ward. A Contrast-Based Scalefactor for Luminance Display. In P. Heckbert, editor, *Graphics Gems IV*. Academic Press, 1994.

[14] G. Ward. The RADIANCE Lighting Simulation and Rendering System. Proc. of SIGGRAPH '94 (Orlando,FL, Jul. 24-29). *Computer Graphics, Annual Conference Series*, 1994.