CS5630 Physically Based Realistic Rendering

Steve Marschner

“Spring” 2026

04 Monte Carlo Integration

Low discrepancy sampling

We saw that the baseline convergence of MC integration is $N^{-\frac{1}{2}}$

- getting this convergence rate requires very little cooperation from the integrand

For “easy” cases we can do better

- easy means smooth integrands in low dimensions

Generally the idea is to use samples that are “less clumpy”

- stratified sampling
- blue noise sampling
- Quasi Monte Carlo (QMC) sampling

Variance

Variance measures expected variation about the mean

- we measure squared deviation since we want it to be positive on both sides
- $\mathrm{Var}\{X\} = \sigma^2\{X\} = E\left\{(X - E\{X\})^2\right\}$

Standard deviation is the square root of variance (root-mean-square variation)

- $\sigma\{X\} = \sqrt{\sigma^2\{X\}}$
- it is the intuitive measure of “degree of uncertainty” in the units of X

Some handy facts

- $\sigma^2\{X\} = E\{X^2\} - E\{X\}^2$
- $\sigma\{aX\} = a\sigma\{X\}$, therefore $\sigma^2\{aX\} = a^2\sigma^2\{X\}$

Statistical independence

Two r.v.s X and Y can be thought as a single pair-valued r.v. (X, Y)

The distribution of this pair is the joint distribution of X and Y

- call its probability density $p(x, y)$

X and Y are independent iff. $p(x, y) = p(x)p(y)$

- that is, the joint pdf is separable into the product of two one-variable pdfs

For independent r.v.s, some handy things are true:

- $E\{XY\} = E\{X\}E\{Y\}$ — for X, Y independent
- $\sigma^2\{X + Y\} = \sigma^2\{X\} + \sigma^2\{Y\}$ — for X, Y independent
- but always remember these things are not true in general!

Baseline Monte Carlo convergence

Estimator with N independent samples

$$G_N = \frac{1}{N}\sum_{i=1}^{N} g(x_i) \quad \text{where} \quad x_i \sim p$$

Variance of sum is N times the variance of the one-sample estimator

$$\sigma^2\left\{\sum_{i=1}^{N} g(x_i)\right\} = \sum_{i=1}^{N} \sigma^2\{g\} = N\sigma^2\{g\}$$

Variance of G_N is N^{-2} times the variance of the sum

$$\sigma^2\{G_N\} = \frac{\sigma^2\{g\}}{N} \qquad \sigma\{G_N\} = \frac{\sigma\{g\}}{\sqrt{N}}$$

Improving the convergence rate

…requires using non-independent samples

- to invalidate the proof on the previous slide

One approach: stratified sampling

- divide domain up into N equal-area parts, place one sample uniformly in each
- for smooth integrands eventually gets better convergence (dimension dependent)

