Popular Sampling Patterns

Fourier Analysis of Numerical Integration in Monte Carlo Rendering

Wojciech Jarosz
wjarosz@dartmouth.edu

Dartmouth Visual Computing Lab

Render the Possibilities SIGGRAPH 2016
Recall: Monte Carlo Integration

\[ I = \int_D f(x) \, dx \]
Recall: Monte Carlo Integration

\[ I = \int_D f(x) \, dx \]
Recall: Monte Carlo Integration

\[ I = \int_{D} f(x) \, dx \]
Recall: Monte Carlo Integration

\[ I = \int_{-\infty}^{\infty} f(x) \, dx \]
\[ \approx \int_{-\infty}^{\infty} f(x) S(x) \, dx \]
Recall: Monte Carlo Integration

\[ I = \int_D f(x) \, dx \]

\[ \approx \int_D f(x) S(x) \, dx \]

\[ S(x) = \frac{1}{N} \sum_{k=1}^{N} \delta(x - x_k) \]
Recall: Monte Carlo Integration

\[ I = \int_D f(x) \, dx \]

\[ \approx \int_D f(x) S(x) \, dx \]

\[ S(x) = \frac{1}{N} \sum_{k=1}^{N} \delta(x - x_k) \]
Recall: Monte Carlo Integration

\[ I = \int_{D} f(x) \, dx \]

\[ \approx \int_{D} f(x) S(x) \, dx \]

\[ S(x) = \frac{1}{N} \sum_{k=1}^{N} \delta(x - x_k) \]

How to generate the locations \( x_k \)?
Independent Random Sampling

for (int k = 0; k < num; k++)
{
    samples(k).x = randf();
    samples(k).y = randf();
}
Independent Random Sampling

for (int k = 0; k < num; k++)
{
    samples(k).x = randf();
    samples(k).y = randf();
}
Independent Random Sampling

```c
for (int k = 0; k < num; k++)
{
    samples(k).x = randf();
    samples(k).y = randf();
}

✔ Trivially extends to higher dimensions
```
Independent Random Sampling

```cpp
for (int k = 0; k < num; k++)
{
    samples(k).x = randf();
    samples(k).y = randf();
}
```

✔ Trivially extends to higher dimensions
✔ Trivially progressive and memory-less
Independent Random Sampling

```c
for (int k = 0; k < num; k++)
{
    samples(k).x = randf();
    samples(k).y = randf();
}
```

- ✔ Trivially extends to higher dimensions
- ✔ Trivially progressive and memory-less
- ✗ Big gaps
Independent Random Sampling

for (int k = 0; k < num; k++)
{
    samples(k).x = randf();
    samples(k).y = randf();
}

✔ Trivially extends to higher dimensions
✔ Trivially progressive and memory-less
✘ Big gaps
✘ Clumping
Recall: Fourier theory

Fourier transform: \[ \hat{f}(\omega) = \int_D f(x) e^{-2\pi i \omega x} \, dx \]
Recall: Fourier theory

Fourier transform:

\[
\hat{f}(\omega) = \int_D f(\vec{x}) e^{-2\pi i \omega \cdot \vec{x}} \, d\vec{x}
\]
Recall: Fourier theory

Fourier transform: \[ \hat{f}(\vec{\omega}) = \int_D f(\vec{x}) e^{-2 \pi i (\vec{\omega} \cdot \vec{x})} \, d\vec{x} \]

Sampling function: \[ \hat{S}(\vec{\omega}) = \int_D S(\vec{x}) e^{-2 \pi i (\vec{\omega} \cdot \vec{x})} \, d\vec{x} \]
Recall: Fourier theory

Fourier transform:
\[ \hat{f}(\omega) = \int_D f(x) e^{-2 \pi i (\omega \cdot x)} \, dx \]

Sampling function:
\[ \hat{S}(\omega) = \int_D \frac{1}{N} \sum_{k=1}^{N} \delta(|x - \bar{x}_k|) e^{-2 \pi i (\omega \cdot x)} \, dx \]
Recall: Fourier theory

