Master’s Thesis

Efficient and Effective
Stratification-Based Technique
for Stochastic Sampling

Jieun Ko

Department of Electrical and Computer Engineering
The Graduate School
Sungkyunkwan University
Efficient and Effective
Stratification-Based Technique
for Stochastic Sampling

Jieun Ko

Department of Electrical and Computer Engineering
The Graduate School
Sungkyunkwan University
Efficient and Effective
Stratification-Based Technique
for Stochastic Sampling

Jieun Ko

A Master’s Thesis Submitted to the Department of
Electrical and Computer Engineering
and the Graduate School of Sungkyunkwan University
in partial fulfillment of the requirements
for the degree of Master of Science in Engineering

October 2020

Supervised by
Sungkil Lee
Major Advisor
This certifies that the Master’s thesis of Jieun Ko is approved.

Committee Chair : Jaepil Heo

Committee Member : Jaemin Jo

Major Advisor : Sungkil Lee

The Graduate School
Sungkyunkwan University
December 2020
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>ii</td>
</tr>
<tr>
<td>List of Equation</td>
<td>ii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>ii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Related Work</td>
<td>5</td>
</tr>
<tr>
<td>2.1. Stochastic Sampling</td>
<td>5</td>
</tr>
<tr>
<td>2.2. Poisson disk Sampling</td>
<td>6</td>
</tr>
<tr>
<td>2.3. Stratified Sampling</td>
<td>7</td>
</tr>
<tr>
<td>2.4. Progressive Sampling</td>
<td>8</td>
</tr>
<tr>
<td>2.5. Low-Discrepancy Sampling</td>
<td>8</td>
</tr>
<tr>
<td>3. Parallelization of Progressive Jittered Sampling Using GPU</td>
<td>9</td>
</tr>
<tr>
<td>3.1. Random Number Generation</td>
<td>11</td>
</tr>
<tr>
<td>3.2. Initialization</td>
<td>12</td>
</tr>
<tr>
<td>3.3. Parallelization</td>
<td>13</td>
</tr>
<tr>
<td>3.4. Results</td>
<td>15</td>
</tr>
<tr>
<td>3.4.1. Performance</td>
<td>15</td>
</tr>
<tr>
<td>3.4.2. Spectral and Pattern Analysis</td>
<td>17</td>
</tr>
<tr>
<td>3.4.3. Rendering Quality</td>
<td>21</td>
</tr>
<tr>
<td>3.5. Discussion</td>
<td>24</td>
</tr>
<tr>
<td>3.6. Limitation and Future Work</td>
<td>25</td>
</tr>
<tr>
<td>4. Effective Stratified Sampling Using Dart</td>
<td>26</td>
</tr>
<tr>
<td>4.1 Stratification</td>
<td>27</td>
</tr>
<tr>
<td>4.2 Generation of Sampling Area by Dart Throwing</td>
<td>27</td>
</tr>
<tr>
<td>4.3 Generation of Samples</td>
<td>30</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.4. Results</td>
<td>33</td>
</tr>
<tr>
<td>4.4.1. Performance</td>
<td>33</td>
</tr>
<tr>
<td>4.4.2. Spectral Analysis</td>
<td>35</td>
</tr>
<tr>
<td>4.4.3. Rendering Quality</td>
<td>38</td>
</tr>
<tr>
<td>4.5. Discussion and Limitations</td>
<td>42</td>
</tr>
<tr>
<td>5. Conclusion</td>
<td>43</td>
</tr>
<tr>
<td>References</td>
<td>44</td>
</tr>
<tr>
<td>Korean Abstract</td>
<td>46</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>34</td>
</tr>
</tbody>
</table>

List of Equations

<table>
<thead>
<tr>
<th>Equation</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>23</td>
</tr>
<tr>
<td>11</td>
<td>26</td>
</tr>
<tr>
<td>12</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>31</td>
</tr>
<tr>
<td>14</td>
<td>34</td>
</tr>
<tr>
<td>15</td>
<td>37</td>
</tr>
<tr>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>17</td>
<td>41</td>
</tr>
</tbody>
</table>
Abstract

Efficient and Effective Stratification-Based Technique for Stochastic Sampling

Sampling techniques are widely used in computer graphics. By adding sample pixels in the area and averaging them, aliasing can be reduced. So the number of samples and its distribution are the key factors of image quality. The higher the number, the better image is generated. The sampling patterns with blue-noise property avoid aliasing which makes better results. There have been many studies to address these problems, increasing the sampling efficiency, or generating the sampling patterns with matching properties.

