Adaptive Filtering for Progressive Monte Carlo Image Rendering

Frank Suykens, Yves D. Willems

Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A, B3001 Heverlee
Belgium
Frank.Suykens@cs.kuleuven.ac.be

ABSTRACT

Image filtering is often applied as a post-process to Monte Carlo generated pictures, in order to reduce noise. In this paper we present an algorithm based on density estimation techniques that applies an energy preserving adaptive kernel filter to individual samples during image rendering. The used kernel widths diminish as the number of samples goes up, ensuring a reasonable noise versus bias trade-off at any time. This results in a progressive algorithm, that still converges asymptotically to a correct solution. Results show that general noise as well as spike noise can effectively be reduced. Many interesting extensions are possible, making this a very promising technique for Monte Carlo image synthesis.

Keywords: global illumination, Monte Carlo, density estimation, bidirectional path tracing, image filtering

1 INTRODUCTION

Monte Carlo techniques are frequently used for physically based image synthesis. Methods like path tracing and bidirectional path tracing are relatively simple and are able to solve the global illumination problem for very general scenes. The basis of Monte Carlo methods is stochastic sampling and therefore errors in the resulting images show up as noise. The most common and easiest way to reduce this noise, is to take more samples, which in the limit will result in a perfect image. However sampling in Monte Carlo rendering involves tracing paths through the scene, and these are very costly operations. Computing an image where all the noise is reduced to a visibly acceptable level, can take an awful lot of time.

Pure Monte Carlo methods compute each pixel in an image independently, ignoring any coherence between neighbouring pixels A substantial amount of research has been directed towards designing filtering procedures that exploit this coherence. Most of these filters are applied as a post-process after the image has been computed with a certain amount of samples per pixel.

Note that filtering an image does not necessarily produce a more accurate image. There is always a trade-off between noise and bias involved. For example using a simple blur filter as a post-process, may reduce some of the noise but it also blurs edges in the image, which may be as visually distracting as the noise.

In this paper we present an energy preserving filtering procedure, based on density estimation techniques, with the following characteristics:

- A variable kernel filter is applied, not to the pixels but to the individual samples themselves. The value of a pixel is an average of a number of samples, but not all these samples are equally 'bad'. The kernel width is based on some 'badness' criterion for the sample, allowing to spread out bad samples but leave the good samples within the same pixel localised.
- The kernel filters are applied during image creation and not as a post-process. The used kernel widths diminish as the number of samples goes up, ensuring a reasonable noise versus bias trade-off at any time. This results in a progressive algorithm, that still converges asymptotically to a correct solution.

In the next section we will discuss some existing filtering methods that have been used for Monte Carlo rendering and indicate the differences with our method. Section 3 will explain the density estimation techniques upon which this work was based. In section 4 our filtering algorithm is presented followed by some results in section 5. Although good results have been obtained, there are still a lot of improvements and some open questions that need further research. These will be discussed together with the conclusion in section 6.

2 PREVIOUS WORK

Filtering out the noise in an image is a very tempting idea, especially since the human eye and brain is very good at determining how the image should look despite of the noise.

Lee and Redner [Lee90] proposed the use of non-linear median and alpha trimmed filters to eliminate spike noise in stochastically rendered images. The spike noise pixels are thrown out, making this a non energy preserving filter. In fact any filter applied after tone mapping is not energy preserving.

Rushmeier and Ward [Rushm94] present an energy preserving non-linear filter that also targets spike-like noise, but spreads it out over a number of neighbouring pixels depending on a variance estimate.

Jensen [Jense95] applies filters only to the light transport that was scattered diffusely at least twice, under the assumption that this forms the source of most of the noise and in order not to blur features caused by the other part of the light transport.

McCool [McCoo99] recently investigated the use of anisotropic diffusion for noise reduction. The noise is averaged out using diffusion equations but edges and textures can be preserved. It is an energy preserving technique.

All previous methods work as a preprocess. Dutré et al. [Dutre93] used a gaussian kernel around visible path vertices in object space for particle tracing. However a fixed kernel width was used and expensive eye rays were needed to evaluate the kernel for affected pixels.

Our method includes aspects from several of these previous methods but still is significantly different from them in that it uses an *energy preserving* filter with *variable kernel width* for *individual samples* (in image space) and is applied during rendering. The *progressiveness* of our solution is one of the main differences with the previous work.