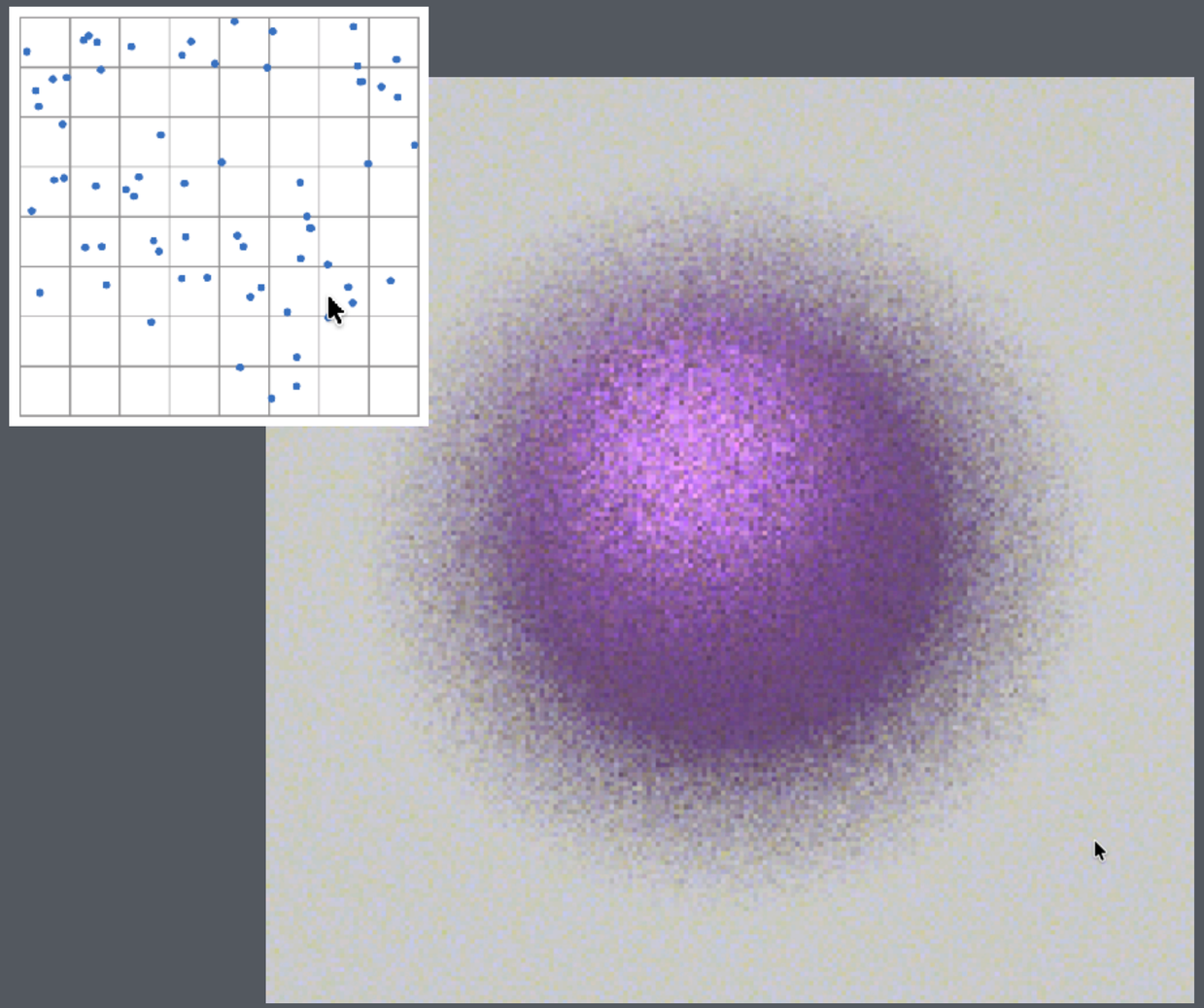
Another approach: blue noise sampling

- use Fourier spectrum of sample pattern as a design tool
- aim for patterns lacking low frequencies — hence “blue” by analogy to light spectra
- various schemes for generating and storing patterns, or piecing together on the fly

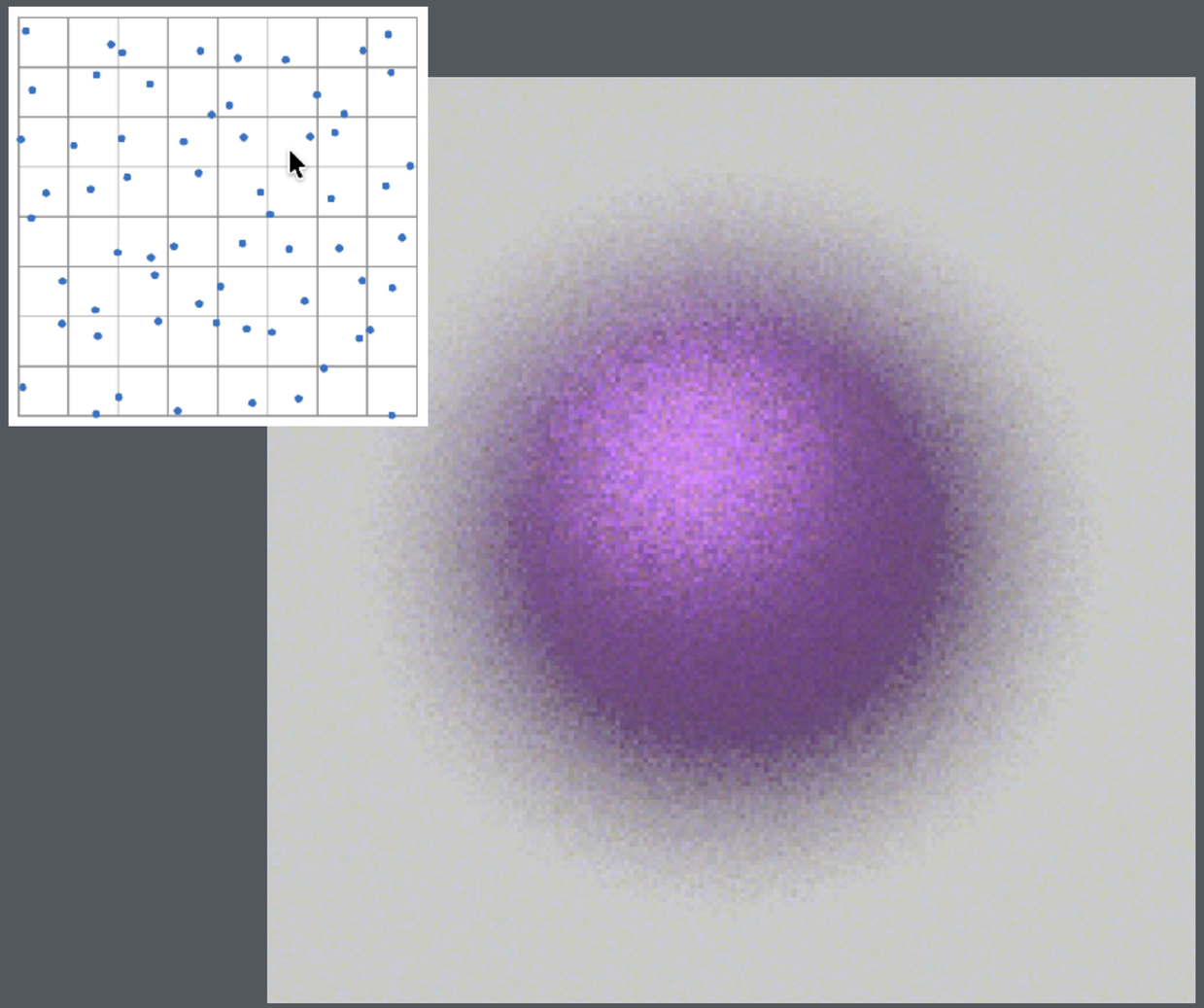
Software engineering goal: drop-in replacement for random()

