The problem of Monte Carlo rendering noise

Abstract

When rendering using Monte Carlo methods, either a large amount of samples are necessary or noise will be present in the image. A lot of methods have already tried to tackle this problem including adaptive sampling, reconstruction techniques and advanced image filtering techniques. In this paper I will recapitulate the work I have done for my thesis so far. The focus will be on the RPF algorithm called random parameter filtering (RPF) as proposed in [Sen and Darabi 2011b]. As this algorithm is based on a cross bilateral filter, this concept will be explained and discussed as well. Other methods that handle the noise caused by Monte Carlo rendering at low sampling rates were investigated as well and discussed as Related Work.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing;

Keywords: monte carlo rendering, filter, distribution effects, global illumination

1 Introduction

Computer Graphics have evolved from 2D to almost realistic 3D during the last decennia. To reach this realism in graphics, Global Illumination algorithms became very important as these incorporate the indirect light caused by the surroundings. These algorithms require large numbers of calculations and thus a large rendering time, especially when results without noise are expected. To lower this rendering time, Monte Carlo methods are often used because they can be used to evaluate multidimensional functions in an unbiased way. Complex and photorealistic images can be created using these Monte Carlo methods. In the case of (Monte Carlo) rendering, the multidimensional integrals that needs to be solved at every pixel of the image are: integration of the radiance over the aperture of the camera (u,v), integration over the time the shutter is open ($t_0 \rightarrow t_1$) and over the area of the pixel $((i-1/2, j-1/2) \rightarrow (i+1/2, j+1/2)$:

$$I(i,j) = \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \cdots \int_{-1}^{1} \int_{-1}^{1} \int_{t_0}^{t_1} f(x, y, \cdots, u, v, t)\, dt\, dv\, du\, dy\, dx.$$

As Monte Carlo methods take random samples to evaluate a function, a lot of samples are usually necessary to evaluate the function precisely and thus a lot of noise will be present when using a low number of samples. One obvious solution to this problem is to use more samples, but as the rendering time increases dramatically with the amount of samples this is not always an option. Adaptive sampling algorithms offer a way to distribute the available numbers of samples in the best possible way across the image. This can be done by allocating more samples to regions with difficult light effects. Reconstruction techniques take the image with a fixed amount of samples and try to use these available samples as much as possible, across pixels and even across frames. Another technique that can be used is the focus of this paper and is basically an advanced image filter, it uses not only the color computed at each sample of the image to filter it, but also other parameters that are available during the rendering process.
In the related work section I will describe all the different related papers I researched before deciding to research the RPF algorithm in more detail. The subsequent sections describe the concept of bilateral filter after which the RPF algorithm is explained in detail. To conclude the results describe my current results followed by a discussion of these results and a conclusion.

2 Related Work

More information about Monte Carlo methods can be found here [Kalos and Whitlock 2009], while more information about advanced rendering can be found here [Dutré et al. 2006] and here [Pharr and Humphreys 2010]. Some recent work that try to tackle the problem of noise at low sampling rates when using Monte Carlo rendering are discussed below.

Adaptive Rendering with Non-Local Means Filtering:
The following algorithm found in [Rousselle et al. 2012] describes an adaptive sampling algorithm for Monte Carlo rendering. The first step of the algorithm distributes a given budget of samples over the image. In the second step the image is filtered with a variant of the NL-means filter which is a generalisation of the bilateral filter. This filter considers distances between pairs of pixel values to compute filter weights, the term non-local is caused by the fact that the set of samples that contribute to one output pixel can come from a large region in the input image. In the third step of the algorithm the remaining error in the filtered image is estimated to drive the adaptive sampling in the next iteration step.

Multidimensional Adaptive Sampling and Reconstruction for Ray Tracing:
A combination of adaptive sampling and reconstruction is proposed in [Hachisuka et al. 2008]. This paper introduces a new sampling strategy for ray tracing. The samples on which the strategy operates are generated from the rendering equation directly and are thus not generated through Monte Carlo sampling. Additionally this algorithm uses all previously generated samples to generate a new sample. After sampling this high-dimensional function, a reconstruction is made by integrating the function over all but the image dimensions.