3 DENSITY ESTIMATION

3.1 Standard density estimation

Our work is partly based on density estimation which is a technique for recovering a probability density function (pdf) from a number of observed samples of this function. There are three main approaches in density estimation: the histogram method, nearest neighbour methods and kernel density estimation [Silve86]. Our method uses a form of kernel density estimation.

Given an unknown pdf $p(x)$ and N observations or samples x_i generated by this pdf, a standard kernel density estimator for $p(t)$ is given by [Silve86] :

$$\hat{p}(t) = \frac{1}{Nh^d} \sum_{i=1}^{N} \mathcal{K}(\frac{t - x_i}{h}) \qquad (1)$$

with $\mathcal{K}$ a certain kernel function, h the kernel width or bandwidth and d the dimension of the domain of p. The estimator can be seen as if every sample x_i is spread out by the kernel function $\mathcal{K}$.

The estimated value of this estimator is the original pdf $p(t)$ convoluted by the kernel function $\mathcal{K}((t-x)/h)$. So the result will always have a bias depending on the kernel $\mathcal{K}$ and the width h.

Expressions for variance and bias of the estimator (1) can be derived [Silve86] and learn us that in order to reduce variance a large h should be chosen, but for bias reduction a small h should be taken. The width of the kernel thus allows to trade bias for noise and vice versa, and it should be chosen very carefully. The choice of kernel width is in fact much more important than the choice of the kernel shape.

Variable kernel density estimation uses a different kernel width h_i for every observed sample x_i. Intuitively a smaller kernel width could be chosen where the density of observation is large and a wider kernel where only few observations are found.

3.2 Density estimation in rendering

Density estimation has been used in rendering mainly to reconstruct diffusely reflected illumination on surfaces in a scene.

Heckbert [Heckb90] first noted that reconstructing this illumination is in fact a density estimation problem. He uses the histogram method to construct adaptive radiosity textures. Shirley et al. [Shirl95], Myszkowski [Myszk97], and Walter et al. [Walte97] use kernel density estimation to reconstruct diffuse illumination on surface after storing surface hits generated by path tracing. Myszkowski and Walter [Walte98] use an adaptive kernel width based on the density of the samples. Jensen [Jense96] uses a nearest neighbour method when reconstructing illumination from a photon map. Most of this illumination however is only used indirectly in a gather pass, allowing a less accurate storage and reconstruction.

3.3 Density estimation for multiple importance sampling

In our examples we will use a density estimation approach on the image plane (dimension $d = 2$), where samples are generated using bidirectional path tracing [Lafor93, Veach94]. Note that we need to reconstruct an arbitrary illumination function and not a pure pdf. Also note that that the samples on screen originate from a number of *different* pdf's using multiple importance sampling[1] [Veach95].

Moreover do the samples obtained in the image plane originate from a much higher dimensional sampling procedure (tracing paths), so that even samples with the same image plane position can have a vastly different value. The image plane function radiance $f(x)$ that we want to reconstruct is in fact the result of an integration over all possible transport paths that go through this particular image position x. We will denote full transport paths going through x as a capital X and similarly we will denote functions defined on a full transport path by a capital letter. For example the contribution of a path X to the corresponding image position x is denoted by $F(X)$.

With respect to this rendering context we need to slightly modify the kernel density estimator:

Suppose we want to reconstruct a 2D image plane function f and do this by taking N_k samples or paths $X_{k,i}$ from a set of pdf's P_k. Due to the multiple importance sampling, each sample $X_{k,i}$ has to be assigned a weight $W_k(X_{k,i})$ to get an unbiased solution. The constraint $\sum_k W_k(X) = 1$ ensures this unbiasedness (see [Veach95] for more information). Now the modified variable kernel density estimator becomes:

$$\begin{aligned}\hat{f}(t) &= \sum_k \frac{1}{N_k} \sum_i \frac{W_k(X_{k,i})F(X_{k,i})}{P_k(X_{k,i})} \cdot \\ &\qquad h^{-2}(X_{k,i})\mathcal{K}(\frac{t - x_{k,i}}{h(X_{k,i})}) \qquad (2)\end{aligned}$$

Using $N = \sum_k N_k$ and $G_{k,i} = \frac{N W_k(X_{k,i})F(X_{k,i})}{N_k P_k(X_{k,i})}$ this can be written more compactly as:

$$\hat{f}(t) = \frac{1}{N} \sum_{k,i} G_{k,i} h^{-2}(X_{k,i})\mathcal{K}(\frac{t - x_{k,i}}{h(X_{k,i})}) \qquad (3)$$

[1]Paths are traced from eye and light simultaneously, and every vertex from the eye path is connected to every vertex of the light path. Each connection results in a different pdf.