- but both these approaches require knowing the number of samples up front

Stratified point sets and their effects



independent



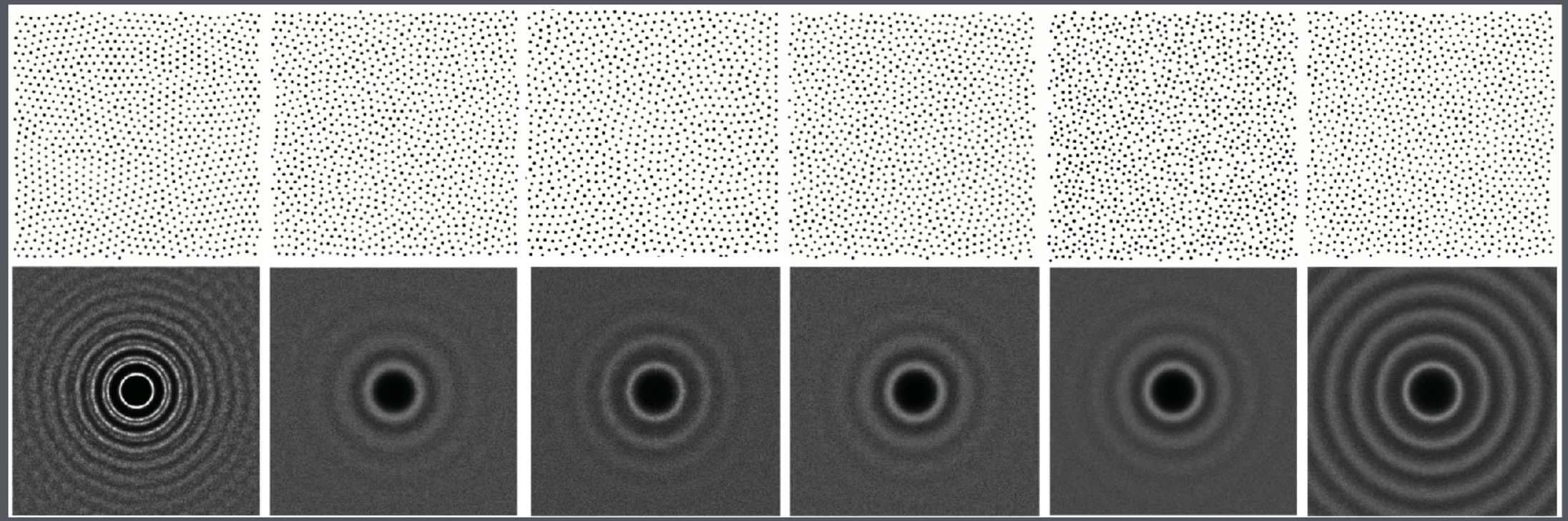
grid stratified

Convergence with stratification

If we're integrating over a unit volume domain with $p(x) = 1$

- volume of each stratum is $1/N$
- pdf for each sample is $p_i(x) = N$
- estimator for each stratum is $g_i(x_i) = \dfrac{f(x_i)}{p_i(x_i)} = \dfrac{g(x_i)}{N}$
- sum of individual estimates is $G'_N = \displaystyle\sum_{i=1}^{N} g_i(x_i) = \frac{1}{N}\sum_{i=1}^{N} g(x_i)$ — so the code looks the same
- key thing: for smooth integrands the variance of g_i is less than the variance of g
- when things look nice, variance scales as the square of the diameter of the strata
- leads to a hope for $N^{-1.5}$ convergence in 1D or N^{-1} convergence in 2D (nbdemo)

