Lecture 7: Monte Carlo Rendering

CS 6620, Spring 2009

Kavita Bala
Computer Science
Cornell University

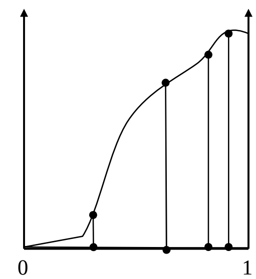
MC Advantages

- Convergence rate of O($\frac{1}{\sqrt{N}}$)
- Simple
  - Sampling
  - Point evaluation
  - Can use black boxes
- General
  - Works for high dimensions
  - Deals with discontinuities, crazy functions,…

Importance Sampling

- Why do we sample by p(x)?
- Why not just uniformly?

- Better use of samples by taking more samples in 'important' regions, i.e. where the function is large



Non-Uniform Samples

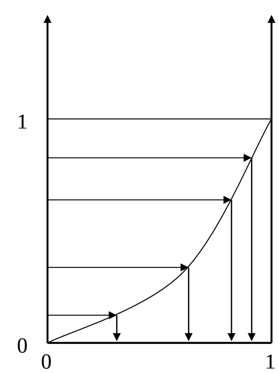
- 1) Choose a normalized probability density function *p(x)*

- 2) Integrate to get a cumulative distribution function *P(x)*:

$$P(x) = \int_0^x p(t)dt$$

- 3) Invert *P*:

$$x = P^{-1}(\xi)$$



Note this is similar to going from y axis to x in discrete case!

Importance Sampling

$$p(x) = \frac{f(x)}{\int_D f(x)}$$

- General principle:
  The closer the shape of p(x) is to the shape of f(x), the lower the variance
- Variance can also increase if p(x) is chosen badly

Cosine distribution

$$f = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \cos\theta \sin\theta d\theta d\phi$$

$$p(\theta,\phi) = \frac{\cos\theta \sin\theta}{\pi}$$

$$CDF(\theta,\phi) = \int_0^{\theta}\int_0^{\phi} \frac{\cos\theta \sin\theta}{\pi} drd\theta = (1-\cos^2\theta)\frac{\phi}{2\pi}$$

$$F(\theta) = 1-\cos^2\theta$$

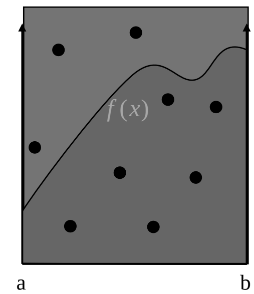
$$F(\phi) = \frac{\phi}{2\pi}$$

$$\phi_i = 2\pi u_1 \qquad \theta_i = \cos^{-1}\sqrt{u_2}$$

Rejection Methods

- Pick ξ_1, ξ_2

$$I = \int_a^b f(x)dx$$



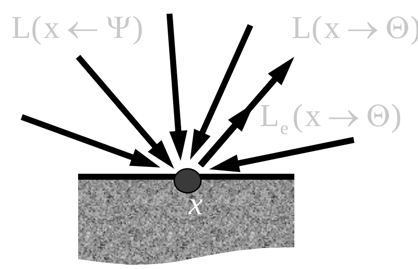
- If $\xi_2 < f(\xi_1)$, select ξ_2
- Is this efficient? What determines efficiency? A(f)/A(rectangle)

MC Advantages

- Convergence rate of O($\frac{1}{\sqrt{N}}$)
- Simple
  - Sampling
  - Point evaluation
  - Can use black boxes
- General
  - Works for high dimensions
  - Deals with discontinuities, crazy functions,…

MC applied to RE

$$L(x \to \Theta) = L_e(x \to \Theta) + \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

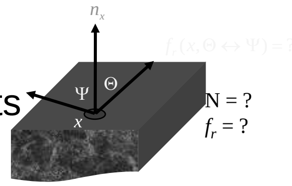


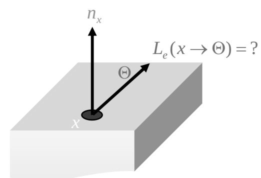
Radiance Evaluation

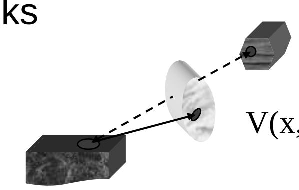
- Many different light paths contribute to single radiance value
  - many paths are unimportant
- Tools we need:
  - generate the light paths
  - sum all contributions of all light paths
  - clever techniques to select important paths