Fourier transform: \[ \hat{f}(\vec{\omega}) = \int_D f(\vec{x}) e^{-2 \pi i (\vec{\omega} \cdot \vec{x})} \, d\vec{x} \]

Sampling function: \[ \hat{S}(\vec{\omega}) = \int_D \frac{1}{N} \sum_{k=1}^{N} \delta(|\vec{x} - \vec{x}_k|) e^{-2 \pi i (\vec{\omega} \cdot \vec{x})} \, d\vec{x} \]

\[ = \frac{1}{N} \sum_{k=1}^{N} e^{-2 \pi i (\vec{\omega} \cdot \vec{x}_k)} \]
Independent Random Sampling

\[
\frac{1}{N} \sum_{k=1}^{N} \delta(|\vec{x} - \vec{x}_k|)
\]

\[
\left| \frac{1}{N} \sum_{k=1}^{N} e^{-2\pi i (\vec{\omega} \cdot \vec{x}_k)} \right|^2
\]
Independent Random Sampling

\[ \frac{1}{N} \sum_{k=1}^{N} \delta(|\vec{x} - \vec{x}_k|) \]

\[ \left| \frac{1}{N} \sum_{k=1}^{N} e^{-2\pi i (\vec{\omega} \cdot \vec{x}_k)} \right|^2 \]
Chapter 5. Popular sampling patterns

Figure 5.6: Illustration of random and some stochastic grid-based sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

5.3 Blue noise

Any sampling pattern with Blue noise characteristics is supposed to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investigated a radially averaged power spectra of various sampling patterns. He advocated three important features for an ideal radial power spectrum; First, its peak should be at the origin. Second, the power spectrum should decrease radially outward from the origin. Third, the power spectrum should be uncorrelated with itself for any two radial vectors.

Independent Random Sampling

\[
\frac{1}{N} \sum_{k=1}^{N} \delta(\bar{x} - \bar{x}_k)
\]

\[
\left| \frac{1}{N} \sum_{k=1}^{N} e^{-2\pi i (\bar{\omega} \cdot \bar{x}_k)} \right|^2
\]
5.3 Blue noise

Any sampling pattern with Blue noise characteristics is supposed to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investigated a radially averaged power spectrum of various sampling patterns. He advocated three important features for an ideal radial power spectrum; First, its peak should be at
advocated three important features for an ideal radial power spectrum; First, its peak should be at 5.3 Blue noise (first two components) are summarised in Figures 5.8. 64 points in 2D. The corresponding sampling power spectra for Halton and Hammersley samples number of total samples is necessary. Figure 5.7 illustrates the Hammersley point set with 16 and for arbitrary dimensions, but due to the first dimension being a regular sampling, knowledge of the sequence is called the Hammersley sequence, which can create a even lower discrepancy point set corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.
5.3 Blue noise

Any sampling pattern with Blue noise characteristics is supposed to be well-distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investigated a radially averaged power spectra of various sampling patterns. He advocated three important features for an ideal radial power spectrum; First, its peak should be at
Independent Random Sampling

\[
\frac{1}{N} \sum_{k=1}^{N} \delta(\lvert \vec{x} - \vec{x}_k \rvert) \quad E \left[ \frac{1}{N} \sum_{k=1}^{N} e^{-2\pi i (\vec{\omega} \cdot \vec{x}_k)} \right]^2
\]
Any sampling pattern with Blue noise characteristics is supposed to be well distributed within the first two components, as summarized in Figures 5.8.

Ulichney [47], who investigated a radially averaged power spectra of various sampling patterns. He advocated three important features for an ideal radial power spectrum: First, its peak should be at a spatial domain without containing any regular structures. The term Blue noise was coined by 64 points in 2D. The corresponding sampling power spectra for Halton and Hammersley samples number of total samples is necessary. Figure 5.7 illustrates the Hammersley point set with 16 and for arbitrary dimensions, but due to the first dimension being a regular sampling, knowledge of the sequence is called the Hammersley sequence, which can create an even lower discrepancy point set.

The corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

$\frac{1}{N} \sum_{k=1}^{N} \delta(|\vec{x} - \vec{x}_k|) \cdot E \left[ \left( \frac{1}{N} \sum_{k=1}^{N} e^{-2 \pi i (\vec{\omega} \cdot \vec{x}_k)} \right)^2 \right]$
Regular Sampling

```cpp
for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + 0.5)/numX;
        samples(i,j).y = (j + 0.5)/numY;
    }
```
Regular Sampling

```c
for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + 0.5)/numX;
        samples(i,j).y = (j + 0.5)/numY;
    }
```

Regular Sampling

```cpp
for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + 0.5)/numX;
        samples(i,j).y = (j + 0.5)/numY;
    }
```

✔ Extends to higher dimensions, but…
Regular Sampling

for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + 0.5)/numX;
        samples(i,j).y = (j + 0.5)/numY;
    }

✔ Extends to higher dimensions, but...

✗ Curse of dimensionality
Regular Sampling

for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + 0.5)/numX;
        samples(i,j).y = (j + 0.5)/numY;
    }

✔ Extends to higher dimensions, but...

✘ Curse of dimensionality

✘ Aliasing
Regular Sampling

for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + 0.5)/numX;
        samples(i,j).y = (j + 0.5)/numY;
    }
Jittered/Stratified Sampling

for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + randf())/numX;
        samples(i,j).y = (j + randf())/numY;
    }
Jittered/Stratified Sampling

for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + randf())/numX;
        samples(i,j).y = (j + randf())/numY;
    }

✔ Provably cannot increase variance
Jittered/Stratified Sampling

for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + randf())/numX;
        samples(i,j).y = (j + randf())/numY;
    }

✔ Provably cannot increase variance
✔ Extends to higher dimensions, but…
Jittered/Stratified Sampling

for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + randf())/numX;
        samples(i,j).y = (j + randf())/numY;
    }

✔ Provably cannot increase variance
✔ Extends to higher dimensions, but...
✘ Curse of dimensionality
for (uint i = 0; i < numX; i++)
    for (uint j = 0; j < numY; j++)
    {
        samples(i,j).x = (i + randf())/numX;
        samples(i,j).y = (j + randf())/numY;
    }

✔ Provably cannot increase variance

✔ Extends to higher dimensions, but...

✘ Curse of dimensionality

✘ Not progressive
Chapter 5. Popular sampling patterns

Figure 5.6: Illustration of random and some stochastic grid-based sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

5.3 Blue noise

Any sampling pattern with Blue noise characteristics is suppose to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who invesigated a radially averaged power spectra of various sampling patterns. He advocated three important features for an ideal radial power spectrum; First, its peak should be at Fourier Analysis of Numerical Integration in Monte Carlo Rendering
Independent Random Sampling

Samples Expected power spectrum Radial mean

Figure 5.6: Illustration of random and some stochastic grid-based sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

sequence is called the Hammersley sequence, which can create a even lower discrepancy point set for arbitrary dimensions, but due to the first dimension being a regular sampling, knowledge of the number of total samples is necessary. Figure 5.7 illustrates the Hammersley point set with 16 and 64 points in 2D. The corresponding sampling power spectra for Halton and Hammersley samples (first two components) are summarised in Figures 5.8.

5.3 Blue noise

Any sampling pattern with Blue noise characteristics is suppose to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investigated a radially averaged power spectra of various sampling patterns. He advocated three important features for an ideal radial power spectrum; First, its peak should be at

Samples Expected power spectrum Radial mean
Monte Carlo (16 random samples)
Monte Carlo (16 jittered samples)
Stratifying in Higher Dimensions

Stratification requires $O(N^d)$ samples

- e.g. pixel (2D) + lens (2D) + time (1D) = 5D
Stratifying in Higher Dimensions

Stratification requires $O(N^d)$ samples

- e.g. pixel (2D) + lens (2D) + time (1D) = 5D
  - splitting 2 times in 5D = $2^5 = 32$ samples
  - splitting 3 times in 5D = $3^5 = 243$ samples!
Stratifying in Higher Dimensions

Stratification requires $O(N^d)$ samples

- e.g. pixel (2D) + lens (2D) + time (1D) = 5D
  - splitting 2 times in 5D = $2^5 = 32$ samples
  - splitting 3 times in 5D = $3^5 = 243$ samples!