Blue noise point sets



Dong-Ming Yan et al. 2015

Stratification in software

We'd like to be able to just call random() but now there is some bookkeeping

- the sampler needs to know how many samples will be needed
- the random numbers are not all equivalent any more
  - numbers that are used the same way need to form a stratified pattern
  - the two coordinates of a point need to be related in a different way
  - separate samples from two 1D stratified patterns does not make a 2D stratified pattern
- for each of N samples we need to be able to generate many numbers—a multidimensional point

A commonly used interface has three central methods

- generate() — asks the sampler to get ready for a new integration
- next() — return the next random number (the next coordinate of the sample)
- advance() — move to the next sample

Quasi Monte Carlo

The estimation of expected values doesn't have to depend on randomness

Alternative property: low discrepancy

- discrepancy = max difference between volume of a box and the fraction of sample in it
- with bound on discrepancy, Monte Carlo integration works with deterministic samples

One class of methods: Halton sequence

- suprising idea: write sample integer as a base-p integer, then reverse digits to a base-p fraction
  - base 2: 1001010 -> 0.0101001; base 3: 0122102 -> 0.2012210
- use such sequences with relatively prime bases on each axis
  - result is an n-D low discrepancy sequence

Hammersly, Sobol' use related ideas

Quasi Monte Carlo sequences

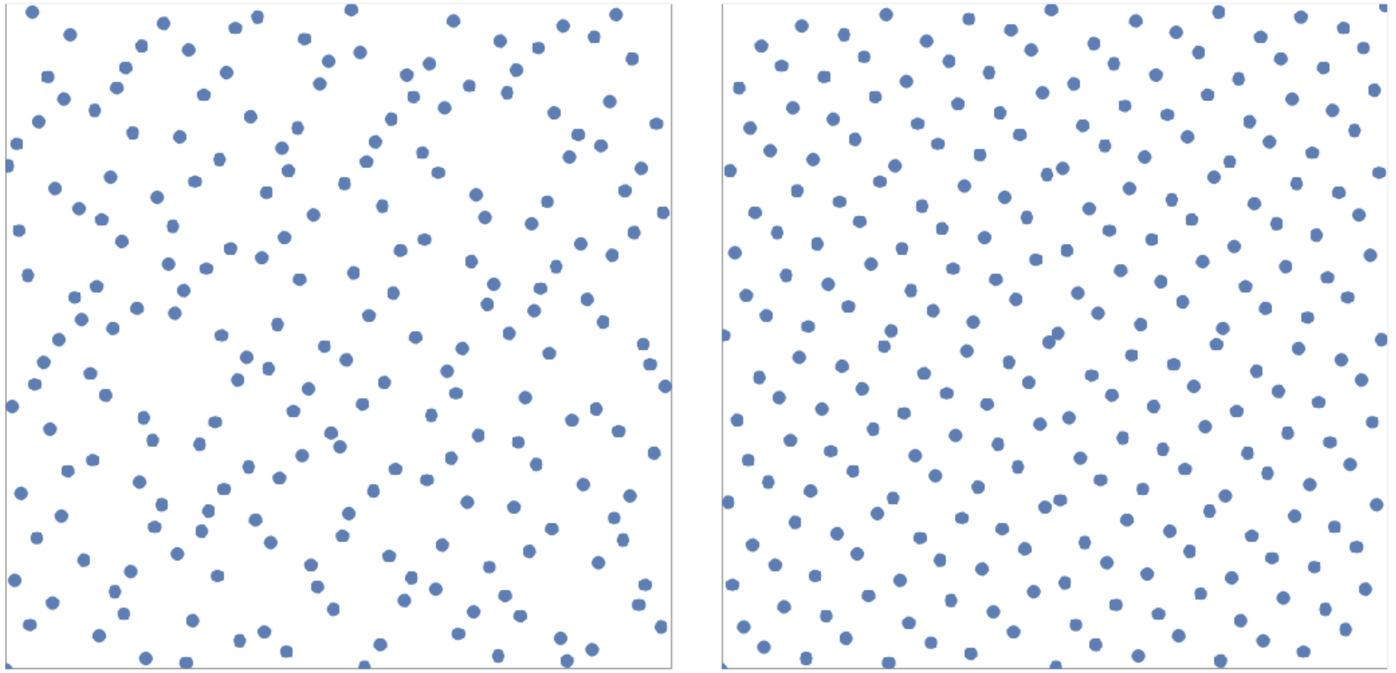


Figure 7.25: The First Points of Two Low-Discrepancy Sequences in 2D. (a) Halton (216 points), (b) Hammersley (256 points).