In this thesis, we study a new approach to parallelize a progressive sampling technique with improved performance by using GPU parallelization. The proposed technique decorrelate the sequential dependency with the progressive jittered sampling pattern to parallelize. This algorithm enables to generate samples with more efficiency than existing progressive jittered sampling. Another approach is to increase the randomness of stratified sampling by determining the sampling area through the dart throwing technique. Unlike previous stratified sampling, the method allows us to render images with improved image quality.

Keywords: Sampling, Anti-aliasing, Blue noise, MCMC
1. Introduction

Stochastic sampling is applied in various fields. These applications range from rendering, imaging, and reconstruction. For example, in the computer graphics community, after ray tracing method was introduced by Whitted, which simulates realistic shading effects [1]. The sampling techniques are widely studied to address its aliasing problem. The choice of a sampling pattern and number of samples is key factor of good rendering quality and speed of computation.

Among the sampling patterns, Poisson disk sampling produces high quality images, while having the same number of samples [2]. However, as the computational cost of Poisson disk sampling is high, jittered sampling (stratified sampling)[2, 3] technique was proposed to increased the sample generation speed. This technique does not satisfy the characteristics of the progressive sample sequence, which ensures that all samples from previous stages are well-distributed state in the process of generating samples step by step due to the nature of sampling in order by being divided into grid structures. Therefore, a technique such as progressive jittered (PJ) [4] that performs sampling gradually has been proposed.

However, jittered sampling has a limit in terms of generating visually pleasing sample patterns. While its computational speed is higher than other
method, because of its grid structure limit its randomness, which results high aliasing than other methods.

There have been many effort to improve the distribution. N-rook sampling [5] or multi-jittered sampling [6] were studied to solve the one-dimensional clumping. However, they did not have blue noise property enough.

In this paper, we present a new efficient progressive jittered sampling in Chapter 3 which improves sample generation speed by parallelization. Then, we present a improved jittered sampling technique by dart throwing in Chapter 4. Unlike previous jittered sampling, our method provides increased randomness while maintaining certain distance from samples. From these advantages, the suggested algorithms can be applied to existing path tracing, ray tracing or soft shadow algorithms to achieve high quality image in time.

The main contributions of this thesis are:

1. A new technique for parallelizing progressive sampling using GPU parallelization. This is the first technique able to decorrelate the progressive jittered sampling pattern and parallelize them. This improved the existing progressive jittered method in terms of performance. See Chapter 3.

2. A stratification-based technique for combining the dart-throwing method with jittering. Randomness is increased than the existing jittered sampling method. See Chapter 4.
1.1 Parallelization of Progressive Jittered Sampling

Anti-aliasing is one of the old problems in the rendering field. One of the ways to solve this problem is to eliminate aliasing by selecting a pixel of an image and then calculate the number average of surrounding pixels. The stochastic sampling technique, a method of selecting pixels around this area, calculates positions of 2D samples using a probabilistic method. The more sample sets are evenly distributed, and the less regular, the better.

Since the resulting image is calculated by averaging surrounding pixels, the number of samples is important. The quality of the image increase as the number of samples increase. However, the CPU-based sampling algorithms increase the calculation time linearly in proportion to the number of samples. So there are many research to increase performance. The performance can be improved by GPU-based parallelization. However, parallelization of progressive sampling is difficult because a sequential dependency occurs when reading previous samples.

In this paper, we present a parallel algorithm that generates samples progressively. Our main idea is to subdivide the sample domain into grid cells and draw samples concurrently by the pattern. Samples are generated by reading four samples in the previous level. After the strata are divided, the sample positions of the steps are calculated in parallel according to the pattern of progressive jittered sampling [4].
1.2 Effective Stratified Sampling Using Dart Throwing

Stratified sampling (jittered sampling) [2, 3] is a sampling strategy that divides the sampling area by uniform strata. It places one sample per grid in a random location within the grid. However, the structure of stratified sampling can cause constrained randomness. This randomness in the pattern produces an irregular sampling pattern which is important for avoiding aliasing [2].

In this paper, we present a stratification-based sampling method in which irregularity is added for rendering image quality. We improve sampling distribution to be closer to the blue noise property. Unlike the previous jittered sampling technique, our sampling model determines the sampling area through dart throwing. The randomness is increased by generating a patch in which the sample is placed at random in the area of a specific size.

To maintain the distance of each sample, we make the sampling area smaller than the sampling patch. The sample is placed in the sampling area as a limited range of sampling patches. Consequently, we simulate the minimum distance property of blue noise. Our method generates 2d power spectra that are more similar to blue noise property than jittered sampling patterns.
2. Related Work

In computer graphics, sampling techniques have been studied widely. We briefly review the main categories of the sampling techniques.