SURE-based Optimization for Adaptive Sampling and Reconstruction:
A similar goal is aimed for by [Li et al. 2012] allthough samples are distributed over the image based on an estimator of the Mean Squared Error (MSE) of the image. The estimator that is used is called Stein's Unbiased Risk Estimator (SURE), it allows to estimate the error of an estimator without knowing the true value that is estimated. A difference with most other algorithms using a filter is that a filterbank is used instead of a single filter, any kind of filter can be added to this filterbank. The authors of the paper have experimented with isotropic Gaussian, cross bilateral and a modified non-local means filter. The algorithm itself goes as follows: A small number of initial samples are rendered first, followed by filtering each pixel with all filters in the filterbank. Next the filtered color with the lowest SURE error is chosen for each pixel and is used in the reconstruction. When more samples are available and thus yet to be distributed over the image, these are used for the pixels with the largest SURE errors after which the

process goes back to filtering each pixel with all the filters from the filterbank.

Adaptive sampling and reconstruction using greedy error minimization:
Another algorithm that does not fix which filter is used for different pixels is described in [Rousselle et al. 2011]. As this algorithm is greedy, it minimizes the function at each stage hoping to reach a global optimum. Given a current sample distribution, a filter that minimizes the pixel error is selected from a discrete set of filters that can be different for each pixel. Given the filter selection, addition samples are distributed to further reduce the MSE. Because the MSE cannot be calculated exactly, the change in MSE is calculated instead. This whole process can be repeated until any chosen termination criterion is met.

Temporal Light Field Reconstruction for Rendering Distribution Effects:
A general reconstruction technique that tries to maximize the image quality based on a given set of samples is given in [Lehtinen et al. 2011]. This technique exploits the dependencies in the temporal light field and allows efficient reuse of samples between different pixels. The effective sampling rate is multiplied by a large factor when using this technique. The paper makes the following four contributions:

1. ***The reconstruction algorithm***
   The input of the algorithm contains a set of samples in the form:
   $$s = \{(x, y, u, v, t) \to (z, v, L)\} \tag{1}$$
   With xy the screen coordinates, uv the lens coordinates, t the time, z the depth, v the 3D motion vector and L the radiance associated with the input sample.
   The reconstruction algorithm then goes as follows:
   (a) The screen coordinates of the input samples are reprojected to the (u,v,t) coordinates of the reconstruction location. Samples that are too far away from the reconstruction location in xy are discarded.
   (b) The returned clusters of samples are grouped into apparent surfaces.
   (c) If multiple apparent surfaces are found, they are sorted front-to-back. Afterwards, it is determined which one covers the reconstruction location.
   (d) The output color is computed by filtering the samples that belong to the covering surface using a circular tent filter.
2. ***Determining visibility consistency***
   By using the key observation that the relative ordering of the sreen positions of samples from a non-overlapping surface never changes under reprojections, they derive a formal criterion named SAMESURFACE to detect when a set of reprojected samples should be filtered together.
3. ***Resolve visibility without explicit surface reconstruction***
   Which surface is visible at the reconstruction location is determined in this technique. The challenge is to distinguish between small holes in the geometry and apparent holes caused by stochastic sampling. To do this a radius R is precomputed, R is the radius of the largest empty circle that is expected to be visible on the xy plane after reprojection. Holes that are smaller than R should be filled because it is beyond the resolution of the input sampling. The visibility is then determined using the following rule. A reconstruction location is covered if it is possible to find three reprojected input samples that form a triangle that covers the reconstruction location and fits inside a circle of radius R.
4. ***Hierarchical query structure***
   The reconstruction algorithm needs to retrieve the input samples that reproject to the vicinity of reconstruction locations (x, y), given (u, v, t) quickly. The input samples are organized into a bounding volume hierarchy, where the extents of the nodes xy are parameterized using u, v and t. When executing a query, the parameterized bounding volume is used to test if the reconstruction location is inside the bounds.

