Image Correction Sampling and Filtering for High-Quality Volumetric Rendering

Sunho Kim, Yo-Sung Ho
School of Electrical Engineering and Computer Science
Gwangju Institute of Science and Technology (GIST)
Gwangju, South Korea
{sunhokim, hoyo}@gist.ac.kr

Abstract — As high-quality volumetric rendering continues to be demanded, image correction has become an important issue. Here, the correction technique should not only increase accuracy, but also be efficient in terms of speed. Multiple sampling or reconstruction filtering techniques can be used to remove noises from rendering artifacts. In this paper, we analyze six multiple sampling and six reconstruction filtering techniques for correcting rendering results, and compare their performances. For sampling, we mainly use samplers based on the random number or the Quasi-Monte Carlo sequence. For the reconstruction filtering process, we employ typical filters used in image processing and some derived filters. Our experimental results show that the low discrepancy sampler is fast and optimal in terms of the PSNR performance, and the tent filter performs relatively fast and accurate filtering.

Keywords—Volumetric rendering, Ray tracing, Random sampling, Quasi-Monte Carlo, Reconstruction filtering

I. INTRODUCTION

For 3D surface rendering, volumetric rendering techniques, such as ray tracing and path tracing, can be used. These techniques first shoot light rays in the image space from the eye or camera’s field of view. Then, the intersection point between the ray and the 3D surface is obtained, and the color information at the intersection is projected into the image space. When the volumetric rendering is performed, generally only one ray is used per pixel.

In general, using only one ray per pixel results in many artifacts and noises. In order to reduce noises or artifacts, multiple rays per pixel can be used to perform the sampling. What we need to consider here is how to sample and how many supplemental rays should be used. It is important to use an efficient sampling sequence because a large number of supplemental rays can provide good results, but there is a problem of increased computational cost. The random number or the Quasi-Monte Carlo method [1] is mainly used as the sampling sequence. Multiple sampling result can be adjusted through the reconstruction filtering as a post-processing operation.

In this paper, we analyze various multiple sampling and reconstruction filtering methods for high-quality rendering. First, we analyze six sampling methods of creating supplemental rays through random-based or Quasi-Monte Carlo-based sequences and perform rendering. In order to correct rendering results, we test six reconstruction filtering methods that are widely used to refine the result of volumetric rendering and compare their experimental results.

II. MULTIPLE SAMPLING METHODS

A. Independent Sampler

Independent sampler, the most basic sample generator, simply generates independent and uniformly distributed pseudorandom numbers. Rendering results refined by this multiple sampler typically take a lot of computational time to converge, as no precautionary measures are taken to avoid sample chunks. When this sampler is rendered to multiple threads or machines, the order of the samples is affected by the operating system scheduler and initialized using a deterministic procedure.

B. Stratified Sampler

The stratified sampler, also known as a jittering, divides the domain into specific numbers of regular grids and generates samples within each grid. Here, the samples generated for each grid are not created at the center of the grid like regular samplers, but at random positions within the grid. A stratified sampler generally has fewer sample clumps and better convergence when compared to an independent sampler. Like independent samplers, however, multicore rendering can cause problems to generate different rendering results in the subsequent process due to non-determinism introduced by the operating system scheduler in general.
C. Halton’s Quasi-Monte Carlo Sampler

First, we need to understand the Quasi-Monte Carlo formula. Unlike the usual Monte Carlo integration formula, Quasi-Monte Carlo can be represented by:

\[ \int_{[0,1]^d} f(u)du = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \]  

(1)

where \( x_i \) is the \( i \)th point of a set of points \( x_1, x_2, ..., x_N \) and \( s \) is a dimension. The way to choose the \( x_i \) in the Quasi-Monte Carlo sampler is different from the conventional Monte Carlo sampling.

The Quasi-Monte Carlo number sequence is designed to reduce sample chunks in an integrated dimension, which can result in a higher convergence order in the rendering. Due to the feature of the deterministic sample, errors can occur as a form of grid or moiré patterns rather than random noise. Of course, the more number of samples can reduce errors.

Discrepancy of the Halton sequences [3] is defined by:

\[ D^*_N(x(1),...,x(N)) \leq C \frac{(\log N)^s}{N} \]  

(2)

where \( C \) is a constant and it depends only on arbitrary co-prime integers which is greater than 1. The Halton sequences cause problems to provide very high-quality point sets that are increasingly relevant at higher dimensions. To improve this problem, the Halton sequences are generally used in combination with scrambling permutations.

D. Hammersley’s Quasi-Monte Carlo Sampler

The Hammersley’s Quasi-Monte Carlo sampler [2] makes very similar form of sequence as the Halton sequence. This sampler produces a very high quality set of points and a sequence is slightly less discrepant and more regular. The discrepancy is defined by:

\[ D^*_N(x(1),...,x(N)) \leq C \frac{(\log N)^{s-1}}{N} \]  

(3)

Like the Halton sequence, we have to scramble points to reduce the pattern of higher dimensional correlations. This sampler produces an odd-looking intermediate result when combined with a rendering technique, such as a path tracing or ray tracing that traces the path originating from the light source. It also cannot be used together with some rendering algorithms such as bi-directional path tracing and energy redistribution path tracing because it does not work well.