Inconvenient for large $d$
- cannot select sample count with fine granularity
Uncorrelated Jitter [Cook et al. 84]
Uncorrelated Jitter [Cook et al. 84]

Compute stratified samples in sub-dimensions
Uncorrelated Jitter [Cook et al. 84]

Compute stratified samples in sub-dimensions

- 2D jittered \((x,y)\) for pixel
Uncorrelated Jitter [Cook et al. 84]

Compute stratified samples in sub-dimensions

- 2D jittered \((x,y)\) for pixel
- 2D jittered \((u,v)\) for lens

\[\begin{array}{ccc}
  x_1, y_1 & x_2, y_2 \\
  x_3, y_3 & x_4, y_4 \\
  u_1, v_1 & u_2, v_2 \\
  u_3, v_3 & u_4, v_4 \\
\end{array}\]
Uncorrelated Jitter [Cook et al. 84]

Compute stratified samples in sub-dimensions

- 2D jittered \((x,y)\) for pixel
- 2D jittered \((u,v)\) for lens
- 1D jittered \((t)\) for time
Uncorrelated Jitter [Cook et al. 84]

Compute stratified samples in sub-dimensions

- 2D jittered \((x,y)\) for pixel
- 2D jittered \((u,v)\) for lens
- 1D jittered \((t)\) for time
- combine dimensions in random order
Depth of Field (4D)

Reference

Random Sampling

Uncorrelated Jitter

Image source: PBRTe2 [Pharr & Humphreys 2010]
Uncorrelated Jitter $\rightarrow$ Latin Hypercube

Stratify samples in each dimension separately
Uncorrelated Jitter → Latin Hypercube

Stratify samples in each dimension separately

- for 5D: 5 separate 1D jittered point sets

<table>
<thead>
<tr>
<th>x</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>x2</td>
<td>x3</td>
<td>x4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>y2</td>
<td>y3</td>
<td>y4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>u</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>u1</td>
<td>u2</td>
<td>u3</td>
<td>u4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>v</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>v2</td>
<td>v3</td>
<td>v4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>t2</td>
<td>t3</td>
<td>t4</td>
<td></td>
</tr>
</tbody>
</table>
Uncorrelated Jitter $\rightarrow$ Latin Hypercube

Stratify samples in each dimension separately

- for 5D: 5 separate 1D jittered point sets

- combine dimensions in random order
Uncorrelated Jitter $\rightarrow$ Latin Hypercube

Stratify samples in each dimension separately

- for 5D: 5 separate 1D jittered point sets
- combine dimensions in random order

Shuffle order
N-Rooks = 2D Latin Hypercube [Shirley 91]

Stratify samples in each dimension separately

- for 2D: 2 separate 1D jittered point sets
- combine dimensions in random order
Latin Hypercube (N-Rooks) Sampling

[Shirley 91]
Latin Hypercube (N-Rooks) Sampling

// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
  for (uint i = 0; i < numS; i++)
    samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
  shuffle(samples(d,:));
// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
Fourier Analysis of Numerical Integration in Monte Carlo Rendering

// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
Latin Hypercube (N-Rooks) Sampling

// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));

Shuffle rows
Latin Hypercube (N-Rooks) Sampling

// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
Latin Hypercube (N-Rooks) Sampling

// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
Latin Hypercube (N-Rooks) Sampling

// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
Latin Hypercube (N-Rooks) Sampling