Reconstruction the Indirect Light Field for Global Illumination:
The algorithm that is described in [Lehtinen et al. 2012] is similar to the former algorithm and is published by almost the same authors. The paper also describes a general reconstruction technique that exploits anisotropy in the light field and permits efficient reuse of input samples between pixels or world-space locations, multiplying the effective sampling rate by a large factor. The main difference between the two algorithms is that this algorithm tries to reconstruct the indirect light field at scene point instead of at points on the lens. That is also why reconstructing an image using this algorithm takes three to four times longer than the former algorithm. The former algorithm uses a 2D hierarchy, but this algorithm needs to reconstruct the incident light field at arbitrary points in the scene, so a true 3D algorithm is needed.

Axis-Aligned Filtering for Interactive Sampled Soft Shadows:
The post-processing step needed in Light Field Reconstruction techniques is very expensive. Other algorithms like the one proposed in [Mehta et al. 2012] has a very simple filtering step by using axis-aligned filters. Because of this extremely simple step, this algorithm can be integrated in real-time raytracers. Adaptive filtering is used because the parameters of the used filters are adjusted depending on the input samples. This algorithm is basically an image filter for noise that is fixed on the soft shadows effect.

High-Quality Spatio-Temporal Rendering using Semi-Analytical Visibility:
A visibility technique with the only purpose to render motion blur with per-pixel anti-aliasing is described in [Gribel et al. 2011]. A number of line samples are used over a rectangular group of pixels that form a 2D spatio-temporal visibility problem together with the time dimension. This problem needs to be solved per line sample. Each group of pixels in the image is rendered separately. For each group a Bounding Volume Hierarchy is used to receive only the geometry that overlaps with the tile. Furthermore each line sample is processed one at a time by calculating what triangles intersect with the sample followed by resolving the depth visibility. When all the triangles have been processed, the final visibility is resolved and for each pixel the contribution of the line samples overlapping the pixel is accumulated to the color of that pixel.

A Theory of Monte Carlo Visibility Sampling:
[Ramamoorthi et al. 2012] tries to lower the amount of noise introduced by Monte Carlo sampling of the visibility term in the rendering equation. By analysing the effectiveness of different non-adaptive Monte Carlo sampling patterns for rendering soft shadows, they search for the pattern with the lowest expected variance for a certain visibility function. These results can lead to a reduction in the needed number of shadow samples without losing

precision by using the best sampling pattern.

Compressive estimation for signal integration in rendering: To conclude with, Monte Carlo methods are not the only methods around that can compute an integral of an unknown function, in [Sen and Darabi 2010] a method like this is presented and applied to anti-aliasing and motion blur effects. This is possible because of the theory of compressed sensing, which allows to reconstruct a signal from a few linear samples if it is sparse in a transform domain. The advantage of this function over Monte Carlo methods is that is needs only a few samples to estimate the function.

3 Bilateral Filter

The Bilateral filter is introduced in [Tomasi and Manduchi 1998], some information can also be found in [Devarajan and Nyikal 2013].
As the name already suggests, a bilateral filter filters an image using two different inputs, the position of pixels and its intensity value. The most important property and reason why the bilateral filter is used very often is that it preserves the edges in the image. To understand the working of a bilateral filter, the concept of a Gaussian filter is explained first.

3.1 Gaussian filter

As an image has two dimensions, the 2D version of the gaussian filter will be explained, in fact this is simply the product of two one dimensional gaussians, one per direction. The equation of a 2D Gaussian filter is defined as:

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

with x and y being the distance from the origin and σ being the standard deviation of the Gaussian distribution. The shape of a Gaussian filter being a 2D bell shaped function.
When an image is filtered using a Gaussian filter, this will lead to blurred images as the values of each pixel will fade into the values of the surrounding pixels.