E. Sobol’s Quasi-Monte Carlo Sampler

The Sobol sequence [4] provides a good set of points for relatively more efficient computations compared to Hammersley and Halton sequences. The Sobol sequence satisfies one of the following two properties. The first property corresponds to the case where there is exactly one drawing in the 2\( ^d \) hypercube for any binary segment of the d-dimensional sequence of length 2\( ^d \), and the second property subdivides each 4\( ^d \) hypercube for any binary segment of the d-dimensional sequence of length 4\( ^d \), resulting in exactly one drawing in each 4\( ^d \) hypercube.

When using this sampler to perform parallel block-based rendering, the sequence is internally enumerated. However, there is a disadvantage that it is sensitive to the pattern artifact of the resultant image. To minimize this, the Sobol sequence typically uses a power of two sub-samples per pixel.

F. Low Discrepancy Sampler

This sampler is a combined version between the Quasi-Monte Carlo sequence with a pseudorandom number generator based on the Kollig and Keller’s method [5]. The low discrepancy sampler makes the sequence using the following way. First, all of the individual 2D sample dimensions are filled using the same (0, 2)-sequence. Due to internal storage costs, low discrepancy samples are only provided up to a certain dimension, after which independent sampling takes over [9]. Fig. 2 shows the sampling result compared to the independent sampler. Unlike the independent sampler, the low discrepancy sampler shows a little bit more regular pattern.

![Fig. 2. Comparison of sequences made by the independent sampler(left) and a low discrepancy sampler(right).](image)

III. RECONSTRUCTION FILTERING METHODS

A. Box Filter

Box filters, which are also widely used in the field of image processing, have the fastest speed, but have a disadvantage that aliasing usually occurs. Therefore, it is evaluated that the performance of the filtering algorithm for the rendering result is the lowest. Besides, there is another problem that the user performs filtering by specifying a specific window size, and blurring occurs depending on the window size.

B. Tent Filter

A tent filter, also known as a triangle filter, uses a triangular kernel and is often used in path tracing. This filter generates less aliasing than the box filter and does not cause ringing effect. This filter can be useful in rendering scenes where brightness discontinuity occurs rapidly, bilinear interpolation method, spline interpolation method, and so on.

C. Gaussian Filter

A window-based Gaussian filter with configurable standard deviation can be used not only for image processing but also for refining rendering results. Since the Gaussian function-based high-level calculation is used, the ringing
effect does not occur and good results can be obtained. However, compared to box filters, Gaussian filters can also experience blurring problems. In some cases, this blurring problem occurs more often than box filters. The two-dimensional Gaussian function is defined by:

\[ g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \] (4)

D. Mitchell-Netravali Filter

The Mitchell-Netravali filter [7] is one of the separable cubic spline reconstruction filters with four pixel wide carriers. To perform this filter, the filtering is first excepts the portion corresponding to the improper filter such as the discontinuity curve, then performed the filtering process based on the following equations:

\[
b(x) = \begin{cases} 
\frac{1}{6} & \text{if } x < 1 \\
\frac{1}{2} & \text{if } 1 \leq |x| < 2 \\
0 & \text{otherwise}
\end{cases}
\] (5)

where B and C are constants. This filter can be a complementary companion to the problems of sharpness and ringing.

E. Catmull-Rom Filter

The Catmull-Rom filter [8] is a modified version of the Mitchell-Netravali filter. This filter is also adjusted to give a higher sharpness instead of increasing the sensitivity to ringing. The Catmull-Rom filter is defined by:

\[
f(t) = \begin{cases} 
t^3 + 5t^2 + 8t + 4 & \text{if } -2 \leq t < -1 \\
-3t^3 - 5t^2 + 2 & \text{if } -1 \leq t < 0 \\
3t^3 - 5t^2 + 2 & \text{if } 0 \leq t < 1 \\
-3t^3 + 5t^2 - 8t + 4 & \text{if } 1 \leq t \leq 2
\end{cases}
\] (6)

F. Lanczos Sinc Filter

The Lanczos Sinc filter [6] is the theoretically optimal low-pass filter. This filter is generally considered one of the most useful filters because it can produce sharp, high-quality output. However, there are fatal disadvantages such as ringing effect in the discontinuous areas, and problems arise such as artifacts in the corner rendering. It also shows the slowest computational speed because it is based on many calculations. The Lanczos Sinc filter is defined by:

\[
L(x) = \begin{cases} 
\frac{1}{\pi^2x^2} \sin(\pi x/a) & \text{if } x = 0, \\
\frac{a \sin(\pi x/a)}{\pi^2x^2} & \text{if } -a \leq x < a \text{ and } x \neq 0, \\
0 & \text{otherwise.}
\end{cases}
\] (7)

IV. EXPERIMENTAL RESULTS

In order to check the performance of noise reduction using multiple sampling and reconstruction filtering, we have compared six methods, respectively. All sampling and filtering methods were analyzed and experimented based on the Mitsuba renderer [9]. Before finding optimal multiple sampling and reconstruction filtering, we used the path tracing method for the volumetric rendering process.

