Monte-Carlo and Importance Sampling

Phani Srikar

January 5, 2025

1 Introduction

In computer graphics, numerical integration is essential for solving equations like global illumination, ray tracing, and the rendering equation, numerical methods are more efficient with discrete machine like computers over analytical methods. Efficient discrete sampling methods determine the accuracy and performance of these techniques and help reduce noise.

In this article, we explore various sampling methods, starting from Monte Carlo methods to more sophisticated techniques like Importance Sampling and Quasi-Random sampling functions like Halton and Hammersley sequences.

By the end, you and me can understand the different sampling techniques, their trade-offs, how and when to use them. I will try to provide derivations, code, references and examples wherever I can.

2 Monte Carlo Sampling

Monte Carlo sampling uses statistical random sampling to evaluate integrals, by trying to reduce variance as much as possible. **It is particularly useful in higher-dimension domains where we cannot estimate the PDF of the domain we are operating on.**

Example: Estimating the Integral of $f(x) = x^2$

Consider the integral of the function $f(x) = x^2$ over the interval $[0, 1]$, which we want to estimate using Monte Carlo sampling. The analytical solution to this is:

$$I = \int_0^1 x^2 \, dx = \left[\frac{x^3}{3}\right]_0^1 = \frac{1}{3} \approx 0.3333.$$

To estimate this integral using Monte Carlo sampling, we randomly sample N points $x_1, x_2, \ldots, x_N$ uniformly from the interval $[0, 1]$, and evaluate the function $f(x) = x^2$ at each of these points. The integral can then be approximated by averaging the function values at these random points and then scaling by the length of the interval $b - a$, where $a = 0$ and $b = 1$.

The Monte Carlo estimate of the integral is:

$$I \approx \frac{b-a}{N} \sum_{i=1}^{N} f(x_i)$$

where:

- $x_1, x_2, \ldots, x_N$ are uniformly sampled points from the interval $[0, 1]$
- $f(x_i) = x_i^2$ is the value of the function at each point
- $\frac{b-a}{N}$ is the scaling factor of the domain interval

For example, if we sample 5 random points $x_1 = 0.2, x_2 = 0.8, x_3 = 0.3, x_4 = 0.7, x_5 = 0.5$, we compute the function values as:

$$f(x_1) = 0.2^2 = 0.04, \quad f(x_2) = 0.8^2 = 0.64, \quad f(x_3) = 0.3^2 = 0.09, \quad f(x_4) = 0.7^2 = 0.49, \quad f(x_5) = 0.5^2 = 0.25.$$

The Monte Carlo estimate of the integral is the average of these values:

$$I \approx \frac{(1-0)}{5}(0.04 + 0.64 + 0.09 + 0.49 + 0.25) = 0.302.$$

This is an approximation of the exact integral $\frac{1}{3}$. As N increases, the Monte Carlo approximation will converge to the exact value of the integral.

Cons: High variance

While Monte Carlo sampling is a powerful technique, it can still suffer from high variance due to random fluctuations in the sampled points. In this example, some regions of the interval might have more points clustered together, leading to inaccurate estimates in those regions. This is known as **clumping** of samples.

Consider a scenario where you sample a lightmap, it can lead to clumping of samples in regions where there are less high-frequency light sources, this can lead to loss of information and cause aliasing of high-frequency samples. A PDF *(probability distribution function)* weight in this case would yield more accurate approximation.

3 Importance Sampling

Importance sampling optimizes Monte Carlo by focusing samples in regions where the integrand contributes most to the result. Instead of uniform sampling, we sample from a distribution $p(x)$ that approximates the shape of the function $f(x)$. **Basically slapping Monte-Carlo sampling with a PDF *(probability distribution function)* reduces the variance and convergence rate.**

3.1 Derivation:

Adjusting the Integral, when sampling from $p(x)$, the integral becomes:

$$I = \int \frac{f(x)}{p(x)} p(x) dx.$$

The Monte Carlo approximation is then:

$$I \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)},$$

where $x_i \sim p(x)$.

3.2 Example: Cosine-Weighted Hemisphere Sampling

In diffuse shading, sampling directions which are proportional to $\cos(\theta)$ reduces variance. The PDF for cosine-weighted sampling is:

$$p(\omega_i) = \frac{\cos(\theta)}{\pi}.$$

The Monte Carlo estimate then becomes (in the rendering equation):

$$L_o \approx \frac{1}{N} \sum_{i=1}^{N} \frac{L_i(\mathbf{x}, \omega_i) f_r(\omega_i, \omega_o)(\mathbf{n} \cdot \omega_i)}{p(\omega_i)}.$$

Substituting $p(\omega_i)$, with the sample weight simplifies the integral, aiding to achieve more accurate approximation.