3.2 Bilateral filter as two Gaussian filters

A bilateral filter can be explained as being two Gaussian filters that filter in the spatial (the domain filter D) and in the intensity domain (the range filter R) respectively.

$$\begin{aligned} D(x,y) &= e^{-\frac{x^2+y^2}{2\sigma^2}} \\ R(p_i,p_j) &= e^{-\frac{f(p_i)^2+f(p_j)^2}{2\sigma^2}} \end{aligned} \tag{3}$$

The domain filter uses the difference between the positions of the two pixels in both the horizontal direction (x) and the vertical direction (y). The range filter uses the intensity value (obtained by the image function f) of both pixels.
We can now write down the equation to compute final value of a pixel filtered using a bilateral filter.

$$\begin{aligned} b(p_i) &= k^{-1} \sum_{m=0}^{N-1} f(p_m) D(x,y) R(p_i,p_m) \\ &\text{with} \\ k &= \sum_{m=0}^{N-1} D(x,y) R(p_i,p_m) \end{aligned} \tag{4}$$

x and y are again the difference in the positions of the pixels p_i and p_m and N is the size of the neighboorhood taken into account for each pixel. The value of the pixel is multiplied with the two filter weights and then summed over all the samples in the neighboorhood, the k^{-1} term makes sure the filter does not introduce more intensity into the image. A nice property of the bilateral filter is that when the variance of the range filter is set to infinity, the bilateral filter acts like a Gaussian filter.

3.3 Cross Bilateral Filter

The RPF algorithm that is described in the following section, is based on a more advanced version of a bilateral filter, called the cross bilateral filter. A Cross bilateral filter does not only filter in the spatial and the intensity domain but adds another filtering function to the filter. This added function is a function just like the image function and returns a value for each position in the image. An example of this function are the world position at the first intersection point of the ray traced through that pixel, allthough other features like the normal are also suitable.

4 Random Parameter Filtering algorithm

4.1 General information

The algorithm Random parameter filtering introduced in [Sen and Darabi 2011b] tries to reduce the noise caused by the random parameters used in Monte Carlo integration. This noise reduces when the sampling rate increases, but this also increases the rendering time which is often not desirable. A Monte Carlo rendering system calculates the colors of the pixels using a function like this:

$$c_i \Leftarrow f(p_{i,1}, p_{i,2}; r_{i,1}, r_{i,2}, \cdots, r_{i,n})$$

with f being the function of the scene with inputs p_i being the position of the sample and the random parameters r_i. As the RPF algorithm is a post-process filter and uses a lot more than only the colors of the samples to filter the image, the following data has to be collected by the rendering system and given to the RPF algorithm as input:

$$x_i = \{p_{i,1}, p_{i,2}; r_{i,1}, \cdots, r_{i,n}; f_{i,1}, \cdots, f_{i,m}; c_{i,1}, c_{i,2}, c_{i,3}\}$$

For each sample a vector like this is stored, containing the position of the sample p_i, the random parameters r_i, needed to compute the sample colors c_i and the scene features f_i. These scene features are the extra features on which the cross bilateral filter also filters, in particular the features that are used are the world-space position, normal and texture values for the first intersection point of the ray and the world position and normal for the second intersection when using path tracing. When a feature is not available for a certain sample it is replaced with a zero.

The algorithm itself computes the statistical dependency between the outputs and the inputs of the rendering system. This information

is then used in the filter to adjust the importance of the sample value (i.e. the filter weight).
An example about an image that is rendered with depth of field can clarify this process: regions that are in focus are free of noise because a ray that goes through the lens from a focused point on the image will land on the same point in the scene, regardless of where on the lens the random point is located. For pixels out of focus on the other hand, the color of the sample depends on the random lens position.

4.2 Implementation of the algorithm

A detailed description on how the RPF algorithm is implemented can be found in [Sen and Darabi 2011a]. The algorithm consists of three big parts which will be explained seperately.