In order to find the optimal number of sampling rays first, we have changed the number of sampling rays to 64, 128, 256, and 512 and compared their speed and accuracy. In order to check the accuracy, we calculate peak signal-to-noise ratio (PSNR):

\[
PSNR = 10\log_{10}\frac{255^2}{MSE}
\] (8)

where

\[
MSE = \frac{1}{X Y} \sum_{i=1}^{Y} \sum_{j=1}^{X} (z_{f(i,j)} - z_{o(i,j)})^2
\] (9)

### TABLE I. COMPARISON RESULTS OF THE NUMBER OF RAYS

<table>
<thead>
<tr>
<th>Number of Rays</th>
<th>PSNR</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 rays</td>
<td>32.02</td>
<td>2.15</td>
</tr>
<tr>
<td>128 rays</td>
<td>32.80</td>
<td>3.93</td>
</tr>
<tr>
<td>256 rays</td>
<td>33.60</td>
<td>7.89</td>
</tr>
<tr>
<td>512 rays</td>
<td>34.40</td>
<td>15.80</td>
</tr>
</tbody>
</table>

(Unit: db(PSNR), min(Time))

Fig. 3. Noise reduction results using 64 rays (top) and 512 rays (bottom)

Based on Table 1 and Fig. 3, we set the number of sampling rays to 512 because this number can refine a lot of noises. Then, six multiple sampling algorithms and six reconstruction filtering algorithms were performed, based on the same number of supplemental rays and the same path tracing method. For performance comparison, the execution time of the whole rendering process is checked and the PSNR shown above were applied to the rendering result. Table 2 shows the results of six multiple sampling methods, and Table 3 shows the results of six reconstruction filtering methods.

Generally, PSNR for sampling results showed similar performance except for the independent sampler that recorded the lowest value. However, in terms of the computational time, low discrepancy sampler showed the fastest performance and showed the best sampling performance. In filtering results,
filters proposed in [6, 7, 8] showed numerically superior performance, but showed a fatal problem in generating artifacts in the mirror portion, as shown Fig. 4. Therefore, it was confirmed that the performance of the tent filter was superior to those of the other filters.

![Image of artifacts](image1)

Fig. 4. Artifacts occurred case from filtering results

![Image of noise reduction](image2)

Fig. 5. Noise reduction results using low discrepancy sampler and tent filter

### TABLE II. COMPARISON RESULTS OF SIX SAMPLING METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>bathroom2</th>
<th>classroom</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>Time</td>
</tr>
<tr>
<td>Independent sampler</td>
<td>34.30</td>
<td>15.69</td>
</tr>
<tr>
<td>Low discrepancy sampler</td>
<td>34.40</td>
<td>15.83</td>
</tr>
<tr>
<td>Stratified sampler</td>
<td>34.42</td>
<td>16.33</td>
</tr>
<tr>
<td>Halton QMC</td>
<td>34.40</td>
<td>16.30</td>
</tr>
<tr>
<td>Hammersley QMC</td>
<td>34.42</td>
<td>16.58</td>
</tr>
<tr>
<td>Sobol QMC</td>
<td>34.44</td>
<td>17.41</td>
</tr>
</tbody>
</table>

(Units: dB(PSNR), min(Time))

### TABLE III. COMPARISON RESULTS OF SIX FILTERING METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>bathroom2</th>
<th>classroom</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>Time</td>
</tr>
<tr>
<td>Box filter</td>
<td>33.29</td>
<td>15.86</td>
</tr>
<tr>
<td>Tent filter</td>
<td>34.39</td>
<td>15.81</td>
</tr>
<tr>
<td>Gaussian filter</td>
<td>34.71</td>
<td>16.16</td>
</tr>
<tr>
<td>Mitchell-Netravali filter</td>
<td>34.08</td>
<td>15.99</td>
</tr>
<tr>
<td>Catmull-Rom filter</td>
<td>33.87</td>
<td>16.05</td>
</tr>
<tr>
<td>Lanczos Sinc Filter</td>
<td>34.76</td>
<td>16.31</td>
</tr>
</tbody>
</table>

(Units: dB(PSNR), min(Time))

### V. CONCLUSION

In this paper, we have compared and analyzed the method of performing sampling and filtering as post-processing of the rendering result obtained by the volumetric rendering. As the sampling method, we have analyzed random-based sampling, such as independent sampling and stratified sampling, and the Quasi-Monte Carlo-based sampling method, such as low discrepancy sampling, the Hammersley’s method, the Halton’s method, and the Sobol’s method. As the filtering method, we analyzed the Mitchell-Netravali filter, the Catmull-Rom filter, and the Lanczos Sinc filter as well as the box filter, the Gaussian filter and the tent filter, which are widely used for image processing. Experimental results show that the low discrepancy sampler is faster than the other samplers of similar PSNR performance by about 30 seconds, and is the most optimal sampler. Unlike several filters that generate artifacts in the boundary area, the tent filter performs fast and accurate filtering without generating any annoying artifacts.

### Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) (No. 2011-0030079).

### References