Efficient Monte Carlo and Quasi-Monte Carlo Rendering Techniques

Alexander Keller, Thomas Kollig, Mateu Sbert, László Szirmay-Kalos

Abstract

The tutorial reviews advanced soft- and hardware rendering techniques that are based on Monte Carlo, quasi-Monte Carlo, and randomized quasi-Monte Carlo methods. The morning session explains the basic theoretical concepts along with the practical algorithmic aspects of Monte Carlo, quasi-Monte Carlo and randomized quasi-Monte Carlo integration. We also focus on error reduction techniques emphasizing importance, correlated, and Metropolis sampling. After reviewing the equations of image synthesis and global illumination in the continuous and discrete setting, the afternoon session is devoted to the practical application of the aforementioned sampling techniques in rendering algorithms. The tutorial presents the advanced tricks of gathering, shooting, and bidirectional random walk methods, and a strikingly simple implementation of the Metropolis light transport algorithm. Concerning improved efficiency, techniques based on reusing light paths are presented including applications like e.g. instant radiosity and photon mapping. The tutorial is completed by production quality quasi-Monte Carlo rendering techniques for anti-aliasing, parallelization, deterministic RenderMan, distribution ray tracing, and interactive global illumination. A basic understanding of rendering terms, equations, and techniques is assumed.

1. Lecturers

Alexander Keller, University of Ulm

Alexander Keller is a professor in computer graphics at the University of Ulm, Germany. He received his Ph.D. with distinction in 1997 at the University of Kaiserslautern. Based on this work he designed and developed the quasi-Monte Carlo techniques behind the rendering software mental ray which is the backend renderer of Maya, 3d Studio Max, CATIA, and many others. Due to its superior performance this rendering software received a technical achievement award in 2003. Alexander Keller is continuously publishing and developing highly efficient quasi-Monte Carlo rendering techniques for almost 10 years now. He had been invited to the CalTech, the ETH Zürich, and the Saarland University for giving courses on his quasi-Monte Carlo techniques.

Contact:

Abt. Medieninformatik, Geb. O27/338
Universität Ulm
D-89069 Ulm, Germany

keller@informatik.uni-ulm.de
medien.informatik.uni-ulm.de/∼keller

Thomas Kollig, University of Kaiserslautern

Thomas Kollig is an expert in Monte Carlo and quasi-Monte Carlo integration. Following several successful international publications, his Ph.D. thesis on randomized quasi-Monte Carlo methods for photorealistic image synthesis is close to submission.

Contact:

Fachbereich Informatik, Geb. 36/208
Universität Kaiserslautern
D-67653 Kaiserslautern, Germany

kollig@informatik.uni-kl.de
www.uni-kl.de/AG-Heinrich/Thomas.html

Mateu Sbert, University of Girona

Mateu Sbert is an associate professor in computer science at the University of Girona. He received a M.Sc. in theoretical physics (1977) at the University of Valencia, a M.Sc. in mathematics (statistics and operations research, 1983) at U.N.E.D. University (Madrid) and his Ph.D. in computer science at the U.P.C. (Universitat Politecnica de Catalunya, 1997, Best Ph.D. Award). Mateu Sbert's research interests include the application of Monte Carlo, integral geometry

and information theory techniques to radiosity, global illumination and image based rendering. He has authored or co-authored about 60 papers in his areas of research and served as a member of program committee in several Spanish and international conferences. Mateu Sbert co-organized the 2001 Dagstuhl Seminar No. 01242, entitled "Stochastic Methods in Rendering".

Contact:

Campus Montilivi-Edifici
University of Girona
PIV 17071 Girona, Spain

mateu@ima.udg.es
ima.udg.es/∼mateu

László Szirmay-Kalos, Budapest University of Technology

László Szirmay-Kalos is the head of the computer graphics group at the Faculty of Electrical Engineering and Information Technology at the Budapest University of Technology and Economics. He received his Ph.D. in 1992 and full professorship in 2001 in computer graphics. His research area is Monte-Carlo global illumination algorithms and he published more than a hundred papers.

Contact:

Magyar Tudósok krt. 2.
Budapest University of Technology
Budapest, H-1117, Hungary

szirmay@iit.bme.hu
www.iit.bme.hu/∼szirmay

2. Syllabus

9.30 - 11.00 Part I

Introduction (Szirmay-Kalos) Why you should use Monte-Carlo and quasi-Monte Carlo integration in your renderer

Monte Carlo integration (Sbert) Variance Reduction: importance sampling, partial analytic integration, correlated sampling, weighted sampling, multiple importance sampling

Quasi-Monte Carlo integration (Keller) Discrepancy and discrete density approximation, algorithms for low discrepancy sampling points, structure of low discrepancy sampling points

11.00 - 11.30 Coffee break

11.30 - 13.00 Part II

Randomized quasi-Monte Carlo integration (Keller) Randomized low discrepancy sampling

Efficient multidimensional sampling (Kollig)

Metropolis sampling (Szirmay-Kalos)

Random walks in the radiosity context (Sbert)

13.00 - 14.30 Lunch break

14.30 - 16.00 Part III

Multipath algorithms (Sbert) A random walk approach with global lines

Random walks (Szirmay-Kalos) The general setting, the art of path building and reuse: distributed ray-tracing, path tracing, light tracing, bi-directional path tracing, photon map, instant radiosity, virtual light sources, resuing path, discontinuity buffer, Metropolis light transport

Stochastic iteration algorithms (Szirmay-Kalos)

16.30 - 18.00 Part IV

Bidirectional path tracing (Kollig) Efficiency from randomized low discrepancy sampling

Quasi-Monte Carlo rendering techniques (Keller) Interleaved sampling and parallelization, efficient volume rendering, strictly deterministic sampling in RenderMan, strictly deterministic path and distribution ray tracing, interactive global illumination

Monte Carlo and Beyond

Course Notes

Alexander Keller

keller@informatik.uni-ulm.de

This course was first held at the Caltech July 30th through August 3rd, 2001.

Early 2002 it was held at the ETH Zürich.

’For every randomized algorithm, there is a clever deterministic one.’

Harald Niederreiter, Claremont, 1998.

’For every randomized algorithm, there is a clever deterministic one.’

Harald Niederreiter, Claremont, 1998.

- no real random on classical deterministic computers
- real random by measuring quantum registers

Monte Carlo and Beyond

- **MC**: Monte Carlo
  - random sampling

- **QMC:** Quasi-Monte Carlo integration
  - low-discrepancy sampling by deterministic nets, sequences, and lattices

Monte Carlo and Beyond

- **MC**: Monte Carlo
  - random sampling

- **QMC:** Quasi-Monte Carlo integration
  - low-discrepancy sampling by deterministic nets, sequences, and lattices

- **RQMC = MC:** Monte Carlo extensions of quasi-Monte Carlo
  - random field synthesis on good lattice points
  - randomized quasi-Monte Carlo integration

Monte Carlo and Beyond

- **MC**: Monte Carlo
  - random sampling

- **QMC:** Quasi-Monte Carlo integration
  - low-discrepancy sampling by deterministic nets, sequences, and lattices

- **RQMC = MC:** Monte Carlo extensions of quasi-Monte Carlo
  - random field synthesis on good lattice points
  - randomized quasi-Monte Carlo integration

- **DRQMC = QMC:** Derandomized randomized quasi-Monte Carlo integration

Applications in Computer Graphics

- **MC**: Industry standard RenderMan by PIXAR
  - stratified random sampling

Applications in Computer Graphics

- **MC**: Industry standard RenderMan by PIXAR
  - stratified random sampling

- **QMC**: Derandomized RenderMan
  - new graphics hardware

Applications in Computer Graphics

- **MC**: Industry standard RenderMan by PIXAR
  - stratified random sampling

- **QMC**: Derandomized RenderMan
  - new graphics hardware

- **RQMC**: Ocean wave synthesis
  - discrete Fourier transform independent of dimension

  RQMC: Error estimation for bidirectional path tracing
  - simpler algorithms

Applications in Computer Graphics

- **MC**: Industry standard RenderMan by PIXAR
  - stratified random sampling

- **QMC**: Derandomized RenderMan
  - new graphics hardware

- **RQMC**: Ocean wave synthesis
  - discrete Fourier transform independent of dimension

  RQMC: Error estimation for bidirectional path tracing
  - simpler algorithms

- **DRQMC**: Industry standard mental ray by mental images
  - deterministic correlated low discrepancy sampling
  - fastest performance

Reengineering the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient

Reengineering the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient

- Uniformity is sufficient
  - low-discrepancy sampling more efficient

Reengineering the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient
- Uniformity is sufficient
  - low-discrepancy sampling more efficient
- Either stratification or Latin hypercube sampling
  - you can have both and even more...

Reengineering the Classics of Computer Graphics

- Uncorrelated sampling
  - correlated sampling more efficient
- Uniformity is sufficient
  - low-discrepancy sampling more efficient
- Either stratification or Latin hypercube sampling
  - you can have both and even more...
- One dimensional stratified Monte Carlo integration
  - Cranley-Patterson rotations more efficient

Reengineering the Classics of Computer Graphics

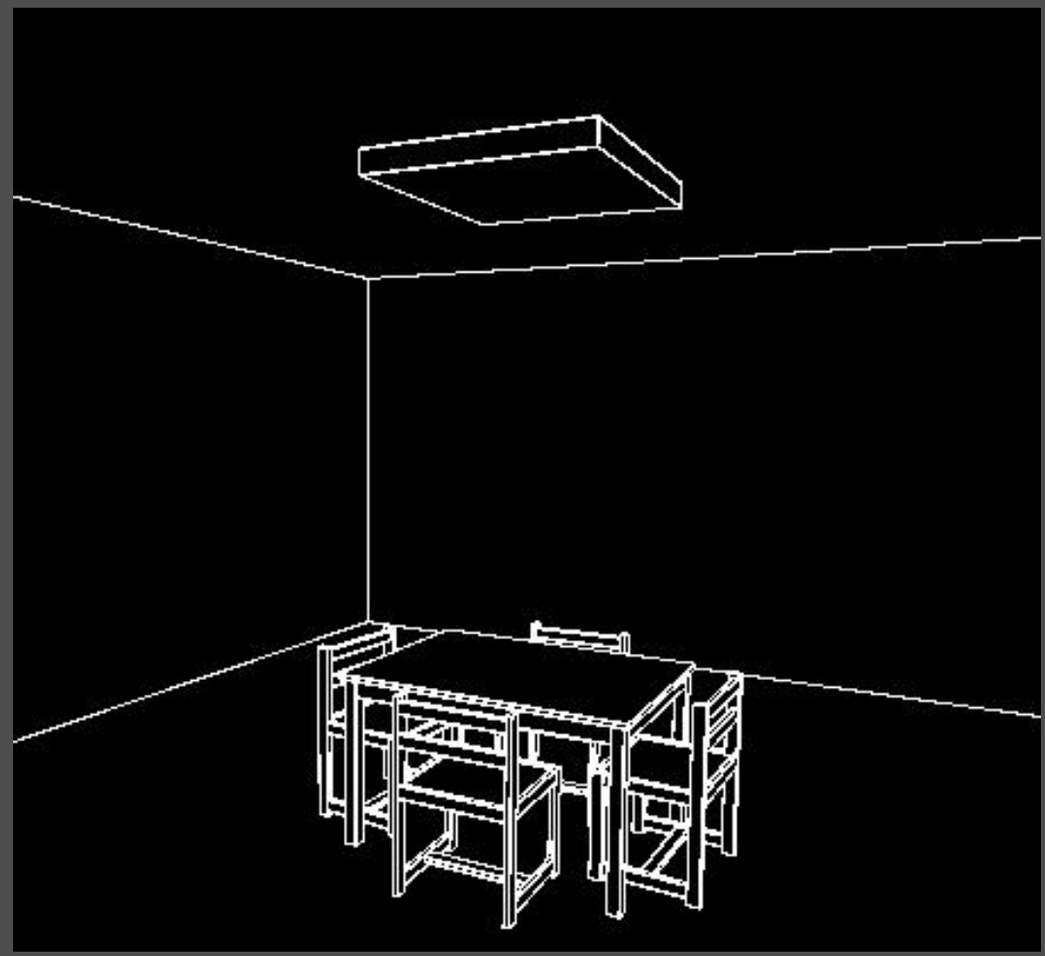
- Uncorrelated sampling
  - correlated sampling more efficient
- Uniformity is sufficient
  - low-discrepancy sampling more efficient
- Either stratification or Latin hypercube sampling
  - you can have both and even more...
- One dimensional stratified Monte Carlo integration
  - Cranley-Patterson rotations more efficient
- Antialiasing only by random sampling
  - deterministic low-discrepancy sampling more efficient