// initialize the diagonal
for (uint d = 0; d < numDimensions; d++)
    for (uint i = 0; i < numS; i++)
        samples(d,i) = (i + randf())/numS;

// shuffle each dimension independently
for (uint d = 0; d < numDimensions; d++)
    shuffle(samples(d,:));
Latin Hypercube (N-Rooks) Sampling
Latin Hypercube (N-Rooks) Sampling
Latin Hypercube (N-Rooks) Sampling
Latin Hypercube (N-Rooks) Sampling

Evenly distributed in each individual dimension
Latin Hypercube (N-Rooks) Sampling

Unevenly distributed in n-dimensions

Evenly distributed in each individual dimension
Chapter 5. Popular sampling patterns

Samples | Expected power spectrum | Radial mean
---|---|---
Random | | |
Jitter | | |
Multi-jitter | | |
N-rooks | | |

Figure 5.6: Illustration of random and some stochastic grid-based sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

5.3 Blue noise

Any sampling pattern with Blue noise characteristics is supposed to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investigated a radially averaged power spectra of various sampling patterns. He advocated three important features for an ideal radial power spectrum: First, its peak should be at
Multi-Jittered Sampling


- combine N-Rooks and Jittered stratification constraints
Multi-Jittered Sampling
Multi-Jittered Sampling

// initialize
float cellSize = 1.0 / (resX*resY);
for (uint i = 0; i < resX; i++)
    for (uint j = 0; j < resY; j++)
    {
        samples(i,j).x = i/resX + (j+randf()) / (resX*resY);
        samples(i,j).y = j/resY + (i+randf()) / (resX*resY);
    }

// shuffle x coordinates within each column of cells
for (uint i = 0; i < resX; i++)
    for (uint j = resY-1; j >= 1; j--)
        swap(samples(i, j).x, samples(i, randi(0, j)).x);

// shuffle y coordinates within each row of cells
for (unsigned j = 0; j < resY; j++)
    for (unsigned i = resX-1; i >= 1; i--)
        swap(samples(i, j).y, samples(randi(0, i), j).y);
Multi-Jittered Sampling

Initialize
Multi-Jittered Sampling

Shuffle x-coords
Multi-Jittered Sampling

Shuffle x-coords
Multi-Jittered Sampling

Shuffle x-coords
Multi-Jittered Sampling

Shuffle x-coords
Multi-Jittered Sampling
Multi-Jittered Sampling

Shuffle y-coords
Multi-Jittered Sampling

Shuffle y-coords
Multi-Jittered Sampling

Shuffle y-coords
Multi-Jittered Sampling

Shuffle y-coords
Multi-Jittered Sampling

Shuffle y-coords
Multi-Jittered Sampling (Projections)
Multi-Jittered Sampling (Projections)
Multi-Jittered Sampling (Projections)
Multi-Jittered Sampling (Projections)
Multi-Jittered Sampling (Projections)

Evenly distributed in each individual dimension
Multi-Jittered Sampling (Projections)

Evenly distributed in 2D!

Evenly distributed in each individual dimension
Chapter 5. Popular sampling patterns

Random

Jitter

Multi-jitter

N-rooks

Figure 5.6: Illustration of random and some stochastic grid-based sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

5.3 Blue noise

Any sampling pattern with Blue noise characteristics is supposed to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investigated a radially averaged power spectra of various sampling patterns. He advocated three important features for an ideal radial power spectrum; First, its peak should be at a specific frequency, Second, the power should decrease quickly with increasing frequency, and Third, the power should be constant at higher frequencies. The corresponding sampling power spectra for Halton and Hammersley samples (first two components) are summarised in Figures 5.8.
Chapter 5. Popular sampling patterns

Figure 5.6: Illustration of random and some stochastic grid-based sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

5.3 Blue noise

Any sampling pattern with Blue noise characteristics is supposed to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investigated a radially averaged power spectrum of various sampling patterns. He advocated three important features for an ideal radial power spectrum; First, its peak should be at a specific frequency, second, the power should decrease as the frequency increases, and third, the power should converge to a constant value as the frequency approaches infinity.