2.1 Stochastic Sampling

After ray tracing method introduced by Whitted [1], many studies were proposed to address the aliasing artifact. Dippe and Wold pioneered by analyzing sampling technique in the frequency aspect [2]. Cook showed that the regularly spaced samples make the noises [7]. Shirley showed the relationship of the sample point distribution measure and its result image error [8]. Aliasing artifacts showed in regular sampling and random patterns reduced those artifacts. As shown, high-frequency noise is better visually pleasing than aliasing which is called blue noise. Poisson disk sampling is one of the trials to achieve those properties.
2.2 Poisson Disk Sampling

Poisson disk sampling is a typical sampling with blue noise characteristics. These pattern distributions are similar to that of eye cells in the retina which visually pleasing images can be obtained [9]. The most basic algorithm for generating Poisson disk sample sets is the dart throw [7] sampling technique. This technique randomly generates points in the sampling area, determines whether there is a sample within a certain distance radius of the sample, and calculates it repeatedly until the target area is filled. In this sampling method, the distance calculation and rejection process are repeated. Also, as the process progresses, the density of the sample increases, the chance to find the right sample is being lowered. This results in high computation time. Various studies have been conducted to increase this efficiency, parallel Poisson disk sampling [10] is one of them.

![Figure 1. An example of jittered sampling progress.](image-url)
2.3 Stratified Sampling

Stratified sampling is a method of a process that subdivide the sampling domain into regular cells and place samples into each subdivided part. Dippe et al. proposed jittered sampling which places one sample in each stratum randomly [2]. However, the jittering technique suffers clumping when the dimension decreased to 1 dimension. To address this problem, Shirley et al. proposed multi-jittered sampling [6]. Their idea is to extend the jittered sampling pattern with the N-rook pattern. The method makes samples aligned when it is projected onto X-axis or Y-axis. Kensler et al. improved this method, proposing correlated multi-jittered sampling which makes more well-distributed samples [11]. However, as they are bounded in the stratified structure, these patterns did not possess the minimum distance property of blue noise.

Figure 2. An example of jittered sampling progress.
2.4 Progressive Sampling

The sequences are progressive if the distribution of the sample is well distributed as the number of samples increases. For example, a random sampling pattern is generated with uniformly distributed pseudo-random numbers. Random sampling exhibits high aliasing error. Halton proposed sample sequences based on co-prime numbers [12]. Sobol proposed a sample sequence which is based on base-2 computation [13, 14].

![Figure 3. An example of Halton sequence sampling progress.](image)

2.5 Low-Discrepancy Sequence

Discrepancy is a number that describes the distribution of the sample pattern. The low discrepancy sequence properties of uniformity and equidistribution [15]. Shirley showed a relation between discrepancy and image quality [8]. These sequences are generated by a deterministic mathematical formula. Low-discrepancy point sets have stratification property [5]. However, these patterns tend to have regularity.
3. Parallelization of Progressive Jittered Sampling Using GPU

In this section, we present our method for parallelization of progressive jittered sampling. For our purpose, we generate random numbers in the CPU and copy them to the random number table in the GPU. The as memories are copied directly, the distribution of the random numbers are unchanged.

Since we read 4 samples for each process. We first generate 4 samples in the initialization process. After, we generate samples by reading the 4 samples in the previous step. The sampling area is divided into four parts to calculate the sampling positions of each step in parallel according to the pattern of progressive jittered sampling.

This chapter describes the detailed process of our method. First, we describe how we generate random numbers (Chapter 3.1). Then, show the initialization and parallelization process (Chapter 3.2 and Chapter 3.3). Fig. 4 shows an overview of the proposed algorithm of our method.
Figure 4. Overview of our algorithm.
3.1 Random Number Generation

Stochastic sampling decides the position of the sample according to the distribution of random numbers. In our study, we use uniformly distributed random numbers to place samples. Our algorithm uses a random number generator function from the CPU. However, the random numbers need to be in GPU memory for GPU parallelization. Hence, we make a random number table and generate multiple random numbers with a generator, and put it in the table. The generated random numbers are copied into the texture to use the numbers in the GPU. We assume the distribution of the random numbers is the same.
3.2 Initialization

To calculate the position of a sample, our algorithm reads four samples of the previous level and calculates the next samples in parallel. Figure 5 shows the initialization and subsequent parallelization steps. In the parallelization process, the first four samples are generated in the initialization process. In the next level, samples are calculated in units of 4.

Figure 5. An illustration of sample generation method.
3.3 Parallelization

In the case of Progressive jittered sampling, the calculation is dependent on the previous result that produces the next sample which is using the previous sample. This is the reason why parallelization is difficult. To solve the sequential dependence between these algorithms, our algorithm reads the four sample calculated in the previous step. Each of these four samples generates four new samples in parallel.