Monte Carlo and Beyond

- **Principles of rendering algorithms**
- Monte Carlo integration
- Quasi-Monte Carlo points
- Quasi-Monte Carlo integration
- Monte Carlo extensions of quasi-Monte Carlo
- Application to computer graphics

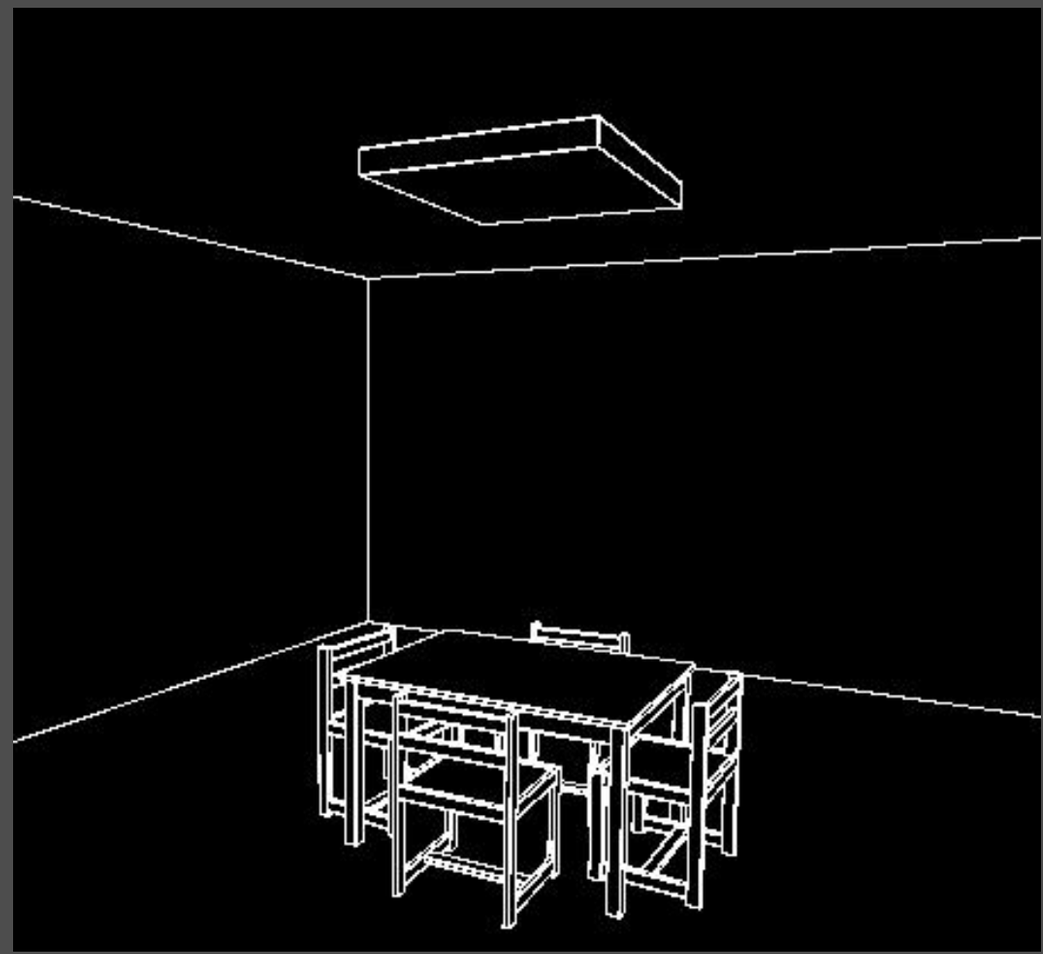
Scene Geometry

- Scene surface $\partial V := \cup_{i=1}^{K} S_i$
  - S_i surface primitive, e.g. triangle
- Scene $V :=$ BoundingBox(∂V)



Scene Geometry

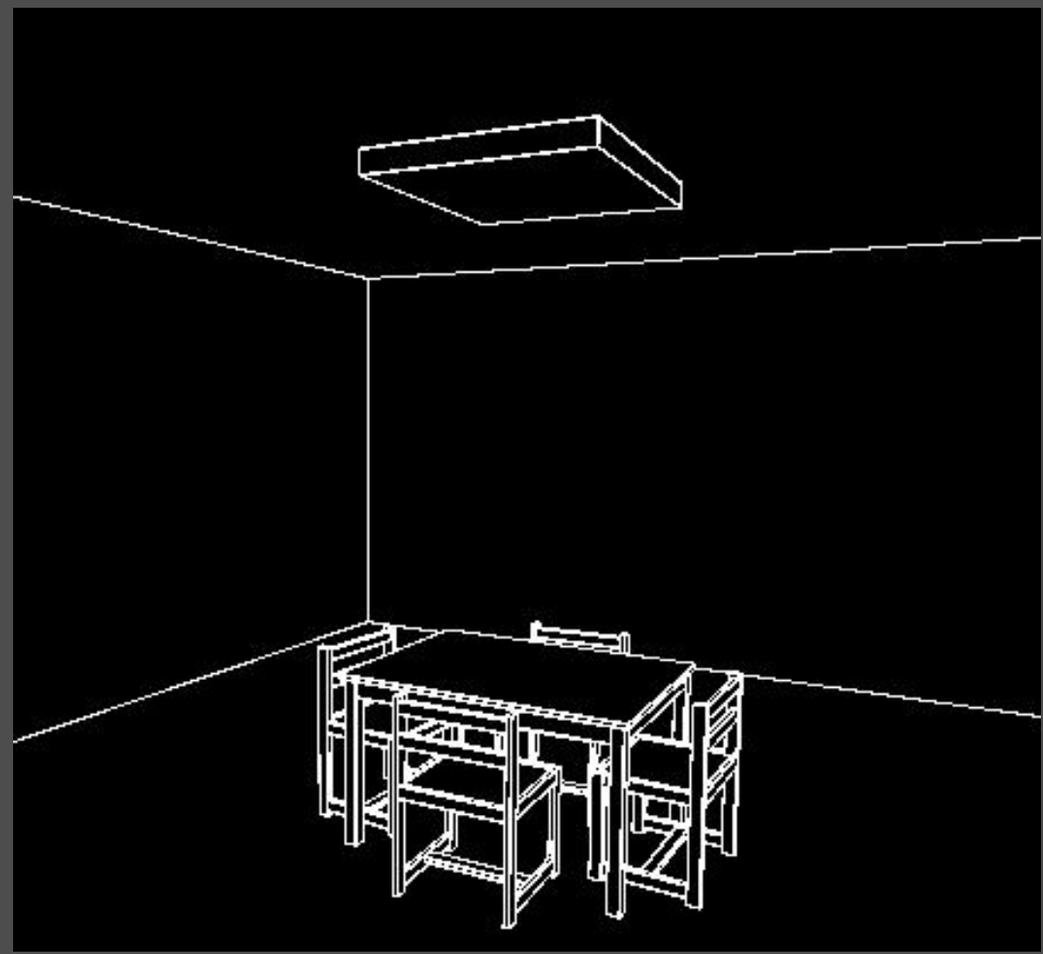
- Scene surface $\partial V := \cup_{i=1}^{K} S_i$
  - S_i surface primitive, e.g. triangle
- Scene $V :=$ BoundingBox(∂V)



- Set Ω of all unit directions ω
  - surface of a unit sphere
- Surface normal $\hat{n} : \partial V \rightarrow \Omega$

Scene Geometry

- Scene surface $\partial V := \cup_{i=1}^{K} S_i$
  - S_i surface primitive, e.g. triangle
- Scene $V :=$ BoundingBox(∂V)

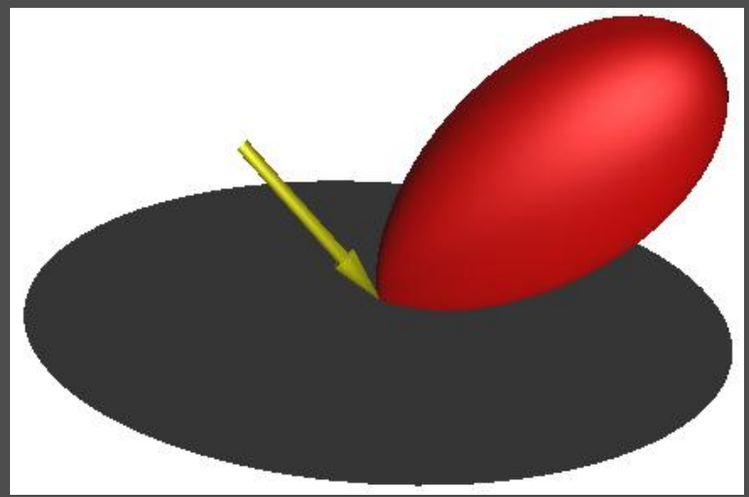


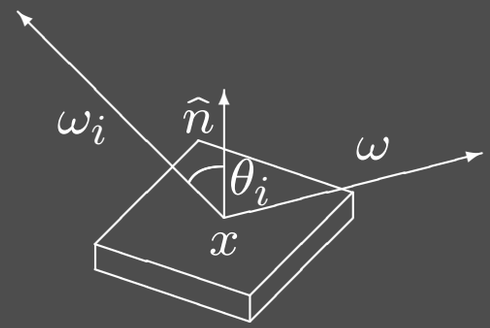
- Set Ω of all unit directions ω
  - surface of a unit sphere
- Surface normal $\hat{n} : \partial V \rightarrow \Omega$

- Ray $(x, \omega) \in V \times \Omega$
- Hitpoint $h : V \times \Omega \rightarrow \partial V \cup \{\infty\}$
  - first surface point hit, when shooting a ray from x into direction ω

Interaction of Light and Matter

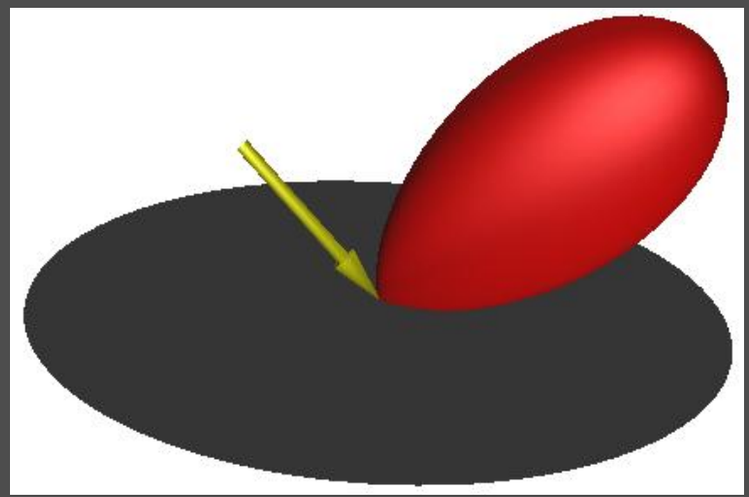
- Bidirectional scattering distribution function $f_s(\omega_i, x, \omega) : \Omega \times \partial V \times \Omega \to \mathbb{R}_0^+$
  - may depend on wavelength
  - Helmholtz reciprocity principle $f_s(\omega_i, x, \omega) = f_s(\omega, x, \omega_i)$