So the only difference with estimator (1) is the introduction of a weight $G_{k,i}$ for every sample as well as dependency of h on the sample (or path) X. It can be shown easily that the expected value of this estimator is also a convolution of the target function with the kernel.

3.4 Kernel shape selection

The shape of the kernel $\mathcal{K}$ determines how samples are spread out into their neighbourhood. We use the fairly standard Epanechnikov kernel which is given by:

$$\mathcal{K}(x) = \begin{cases} \frac{2}{\pi}(1 - |x|^2) & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

This kernel has some desirable properties concerning the mean integrated square error of the estimator (see [Silve86, p40]). Other kernels like a gaussian can be used if the estimated function has certain smoothness requirements, but since the radiance on the screen can be an arbitrary discontinuous function, we see no benefits in using another kernel than the Epanechnikov.

3.5 Kernel width selection

Even more important than the kernel shape is a good selection of the kernel width or bandwidth. For every sample $X_{k,i}$ a choice has to be made for $h(X_{k,i})$. Two important factors that determine our heuristic for the kernel width are the density and weight of the samples, and the number of samples used in the estimation. We will discuss these in the next two subsections.

3.5.1 Density and weight of samples

In standard density estimation (see equation 1), where each sample has an equal weight, the density of the samples is directly related to the width of the kernel that should be used. A lower density requires a wider kernel in order to reduce variance to an even level over the domain. This is shown schematically in figure 1.

Looking at this figure another relation becomes obvious: where the estimated pdf $p(x)$ is large a narrow kernel is used and where $p(x)$ is small a wide kernel is used. This is equivalent to relation between the density of the samples and kernel widths, since the samples are distributed according to $p(x)$.

In [Abram82] Abramson suggests taking the kernel width $h(x_i)$ proportional to the *inverse square root of the estimated function value at the sample position*:

$$h(x_i) \sim (p(x_i))^{-\frac{1}{2}} \qquad (5)$$

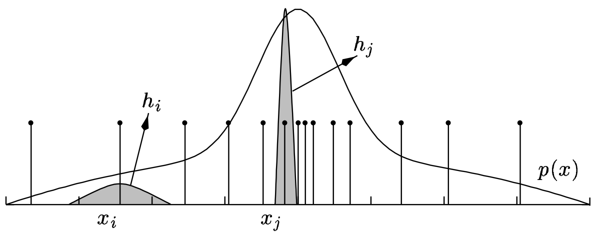

Figure 1: Variable kernel density estimation: the chosen kernel width is related to the density of the samples, which corresponds to a dependency on the target function $p(x)$. Where $p(x)$ is small a wider kernel is used.

This choice, as Abramson proves, has some desirable bias reduction properties. Of course $p(x)$ is to be estimated and therefore not known, so that a pilot estimate for $p(x)$ is required. (Our approach to this problem will be explained in section 4.)

As said, for the image synthesis problem, we estimate a function $f(x)$ using several pdf's $P_k(X)$ and this leads to a different weight $G_{k,i}$ for each sample $X_{k,i}$. The value of G is determined by how good the used importance sampling pdf's fit the function $F(X)$. A big value for G means that W/P is small, but that F is large for that sample. It is well known that bad importance sampling can be a terrible source of noise in the image. Ideally W/P is proportional to F so that every sample has an equal weight. As a result a bigger value of $G_{k,i}$ requires a bigger kernel, as this can be seen as a 'bad' sample. So also for our adapted density estimation framework a similar kernel width dependency holds between sample value and estimated function as shown in figure 2.

We propose an adapted version of the heuristic of Abramson in order to accommodate unequally weighted samples:

$$h(X_{k,i}) \sim \left(\frac{G_{k,i}}{f(x_{k,i})}\right)^{\frac{1}{2}} \tag{6}$$

Note that filling in this dependency into equation 3, shows that the top value of the kernel becomes independent of the weight $G_{k,i}$. Also note that the target image plane function $f(x)$ is used as reference (and not $F(X)$, which we can evaluate exactly), where $G_{k,i}$ acts as a one sample estimate of $f(x)$.