At this time, the additional creation process in one stratum. First, the sample of the previous level is read and the quadrant is divided based on the sample. Then, the sample is added to the diagonal section. Then, in the next step, one of the two remaining quadrants is selected using a random number table to generate a sample. Finally, a sample is created in the diagonal quadrant of the sample.

Through this, a total of four samples are generated. Through these processes, we maintain the progressiveness of the four partitions along with the parallelization operation. Table 1 shows the pseudo-code of our proposed method.
Table 1. Pseudo-code of GPU implementation of our algorithm.

function ParallelProgressiveSamplingGPU

// s : number of samples to generate
// number_of_iteraton : \( \lceil \log_2 s \rceil \)
// construct framebuffer objects map1 and map2
// initialize random numbers to framebuffer
// initialization done at map1, placing 4 samples
foreach number_of_iteration
    bind framebuffer objects map1 and map2
    // use map1 as input texture
foreach sample of previous level in parallel
    subdivide cell into 4 child strata
    read random number frame buffer
    // sample position is decided based on probability
return sample
3.4 Results

3.4.1 Performance

The proposed method is implemented on Intel Core i7-5960X 3.40GHz and NVIDIA GeForce GTX 980 Ti. Figure 6, compares the computation time of our method and progressive jittered [4] method according to the number of samples. The number of samples was increased from 16 to about 4M, and the execution time was measured. The number can be seen in Table 2.

When the number of samples is smaller than about 4k, the performance of the CPU-based algorithm is higher. However, when the number of samples is larger than 16k, the performance of our algorithm is higher than CPU based progressive jittered sampling method. Moreover, the gap of execution time between progressive jittered sampling and our proposed method is becoming larger as the number of samples increases.

This is due to the cost of preparing the GPU. As GPU has computational cost to utilize the resources the result shows it is not efficient when the sampling number is small. However, as the number of samples increases, the performance of the algorithm proposed in this paper is becoming higher.
Figure 6. Performance comparison with progressive jittered sampling and Ours.

<table>
<thead>
<tr>
<th>Number of Samples</th>
<th>PJ[4]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0.003ms</td>
<td>0.54ms</td>
</tr>
<tr>
<td>64</td>
<td>0.007ms</td>
<td>1.036ms</td>
</tr>
<tr>
<td>256</td>
<td>0.021ms</td>
<td>1.537ms</td>
</tr>
<tr>
<td>1024</td>
<td>0.076ms</td>
<td>2.035ms</td>
</tr>
<tr>
<td>4096</td>
<td>0.283ms</td>
<td>2.536ms</td>
</tr>
<tr>
<td>16K</td>
<td>1.158ms</td>
<td>3.03ms</td>
</tr>
<tr>
<td>65.5K</td>
<td>4.309ms</td>
<td>3.534ms</td>
</tr>
<tr>
<td>262K</td>
<td>10.048ms</td>
<td>4.039ms</td>
</tr>
<tr>
<td>1M</td>
<td>31.922ms</td>
<td>4.533ms</td>
</tr>
<tr>
<td>4.19M</td>
<td>148.718ms</td>
<td>5.025ms</td>
</tr>
</tbody>
</table>
3.4.2 Spectral and Pattern Analysis

In Figure 7, we compare our method with various 2D sampling patterns and show each sample pattern's 2D Fourier spectra. Each image of the upper line is 64 samples generated with a 2D sampling pattern of random sampling, Hammersley sampling [16], Halton sampling, Poisson disk sampling, jittered sampling, progressive jittered sampling, and our proposed sampling method. Each image with the bottom line is its 2D Fourier spectra.

The sampling patterns' characteristics can be seen by looking at the 2D power spectrum. The black region in the center denotes that low frequency is removed. The more the center region is removed, has better the blue noise characteristic. The characteristic is well shown with Poisson disk sampling. The random sampling removes some frequency regions. However, it fails to produce the blue noise property.

Hammersley sampling and Halton sampling are examples of a pattern with low-discrepancy characteristics. As it can be observed in the figure, the 2D power spectrum of Hammersley sampling and Halton sampling pattern does not show the blue noise characteristics.

In the case of jittered sampling, it can be seen that the low frequency region in the center of the 2D Fourier spectrum is slightly removed. For the
progressive jittered sampling and our proposed sampling, the center of the 2D power spectrum is equally removed. However, we showed the performance of our method, as in the previous chapter, is better with the same quality.