N-Rooks Sampling

Samples | Expected power spectrum | Radial mean
---|---|---
![Samples](image1.png) | ![Expected power spectrum](image2.png) | ![Radial mean](image3.png)
Chapter 5. Popular sampling patterns

Figure 5.6: Illustration of random and some stochastic grid-based sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

5.3 Blue noise

Any sampling pattern with Blue noise characteristics is supposed to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investigated a radially averaged power spectrum of various sampling patterns. He advocated three important features for an ideal radial power spectrum: First, its peak should be at

...
Poisson-Disk/Blue-Noise Sampling

Enforce a minimum distance between points

Poisson-Disk Sampling:


Random Dart Throwing
Random Dart Throwing
Random Dart Throwing
Random Dart Throwing
Random Dart Throwing
Random Dart Throwing
Random Dart Throwing
5.4 Interpreting and exploiting knowledge of the sampling spectra

Recently [39], it has been shown that the low frequency region of the radial power spectrum (of a given sampling pattern) plays a crucial role in deciding the overall variance convergence rates of sampling patterns used for Monte Carlo integration. Since blue noise sampling patterns contain almost no radial energy in the low frequency region, they are of great interest for future research to obtain fast results in rendering problems. Surprisingly, Poisson Disk samples have shown the convergence rate of $O(N^{-1})$ which is the same as given by purely random samples. This can be explained by looking at the low frequency region in the radial power spectrum of Poisson Disk samples (Fig. 5.9) which is not zero. The importance of the shape of the radial mean power spectrum in the low frequency region demands methods and algorithms that could eventually allow sample generation directly from a target Fourier spectrum.

5.4.1 Radially-averaged periodograms

Figures 5.6, 5.8 and 5.9 depict radially averaged periodograms of the various sampling strategies described in this chapter. These spectra reveal two important characteristics of estimators built using the corresponding sampling strategies.
Blue-Noise Sampling (Relaxation-based)
Blue-Noise Sampling (Relaxation-based)

1. Initialize sample positions (e.g. random)
Blue-Noise Sampling (Relaxation-based)

1. Initialize sample positions (e.g. random)
2. Use an iterative relaxation to move samples away from each other.
5.3.3 Tiling-based methods

There are some tile-based approaches that can be used to generate blue noise samples. Tile-based methods overcome the computational complexity of dart-throwing and/or relaxation-based approaches in generating blue noise sampling patterns. In the computer graphics community, two tile-based approaches are well known: First approach uses a set of precomputed tiles \([10, 25]\), with each tile composed of multiple samples, and later use these tiles, in a sophisticated way, to pave the sampling domain. Second approach employed tiles with one sample per tile \([34, 33, 49]\) and uses some relaxation-based schemes, with look-up tables, to improve the overall quality of samples.

Although many blue noise sample generation algorithms exist, none of them are easily extendable to higher dimensions \((>3)\).

5.4 Interpreting and exploiting knowledge of the sampling spectra

Recently \([39]\), it has been shown that the low frequency region of the radial power spectrum (of a given sampling pattern) plays a crucial role in deciding the overall variance convergence rates of sampling patterns used for Monte Carlo integration. Since blue noise sampling patterns contain almost no radial energy in the low frequency region, they are of great interest for future research to obtain fast results in rendering problems. Surprisingly, Poisson Disk samples have shown the convergence rate of \(O(N^{-1})\) which is the same as given by purely random samples. This can be explained by looking at the low frequency region in the radial power spectrum of Poisson Disk samples (Fig. 5.9) which is not zero. The importance of the shape of the radial mean power spectrum in the low frequency region demands methods and algorithms that could eventually allow sample generation directly from a target Fourier spectrum.