3.5.2 Number of Samples

The kernel width should also depend on the number of samples used. As this number increases the kernel width should go to 0 in order to reduce bias in the solution. Asymptotically for ($N \rightarrow \infty$) a perfect solution can be obtained.

For fixed kernel widths an optimal N dependency can be derived with regard to a mean squared error

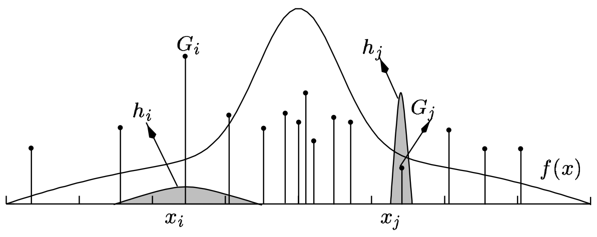

Figure 2: Variable kernel density estimation: for unequal sample weights the chosen kernel width can still be related to the target function $f(x)$. Large sample values with respect to $f(x)$ require wider kernels.

metric [Silve86]:

$$h \sim \left(\frac{1}{N}\right)^{\frac{1}{5}} \tag{7}$$

In graphics however a *square root dependency* has been used more often [Shirl95, Walte97].

In our experience the first rule tends to overblur discontinuities in the image, for which the eye is quite sensitive, while the second rule leads to noticeable variance even for a high number of samples. Several in-between exponents were tried and for the moment we will denote the dependency by a general exponent α.

3.5.3 Final heuristic

We chose the following final heuristic for the kernel width:

$$h(X_{k,i}) = C \cdot \left(\frac{1}{N}\right)^{\alpha} \cdot \left(\frac{G_{k,i}}{f(x_{k,i})}\right)^{\frac{1}{2}} \tag{8}$$

The constant C is a reference kernel width that defines a base value for how much samples should be spread out.

4 THE ALGORITHM

In this section we will give a detailed overview of the complete filtered image synthesis algorithm.

The basis of the algorithm is tracing (bidirectional) paths and for each path that has a contribution to the image plane, an appropriate kernel width is determined. This sample is then 'splatted' into an image buffer using that kernel.

To determine the kernel width for the samples, we need to know an approximate value for the image plane function $f(x)$ (see equation 8). The proposed

```
• Trace Paths ($N_0$ samples/pixel) and store screen hits ($x_i$, $G_i$)
• Image $f'_0$ = Fixed width kernel density estimation (width: $C' \cdot N_0^{-1/2}$) using stored hits
• Image $f_0$ = Variable width kernel density estimation (width: eq. 8) using $f'_0$ as reference
• Dispose stored hits
• $N_{total} = N_0$
• **For** $j = 1$ **to** number of iterations J
    – Choose $N_j$
    – $N_{total} += N_j$
    – $f_j = \frac{N_{total} - N_j}{N_{total}} \cdot f_{j-1}$ (Scale down)
    – $f_j +=$ Trace paths ($N_j$ samples/pixel) and apply var. kernel using $f_{j-1}$ as reference
• $f_J$ is the final image
```

Figure 3: Overview of the progressive filter & render algorithm

algorithm uses a number of iterations and in each iteration j an new approximation $f_j(x)$ (an image) is made using f_{j-1} as a reference for kernel width determination.

To start up the algorithm, an initial batch of paths (N_0 samples per pixel[2]) are traced and the hits on screen are all stored. Only storage of screen position $x_{k,i}$ and sample weight $G_{k,i}$ is necessary. Using these hits a first approximation f'_0 is made using *fixed kernel* density estimation (as in [Abram82]) with a kernel size $h = C' \cdot N_0^{-1/2}$. The constant C' should be large enough to eliminate most of the variance in the image. We use values so that the kernel would have a diameter of 8-10 pixels in the image (for $N_0 = 1$).

Using the same stored samples, a second approximation f_0 is constructed using variable kernel widths, using f'_0 as the reference image. In our implementation the constant C in this estimate is set to get a spread of 4-8 samples per pixel (when $N_0 = 1$ and $G/f = 1$). The value of C can be varied to exchange bias for variance and vice versa. It stays constant during all subsequent iterations.

[2]We use 'samples per pixel' for N appearing in the kernel width heuristic. Normally one would use total number of samples as in the density estimator (3) but the difference is only a constant factor (number of pixels in the image) and we have moved it to the the constant C. This allows us to specify C in terms of a number of pixels.