In Figure 8, we show the sampling distribution of jittered sampling, progressive jittered sampling, and our proposed sampling method using 16, 64, 265, and 1024 samples. In Figure 8, we can observe that our method has the same quality and distribution as jittered sampling or progressive sampling. However, in the previous chapter (Chapter. 3.4.1) we showed our proposed method has better performance than progressive jittered sampling.
Figure 7. Comparison of various sampling pattern and its power spectrum results.
Figure 8. Comparison of spatial sampling pattern with 16, 64, 256 and 1024 samples.
3.4.3 Rendering Quality

We now analyze the rendering quality of five sampling patterns. PSNR (Peak Signal-to-noise ratio) and SSIM (Structural Similarity Index) are used for the quality assessment [17]. In Figure 9, the images are generated using the path tracing method with the Cornell box model. The reference image is generated using 256 samples per pixel (spp), 1 bounce/path in Figure 9 (a). The other images are tested using 1 bounce/path, 10 spp.

In Figure 9, the image is rendered with the Poisson disk sampling pattern (b) and Halton sequences (c). Figure 9 (d) shows the image generated by Hammersley sequences. The image (e) is generated using random sampling and (f) is produced with our method. The image rendered with the Poisson disk sampling method’s PSNR is 44.64dB and SSIM is 0.9789. The image with the Halton sequence and Hammersley sequence resulted in 44.61dB for PSNR and 0.9787 for SSIM. The lowest number was resulted by the image with random sampling, which is 44.45dB for PSNR and 0.9782 for SSIM. The image generated using random sampling is most noisy as it resulted in the lowest PSNR.
Figure 9. A rendering comparison of path tracing images of five sampling patterns. (a) Poisson disk sampling method as a reference (205 spp.). (b) Poisson disk sampling (10 spp.). (c) Halton sequence (10 spp.). (d) Hammersley sequence (10 spp.). (e) Random sampling (10 spp.). (f) Ours (10 spp.).
In Figure 10, we plot PSNR with increasing 5, 10, 15, 20 and 25 sample points using various sampling algorithm methods. In Figure 10, the images are generated using the path tracing method with the Stanford lucy model. The reference image is generated using 256 samples per pixel (spp).

The quality of image resulted highest mostly in Poisson disk sampling and lowest on random sampling as shown in Figure 10. The visual and numerical results in Figure 9 and Figure 10 indicate that our patterns closely compete with the Halton sequence and Hammersley sequence and outperform the random sampling pattern.

![Figure 10. A PSNR plot of the renderings of a scene using five sampling patterns. The reference image (left) and the PSNR results as sample count (right).](image)
3.5 Discussions

In this study, we proposed the parallel progressive jittered algorithm which is based on GPU. We removed the sequential dependence of progressive jittered sampling. So, we can compute sample position concurrently. To compute the position of samples, we read the samples from the previous step.

The jittered sampling was not able to reach its optimal distribution. The jittered sampling positions one sample per strata. Since its sample position is calculated sequentially. It shows high error until it gets maximal. Progressive jittered sampling was proposed to address this problem.

We show improvement in terms of performance, compared to the progressive jittered sampling. We compared the performance of the progressive jittered method to our technique and showed a performance increase. However, we maintained the equal quality of sampling patterns.
3.6 Limitation and Future Work

The main contribution of our method is that it is the first algorithm trying to parallelize the progressive jittered method. By dividing sequences by 4 units, and use them as 4 strata to place the sample progressively. As a result, we showed our method has higher performance than progressive jittered sampling. However, our sampling method showed similar 2D power spectra with progressive jittered. Moreover our method able to generate the same image quality with higher performance.

However, our method has several limitations. As our method used a random table to generate random numbers, there are performance limitations than generating random numbers in the GPU. For future work, we can optimize performance.

Since our method extends jittered sampling, our sampling method has a clumping problem when dimension reduced to 1 dimension. The jittered sampling place one sample per strata. However, there is no distance limitation between the samples when they are just on the edge of strata.
4. Effective Stratified Sampling Using Dart Throwing

In this section, we propose a method to increase the randomness of stratified sampling by determining the sampling area in the stratified structure through the dart throwing technique. Unlike the existing method of sampling in a fixed grid, the area where the sample will be placed is randomly selected. After determining the area to be sampled in this way, the location of the sample is randomly determined by limiting the area range. It can be seen that this technique exhibits higher blue noise characteristics compared to the conventional lattice structure sampling. Figure 11 shows an overview of our process.

Figure 11. Overview of improved stratified sampling using dart throwing.
4.1 Stratification

This technique divides the entire area where the sample will be located in a grid structure. The grid is divided by $4^n (n = 1, 2, 3 \ldots)$ to ensure that the area where the sample is placed has the size of $1 \times 1$ and has a total area of 1. The area to be placed through dart throwing will be doubled in the horizontal direction and doubled in the vertical direction.