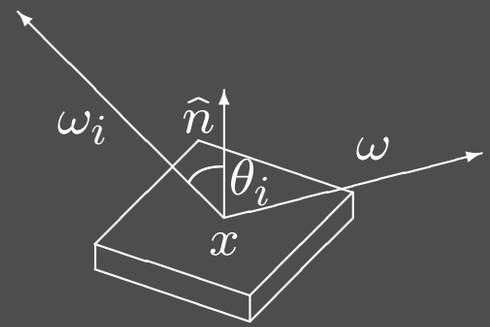




Interaction of Light and Matter

- Bidirectional scattering distribution function $f_s(\omega_i, x, \omega) : \Omega \times \partial V \times \Omega \to \mathbb{R}_0^+$
  - may depend on wavelength
  - Helmholtz reciprocity principle $f_s(\omega_i, x, \omega) = f_s(\omega, x, \omega_i)$



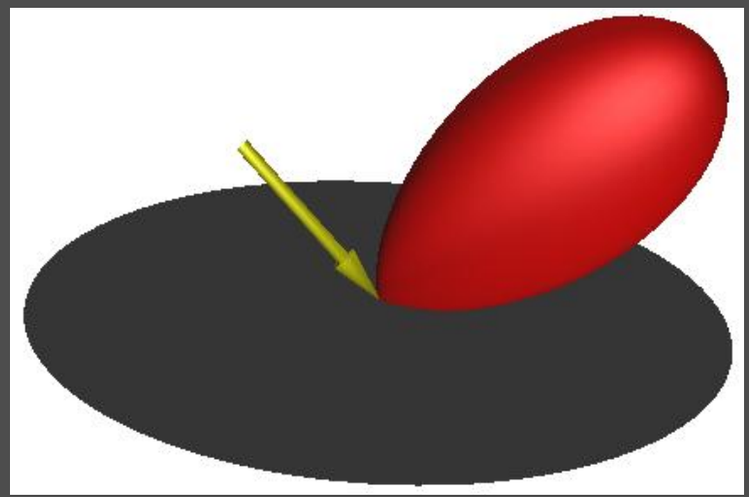


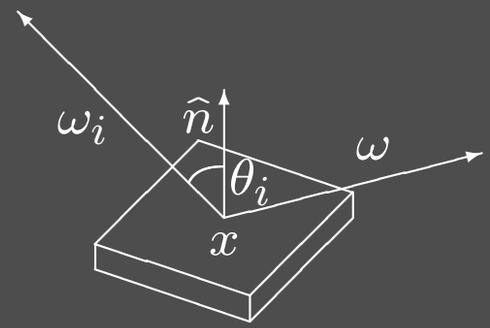
- Scattered radiance

$$L_s(x, \omega) = \int_\Omega f_s(\omega_i, x, \omega) L_i(x, \omega_i) |\hat{n}(x) \cdot \omega_i| d\omega_i$$

Interaction of Light and Matter

- Bidirectional scattering distribution function $f_s(\omega_i, x, \omega) : \Omega \times \partial V \times \Omega \to \mathbb{R}_0^+$
  - may depend on wavelength
  - Helmholtz reciprocity principle $f_s(\omega_i, x, \omega) = f_s(\omega, x, \omega_i)$





- Scattered radiance

$$\begin{aligned} L_s(x,\omega) &= \int_\Omega f_s(\omega_i, x, \omega) L_i(x, \omega_i) |\hat{n}(x) \cdot \omega_i| d\omega_i \\ &= \int_\Omega f_s(\omega_i, x, \omega) L_i(x, \omega_i) \cos\theta_i d\omega_i \end{aligned}$$

- Integral operator shorthand

$$L_s = T_{f_s} L_i$$

Vacuum Radiance Transport

- Emitted radiance $L_e(x, \omega) : \partial V \times \Omega \rightarrow \mathbb{R}_0^+$

Vacuum Radiance Transport

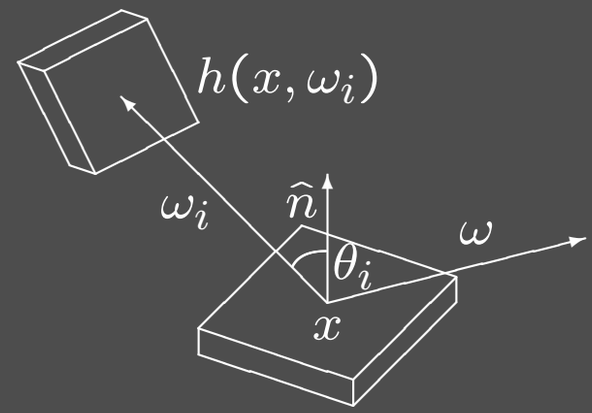
- Emitted radiance $L_e(x, \omega) : \partial V \times \Omega \rightarrow \mathbb{R}_0^+$
- Looking for $L(x, \omega) : V \times \Omega \rightarrow \mathbb{R}_0^+$
  - usually in RGB color space

Vacuum Radiance Transport

- Emitted radiance $L_e(x,\omega) : \partial V \times \Omega \rightarrow \mathbb{R}_0^+$
- Looking for $L(x,\omega) : V \times \Omega \rightarrow \mathbb{R}_0^+$
  - usually in RGB color space
  - in vacuum $L(x,\omega) = L(h(x,-\omega),\omega)$

  $\Rightarrow$ sufficient to consider radiance for $x \in \partial V$

$$L(x,\omega) = L_e(x,\omega) + L_s(x,\omega)$$
$$L_i(x,\omega_i) = L(h(x,\omega_i),-\omega_i)$$

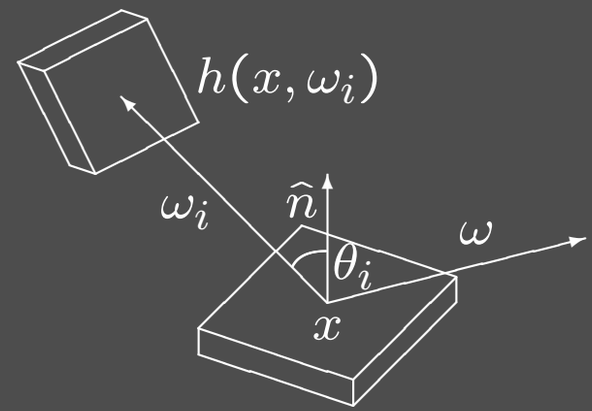


Vacuum Radiance Transport

- Emitted radiance $L_e(x,\omega) : \partial V \times \Omega \rightarrow \mathbb{R}_0^+$
- Looking for $L(x,\omega) : V \times \Omega \rightarrow \mathbb{R}_0^+$
  - usually in RGB color space
  - in vacuum $L(x,\omega) = L(h(x,-\omega),\omega)$

  $\Rightarrow$ sufficient to consider radiance for $x \in \partial V$

$$\begin{aligned} L(x,\omega) &= L_e(x,\omega) + L_s(x,\omega) \\ L_i(x,\omega_i) &= L(h(x,\omega_i),-\omega_i) \end{aligned}$$



- Radiance integral equation

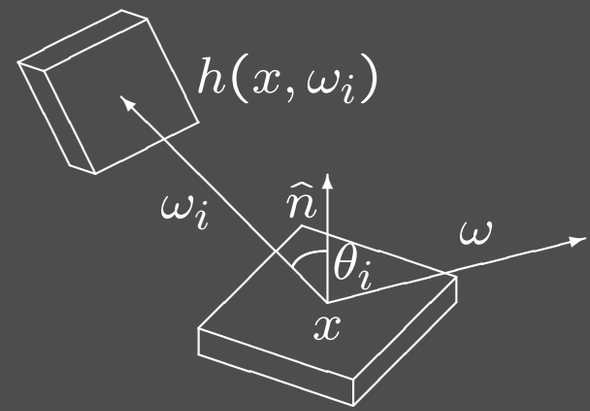
$$\begin{aligned} L(x,\omega) &= L_e(x,\omega) + \int_\Omega f_s(\omega_i,x,\omega) L(h(x,\omega_i),-\omega_i) \cos\theta_i d\omega_i \\ L &= L_e + T_{f_s} L \end{aligned}$$

Vacuum Radiance Transport

- Emitted radiance $L_e(x,\omega) : \partial V \times \Omega \to \mathbb{R}_0^+$
- Looking for $L(x,\omega) : V \times \Omega \to \mathbb{R}_0^+$
  - usually in RGB color space
  - in vacuum $L(x,\omega) = L(h(x,-\omega),\omega)$

  $\Rightarrow$ sufficient to consider radiance for $x \in \partial V$

$$\begin{aligned} L(x,\omega) &= L_e(x,\omega) + L_s(x,\omega) \\ L_i(x,\omega_i) &= L(h(x,\omega_i),-\omega_i) \end{aligned}$$



- Radiance integral equation

$$\begin{aligned} L(x,\omega) &= L_e(x,\omega) + \int_\Omega f_s(\omega_i,x,\omega) L(h(x,\omega_i),-\omega_i) \cos\theta_i d\omega_i \\ L &= L_e + T_{f_s} L \end{aligned}$$

- Neumann series, convergent if $\|T_{f_s}^\alpha\| < 1$

$$\begin{aligned} L &= L_e + T_{f_s} L_e + T_{f_s}^2 L_e + \cdots \\ &= \sum_{i=0}^{\infty} T_{f_s}^i L_e =: (I - T_{f_s})^{-1} L_e \end{aligned}$$

Image Synthesis

- Flux responsivity $W : V \times \Omega \to \mathbb{R}$
- Measurement

$$\begin{aligned} \int_V \int_\Omega W(x,\omega) L(x,\omega) d\omega dx &=: \langle W, L \rangle \\ &= \langle W, (I - T_{f_s})^{-1} L_e \rangle \end{aligned}$$

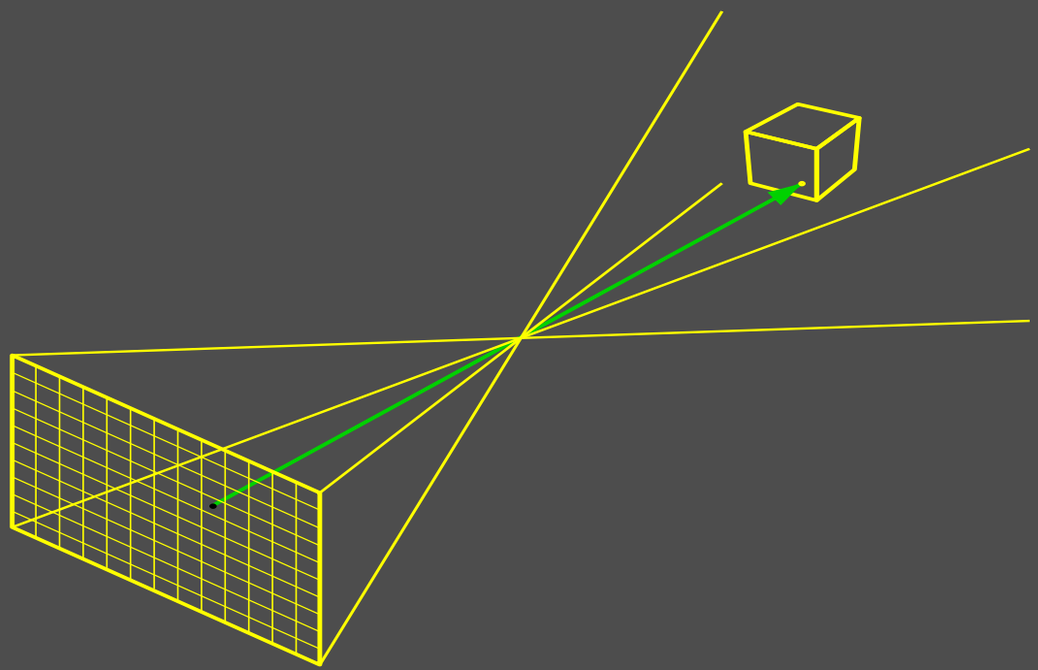
Image Synthesis

- Flux responsivity $W : V \times \Omega \to \mathbb{R}$
- Measurement

$$\int_V \int_\Omega W(x,\omega) L(x,\omega) d\omega dx \quad =: \quad \langle W, L \rangle$$

$$= \quad \langle W, (I - T_{f_s})^{-1} L_e \rangle$$

- Example: Pixelsensors $W_{m,n}$ of a pinhole camera
  - detects average radiance passing through a pixel