4 Quasi-Random Sequences

Quasi-random (aka low discrepancy) sequences are deterministic sequences that uniformly cover a given domain with sufficient randomization and good variance and fast to compute. This section will derive and explain the relationship between the Radical Inverse function, Van der Corput sequence, Hammersley sequence, and Halton sequence.

4.1 Derivation: Radical Inverse function using arithmetic progression in a prime base system

Let p be a prime number. Any non-negative integer k can be expressed in the base p as:

$$k = a_0 + a_1 p + a_2 p^2 + \cdots + a_m p^m,$$

where $0 \leq a_i < p$ are the digits of k in base p. The summation above represents an arithmetic progression where each term contributes to the positional representation of digits of k.

To normalize this representation into the unit interval $[0, 1)$, we divide k by increasing powers of p, creating a fractional expansion:

$$\phi_p(k) = \frac{a_0}{p} + \frac{a_1}{p^2} + \frac{a_2}{p^3} + \cdots + \frac{a_m}{p^{m+1}}.$$

This function $\phi_p(k)$ is known as the **radical inverse function** in base p.

The radical inverse function $\phi_p(k)$ essentially mirrors the digits of k in base p and places them after the decimal point. For a given base p:

$$\phi_p(k) = \sum_{i=0}^{m} \frac{a_i}{p^{i+1}},$$

where a_i are the coefficients (digits) in the base-p expansion of k.

Example (Base 2): For $k = 5$, its binary representation is 101_2:

$$\phi_2(5) = \frac{1}{2^1} + \frac{0}{2^2} + \frac{1}{2^3} = 0.625.$$

The radical inverse function is the core building block for constructing quasi-random sequences such as the Van der Corput, Hammersley, and Halton sequences.

4.2 Van Der Corput Sequence

The **Van Der Corput sequence** is a one-dimensional low-discrepancy sequence constructed directly from the Radical Inverse function. For any given base p:

$$x_k = \phi_p(k), \quad k = 0, 1, 2, \ldots$$

Properties:

- 1D and uniformly distributed in $[0, 1)$.
- Useful for basic low-discrepancy sampling in one dimension.

Example (Base 2): For $k = 0, 1, 2, 3, 4, \ldots$:

$$x_k = \{0, 0.5, 0.25, 0.75, 0.125, \ldots\}.$$

4.2.1 Code

```
f32 RadicalInverseVanDerCorputSample(u32 index, u32 base) {
    f32 inverseBase = 1.0f / base;
    f32 result = 0.0f;
    f32 fraction = inverseBase;

    while (index > 0) {
        result += (index % base) * fraction;
        index /= base;
        fraction *= inverseBase;
    }
    return result;
}
```

Listing 1: Radical Inverse Function

4.3 Halton Sequence

The **Halton sequence** is a generalization of the Van Der Corput sequence to higher dimensions, using distinct prime bases for each dimension. For a d-dimensional sequence:

$$H(n) = \mathbf{x}_k = (\phi_{p_1}(k), \phi_{p_2}(k), \ldots, \phi_{p_d}(k)),$$

where $p_1, p_2, \ldots, p_d$ are the first d prime numbers and $\phi_p(n)$ is the radical inverse function in base p.

The Halton sequence generates points in multi-dimensional space using co-prime bases (e.g., 2, 3, 5 for dimensions 1, 2, and 3). For dimension d

$$H(n) = \Big(\phi_2(n), \phi_3(n), \ldots, \phi_p(n)\Big)$$

4.3.1 Code

```
f32 HaltonSequenceSample(u32 index, u32 base)
{
        return RadicalInverseVanDerCorputSample(index, base);
}
```

Listing 2: Halton Sequence

4.4 Hammersley Sequence

The **Hammersley sequence** is a modified variant of the Halton sequence. It combines the radical inverse function for $d-1$ dimensions with a uniform scaling for the first dimension. For a sequence of N samples in d-dimensions:

$$\mathbf{x}_k = \left(\frac{k}{N}, \phi_{p_1}(k), \phi_{p_2}(k), \ldots, \phi_{p_{d-1}}(k) \right),$$

where $p_1, p_2, \ldots, p_{d-1}$ are distinct prime bases.