5.4.1 Radially-averaged periodograms

Figures 5.6, 5.8 and 5.9 depict radially averaged periodograms of the various sampling strategies described in this chapter. These spectra reveal two important characteristics of estimators built using the corresponding sampling strategies.
5.3.3 Tiling-based methods

There are some tile-based approaches that can be used to generate blue noise samples. Tile-based methods overcome the computational complexity of dart-throwing and/or relaxation-based approaches in generating blue noise sampling patterns. In the computer graphics community, two tile-based approaches are well known: first, a set of precomputed tiles [10, 25], with each tile composed of multiple samples, are used in a sophisticated way to pave the sampling domain. Second, an approach employs tiles with one sample per tile [34, 33, 49] and uses some relaxation-based schemes, with look-up tables, to improve the overall quality of samples. Although many blue noise sample generation algorithms exist, none of them are easily extendable to higher dimensions (>3).

5.4 Interpreting and exploiting knowledge of the sampling spectra

Recently [39], it has been shown that the low frequency region of the radial power spectrum (of a given sampling pattern) plays a crucial role in deciding the overall variance convergence rates of sampling patterns used for Monte Carlo integration. Since blue noise sampling patterns contain almost no radial energy in the low frequency region, they are of great interest for future research to obtain fast results in rendering problems. Surprisingly, Poisson Disk samples have shown the convergence rate of $O(N^{-1})$ which is the same as given by purely random samples. This can be explained by looking at the low frequency region in the radial power spectrum of Poisson Disk samples (Fig. 5.9) which is not zero. The importance of the shape of the radial mean power spectrum in the low frequency region demands methods and algorithms that could eventually allow sample generation directly from a target Fourier spectrum.

5.4.1 Radially-averaged periodograms

Figures 5.6, 5.8 and 5.9 depict radially averaged periodograms of the various sampling strategies described in this chapter. These spectra reveal two important characteristics of estimators built using the corresponding sampling strategies.

---

Poisson Disk Sampling

- Samples
- Expected power spectrum
- Radial mean
Low-Discrepancy Sampling

Deterministic sets of points specially crafted to be evenly distributed (have low discrepancy).

Entire field of study called Quasi-Monte Carlo (QMC)
The Van der Corput Sequence

Radical Inverse $\Phi_b$ in base 2

Subsequent points “fall into biggest holes”
The Van der Corput Sequence

Radical Inverse $\Phi_b$ in base 2

Subsequent points “fall into biggest holes”

<table>
<thead>
<tr>
<th>$k$</th>
<th>Base 2</th>
<th>$\Phi_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$0.1 = 1/2$</td>
</tr>
</tbody>
</table>
The Van der Corput Sequence

Radical Inverse $\Phi_b$ in base 2

Subsequent points “fall into biggest holes”

<table>
<thead>
<tr>
<th>$k$</th>
<th>Base 2</th>
<th>$\Phi_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.1 = 1/2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>.01 = 1/4</td>
</tr>
</tbody>
</table>
The Van der Corput Sequence

Radical Inverse $\Phi_b$ in base 2

Subsequent points “fall into biggest holes”

<table>
<thead>
<tr>
<th>$k$</th>
<th>Base 2</th>
<th>$\Phi_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.1 = 1/2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>.01 = 1/4</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>.11 = 3/4</td>
</tr>
</tbody>
</table>
The Van der Corput Sequence

Radical Inverse $\Phi_b$ in base 2

Subsequent points “fall into biggest holes”

<table>
<thead>
<tr>
<th>$k$</th>
<th>Base 2</th>
<th>$\Phi_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.1 = 1/2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>.01 = 1/4</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>.11 = 3/4</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>.001 = 1/8</td>
</tr>
</tbody>
</table>
The Van der Corput Sequence

Radical Inverse $\Phi_b$ in base 2

Subsequent points “fall into biggest holes”

<table>
<thead>
<tr>
<th>$k$</th>
<th>Base 2</th>
<th>$\Phi_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.1 = 1/2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>.01 = 1/4</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>.11 = 3/4</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>.001 = 1/8</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>.101 = 5/8</td>
</tr>
</tbody>
</table>
The Van der Corput Sequence

Radical Inverse $\Phi_b$ in base 2

Subsequent points “fall into biggest holes”