The Global Illumination Problem in Vacuum

- Given the
  - scene surface ∂V,
  - scattering properties f_s,
  - radiance emission L_e, and
  - a sensor W
- compute

$$\langle W, (I - T_{f_s})^{-1} L_e \rangle$$

$\Rightarrow$ the global illumination problem is reduced to an integration problem

Principles of Rendering Algorithms

- **Pipeline:** Transformation, opt. cull, shade, clip, rasterize with Z-buffer
  - Rasterization hardware
    - e.g. nVidea, ATI, Matrox

Principles of Rendering Algorithms

- **Pipeline:** Transformation, opt. cull, shade, clip, rasterize with Z-buffer
  - Rasterization hardware
    - e.g. nVidea, ATI, Matrox

- **Pipeline:** Split, cull, dice, shade micro-polygons, cull, cast rays with Z-buffer
  - RenderMan (REYES) ray caster
    - PIXAR, California

Principles of Rendering Algorithms

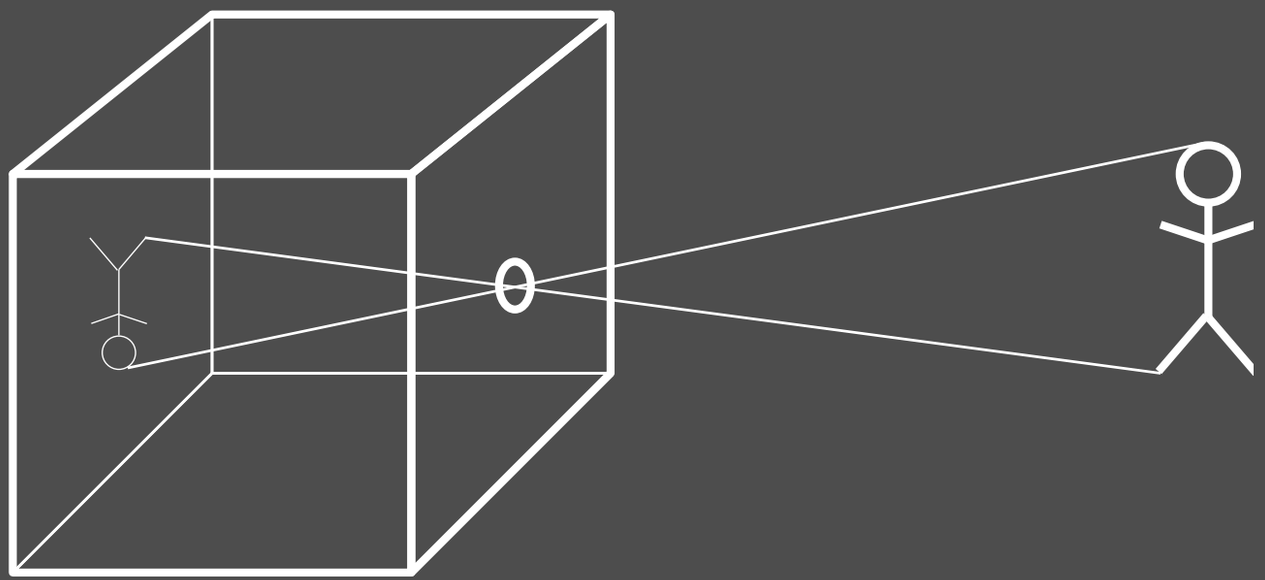
- **Pipeline:** Transformation, opt. cull, shade, clip, rasterize with Z-buffer
  - Rasterization hardware
    - e.g. nVidea, ATI, Matrox

- **Pipeline:** Split, cull, dice, shade micro-polygons, cull, cast rays with Z-buffer
  - RenderMan (REYES) ray caster
    - PIXAR, California

- **Pipeline:** Trace ray by culling, shade, recurse $\Rightarrow$ **no streaming**
  - Entropy (BMRT, Torro) ray tracer with analytic anti-aliasing
    - Exluna, California
  - mental ray
    - mental images, Berlin

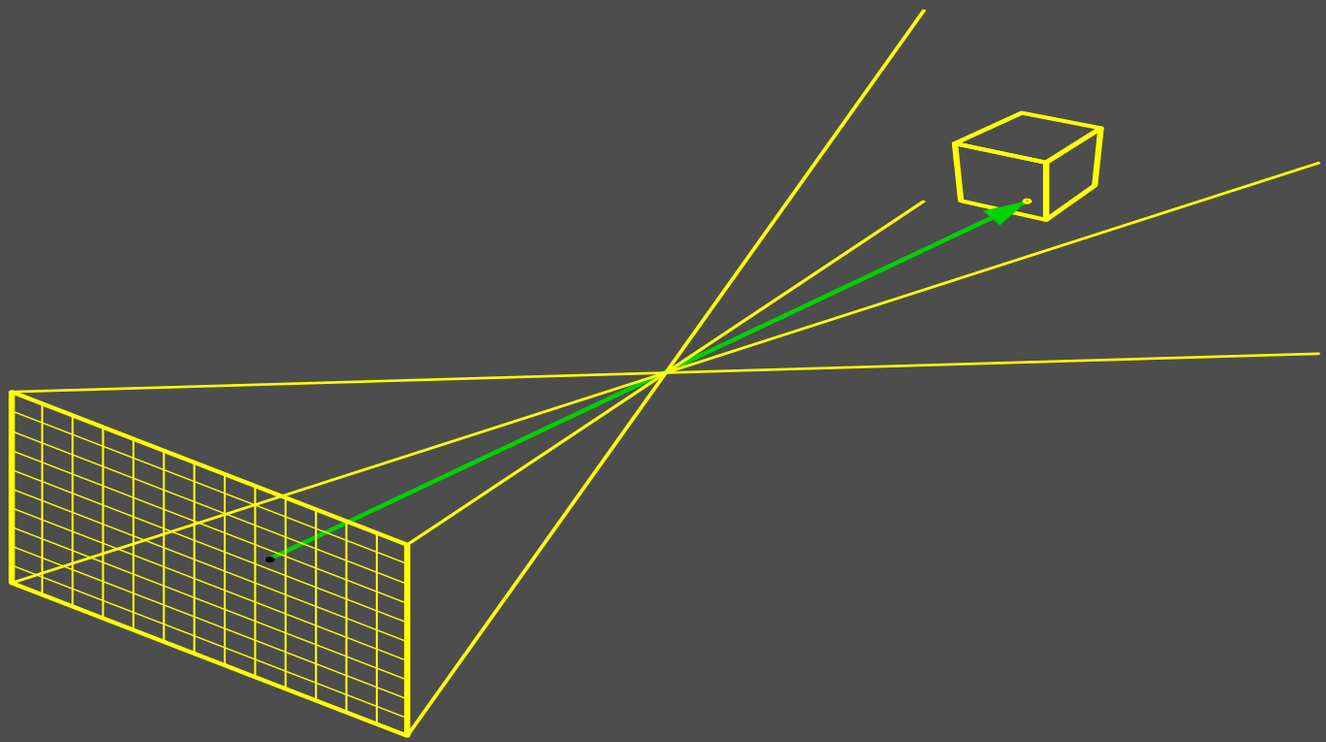
The Pinhole Camera: Camera Obscura

- Central projection onto image plane



Ray Tracing

- Image: Matrix of pixels
  - Pix-el = **Pict**ure **El**ement
- **1980:** Turner Whitted: An Improved Illumination Model for Shaded Display.



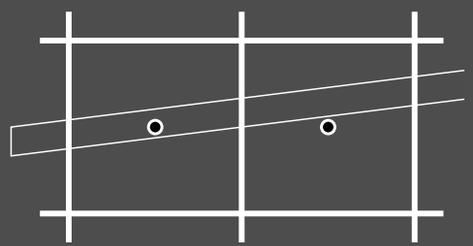
- Trace ray from center of pixel through focal point into the scene

A simple Ray Tracing Program: Sampling

```
#include ”Graphics.h”

int main(int Parameters, char **Parameter)
{
    Image* Picture;
    Color Sample;
    Color SumOfSamples;

    Initialize(Parameters, Parameter);
    Picture = new Image(SizeX, SizeY);

    for (int x = 0; x < SizeX; x++)
        for (int y = 0; y < SizeY; y++)
        {
            Sample = Shade(x + 0.5, y + 0.5);
            Picture→Pixel(x, y) = Sample;
        }

    SaveImage(Picture);

    return 0;
}
```

Observation: Aliasing

- The image contains jaggies.



Anti-Aliasing by Supersampling

- In fact the pixel is an area, not a point !

  $\Rightarrow$ pixel color is average not a single sample

$$\text{pixel color} = \frac{1}{|P|}\int_P L(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} L(x_i)$$

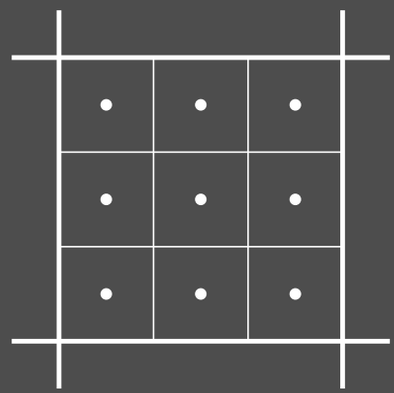
Anti-Aliasing by Supersampling

- In fact the pixel is an area, not a point !