4.2 Generation of Sampling Area by Dart throwing

The stratified structure of the jittered sampling pattern reduces the irregularity of the sample, although it makes samples more evenly distributed over the entire area. As it divides the sampling domain uniformly and assigns one sample per lattice. This would make a distance from the blue-noise property.

To solve this problem, we applied to propose a method which is inspired by one of the existing Poisson disk sampling techniques, the dart throwing technique. The dart throwing technique throws a sample and checks other samples in a distance radius. If there is a sample in it, the generated sample is deleted, and if there is no sample in the radius, the new point is added.
Our proposed algorithm projects the characteristics of the dart throwing technique onto a 2D grid. Instead of checking all samples in radius distance, we use a sampling patch. Figure 12 below shows the process. The sampling patches are selected so that they do not overlap in a radius that is a certain distance from the selected point. Among the generated grid structures, the point where the grid lines intersect horizontally and vertically is used as the reference coordinate. First, a number is assigned to each horizontal grid line and a vertical grid line. After that, a random integer is generated and used as the center coordinate of the segment where the sample will be located. After

![Figure 12. Dart throwing method projected to 2D grid](image)
determining the center of the division, the area within the radius is calculated according to the size of the defined division. Through this, the position on the grid where the section is already located is not selected again. Through this process, a dart throwing effect that prevents creating a sample within a specific distance can be obtained by creating an area randomly on the grid based on the coordinates selected by the sample and maintaining a specific area. Table 3 shows the pseudo-code of our proposed method.

Table 3. Pseudo-code of sampling area method.

```
function SamplingAreaByDartThrowing
    //s : number of samples to generate
    //c : number of strata
    //construct array for samples s x s
    //construct array for strata c x c
    while array_for_strata not fully flagged
        // generate 2 dimensional random number
        // which ranges 0 to s and use it as strata_Coordinate.
        if strata_Coordinate not in array_for_samples
            then foreach stata_Coordinate and its around
                // flag array_for_strata
            return array_for_samples
```
4.3 Generation of Samples

In the stratified sampling, one sample is randomly generated and placed in each grid. However, depending on how each sample is arranged, the distance between the samples is too close or too far at the edge of strata, resulting in a phenomenon that the samples are concentrated to one side.

We propose a method to sample only within a central area of the sampling section. In this way, the distance between samples can be maintained as shown in Figure 13. In Figure 13 (a), we can see jittered sampling has a minimum distance between samples along edges and has a maximum distance across the sampling area. However, as we limit our sampling area, we can limit the distance between samples as shown in (b).

In this paper, the sample generation uses jittering to determine the location of the final sample. The formula is shown as follows (1).

\[ q_i = p_i + w \cdot \xi, \quad [\xi \in -0.5, 0.5], \quad (1) \]

\( \xi \) is a uniform random distribution function. \( p_i \) is the center coordinate of the division determined through dart throwing and \( w \) is the size of half the width of the division. \( q_i \) is the coordinates of the final placed sample. Through this, each
point is arranged so as not to deviate within the area of 1/4 of the division based on the center of the division arranged through the dart throw, so that distance between samples are limited. Table 4 shows the pseudo-code of our proposed method.

Figure 13. Comparison of the sampling distance of limiting sampling domain, (a) with limitation and (b) without limitation. The yellow line suggests the possible minimum distance and the red suggests the maximum.
Table 4. Pseudo-code of our algorithm.

```
function SamplingByDartThrowing
  //s : generated sampling area array
  //p : generated sample array
  //Initialize Array
  s = SamplingAreaByDartThrowing()

  while number of sample to generate
    // generate 2 dimensional random number(-0.5~0.5)
    //p = s + rand() *1.0f
```

- 32 -
4.4 Results

4.4.1 Performance

The proposed method is implemented on Intel Core i7-7700HQ 3.40GHz. Figure 14, compares computation time of stratified sampling method, Poisson disk sampling and our proposed method according to the number of samples. The number of samples was increased from 40, 200, 700 to about 1M, and the execution time was measured.

When the number of samples 40, the execution time of stratified sampling, ours, and Poisson disk sampling is 0.001ms, 2.363ms, and 10.602ms. The execution time is 0.104ms, 490.439ms, and 597.99ms when using about 1M samples. It can be observed time difference between the Poisson disk and our proposed method decreases as the number of samples increase. We can see the performance of our technique and Poisson disk sampling drops when the number of samples increases. This is due to the limitation of the dart throwing technique. However, we can observe our method performance is better than the Poisson disk sampling method.
Figure 14. Performance comparison with various sampling method with Ours.