<table>
<thead>
<tr>
<th>$k$</th>
<th>Base 2</th>
<th>$\Phi_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.1 = 1/2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>.01 = 1/4</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>.11 = 3/4</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>.001 = 1/8</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>.101 = 5/8</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>.011 = 3/8</td>
</tr>
</tbody>
</table>
The Van der Corput Sequence

Radical Inverse $\Phi_b$ in base 2

Subsequent points “fall into biggest holes”

<table>
<thead>
<tr>
<th>$k$</th>
<th>Base 2</th>
<th>$\Phi_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.1 = 1/2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>.01 = 1/4</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>.11 = 3/4</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>.001 = 1/8</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>.101 = 5/8</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>.011 = 3/8</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>.111 = 7/8</td>
</tr>
</tbody>
</table>
The Van der Corput Sequence

Radical Inverse $\Phi_b$ in base 2

Subsequent points “fall into biggest holes”

<table>
<thead>
<tr>
<th>$k$</th>
<th>Base 2</th>
<th>$\Phi_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.1 = 1/2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>.01 = 1/4</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>.11 = 3/4</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>.001 = 1/8</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>.101 = 5/8</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>.011 = 3/8</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>.111 = 7/8</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Halton and Hammersley Points

Halton: Radical inverse with different base for each dimension:

$$\tilde{x}_k = (\Phi_2(k), \Phi_3(k), \Phi_5(k), \ldots, \Phi_{p_n}(k))$$
Halton and Hammersley Points

Halton: Radical inverse with different base for each dimension:

\[ \vec{x}_k = (\Phi_2(k), \Phi_3(k), \Phi_5(k), \ldots, \Phi_{p_n}(k)) \]

- The bases should all be relatively prime.
Halton and Hammersley Points

**Halton:** Radical inverse with different base for each dimension:

\[ \vec{x}_k = (\Phi_2(k), \Phi_3(k), \Phi_5(k), \ldots, \Phi_{p_n}(k)) \]

- The bases should all be relatively prime.
- Incremental/progressive generation of samples
Halton and Hammersley Points

Halton: Radical inverse with different base for each dimension:

$$\vec{x}_k = (\Phi_2(k), \Phi_3(k), \Phi_5(k), \ldots, \Phi_{p_n}(k))$$

- The bases should all be relatively prime.
- Incremental/progressive generation of samples

Hammersley: Same as Halton, but first dimension is $k/N$:

$$\vec{x}_k = \left(\frac{k}{N}, \Phi_2(k), \Phi_3(k), \Phi_5(k), \ldots, \Phi_{p_n}(k)\right)$$
Halton and Hammersley Points

**Halton:** Radical inverse with different base for each dimension:

\[ \vec{x}_k = (\Phi_2(k), \Phi_3(k), \Phi_5(k), \ldots, \Phi_{p_n}(k)) \]

- The bases should all be relatively prime.
- Incremental/progressive generation of samples

**Hammersley:** Same as Halton, but first dimension is \(k/N\):

\[ \vec{x}_k = (k/N, \Phi_2(k), \Phi_3(k), \Phi_5(k), \ldots, \Phi_{p_n}(k)) \]

- Not incremental, need to know sample count, \(N\), in advance
The Hammersley Sequence

1 sample in each “elementary interval”
The Hammersley Sequence

1 sample in each “elementary interval”
The Hammersley Sequence

1 sample in each “elementary interval”
The Hammersley Sequence

1 sample in each “elementary interval”
The Hammersley Sequence

1 sample in each “elementary interval”
The Hammersley Sequence

1 sample in each “elementary interval”
Monte Carlo (16 random samples)
Monte Carlo (16 jittered samples)
Scrambled Low-Discrepancy Sampling
More info on QMC in Rendering

S. Premoze, A. Keller, and M. Raab.  
Advanced (Quasi-) Monte Carlo Methods for Image Synthesis.  
In SIGGRAPH 2012 courses.
How can we predict error from these?