  $\Rightarrow$ pixel color is average not a single sample

$$\text{pixel color} = \frac{1}{|P|}\int_P L(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} L(x_i)$$

- Multiple samples instead of only pixel center

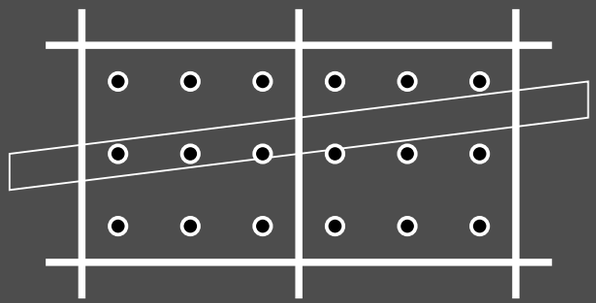


axis-aligned, regular grid

Supersampling

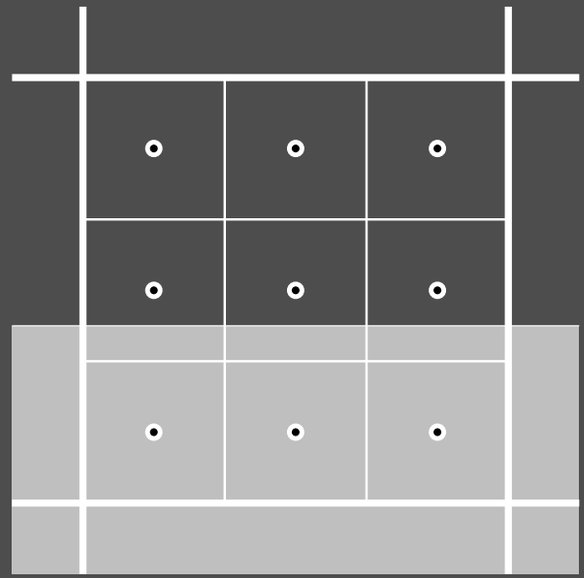
```
⋮
for (int x = 0; x < SizeX; x++)
    for (int y = 0; y < SizeY; y++)
    {
        SumOfSamples = Black;

        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
            {
                Sample = Shade(x + ((double) i + 0.5) / 3.0, y + ((double) j + 0.5) / 3.0);
                SumOfSamples = SumOfSamples + Sample;
            }

        Picture→Pixel(x, y) = SumOfSamples / 9;
    }
⋮
```

Observation 1: Reduced Aliasing



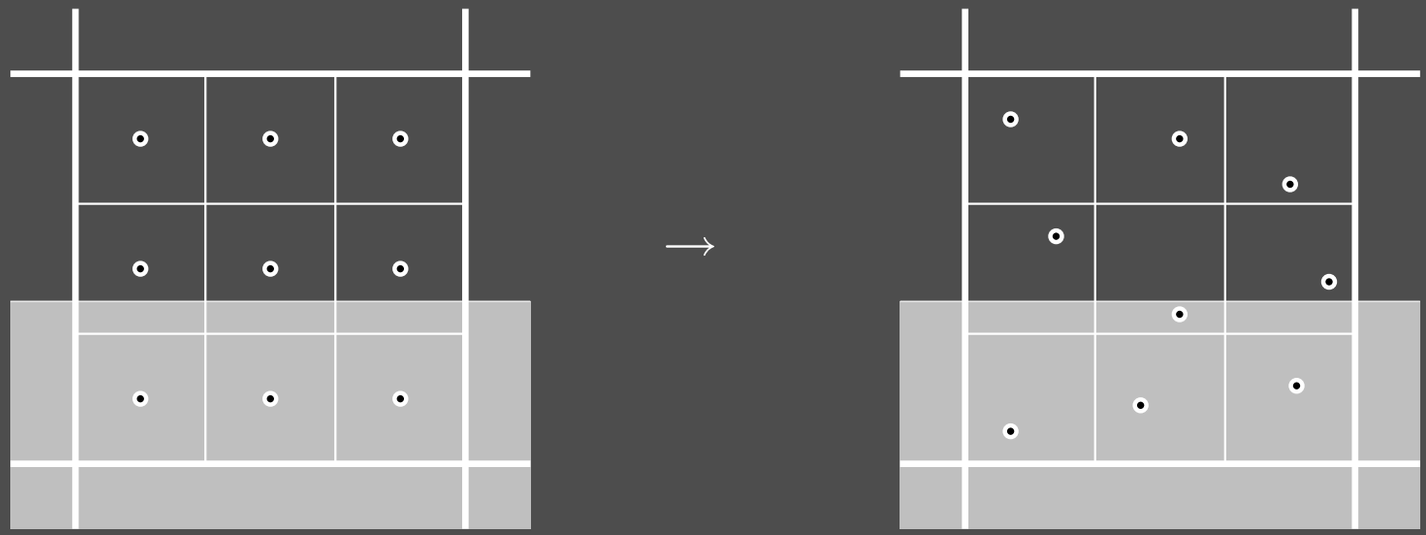
Observation 2: Still Aliasing

- since the 9 points can behave like only 3



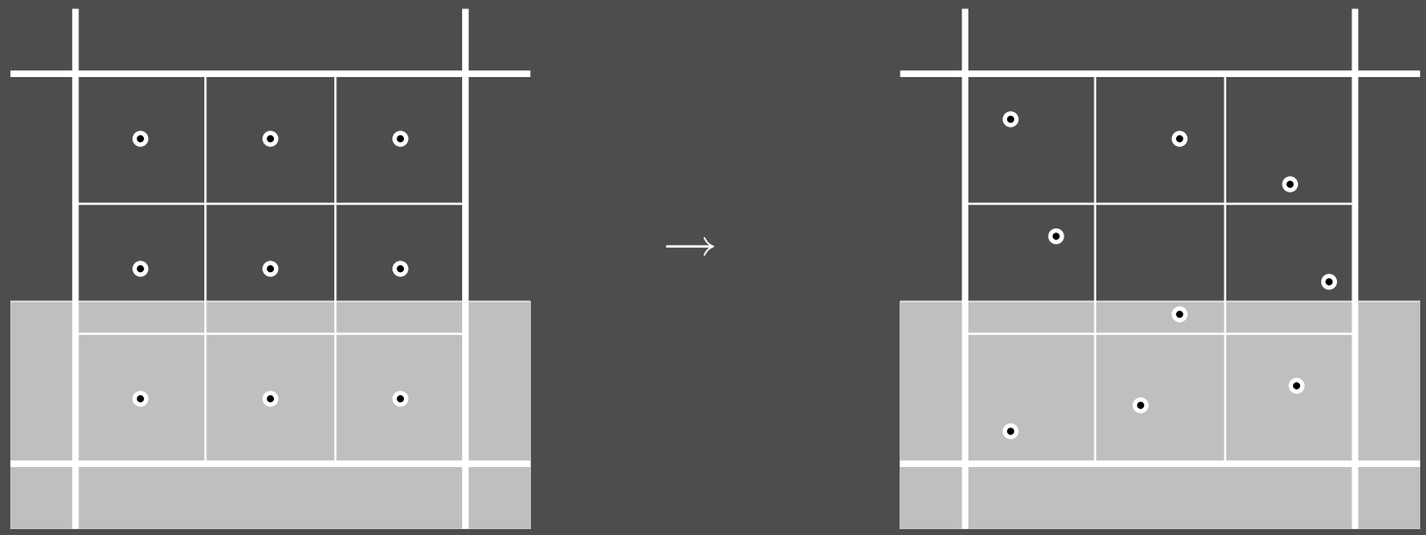
Introducing Randomness

- Jittering



Introducing Randomness

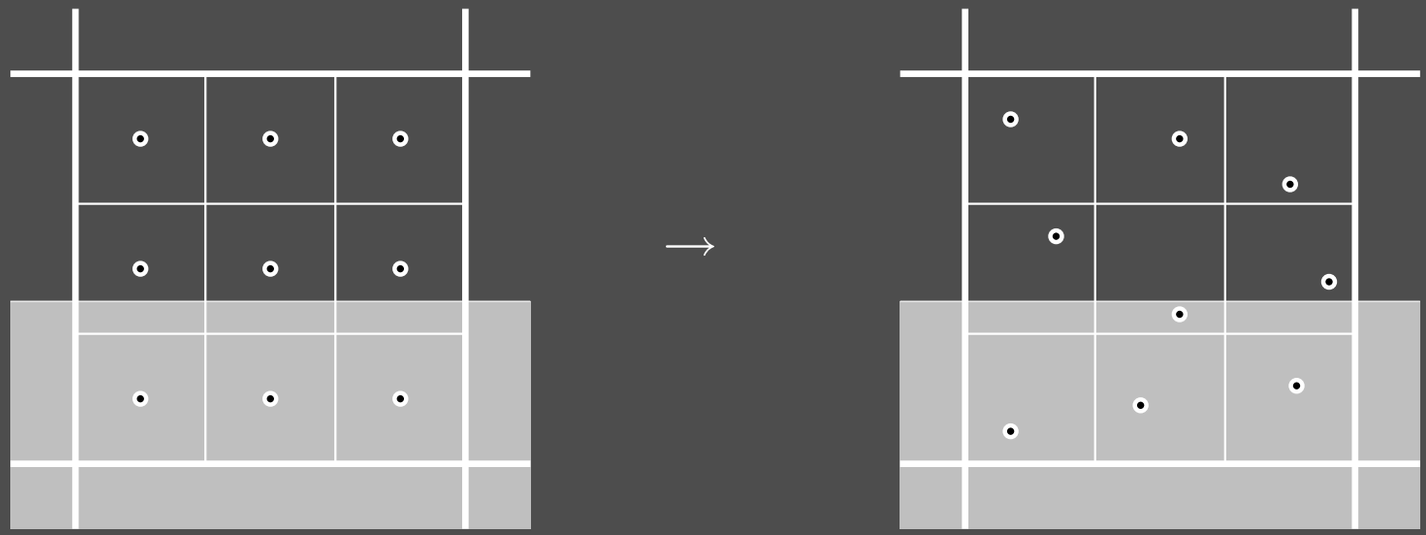
- Jittering



$\Rightarrow$ use random numbers

Introducing Randomness

- Jittering



$\Rightarrow$ use random numbers

- Estimation by throwing the dice is superior !

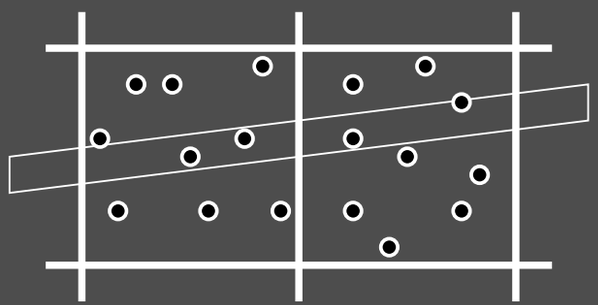
  $\Rightarrow$ Monte Carlo algorithms

Stochastic Supersampling

```
⋮
for (int x = 0; x < SizeX; x++)
    for (int y = 0; y < SizeY; y++)
    {
        SumOfSamples = Black;

        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
            {
                Sample = Shade(x + ((double) i + drand48()) / 3.0,
                               y + ((double) j + drand48()) / 3.0);
                SumOfSamples = SumOfSamples + Sample;
            }

        Picture→Pixel(x, y) = SumOfSamples / 9;
    }
⋮
```

Antialiasing by Stochastic Supersampling

- Noise instead of aliasing



Monte Carlo and Beyond

- Principles of rendering algorithms
- **Monte Carlo integration**
  - Simulation of random variables and fields
  - Monte Carlo integration
  - Method of dependent tests
  - Multilevel method of dependent tests
  - Dependent sampling
  - Replication heuristics
  - Regularization of the samples
- Quasi-Monte Carlo points
- Quasi-Monte Carlo integration
- Monte Carlo extensions of quasi-Monte Carlo
- Application to computer graphics

Probability Spaces, Random Variables and Random Fields

- **Definition:** A ***probability space*** is given by a set $\Omega = \{\omega_1, \omega_2, \ldots\}$ of ***elementary events*** ω_i, where each elementary event is assigned a probability with

  $$0 \leq \textbf{Prob}(\omega_i) \leq 1 \qquad \text{and} \qquad \sum_{\omega \in \Omega} \textbf{Prob}(\omega) = 1.$$

  $E \subseteq \Omega$ is called ***event*** with

  $$\textbf{Prob}(E) = \sum_{\omega \in E} \textbf{Prob}(\omega).$$

Probability Spaces, Random Variables and Random Fields

- **Definition:** A ***probability space*** is given by a set $\Omega = \{\omega_1, \omega_2, \ldots\}$ of ***elementary events*** ω_i, where each elementary event is assigned a probability with

  $$0 \leq \textbf{Prob}(\omega_i) \leq 1 \qquad \text{and} \qquad \sum_{\omega \in \Omega} \textbf{Prob}(\omega) = 1.$$

  $E \subseteq \Omega$ is called ***event*** with

  $$\textbf{Prob}(E) = \sum_{\omega \in E} \textbf{Prob}(\omega).$$

- **Definition:** Given a probability space on the set of elementary events Ω, a mapping

  $$\begin{aligned} X : \Omega &\rightarrow \mathbb{R} \\ \omega &\mapsto X_\omega \end{aligned}$$

  is called a ***random variable***. X_ω is called a ***realization***.