### Table 5. Sampling Performance Comparison (ms)

<table>
<thead>
<tr>
<th>Number of Samples</th>
<th>Jittered</th>
<th>Poisson Disk</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.001ms</td>
<td>10.602ms</td>
<td>2.363ms</td>
</tr>
<tr>
<td>200</td>
<td>0.002ms</td>
<td>20.618ms</td>
<td>4.319ms</td>
</tr>
<tr>
<td>750</td>
<td>0.005ms</td>
<td>31.387ms</td>
<td>5.095ms</td>
</tr>
<tr>
<td>3000</td>
<td>0.026ms</td>
<td>282.94ms</td>
<td>38.429ms</td>
</tr>
<tr>
<td>5500</td>
<td>0.048ms</td>
<td>307.38ms</td>
<td>114.016ms</td>
</tr>
<tr>
<td>6500</td>
<td>0.056ms</td>
<td>312.99ms</td>
<td>116.87ms</td>
</tr>
<tr>
<td>8000</td>
<td>0.073ms</td>
<td>320.51ms</td>
<td>237.03ms</td>
</tr>
<tr>
<td>12200</td>
<td>0.104ms</td>
<td>597.99ms</td>
<td>490.439ms</td>
</tr>
</tbody>
</table>
4.4.2 Spectral Analysis

Figure 15 shows the comparison of random sampling, Hammersley sampling, Halton sampling, Poisson disk sampling, stratified sampling, and our proposed method. The sample pattern generated with 75 samples of each sampling technique was shown in the first column. We show the estimated density of the 2D Fourier spectrum of each algorithm and its radial mean. The density of the 2D Fourier spectrum was estimated using about 4k samples of each sampling method.

The blue noise characteristic can be observed at the 2D power spectrum. The black holes in the center of the 2D power spectrum show that the low frequency of the image is well removed. This characteristic is visible at Poisson disk sampling. The center of the 2D power spectrum has been removed. Moreover, we can observe samples are well-distributed. On the contrary, in random sampling, the 2D power spectrum shows that some frequencies are removed. However, low frequency regions are not well removed. Samples are randomly distributed which exhibit some clumping. Hammersley sampling and Halton sampling is an example of a pattern with low-discrepancy characteristics, as shown in the 2D power spectrum, they do not show blue noise characteristics.
The low frequency region in the center is slightly removed in the jittered sampling. However, it does not show a strong blue noise characteristic. So clumpings of some samples are shown in the sampling pattern. On the other hand, our proposed sampling pattern shows better distributions. From the center of the 2D power spectrum, we can observe that low frequency is more properly removed than jittered sampling.
Figure 15. Spectral analysis of various sample sequence
4.4.3 Rendering Quality

We analyze the rendering quality of our proposed sampling patterns. PSNR (Peak Signal-to-noise ratio) are used for the quality assessment [14]. The images are generated using the path tracing method. The reference image is generated using 400 samples per pixel (spp), 1 bounce/path in Figure 16 and Figure 17.

In Figure 16, we compare the result of path tracing rendering images using a different number of samples. We used 13, 24, and 30 samples per pixel for rendering Stanford lucy. We used 13, 23, and 30 samples per pixel for rendering the cornel box. The number of samples in the testing is slightly different since our method generates samples randomly.

The image comparison Figure 16 indicates an image generated by our method is better than stratified sampling. The PSNR of Stanford lucy images with jittered sampling is 75.19dB, 77.69dB, and 78.63dB each using 13, 24, and 30 samples. On the contrary, the PSNR of rendered images with our method is 75.22dB, 77.78dB, and 78.69dB generated with 13, 24, and 30 samples each. The biggest increase was made when an image is generated with 24 samples, 77.69dB, and 77.78dB. The 0.09dB is increased in the scene.

Besides, the PSNR of cornel box images with jittered sampling is 72.37dB,
75.86dB, and 76.94dB each using 13, 23, and 30 samples. The number of PSNR resulted in our method are 72.48dB, 75.89dB, and 76.99dB, using images generated with 13, 23, and 30 samples each. When an image is generated with 10 samples the biggest difference was made, 72.37dB and 72.48dB. The largest increase was 0.11dB.