Algorithm 1 RPF algorithm

```
filterSize[] = $\{55, 35, 17, 7\}$
for $t = 0$ to $filterSize.length$ do
    for all pixels P in image I do
        Preprocess samples in neighboorhood N of P based on filterSize[t]
        Compute statistical dependencies of sample color and features on inputs $p_i$ and $r_i$, use them to compute filter weights.
        Filter sample colors in P using the weighted cross bilateral filter.
    end for
end for
Box filter all samples of each pixel to compute the final pixel color
return filtered image
```

4.2.1 Multiple iterations

While a filter with a neighboorhood size of the entire image would be desirable, this is not feasible because of the computation overload (to filter one sample all the samples in the neighboorhood have to be used). To solve this problem multiple iterations with different neighboorhood sizes are used. By experimenting, the authors found four iterations with the following sizes to be sufficient: 55,35,17 and 7. By going from a large to a small size, the low frequency noise is filtered first followed by filtering more and more detail noise.

4.2.2 Preprocess samples in Neighboorhood N of P based on the filtersize

In this part of the algorithm a vector of samples is created to use in the computation of the filter weights. Because the number of samples increases as $O(N^2)$ with the block size, a smaller number of random samples are chosen from within the neighboorhood. These samples are then clustered, which means that the random sample is only added to the neighboorhood if all of its scene features are within three standard deviations of the mean for the pixel. This test is only done when the standard deviation of that scene feature for the pixel is bigger than 0.1, this makes sure that not all samples are thrown away when the variance is very small.
Before the statistical dependencies are computed for this newly created neighboorhood, all the elements in the sample vector are normalized by removing the mean and deviding by the standard deviation of all samples in the neighboorhood. This makes sure that all features get an equal weight when calculating dependencies.

4.2.3 Compute statistical dependencies and the filter weights

First Mutual information is explained as this technique is used to calculate the statistical dependencies between a sample feature and the inputs to the Monte Carlo system.

Mutual Information The technique of mutual information is described in [Cover and Thomas 2006] as a measure of the amount of information that one random variable contains about another random variable. The mutual information of two random variables X and Y with a joint probability mass function $p(x, y)$ and marginal probability mass function $p(x)$ and $p(y)$ is given by:

$$I(X,Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)log\frac{p(x,y)}{p(x)p(y)}$$

To implement the computation of the mutual information between two vectors, the histogram of each of them as well as their joint histogram is computed. The marginal histograms are computed by making all the values positive first by subtracting the minimum element. The elements are quantized to integers next. The number of times a value occurs is then counted after which these values are divided by the length of the vector. The joint histogram is implemented very similarly based on the implementation found in [Peng 2007] (again using the positive and quantized vectors). The number of times each couple of values occurs is calculated using a 2D array after which each element is divided by the length of the vectors.

Computing the filter weights Two vectors with filter weights are computed by the algorithm, α and β. α contains one value for each color channel which are the weights used for the color term. β contains one value for each scene feature which are the weights used for the feature term. After the computation of the dependencies of the colors of the samples with the random parameters, the positions and the scene features, the filter weights that are used for the color values can be computed with the following equations:

$$\begin{aligned}\alpha_k = &\quad max(1 - 2(1 + 0.1t)W^r_{c,k}, 0)\\ \beta_k = &\quad W^{f,k}_c max(1 - (1 + 0.1t)W^r_{f,k}, 0)\end{aligned} \tag{5}$$

with $W^r_{c,k}$ the fractional contribution of the random parameters on the k^{th} color channel, t the number of the block size, $W^r_{f,k}$ the fractional contribution of the random parameters on the k^{th} scene feature and $W^{f,k}_c$ the fractional contribution of the k^{th} scene feature on the color. The normalized dependency on the random parameters gets more weight as the block size goes down with each iteration.