Assumptions: black boxes

- Can query the scene geometry and materials
  - surface points
  - light sources
  - visibility checks
  - tracing rays







V(x,z) = 0 or 1

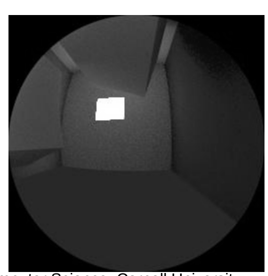
Rendering Equation

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

function to integrate over all incoming directions over the hemisphere around x

Value we want

$= L_e + \int_{\Omega_x}$



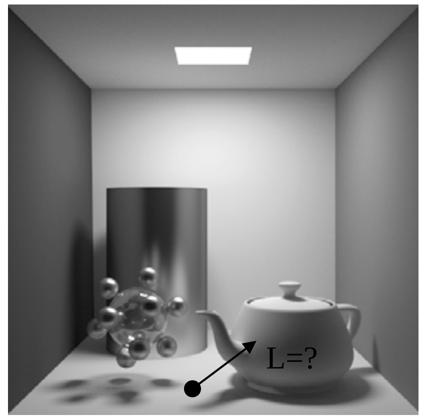
$\cdot f_r \cdot \cos$

How to compute?

$L(x \to \Theta)$ = ?

Check for $L_e(x \to \Theta)$

Now add $L_r(x \to \Theta)$ =



$$\int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

How to compute?

- Monte Carlo!
- Generate random directions on hemisphere Ω_x, using pdf $p(\Psi)$

$$L(x \to \Theta) = \int_{\Omega_x} f_r(\Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

$$\langle L(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(\Psi_i \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi_i) \cdot \cos(\Psi_i, n_x)}{p(\Psi_i)}$$

How to compute?

Generate random directions Ψ_i

$$\langle L \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(\ldots) \cdot L(x \leftarrow \Psi_i) \cdot \cos(\ldots)}{p(\Psi_i)}$$

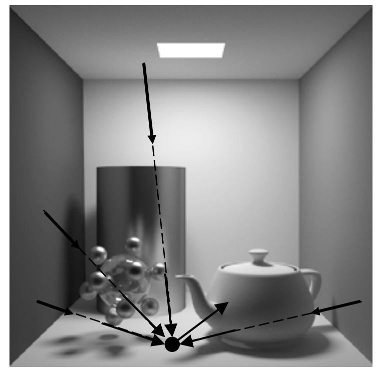
- evaluate brdf
- evaluate cosine term
- evaluate $L(x \leftarrow \Psi_i)$

How to compute?

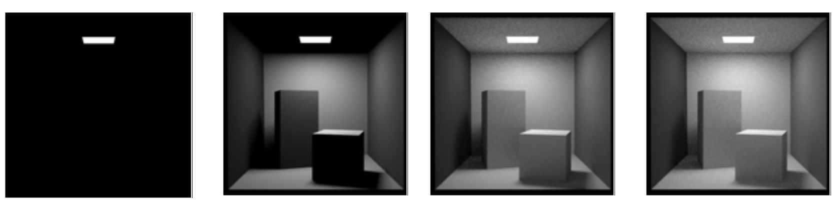
- evaluate $L(x \leftarrow \Psi_i)$?
- Radiance is invariant along straight paths
- $vp(x, \Psi_i)$ = first visible point
- $L(x \leftarrow \Psi_i) = L(vp(x, \Psi_i) \rightarrow \Psi_i)$

How to compute? Recursion ...

- Recursion ….
- Each additional bounce adds one more level of indirect light
- Handles ALL light transport
- “Stochastic Ray Tracing”



When to end recursion?



- Contributions of further light bounces become less significant
- If we just ignore them, estimators will be biased!