Probability Spaces, Random Variables and Random Fields

- **Definition:** A ***probability space*** is given by a set $\Omega = \{\omega_1, \omega_2, \ldots\}$ of ***elementary events*** ω_i, where each elementary event is assigned a probability with

  $$0 \leq \textbf{Prob}(\omega_i) \leq 1 \qquad \text{and} \qquad \sum_{\omega \in \Omega} \textbf{Prob}(\omega) = 1.$$

  $E \subseteq \Omega$ is called ***event*** with

  $$\textbf{Prob}(E) = \sum_{\omega \in E} \textbf{Prob}(\omega).$$

- **Definition:** Given a probability space on the set of elementary events Ω, a mapping

  $$\begin{aligned} X : \Omega &\rightarrow \mathbb{R} \\ \omega &\mapsto X_\omega \end{aligned}$$

  is called a ***random variable***. X_ω is called a ***realization***.

- **Definition:** A ***random field*** (also called ***random function***)

  $$\begin{aligned} X : \Omega &\rightarrow C(s, d) \\ \omega &\mapsto X_\omega \end{aligned}$$

  maps the space of elementary events Ω into the space of continuous functions $C(s, d)$. If $s = 1$ the random fields can be called ***random process***.

Discrete Random Variables

- **Definition:** If the probability space Ω is finite or countable, the random variable X is ***discrete***.

$$\begin{aligned} P_X : \mathbb{R} &\to [0,1] \\ x &\mapsto \textbf{Prob}(X \leq x) = \sum_{x' \leq x} \textbf{Prob}(X = x') \end{aligned}$$

  is called ***cumulative distribution function (cdf)*** of the random variable X.

Continuous Random Variables

- **Definition:** A ***continuous random variable*** X and its underlying (real) probability space are defined by an integrable density function

  $$p_X : \mathbb{R} \to \mathbb{R}_0^+$$

  with the property $\int_{\mathbb{R}} p_X(x)dx = 1$. A set $A \subseteq \mathbb{R}$ that can be built by the union $A = \cup_k I_k$ of countably many pair-wise disjoint intervals of arbitrary kind (open, closed, half-open, one-sided infinite) is called ***event***. X takes a value from A with

  $$\textbf{Prob}(A) = \int_A p_X(x)dx = \sum_k \int_{I_k} p_X(x)dx.$$

  The ***cumulative distribution function (cdf)*** is

  $$P_X(x) = \textbf{Prob}(X \leq x) = \textbf{Prob}(\{t \in \mathbb{R} | t \leq x\}) = \int_{-\infty}^{x} p_X(t)dt.$$

- Properties of the cumulative distribution function
  - monotonicity and continuity
  - $\lim_{x \to -\infty} P_X(x) = 0$
  - $\lim_{x \to \infty} P_X(x) = 1$

- Properties of the cumulative distribution function
  - monotonicity and continuity
  - $\lim_{x \to -\infty} P_X(x) = 0$
  - $\lim_{x \to \infty} P_X(x) = 1$
- **Corollary:** Any differentiable function P that fulfills the above properties can be assigned a probability density function by

$$p = P'(x).$$

Uniform Distribution $\mathcal{U}$ on $[0,1)^s$

- Probability density function

$$p_{\mathcal{U}}(x) = \begin{cases} 1 & x \in [0,1)^s \\ 0 & \text{else} \end{cases}$$

Uniform Distribution $\mathcal{U}$ on $[0,1)^s$

- Probability density function

$$p_{\mathcal{U}}(x) = \begin{cases} 1 & x \in [0,1)^s \\ 0 & \text{else} \end{cases}$$

- Requirements for simulation, i.e. realization
  - fast, deterministic algorithms
  - mimic independence

    $\Rightarrow$ pseudo-random numbers

Uniform Distribution $\mathcal{U}$ on $[0,1)^s$

- Probability density function

$$p_{\mathcal{U}}(x) = \begin{cases} 1 & x \in [0,1)^s \\ 0 & \text{else} \end{cases}$$

- Requirements for simulation, i.e. realization
  - fast, deterministic algorithms
  - mimic independence

  $\Rightarrow$ pseudo-random numbers

- Example: Linear congruential generators (starting value z_0)

$$\begin{aligned} z_{i+1} &= (az_i + c) \bmod m \qquad \in \{0, \ldots, m-1\} \\ \xi_{i+1} &= \frac{z_{i+1}}{m} \end{aligned}$$

Uniform Distribution $\mathcal{U}$ *on* $[0,1)^s$

- Probability density function

$$p_{\mathcal{U}}(x) = \begin{cases} 1 & x \in [0,1)^s \\ 0 & \text{else} \end{cases}$$

- Requirements for simulation, i.e. realization
  - fast, deterministic algorithms
  - mimic independence

    $\Rightarrow$ pseudo-random numbers

- Example: Linear congruential generators (starting value z_0)

$$\begin{aligned} z_{i+1} &= (az_i + c) \bmod m \qquad \in \{0, \ldots, m-1\} \\ \xi_{i+1} &= \frac{z_{i+1}}{m} \end{aligned}$$

  - discrete subset of $[0,1)$
  - finite period
  - choice of a, c, m crucial for good statistical properties
  - parallelization difficult

The Inversion Method

- Given a density $p(x) > 0$ on $[0, 1]$ generate samples y that are p-distributed
- Determine

$$x = P(y) = \frac{\int_0^y p(\tau) d\tau}{\int_0^1 p(\tau) d\tau} \in [0, 1]$$

and use

$$y_i = P^{-1}(x_i)$$

if P is invertible.

The Multidimensional Inversion Method

- For $p(x) > 0$ for $x \in I^s$ and $\int_{I^s} p(x)dx < \infty$ realize p-distributed samples

  $$P^{-1}(x) := (y^{(1)}, \ldots, y^{(s)}) = y$$

  from $x \sim \mathcal{U}$ by successively determining

  $$\begin{aligned} y^{(1)} \quad &\text{using} \quad x^{(1)} = F_1(y^{(1)}) \,, \\ y^{(2)} \quad &\text{using} \quad x^{(2)} = F_2(y^{(1)}, y^{(2)}) \\ &\vdots \end{aligned}$$

  using the bijections

  $$F_j(t_1, \ldots, t_j) := \frac{\int_0^{t_j} \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}{\int_0^1 \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}$$

The Multidimensional Inversion Method

- For $p(x) > 0$ for $x \in I^s$ and $\int_{I^s} p(x)dx < \infty$ realize p-distributed samples

  $$P^{-1}(x) := (y^{(1)}, \ldots, y^{(s)}) = y$$

  from $x \sim \mathcal{U}$ by successively determining

  $$\begin{aligned} y^{(1)} \quad &\text{using} \quad x^{(1)} = F_1(y^{(1)}) \,, \\ y^{(2)} \quad &\text{using} \quad x^{(2)} = F_2(y^{(1)}, y^{(2)}) \\ &\vdots \end{aligned}$$

  using the bijections

  $$F_j(t_1, \ldots, t_j) := \frac{\int_0^{t_j} \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}{\int_0^1 \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}$$

- If $p(x) = \prod_{j=1}^s p^{(j)}(x^{(j)})$

  $$F_j(t_j) = \frac{\int_0^{t_j} p^{(j)}(\tau) d\tau}{\int_0^1 p^{(j)}(\tau) d\tau}$$

The Multidimensional Inversion Method

- For $p(x) > 0$ for $x \in I^s$ and $\int_{I^s} p(x)dx < \infty$ realize p-distributed samples

  $$P^{-1}(x) := (y^{(1)}, \ldots, y^{(s)}) = y$$

  from $x \sim \mathcal{U}$ by successively determining

  $$\begin{aligned} y^{(1)} \quad &\text{using} \quad x^{(1)} = F_1(y^{(1)}) \,, \\ y^{(2)} \quad &\text{using} \quad x^{(2)} = F_2(y^{(1)}, y^{(2)}) \\ &\vdots \end{aligned}$$

  using the bijections

  $$F_j(t_1, \ldots, t_j) := \frac{\int_0^{t_j} \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}{\int_0^1 \int_0^1 \cdots \int_0^1 p(t_1, \ldots, t_{j-1}, \tau_j, \ldots, \tau_s) d\tau_j \cdots d\tau_s}$$

- If $p(x) = \prod_{j=1}^s p^{(j)}(x^{(j)})$

  $$F_j(t_j) = \frac{\int_0^{t_j} p^{(j)}(\tau) d\tau}{\int_0^1 p^{(j)}(\tau) d\tau}$$

- **Note:** P^{-1} not unique, since there exist many mappings of the unit cube onto itself

Composition Method

- Simulation of composite probability density functions

$$p(x) = \sum_{i=1}^{K} w_i p_i(x) \qquad w_i \in \mathbb{R}^+, \sum_{i=1}^{K} w_i = 1$$

1. fix index i using $\xi \sim \mathcal{U}$

$$\sum_{j=1}^{i-1} w_j \leq \xi < \sum_{j=1}^{i} w_j,$$

   i.e. simulate a discrete random variable with **Prob**$(\omega_i) = w_i$

2. efficiently simulate p_i by

$$\frac{\xi - \sum_{j=1}^{i-1} w_j}{w_i} \in I$$

   using only one random number

Composition Method

- Simulation of composite probability density functions

$$p(x) = \sum_{i=1}^{K} w_i p_i(x) \qquad w_i \in \mathbb{R}^+, \sum_{i=1}^{K} w_i = 1$$

1. fix index i using $\xi \sim \mathcal{U}$

$$\sum_{j=1}^{i-1} w_j \leq \xi < \sum_{j=1}^{i} w_j,$$

   i.e. simulate a discrete random variable with **Prob**$(\omega_i) = w_i$

2. efficiently simulate p_i by

$$\frac{\xi - \sum_{j=1}^{i-1} w_j}{w_i} \in I$$

   using only one random number

- **Note:** The composition method can raise variance.
- Applications: Russian Roulette, stochastic evaluation of sums

Selection Methods

- Neumann rejection method, if $\|p\|_\infty < b < \infty$
  - Choose two independent realizations of uniform random numbers $\xi, \zeta \sim \mathcal{U}$
  - If $p(\xi) > b\zeta$ take ξ as a sample
  - else reject ξ and try again
- Efficiency depends on graph of p

Selection Methods

- Neumann rejection method, if $\|p\|_\infty < b < \infty$
  - Choose two independent realizations of uniform random numbers $\xi, \zeta \sim \mathcal{U}$
  - If $p(\xi) > b\zeta$ take ξ as a sample
  - else reject ξ and try again
- Efficiency depends on graph of p
- Generalized Neumann rejection method
  - density separable, i.e. $p(x) = p_1(x^{(1)}) \cdot p_2(x^{(2)})$
  - multidimensional inversion method on invertible part p_2
  - Neumann rejection method on p_1

Selection Methods

- Neumann rejection method, if $\|p\|_\infty < b < \infty$
  - Choose two independent realizations of uniform random numbers $\xi, \zeta \sim \mathcal{U}$
  - If $p(\xi) > b\zeta$ take ξ as a sample
  - else reject ξ and try again
- Efficiency depends on graph of p
- Generalized Neumann rejection method
  - density separable, i.e. $p(x) = p_1(x^{(1)}) \cdot p_2(x^{(2)})$
  - multidimensional inversion method on invertible part p_2
  - Neumann rejection method on p_1
- Metropolis sampling algorithm
  - construct Markov chain with desired density p as stationary density