If no precautions are taken, the kernel widths h can become so narrow that they would not cover any pixel centre at all, making no contribution to the image. Although in theory a valid solution is still obtained, we have chosen to restrict the kernel width to a minimum of one screen pixel. Note that in standard bidirectional path tracing (BPT) this would correspond to using an one pixel wide Epanechnikov kernel placed in the pixel centre as a weighting function for samples going through this pixel[3].

An overview of the complete algorithm is given in figure 3. Note that it is a relatively simple procedure and that only the first batch of samples must be stored. This in contrast to other density estimation approaches in graphics [Shirl95, Myszk97] where large numbers of hits had to be stored.

5 IMPLEMENTATION & RESULTS

We have implemented the algorithm in RenderPark [Bekae99] using bidirectional path tracing (BPT) for computing the light transport. The method was tested on a scene containing many different illumination features including diffuse, glossy and specular surfaces, direct and indirect caustics, glossy reflections, ...

We have included rendered images of this scene made with the following specs:

- Figure 4: Standard BPT using 4 samples per pixel.
- Figure 5: Standard BPT using 64 samples per pixel. Note the spike noise on the left lighting fixture an in the neighbourhood (reflection from the same fixture).
- Figure 7: The new algorithm using 4 samples per pixel, with exponent $\alpha = 1.5/5$, fixed width constant $C' = 8$ pixels, $N_0 = 1$ sample per pixel, $N_j = 2N_{j-1}$, variable width constant $C = 4$.
- Figure 8: New algorithm using 64 samples per pixel and the same parameters as in the previous figure.
- Figure 6: A 'reference' image rendered with BPT using 2048 samples per pixel. Note that even in this image some noise around the left lighting fixture is present. This noise is in fact caused by highly glossy reflection of the left caustic by the fixture, something that is very hard to compute using BPT.
- Figure 9: Magnifications of a certain part of the images in order to reveal more detail.

[3]Currently we are using a simple box filter in our standard BPT renders

Performance wise the filtered renderings take about 10 to 15 percent more time than the standard bidirectional path tracing images. The 64 samples per pixel filtered image took about 1 hour to compute on an SGI Octane 195Mhz R10000.

Looking at the quality of the images following remarks can be made:

- Distracting spike noise present in the BPT images (even at 2048 samples/pixel) is effectively reduced using the new algorithm. Note especially the scattered yellow spots that originate from reflection of the left lighting fixture.
- Difficult illumination features (e.g. left light fixture and bottom-left glossy reflection of the caustic) appear very noisy in the BPT images. Using the new algorithm these features are adequately blurred so that they look more visually pleasing. However they tend to be overblurred if compared to the reference image, which is another example of bias versus variance trade-off. The overblurring occurs because the used approximate references f_{j-1} underestimate the actual radiance value. Using additional information like surface reflectance or normals, it might be possible to use better (irregularly) shaped kernels to lower this overblurring.
- One of the difficulties in choosing the parameters of the algorithm was preventing excessive blurring of edges. We found that the parameters had to be chosen so that still a small amount of variance (low frequency noise) was allowed in smooth regions, otherwise edges would be too blurry. A promising extension would be to prevent edge blurring by modifying the kernel shape, restricting it at object boundaries.
- Note that the (edge) blurring clearly diminishes as N goes up from 4 to 64 samples.
- A typical problem in density estimation is *boundary bias* that occurs at the boundaries of the domain (screen borders in our case). The cause of this bias is that at the border only samples contribute from one side of the border since samples outside the domain are not generated but could contribute to pixels inside the border. This causes darkening at the edges of the image (slightly visible figure 7). Currently we do not prevent this bias, but well known remedies like kernel mirroring can be easily applied here.

6 CONCLUSION

We presented an adaptive filtered Monte Carlo rendering method. A kernel filter is applied to individual samples using an adaptive kernel width based on a ’goodness’ criterion of the sample. The method is energy preserving and results in a progressive algorithm that still converges to a correct solution.

Results show that a reasonable trade-off between bias and noise can be maintained at any time. As desired difficult illumination features (for the used MC method), that normally result in a high noise level, are blurred more than easy to compute features.

Still many possible extensions and open problems need to be further researched.