Example (2D Hammersley Sequence, Bases 2 and 3): For $N = 5$, the sequence is:

$$\mathbf{x}_k = \left(\frac{k}{5}, \phi_2(k) \right), \quad k = 0, 1, 2, 3, 4.$$

4.4.1 Code

```
f32 HammersleySequenceSample(u32 index, u32 base, u32 totalSamples)
{
    if (base == 1)
         return (f32)index / (f32)totalSamples;
     return RadicalInverseVanDerCorputSample(index, base);
}
```

Listing 3: Hammersley sequence

4.4.2 Hammersely2D

The **Hammersley2D sequence** is a variant of Halton optimized for 2D sampling. For N samples, it generates points, it used the Van Der Corput sequence in base 2 for y dimension:

$$Hammersly2D(n) = \left(\frac{n}{N}, \phi_2(n) \right),$$

$\phi_2(n)$ **cab be optimized for given sample bits using this trick in this blog[5].** This function is tupicaly used when the number of total sampler are known and is used in sampling BRDF/IBL.

4.4.3 Code

```
vec2 HammersleySequence2DSample(u32 index, u32 totalSamples)
{
    return vec2((f32)index / (f32)totalSamples, RadicalInverseVanDerCorputSample(index, 2));
}
```

Listing 4: Hammersley sequence 2D

```
/**
 * Faster HammersleySequence2D(index, DIM = 2) bitwise trick
 * based on http://holger.dammertz.org/stuff/notes_HammersleyOnHemisphere.html
 */

 float RadicalInverseVdCBase2(uint bits) {
     bits = (bits << 16u) | (bits >> 16u);
     bits = ((bits & 0x55555555u) << 1u) | ((bits & 0xAAAAAAAAu) >> 1u);
     bits = ((bits & 0x33333333u) << 2u) | ((bits & 0xCCCCCCCCu) >> 2u);
     bits = ((bits & 0x0F0F0F0Fu) << 4u) | ((bits & 0xF0F0F0F0u) >> 4u);
     bits = ((bits & 0x00FF00FFu) << 8u) | ((bits & 0xFF00FF00u) >> 8u);
     return float(bits) * 2.3283064365386963e-10; // / 0x100000000
 }

vec2 HammersleySequence2DFastSample(u32 index, u32 totalSamples)
{
    return vec2(f32(index) / f32(totalSamples), RadicalInverseVdCBase2(index));
}
```

Listing 5: Fast Hammersley sequence 2D

4.5 Difference Between Hammersley and Halton:

- Hammersley uses a uniform scaling for the first dimension ($\frac{k}{N}$), while Halton uses the radical inverse function for all dimensions.
- Halton sequences are suitable for cases where the total number of samples N is not predetermined.

5 Summary

- Use **Monte Carlo** when you cannot define the PDF of the domain.
- Use **Importance Sampling** when you have a PDF defining the surface of the domain.
- While Monte Carlo and Importance Sampling improve accuracy, they rely on pseudo-random sequences, which can still lead to uneven sample distribution.
- **Quasi-random sequences**, like Halton and Hammersley, ensure low-discrepancy distributions, providing better coverage of the domain.
- **Radical inverse function** forms the foundation for all these quasi-random sequences.
- **Van Der Corput sequence** is a 1D low-discrepancy sequence based on the radical inverse $\phi_p(k)$.
- **Halton sequence** generalizes Van Der Corput to d-dimensions using radical inverse functions in distinct prime bases.
- **Hammersley sequence** extends this to d-dimensions by combining radical inverse functions with uniform scaling in it's first dimension.
- **Hammersley2D sequence** use a uniform scaling of $\frac{k}{N}$ for the first dimension and $\phi_2(k)$ for second dimension, can be optimized using bit tricks.
- *For higher dimension Radial Inverse sequences, use higher prime bases (p) of $\phi_p(k)$.*

Quasi-Random Sampling Functions Usage:

- **Halton Sequence:** Generates low-discrepancy points for multi-dimensional sampling (e.g., BRDF importance sampling or TAA).

- **Hammersley Sequence:** Optimized for 2D sequences, often used for sampling a hemisphere in IBL. **Use it when total no. of samples are known in advance.**
- **Sobol Sequence:** Suitable for higher-dimensional problems, commonly used in global illumination.
- **Stratified Sampling:** Divides the domain into strata to reduce variance in Monte Carlo methods.
- **Blue Noise Sampling:** Ensures even spacing between samples, often used for image sampling and point distributions.

References

[1] https://www.youtube.com/watch?v=N6xZvrLusPI

[2] https://www.youtube.com/watch?v=N6xZvrLusPI

[3] https://en.wikipedia.org/wiki/Importance_sampling

[4] https://ttwong12.github.io/papers/udpoint/udpoint.pdf

[5] http://holger.dammertz.org/stuff/notes_HammersleyOnHemisphere.html