4.2.4 Filter the samples using the weighted cross bilateral filter

After the computation of the filter weights, the filtering of the samples is done simply using algorithm 2

$$\begin{aligned}w_{ij} =exp &\quad (-\frac{1}{2\sigma_c^2}\sum_{1\leq k\leq 3}\alpha_k(c_{i,k} - c_{j,k})^2)*\\ exp &\quad (-\frac{1}{2\sigma_f^2}\sum_{1\leq k\leq m}\beta_k(f_{i,k} - f_{j,k})^2)\end{aligned} \tag{6}$$

Algorithm 2 Filter the samples

```
for all samples i in P do
    $c_i^{''} \leftarrow 0, w \leftarrow 0$
    for all samples j in N do
        Calculate $w_{ij}$ with equation 6
        $c_i^{''} \leftarrow c_i^{''} + w_{ij} c_i^{'}$
        $w \leftarrow w + w_{ij}$
    end for
    $c_i^{''} \leftarrow \frac{c_i^{''}}{w}$
end for
return filtered color of the samples $c^{''}$
```

Because the samples in the neighboorhood are selected using a Gaussian distribution the position term of the bilateral filter could be dropped. The variances of the filter are:

$$\begin{aligned} \sigma_c^2 = \sigma_f^2 = & \quad \frac{\sigma^2}{(1 - W_c^r)^2} \\ \text{with} & \\ \sigma^2 = & \quad 8\sigma_8^2/s \end{aligned} \tag{7}$$

with s the samples per pixel and σ_8^2 the experimentally selected variance of 0.02 for noisy scenes or 0.002 for all others.

5 Results

I implemented a bilateral filter algorithm in C++ at the start to get to know the concept and its details. In figure 1 an image that is unfiltered, one that is filtered with a Gaussian filter and another that is filtered with a bilateral filter, both created using my implementation, can be compared.

Afterwards I implemented the RPF algorithm in C++ also using the PBRT2 renderer from [Pharr and Humphreys 2010]. The algorithm is not implemented in its completeness as it is presented in [Sen and Darabi 2011a]. The following items are different or not implemented in my implementation:

- The algorithm does not handle sample spikes in the high dynamic range differently although in [Sen and Darabi 2011a] it is described how to do it.
- Instead of taking into account all the described scene features, the implementation only uses the normal and world-space coordinates of the first intersection of the ray.

6 Discussion

The reason this paper does not contain resulting images filtered with the RPF algorithm is that the current implementation is not working correctly yet due to an unknown reason. Another problem with the current implementation is that the memory management seems incorrect as the algorithm uses more memory than it should.

These problems will be taken care of during the rest of my thesis as well as an attempt to improve on the RPF algorithm or try to get it to filter out noise caused by probabilistic visibility. A comparison between this implementation and the full RPF algorithm or other algorithms is thus impossible as this implementation is not working correctly yet.

7 Conclusion

Different algorithms to solve the problem of noise in Monte Carlo rendering have been discussed in this paper. Random Parameter Filtering is different from most other algorithms that tackle this problem in the fact that it is a very general algorithm. The statistical dependency which is computed using the theory of Mutual information makes the algorithm useful for any kind of sample feature that can be found. As all the different algorithms show, this is a popular research topic in Computer Graphics at the moment, which makes us hope for even more genious algorithms that solve this problem.

References

COVER, T., AND THOMAS, J. 2006. *Elements of Information Theory*. Wiley.

DEVARAJAN, H., AND NYIKAL, H., 2013. Image scaling and bilateral filtering. `http://scien.stanford.edu/pages/labsite/2006/psych221/projects/06/imagescaling/index2.html`.

[Online, accessed 31 jan 2013].

DUTRÉ, P., BALA, K., AND BEKAERT, P. 2006. *Advanced global illumination*. Ak Peters Series. AK Peters.

GRIBEL, C. J., BARRINGER, R., AND AKENINE-MÖLLER, T. 2011. High-Quality Spatio-Temporal Rendering using Semi-Analytical Visibility. *ACM Transactions on Graphics, 30* (August), 54:1–54:11.

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 Trans. Graph. 27*, 3 (Aug.), 33:1–33:10.

KALOS, M., AND WHITLOCK, P. 2009. *Monte Carlo Methods*. Wiley.

LEHTINEN, J., AILA, T., CHEN, J., LAINE, S., AND DURAND, F. 2011. Temporal light field reconstruction for rendering distribution effects. *ACM Trans. Graph. 30*, 4.