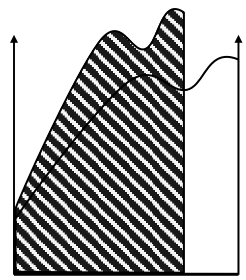
Russian Roulette

Integral

$$I = \int_0^1 f(x)dx = \int_0^1 \frac{f(x)}{P} P dx = \int_0^P \frac{f(y/P)}{P} dy$$

Estimator

$$\langle I_{roulette} \rangle = \begin{cases} \frac{f(x_i)}{P} & \textbf{if } x_i \le P, \\ 0 & \textbf{if } x_i > P. \end{cases}$$



Variance $\sigma_{roulette} > \sigma$

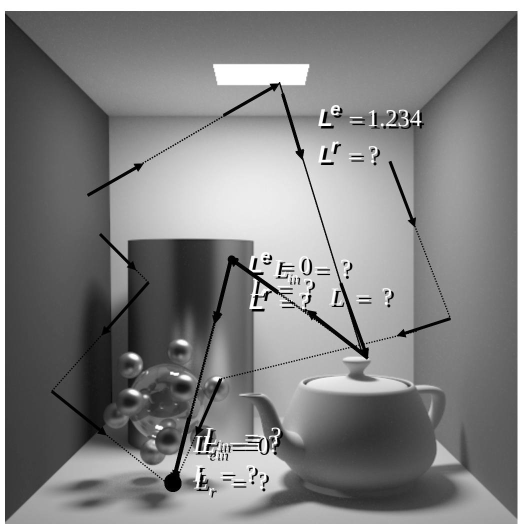
Russian Roulette

- Pick some ‘absorption probability’ α
  - probability 1-α that ray will bounce
  - estimated radiance becomes L/ (1-α)
- E.g. α = 0.9
  - only 1 chance in 10 that ray is reflected
  - estimated radiance of that ray is multiplied by 10
  - instead of shooting 10 rays, we shoot only 1, but count the contribution of this one 10 times

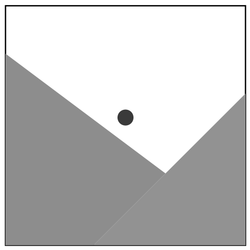
Algorithm so far ...

- Shoot viewing ray through each pixel
- Shoot # indirect rays, sampled over hemisphere
- Terminate recursion using Russian Roulette

Algorithm



Pixel Anti-Aliasing

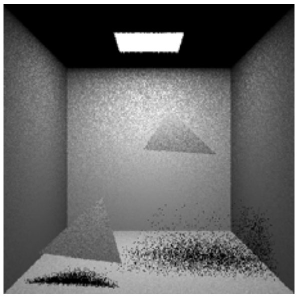


- Compute radiance only at center of pixel: jaggies
- Simple box filter:
- … evaluate using MC

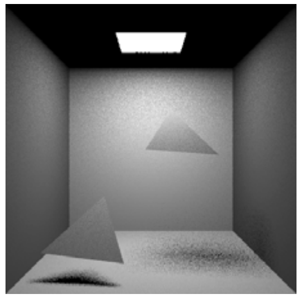
Stochastic Ray Tracing

- Parameters?
  - # starting rays per pixel
  - # random rays for each surface point (branching factor)
- Path Tracing
  - Branching factor == 1

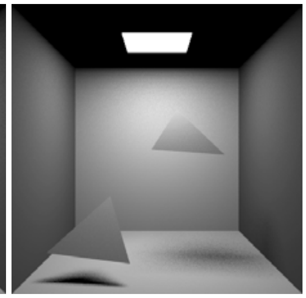
Path tracing



1 ray / pixel



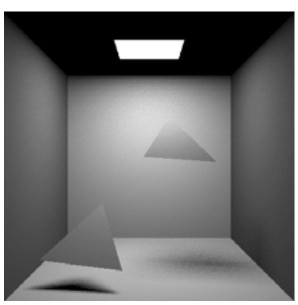
10 rays / pixel



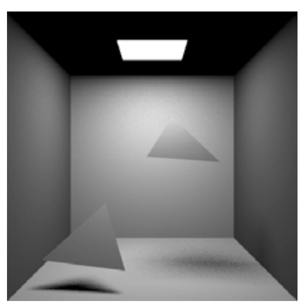
100 rays / pixel

- Pixel sampling + light source sampling folded into one method

Comparison



1 centered viewing ray
100 random shadow rays per viewing ray



100 random viewing rays
1 random shadow ray per viewing ray

Performance/Error

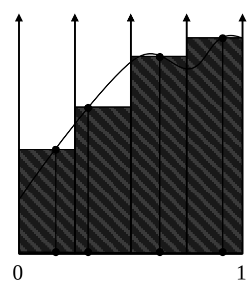
- Want better quality with smaller number of samples
  - Fewer samples/better performance
  - Stratified sampling
  - Quasi Monte Carlo: well-distributed samples