Selection Methods

- Neumann rejection method, if $\|p\|_\infty < b < \infty$
  - Choose two independent realizations of uniform random numbers $\xi, \zeta \sim \mathcal{U}$
  - If $p(\xi) > b\zeta$ take ξ as a sample
  - else reject ξ and try again
- Efficiency depends on graph of p
- Generalized Neumann rejection method
  - density separable, i.e. $p(x) = p_1(x^{(1)}) \cdot p_2(x^{(2)})$
  - multidimensional inversion method on invertible part p_2
  - Neumann rejection method on p_1
- Metropolis sampling algorithm
  - construct Markov chain with desired density p as stationary density
- **Construction dimension**, i.e. random numbers required for one realization
  - now only finite expectation

Special Methods: Normal Distribution $\mathcal{N}(\mu, \sigma)$

- Probability density function

$$f_{\mathcal{N}(\mu,\sigma)}(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - expectation μ
  - variance σ^2

- Trick: Simulate a pair $(X, Y) \sim \mathcal{N}(0, 1) \times \mathcal{N}(0, 1)$

$$f_{\mathcal{N}(0,1)}(x) \cdot f_{\mathcal{N}(0,1)}(y)dxdy = \frac{1}{2\pi} \cdot e^{-\frac{x^2+y^2}{2}}dxdy$$

Special Methods: Normal Distribution $\mathcal{N}(\mu, \sigma)$

- Probability density function

$$f_{\mathcal{N}(\mu,\sigma)}(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - expectation μ
  - variance σ^2

- Trick: Simulate a pair $(X, Y) \sim \mathcal{N}(0, 1) \times \mathcal{N}(0, 1)$

$$f_{\mathcal{N}(0,1)}(x) \cdot f_{\mathcal{N}(0,1)}(y)dxdy = \frac{1}{2\pi} \cdot e^{-\frac{x^2+y^2}{2}} dxdy = \frac{1}{2\pi} \cdot e^{-\frac{r^2}{2}} rdrd\phi$$

Special Methods: Normal Distribution $\mathcal{N}(\mu, \sigma)$

- Probability density function

$$f_{\mathcal{N}(\mu,\sigma)}(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - expectation μ
  - variance σ^2

- Trick: Simulate a pair $(X, Y) \sim \mathcal{N}(0, 1) \times \mathcal{N}(0, 1)$

$$f_{\mathcal{N}(0,1)}(x) \cdot f_{\mathcal{N}(0,1)}(y)dxdy = \frac{1}{2\pi} \cdot e^{-\frac{x^2+y^2}{2}} dxdy = \frac{1}{2\pi} \cdot e^{-\frac{r^2}{2}} rdrd\phi$$

- Polar method (Box-Müller)

$$(X, Y) = \sqrt{-2\ln(1-\xi)} \cdot (\cos 2\pi\nu, \sin 2\pi\nu)$$

  where $\xi, \nu \sim \mathcal{U}$ on $[0, 1)$

Simulation of Periodic Random Fields

- Typical realization procedure of $X : \Omega \to C(s, d)$

1. Realize Gaussian noise on s-dimensional regular grid K

$$\mathbf{N}_\omega(\mathbf{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d, \qquad \mathbf{k} \in K$$

2. Shape noise by spectrum S of phenomenon

$$\hat{\mathbf{X}}_\omega(\mathbf{k}) = S(\mathbf{k})\mathbf{N}_\omega(\mathbf{k})$$

3. Band limited evaluation by fast Fourier transform for each dimension

$$\mathbf{X}_\omega(\mathbf{x}) = \sum_{\mathbf{k} \in K} \hat{\mathbf{X}}_\omega(\mathbf{k}) e^{2\pi i \mathbf{k}^T \cdot \mathbf{x}} \qquad \in C(s, d)$$

Simulation of Periodic Random Fields

- Typical realization procedure of $X : \Omega \to C(s, d)$

  1. Realize Gaussian noise on s-dimensional regular grid K

     $$\mathbf{N}_\omega(\mathbf{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d, \qquad \mathbf{k} \in K$$

  2. Shape noise by spectrum S of phenomenon

     $$\hat{\mathbf{X}}_\omega(\mathbf{k}) = S(\mathbf{k})\mathbf{N}_\omega(\mathbf{k})$$

  3. Band limited evaluation by fast Fourier transform for each dimension

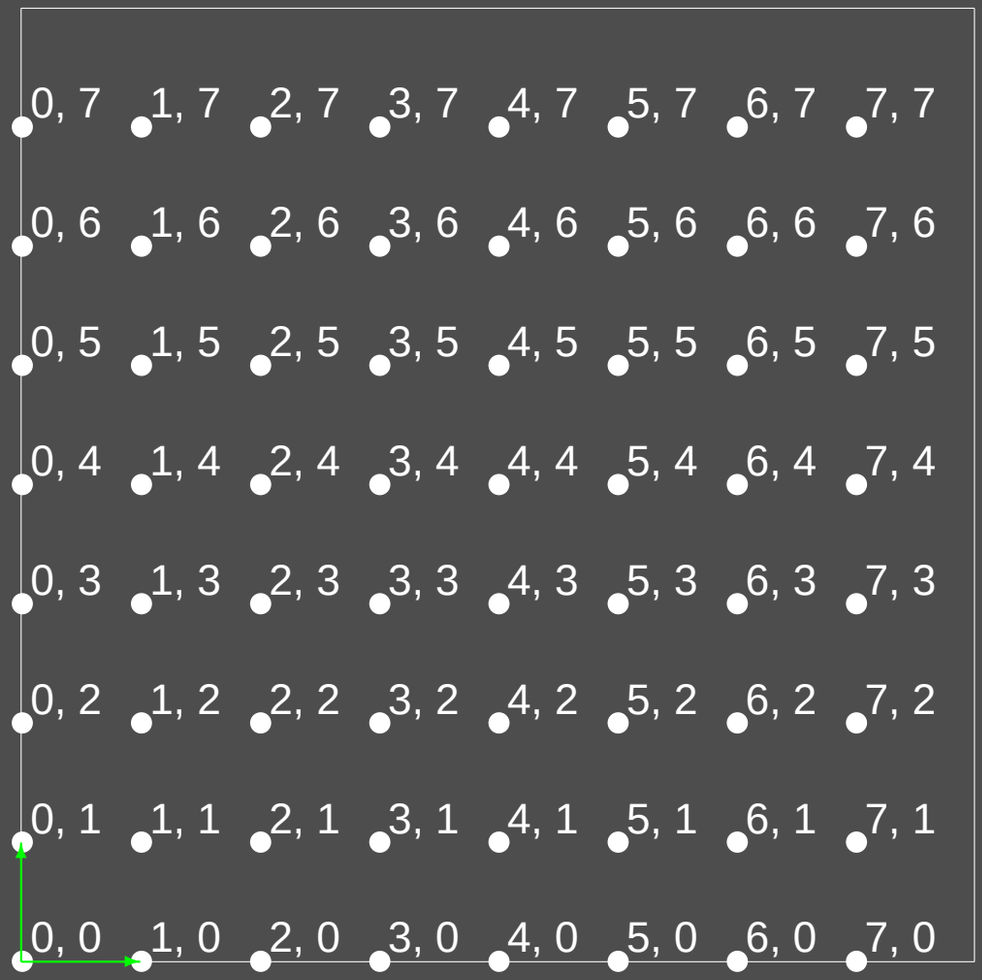
     $$\mathbf{X}_\omega(\mathbf{x}) = \sum_{\mathbf{k} \in K} \hat{\mathbf{X}}_\omega(\mathbf{k}) e^{2\pi i \mathbf{k}^T \cdot \mathbf{x}} \qquad \in C(s, d)$$

- Standard tensor product approach is exponential in $s = \dim \mathbf{x} = \dim \mathbf{k}$

  $\Rightarrow$ **Curse of dimension**

Curse of Dimension from Regular Grids

- Lattices of rank s with $N = n^s$ points from tensor product approach



- $\mathcal{O}\left(n^s \log n\right)$ for s fast Fourier transforms

Curse of Dimension

- **Theorem (Bakhvalov):** *Let C_M^r denote the set of functions on $[0,1)^s$ with r continuous, bounded derivates, i.e.*

$$\left| \frac{\partial^r f(x)}{\partial x_1^{\alpha_1} \cdots \partial x_s^{\alpha_s}} \right| \leq M \text{ for } f \in C_M^r$$

for all $\alpha_1, \ldots, \alpha_s$, such that $\sum_{i=1}^{s} \alpha_i = r$. Then there exists a function $f \in C_M^r$ such that the error of approximating the integral of f using any N point quadrature rule with weights w_i and function values $f(x_i)$ is

$$\left| \int_{[0,1)^s} f(x)dx - \sum_{i=0}^{N-1} w_i f(x_i) \right| > k \cdot N^{-\frac{r}{s}}$$

where the constant $k > 0$ depends on M and r.

Curse of Discontinuities

- Consider

$$f(x) = \begin{cases} 1 & \text{if } x < X^* \\ 0 & \text{if } x \geq X^* \end{cases}$$

  with $x_i = \frac{i}{n}$ and $x_i \neq X^*$. Then

$$\left| \int_0^1 f(x)dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) \right| \sim \frac{1}{n}$$

- $\mathcal{O}\left(N^{-\frac{1}{s}}\right)$ error for s dimensions

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement**
  - **Global information**
    - function classes
  - **Local, partial information**
    - point sampling (standard information)
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement**
  - **Global information**
    - function classes
  - **Local, partial information**
    - point sampling (standard information)
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms
- **Analysis of ϵ-complexity**
  - lower bound by abstract structures
  - upper bound by algorithm

  $\Rightarrow$ match bounds

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement:** Deterministic numerical integration
  - **Global information**
    - function class: $f \in C_M^r([0,1]^s)$
  - **Local, partial information**
    - point sampling (standard information): $f(x)$
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms: $\sum_{i=1}^{N(f)} w_i f(x_i)$
- **Analysis of ϵ-complexity:** $\mathcal{O}(N^{-\frac{r}{s}})$
  - lower bound by abstract structures: Bakhvalov's theorem
  - upper bound by algorithm: Newton-Cotes quadrature formulas

  $\Rightarrow$ matching bounds

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement:** Stochastic numerical integration
  - **Global information**
    - function class: $f \in L^2([0,1]^s)$
  - **Local, partial information**
    - point sampling (standard information): $f(x)$
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms: $\sum_{i=1}^{N(f)} w_i f(x_i)$
- **Analysis of ϵ-complexity:** $\mathcal{O}(N^{-\frac{1}{2}})$
  - lower bound by abstract structures
  - upper bound by algorithm: Monte Carlo integration

  $\Rightarrow$ matching bounds

Information Based Complexity Theory

- **Goal:** Find ϵ-approximations to numerical problems
  - minimal cost algorithm for maximum error ϵ
- **Problem statement:** Stochastic numerical integration
  - **Global information**
    - function class: $f \in C_M^r([0,1]^s)$
  - **Local, partial information**
    - point sampling (standard information): $f(x)$
  - **Model of computation**
    - real number model
    - scalar products as class of algorithms: $\sum_{i=1}^{N(f)} w_i f(x_i)$
- **Analysis of ϵ-complexity:** $\mathcal{O}(N^{-\frac{r}{s}-\frac{1}{2}})$
  - lower bound by abstract structures
  - upper bound by algorithm: Monte Carlo with separation of the main part

  $\Rightarrow$ matching bounds

Monte Carlo Integration

- Principle: Construct random variable with desired functional as expectation
- Numerical integration by **random sampling**

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3\sigma(f)}{\sqrt{N}}\right\}\right) \approx 0.997 \qquad x_i \sim \mathcal{U}$$