- The boundary bias problem at the edges of the image should be solved. This is however more of an implementation issue rather than a fundamental problem.
- Some parameters (C, C', α) in the algorithm need to be configured by hand. An automatic selection is desirable but this is not an easy problem to solve. The results we obtained however were not dramatically dependent on the choice of parameters.
- Currently we are using only the sample values and their impact point on screen for determining the kernel. Other easy to acquire information from the scene or paths could also be used to enhance the images. Some ideas are: using non circular kernel footprints that depend on the surface normals, restricting kernels to one object or surface to prevent edge blurring, ...

These extensions and problems are currently under investigation.

Acknowledgements

Many thanks must go to Olivier Ceulemans for borrowing some of his masters thesis’ code on density estimation.
This research was supported by FWO Grant #G.0263.97

REFERENCES

[Abram82] Ian S. Abramson. On bandwith variation in kernel estimates, a square root law. *The Annals of Statistics*, 10(4):1217–1223, 1982.

[Bekae99] Ph. Bekaert and F. Suykens. *RenderPark, a physically based rendering tool*. K.U. Leuven, http://www.cs.kuleuven.ac.be/∼graphics/, 1999.

[Dutre93] P. Dutre, E. Lafortune, and Y. D. Willems. Monte Carlo Light Tracing with Direct Pixel Contributions. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 128–137, Alvor, Portugal, December 1993.

[Heckb90] Paul Heckbert. Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, volume 24, pages 145–154, August 1990.

[Jense95] Henrik Wann Jensen and Niels Jorgen Christensen. Optimizing Path Tracing Using Noise Reduction Filters. In V. Skala, editor, *Proceedings of the Winter School of Computer Graphics and CAD Systems '95*, Plzen, Czech Republic, February 1995. University of West Bohemia.

[Jense96] Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 21–30, New York, NY, 1996. Springer-Verlag/Wien.

[Lafor93] Eric P. Lafortune and Yves D. Willems. Bi-directional Path Tracing. In H. P. Santo, editor, *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal, December 1993.

[Lee90] Mark E. Lee and Richard A. Redner. A note on the use of nonlinear filtering in computer graphics. *IEEE Computer Graphics and Applications*, 10(3):23–29, May 1990.

[McCoo99] Michael D. McCool. Anisotropic diffusion for monte carlo noise reduction. *ACM Transactions on Graphics*, to appear 1999.

[Myszk97] Karol Myszkowski. Lighting reconstruction using fast and adaptive density estimation techniques. In Julie Dorsey and Phillip Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 251–262, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.

[Rushm94] Holly E. Rushmeier and Gregory J. Ward. Energy preserving non–linear filters. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 131–138. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[Shirl95] Peter Shirley, Bretton Wade, Philip M. Hubbard, David Zareski, Bruce Walter, and Donald P. Greenberg. Global Illumination via Density Estimation. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 219–230, New York, NY, 1995. Springer-Verlag.

[Silve86] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapmann and Hall, New York, NY, 1986.

[Veach94] Eric Veach and Leonidas Guibas. Bidirectional Estimators for Light Transport. In *Fifth Eurographics Workshop on Rendering*, pages 147–162, Darmstadt, Germany, June 1994.

[Veach95] Eric Veach and Leonidas J. Guibas. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Computer Graphics Proceedings, Annual Conference Series, 1995 (ACM SIGGRAPH '95 Proceedings)*, pages 419–428, 1995.

[Walte97] Bruce Walter, Philip M. Hubbard, Peter Shirley, and Donald P. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics*, 16(3):217–259, July 1997.

[Walte98] Bruce Johnathan Walter. *Density Estimation Techniques for Global Illumination*. PhD thesis, Program of Computer Graphics, Cornell University, Ithaca, NY, August 1998.

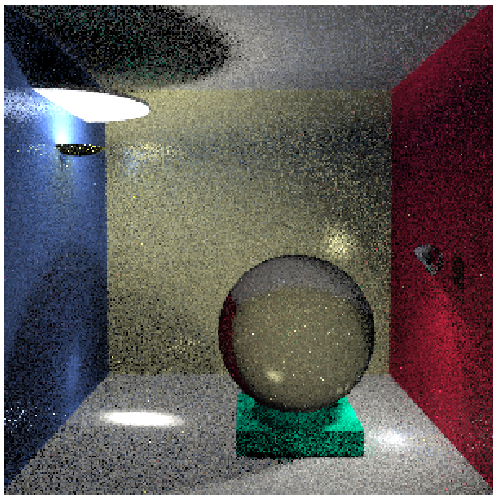
Figure 4: Standard bidirectional path tracing, 4 samples/pixel