- Faster convergence
  - Importance sampling: next-event estimation

Stratified Sampling

- Samples could be arbitrarily close

- Split integral in subparts



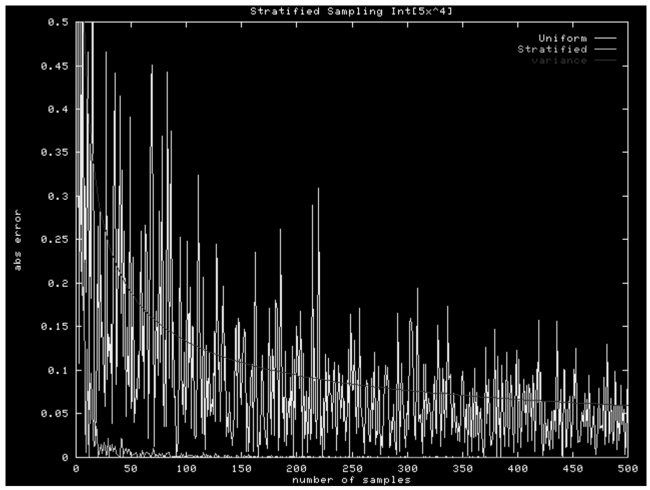
$$I = \int_{X_1} f(x)dx + \ldots + \int_{X_N} f(x)dx$$

- Estimator

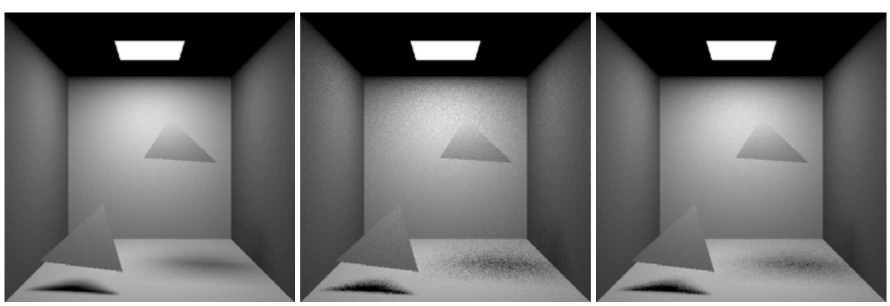
$$\bar{I}_{strat} = \frac{1}{N}\sum_{i=1}^{N}\frac{f(\bar{x}_i)}{p(\bar{x}_i)}$$