LEHTINEN, J., AILA, T., LAINE, S., AND DURAND, F. 2012. Reconstructing the indirect light field for global illumination. *ACM Trans. Graph. 31*, 4 (July), 51:1–51:10.

LI, T.-M., WU, Y.-T., AND CHUANG, Y.-Y. 2012. Sure-based optimization for adaptive sampling and reconstruction. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2012) 31*, 6 (November), 186:1–186:9.

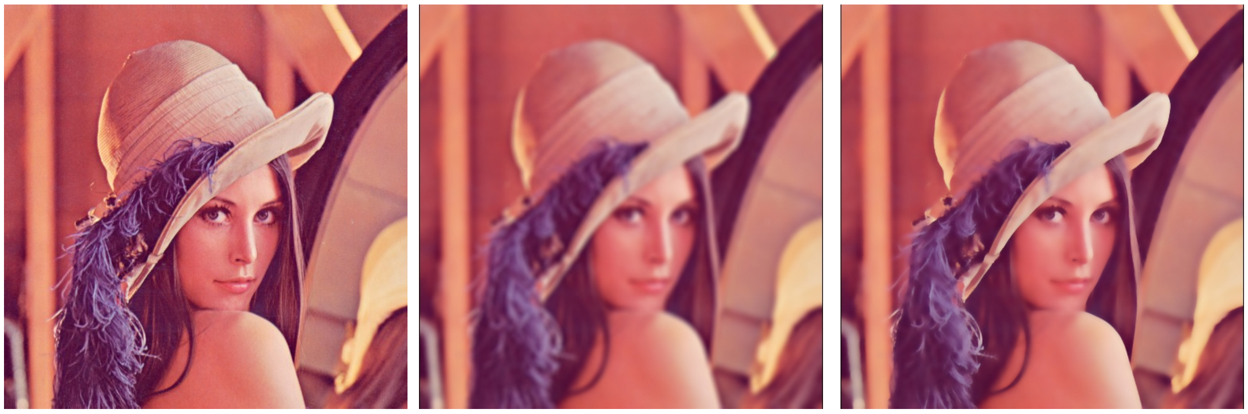
MEHTA, S., WANG, B., AND RAMAMOORTHI, R. 2012. Axis-aligned filtering for interactive sampled soft shadows. *ACM Trans. Graph. 31*, 6 (Nov), 163:1–163:10.

PENG, H., 2007. Mutual information computation. `http://www.mathworks.com/matlabcentral/fileexchange/14888`.

[Online, accessed 31 jan 2013].

PHARR, M., AND HUMPHREYS, G. 2010. *Physically Based Rendering: From Theory to Implementation*. The Morgan Kaufmann series in interactive 3D technology. Elsevier Science.

RAMAMOORTHI, R., ANDERSON, J., MEYER, M., AND NOWROUZEZAHRAI, D. 2012. A theory of monte carlo visibility sampling. *ACM Transactions on Graphics*.



(a) *The famous picture of a woman called called Lena.* **(b)** *Lena filtered using a Gaussian filter* **(c)** *Lena filtered using the bilateral filter*

Figure 1: *Pictures of Lena to compare a Gaussian and a bilateral filter, notice how the bilateral filtered image preserves the edges in the image*

ROUSSELLE, F., KNAUS, C., AND ZWICKER, M. 2011. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph. 30*, 6 (Dec.), 159:1–159:12.

ROUSSELLE, F., KNAUS, C., AND ZWICKER, M. 2012. Adaptive rendering with non-local means filtering. *ACM Trans. Graph. 31*, 6 (Nov.), 195:1–195:11.

SEN, P., AND DARABI, S. 2010. Compressive estimation for signal integration in rendering. *Computer Graphics Forum 29*, 4.

SEN, P., AND DARABI, S. 2011. Implementation of Random Parameter Filtering. Tech. Rep. EECE-TR-11-0004, University of New Mexico, May.

SEN, P., AND DARABI, S. 2011. On Filtering the Noise from the Random Parameters in Monte Carlo Rendering. *ACM Transactions on Graphics (TOG) to appear*.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, 839 –846.