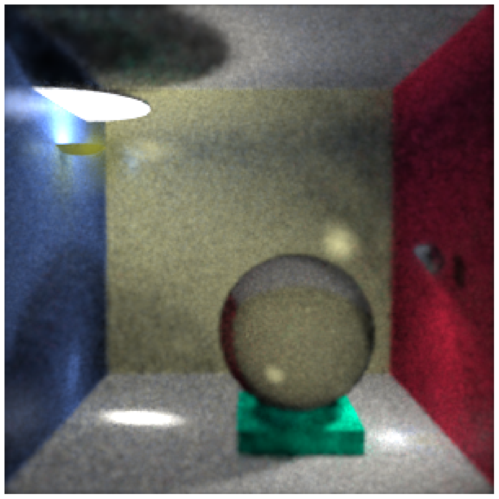
Figure 7: New progressive adaptive filter algorithm, 4 samples/pixel

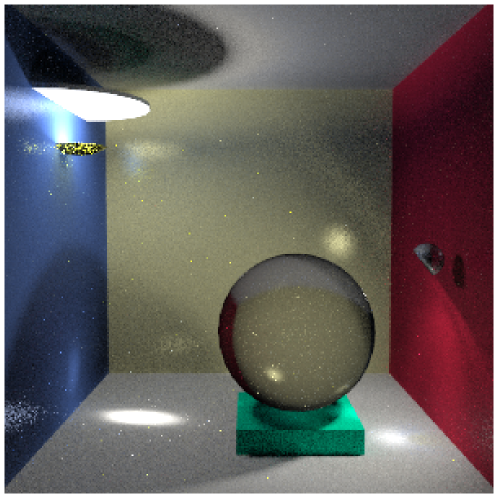
Figure 5: Standard bidirectional path tracing, 64 samples/pixel

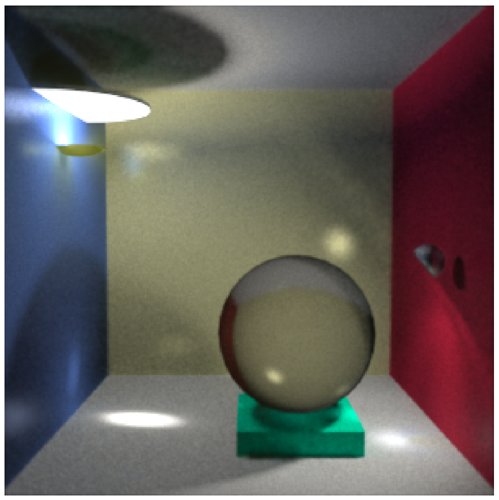
Figure 8: New progressive adaptive filter algorithm, 64 samples/pixel

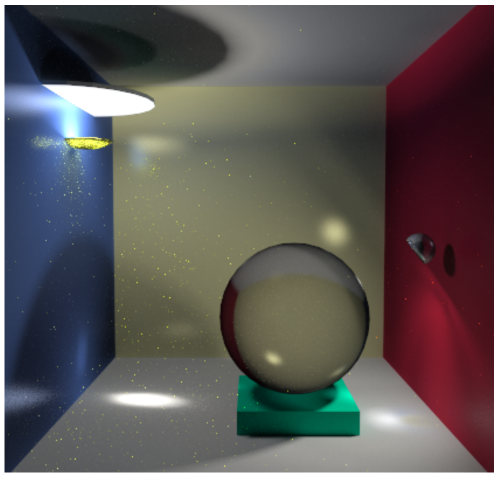
Figure 6: Reference: bidirectional path tracing, 2048 samples/pixel

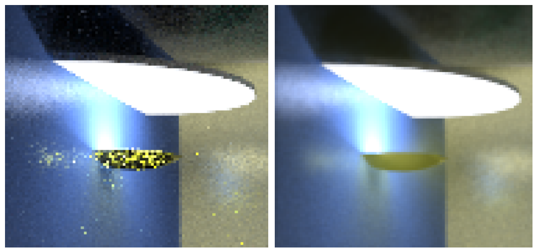
Figure 9: Magnifications of figure 5 (left) and figure 8 (right)