- Variance: $\sigma_{strat} \leq \sigma_{\sec}$

Numerical example



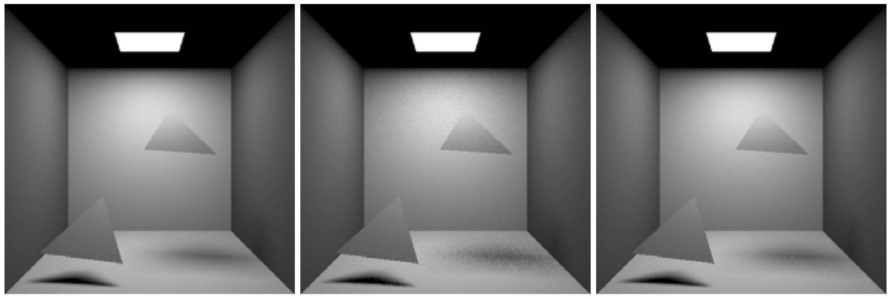
Stratified Sampling



9 shadow rays
not stratified

9 shadow rays
stratified

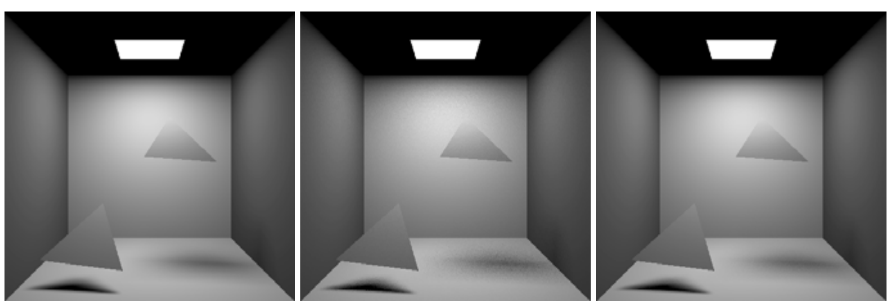
Stratified Sampling



36 shadow rays
not stratified

36 shadow rays
stratified

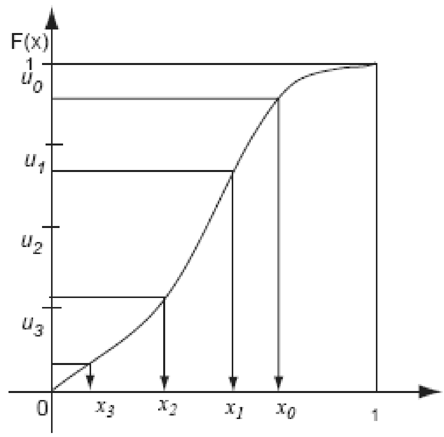
Stratified Sampling



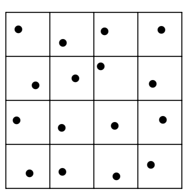
100 shadow rays
not stratified

100 shadow rays
stratified

Stratified and Importance?



2 Dimensions

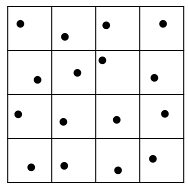


$\rightarrow N^2$ samples

- Problem for higher dimensions
- Sample points can still be arbitrarily close to each other

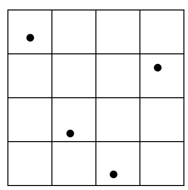
Higher Dimensions

- Stratified grid sampling:



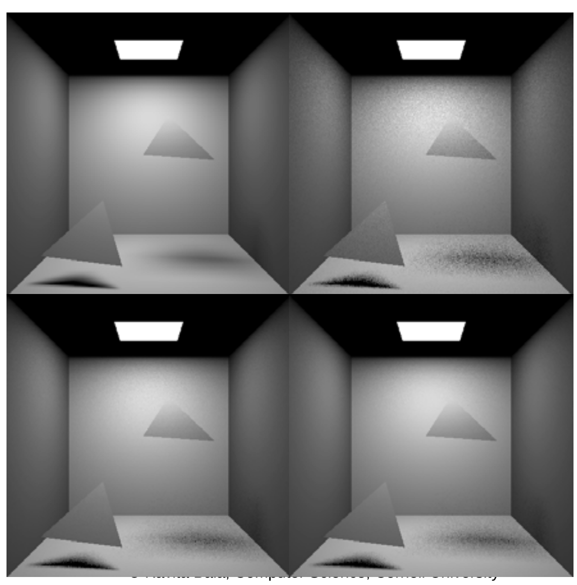
$\rightarrow N^d$ samples

- N-rooks sampling:



$\rightarrow N$ samples

N-Rooks Sampling - 9 rays

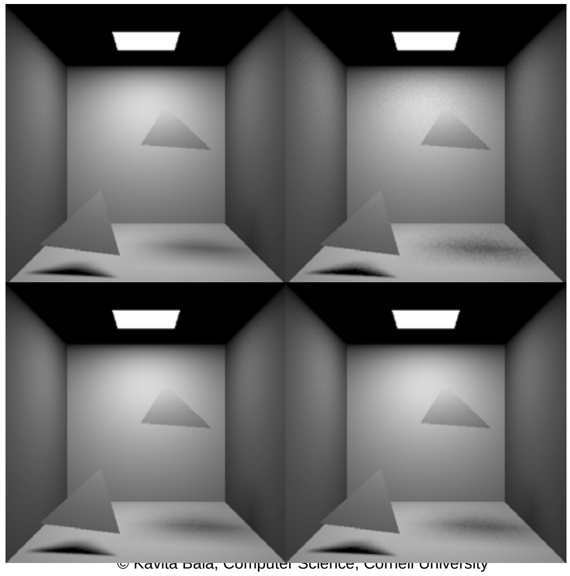


not stratified

stratified

N-Rooks

N-Rooks Sampling - 36 rays



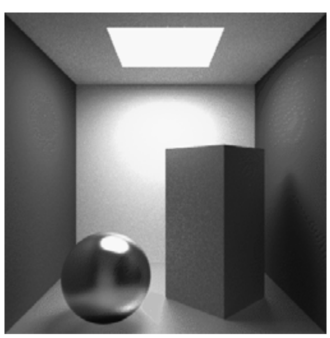
not stratified

stratified

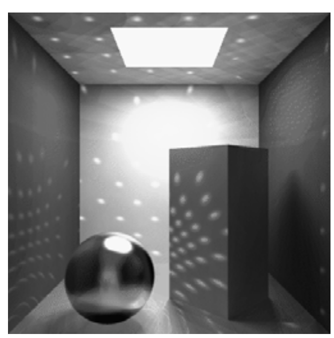
N-Rooks

Other types of Sampling

- How does it relate to regular sampling



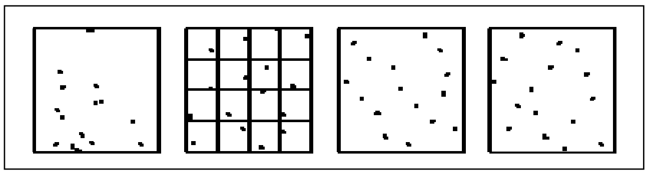
Random sampling



Regular sampling

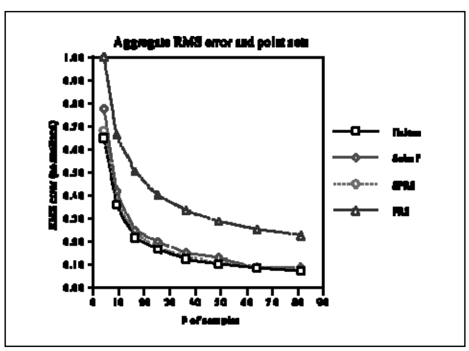
Quasi Monte Carlo

- Eliminates randomness to find well-distributed samples
  - Why? Avoid clumping
  - Why? Has better convergence properties



Random Stratified Sobol Halton

Quasi Monte Carlo



Quasi-Monte Carlo (QMC)

- Notions of variance, expected value don't apply: why?
- Introduce the notion of discrepancy
  - Discrepancy mimics variance
  - Need a low discrepancy sequence
  - E.g., subset of unit interval [0,x]
    - Of N samples, n are in subset
    - Discrepancy: |x-n/N|
  - Mainly: "it looks random"

Example: Halton

- Radical inverse $\phi_p(i)$ for primes *p*
- Reflect digits (base p) about decimal point
  - $\phi_2(i)$: $111010_2 \rightarrow 0.010111$
- Radical inverse function

$$i = \sum_j a_j(i) b^j$$

$$\Phi_b(i) = \sum_j a_j(i) b^{-j-1}$$

Halton

- Sample:
  - Where b_1, b_2, b_3 are primes

$$x_i = (\Phi_{b_1}(i), \Phi_{b_2}(i), \Phi_{b_3}(i), \ldots)$$

  - $x_i = (\phi_2(i), \phi_3(i), \phi_5(i), \phi_7(i), \phi_{11}(i), \ldots.)$

- Discrepancy: $O\left(\frac{(\log N)^d}{N}\right)$

Example: Hammersley

- Say we know what N is ahead of time
- For N samples, a Hammersley point
  - $(i/N, \phi_2(i))$
- For more dimensions:
  - $X_i = (i/N, \phi_2(i), \phi_3(i), \phi_5(i), \phi_7(i), \phi_{11}(i), \ldots.)$

Quasi Monte Carlo

- Converges as fast as stratified sampling
  - Does not require knowledge about how many samples will be used
- Using QMC, directions evenly spaced no matter how many samples are used
- Samples properly stratified-> better than pure MC

Performance/Error

- Want better quality with smaller number of samples
  - Fewer samples/better performance
  - Stratified sampling
  - Quasi Monte Carlo: well-distributed samples
- Faster convergence
  - Importance sampling: next-event estimation