Moreover in Figure 17, we compare the result of rendering images using different sampling patterns. Random sampling, Poisson disk sampling, Halton, Hammersley, jittered sampling and our proposed method were used to render the image. The images are rendered with 51 samples per pixel. We can observe the sampling method with the smallest PSNR is random sampling pattern which is 80.65dB and the sampling pattern with the largest PSNR is Poisson disk sampling which is 80.73dB. The Halton sampling and Hammersley sampling, which are examples of low-discrepancy sampling, their PSNR is 80.66dB and 80.67dB each. PSNR of stratified sampling is 80.68dB. However, our method’s PSNR is 80.70dB. In Figure 17, we can observe our method improved rendering image quality than jittered sampling.
Figure 16. Rendering image comparison with 10, 23 and 30 spp. The reference (left). The rendering result with jittered sampling pattern (upper-right). The rendering result with ours (lower-right).
Figure 17. Rendering image comparison using different sampling patterns. The reference (Top). The rendering result with random, Poisson disk, Halton sampling, Hammersley sampling, jitted sampling, and our proposed sampling pattern (upper-left to lower-right).
4.5 Discussion and Limitation

The primary advantage of our method is that it was able to increase the randomness in the grid structure by applying dart throwing. As a result, we showed our method remove the lower frequency band more effectively than the existing stratified sampling. Since our sampling area is smaller than the sampling patch area, gives chance to have a minimum distance for each sample.

However, our method has several limitations due to its natural fundamentals. As our method study extends stratified sampling, which divides sampling domain uniformly, it shares limitation with existing stratified sampling. That is, the distribution of samples is limited to its grid structure. A stratified sampling algorithm fills strata by one sample sequentially. If the generation of sample sets is not completed the distribution of sample patterns can be restricted. Our proposed method does not guarantee the maximal generation of the sample set when generating samples.

Our method randomly generates and places a sampling patch through dart throwing. As one sample is placed per sampling patch, the number of sampling patch affects the total number of samples. However, the distribution of the sampling patch can be varied as it is randomly generated. So it is difficult to control sample numbers to generate. There is room for improvement through further research.
5 Conclusion

In this thesis, we present two approaches for generating sample points. Firstly, we study a new approach to parallelize a progressive sampling technique with improved performance by using GPU parallelization. Our algorithm improved performance while maintaining the quality of its progressive jittered sampling. The main idea is to generate samples by reading four samples in the previous level. With this approach, we could decorrelate the sample sequence of the progressive jittered sampling pattern. Then, after the strata are divided, the sample positions of the steps are calculated in parallel according to the pattern of progressive jittered sampling.

Also, we presented a method for stratified sampling with increased randomness by determining the sampling area through the dart throwing technique. Compared to the existing jittered sampling technique, our algorithm improved rendering quality with the same number of the sampling point. The main idea of this approach is to determine the sampling area randomly. This gives more randomness to existing jittered sampling. After the sampling area is decided the sample position is randomly determined within the area.
References


논문요약

효율적이고 효과적인 격자구조 기반의 확률적 샘플링

성균관대학교 전자전기컴퓨터공학과 고지은

샘플링 기술은 컴퓨터 그래픽의 다양한 분야에서 사용된다. 렌더링 시 액러싱을 줄이는 방법 중 하나로 샘플링 된 픽셀을 더하여 평균을 내는 방식이 있다. 이 때, 샘플 수와 샘플의 분포가 이미지 품질을 높이는데 중요한 역할을 하며 샘플의 숫자가 높을수록 더 좋은 이미지가 생성되는 경향이 있다. 또한 샘플의 분포가 불규칙성을 띄는 샘플링 패턴은 영상 이미지의 품질을 높이며 액러싱을 방지한다. 이에 따라, 시간 단위에 더 많은 샘플을 생성하여 샘플링 효율성을 높이거나, 특정 샘플 분포를 가진 샘플링 패턴을 생성하기 위해 다양한 연구가 진행되었다. 본 논문에서는 샘플을 생성하는 두 가지 방법을 제시한다. 첫째, 샘플링 효율성을 위해 GPU를 사용한 빌릴화를 통해 향상된 성능으로 점진적 샘플링 기술을 빌릴화하는 새로운 접근 방식을 연구하였다. 또한, 둘째로 본 논문에서는 다양한 다양한 기술을 통해 샘플링 영역을 결정하여 격자구조 샘플링의 무작위성을 높이는 연구를 제안하였다.
점진적 격자구조 샘플링의 효율화를 위한 주요 아이디어는 이전 레벨에서 4개의 샘플을 읽어 각 샘플을 생성하는 것으로 이를 통해 4개의 계층으로 분할한 후 단계의 샘플 위치는 점진적 격자구조 샘플링 패턴에 따라 병렬로 계산하였다. 또 다른 접근법인 다트 던지기 기술을 통해 격자구조 샘플링을 개선하는 방법은 기존의 격자구조 샘플링의 분포를 샘플링의 무작위성을 높이는 것을 통하여 개선한다. 본 연구에서는 샘플링 영역을 무작위로 결정하며, 샘플 간의 거리를 유지하기 위해 샘플링 영역의 크기를 유지하며 그 안에서 샘플을 위치시킨다.

주제어: 샘플링, 안티앨리어싱, 블루노이즈, MCMC
Efficient and Effective Stratification-Based Technique for Stochastic Sampling