Monte Carlo Integration

- Principle: Construct random variable with desired functional as expectation
- Numerical integration by **random sampling**

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3\sigma(f)}{\sqrt{N}}\right\}\right) \approx 0.997 \qquad x_i \sim \mathcal{U}$$

- Simple, independent of dimension and smoothness, only $f \in L^2$
- Problems
  - Noise, slow convergence, difficult parallelization and reproducability
  - No real random numbers

Monte Carlo Integration

- Principle: Construct random variable with desired functional as expectation
- Numerical integration by **random sampling**

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3\sigma(f)}{\sqrt{N}}\right\}\right) \approx 0.997 \qquad x_i \sim \mathcal{U}$$

- Simple, independent of dimension and smoothness, only $f \in L^2$
- Problems
  - Noise, slow convergence, difficult parallelization and reproducability
  - No real random numbers
- Computational complexity

$$N \cdot t_S \cdot \sigma^2(f) = N \cdot t_S \cdot \mathbf{E}\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right|^2$$

Monte Carlo Integration

- Principle: Construct random variable with desired functional as expectation
- Numerical integration by **random sampling**

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3\sigma(f)}{\sqrt{N}}\right\}\right) \approx 0.997 \qquad x_i \sim \mathcal{U}$$

- Simple, independent of dimension and smoothness, only $f \in L^2$
- Problems
  - Noise, slow convergence, difficult parallelization and reproducability
  - No real random numbers
- Computational complexity

$$N \cdot t_S \cdot \sigma^2(f) = N \cdot t_S \cdot \textbf{E}\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right|^2$$

- Increase efficiency, not only variance reduction !!!

$$\frac{1}{t_S \cdot \sigma^2(f)}$$

Error Control

- Unbiased estimator Y

$$\mathbf{E}Y = \int_{I^s} f(x)dx$$

Error Control

- Unbiased estimator Y

$$\mathbf{E}Y = \int_{I^s} f(x)dx$$

- Bias of estimator Y

$$\beta_Y := \mathbf{E}Y - \int_{I^s} f(x)dx$$

Error Control

- Unbiased estimator Y

$$\mathbf{E}Y = \int_{I^s} f(x)dx$$

- Bias of estimator Y

$$\beta_Y := \mathbf{E}Y - \int_{I^s} f(x)dx$$

- Consistent estimator Y

$$\mathbf{Prob}\left(\lim_{N\to\infty} \frac{1}{N}\sum_{i=0}^{N-1} y_i = \int_{I^s} f(x)dx\right) = 1$$

Error Control

- Unbiased estimator Y

$$\mathbf{E}Y = \int_{I^s} f(x)dx$$

- Bias of estimator Y

$$\beta_Y := \mathbf{E}Y - \int_{I^s} f(x)dx$$

- Consistent estimator Y

$$\mathbf{Prob}\left(\lim_{N\to\infty}\frac{1}{N}\sum_{i=0}^{N-1} y_i = \int_{I^s} f(x)dx\right) = 1$$

- Error estimate of the estimate

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right) \approx \frac{1}{N-1}\left[\sum_{i=0}^{N-1}(f(x_i))^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} f(x_i)\right)^2\right]$$

  - adaptive sampling

Correlated Sampling: Separation of the Main Part

- Variance reduction by approximation, method of control variables
- Search g with

$$\|f - g\|_\infty < \tau \in \mathbb{R}^+$$

- Then

$$\int_{I^s} f(x)dx = \underbrace{\int_{I^s} g(x)dx}_{\text{analytical}} + \underbrace{\int_{I^s} f(x) - g(x)dx}_{\text{Monte Carlo}}$$

Correlated Sampling: Separation of the Main Part

- Variance reduction by approximation, method of control variables
- Search g with

$$\|f - g\|_\infty < \tau \in \mathbb{R}^+$$

- Then

$$\int_{I^s} f(x)dx = \underbrace{\int_{I^s} g(x)dx}_{\text{analytical}} + \underbrace{\int_{I^s} f(x) - g(x)dx}_{\text{Monte Carlo}}$$

$$\approx \int_{I^s} g(x)dx + \frac{1}{N}\sum_{i=0}^{N-1} (f(x_i) - g(x_i))$$

Note: The independent evaluation would destroy the advantages of the method.

- Variance of Monte Carlo part

$$\sigma^2(f-g) \leq \int_{I^s} |f(x) - g(x)|^2 dx \leq \tau^2$$

Correlated Sampling: Separation of the Main Part

- Variance reduction by approximation, method of control variables
- Search g with

$$\|f - g\|_\infty < \tau \in \mathbb{R}^+$$

- Then

$$\begin{aligned}\int_{I^s} f(x)dx &= \underbrace{\int_{I^s} g(x)dx}_{\text{analytical}} + \underbrace{\int_{I^s} f(x) - g(x)dx}_{\text{Monte Carlo}} \\ &\approx \int_{I^s} g(x)dx + \frac{1}{N}\sum_{i=0}^{N-1} (f(x_i) - g(x_i))\end{aligned}$$

Note: The independent evaluation would destroy the advantages of the method.

- Variance of Monte Carlo part

$$\sigma^2(f-g) \leq \int_{I^s} |f(x) - g(x)|^2 dx \leq \tau^2$$

- Lower bound $\mathcal{O}\left(N^{-\frac{r}{s}-\frac{1}{2}}\right)$ for $f \in C_M^r([0,1)^s)$ obtained by Newton-Cotes methods

The Method of Dependent Tests

- Principle: Construct random field with desired function as expectation
- Method of dependent tests (parametric Monte Carlo integration)

$$\begin{aligned} g(y) &:= \int_{I^s} f(x,y)dx \\ &\approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \end{aligned}$$

  for integro-approximation problems

The Method of Dependent Tests

- Principle: Construct random field with desired function as expectation
- Method of dependent tests (parametric Monte Carlo integration)

$$\begin{aligned} g(y) &:= \int_{I^s} f(x,y)dx \\ &\approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \end{aligned}$$

  for integro-approximation problems

- Computational complexity

$$N \cdot t_S \cdot \mathbf{E}\left\| \int_{I^s} f(x,y)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \right\|_{L^2}^2$$

The Method of Dependent Tests

- Principle: Construct random field with desired function as expectation
- Method of dependent tests (parametric Monte Carlo integration)

$$\begin{aligned} g(y) &:= \int_{I^s} f(x,y)dx \\ &\approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \end{aligned}$$

  for integro-approximation problems

- Computational complexity

$$N \cdot t_S \cdot \mathbf{E}\left\| \int_{I^s} f(x,y)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y) \right\|_{L^2}^2$$

- **Note: One single set** $(x_i)_{i=0}^{N-1} \subset I^s$ of i.i.d. random samples

  $\Rightarrow$ exploit induced grid structure

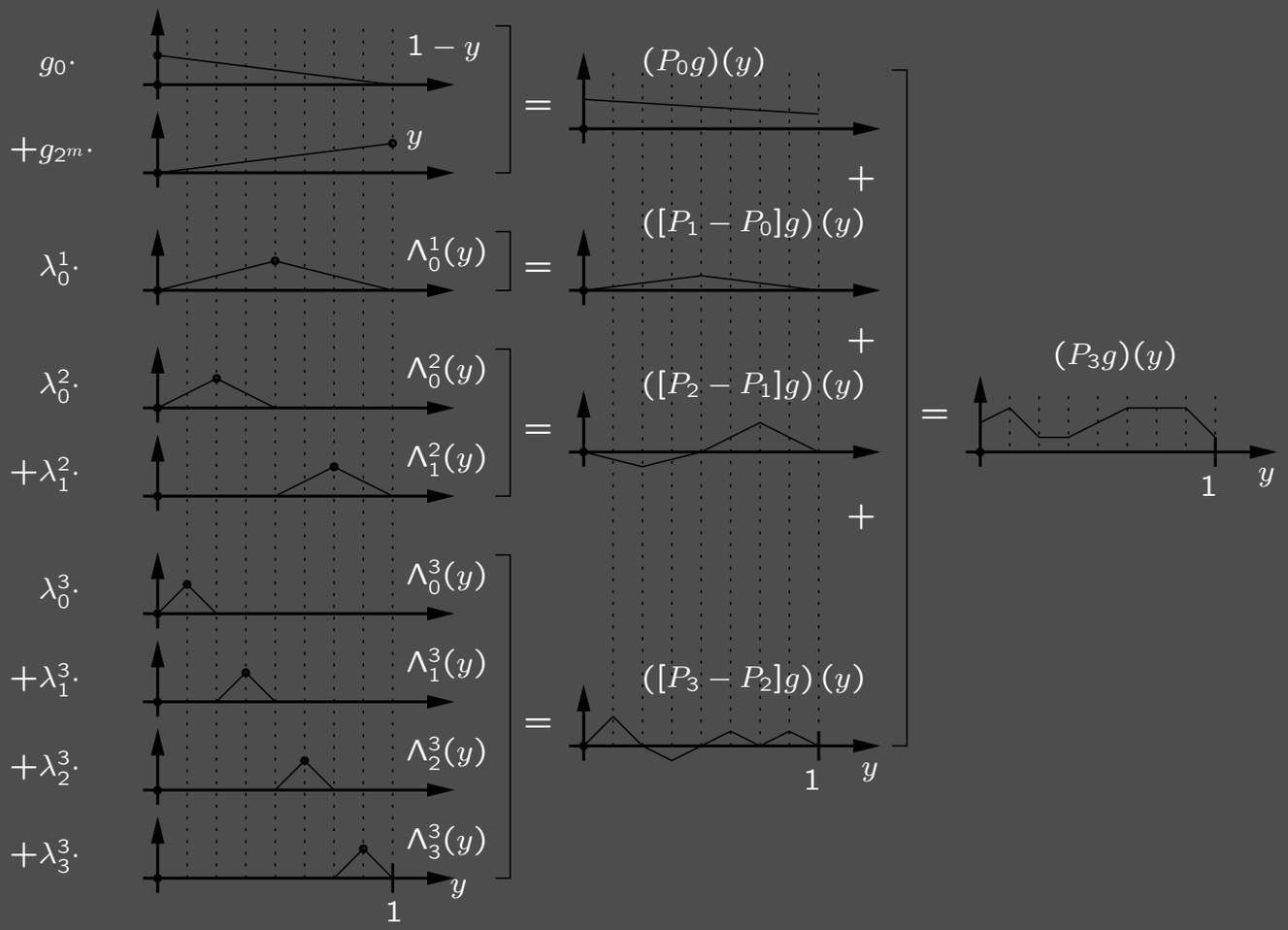
- Examples
  - accumulation buffer
  - multilevel method of dependent tests

Hierarchical Function Representation

- Use multilevel function representation **[Heinrich 1998]**

$$P_m g = P_0 g + \sum_{l=1}^{m} [P_l - P_{l-1}] g$$

for an arbitrary sequence $(P_l)_{l=0}^{m}$ of interpolation operators



Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$

Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$
- Method of dependent tests

$$G_k^l := \frac{1}{N_l} \sum_{i=0}^{N_l - 1} f(x_i, y_k) \text{ with } N_l := N \cdot \frac{2^m + 1}{2^l + 1} \cdot 2^{\alpha l} \cdot \frac{2^\alpha - 1}{2^{\alpha(m+1)} - 1}$$

Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$
- Method of dependent tests

$$G_k^l := \frac{1}{N_l} \sum_{i=0}^{N_l - 1} f(x_i, y_k) \text{ with } N_l := N \cdot \frac{2^m + 1}{2^l + 1} \cdot 2^{\alpha l} \cdot \frac{2^\alpha - 1}{2^{\alpha(m+1)} - 1}$$

- Compute approximation $g_i \approx \hat{g}_i$
  - boundary $g_0 \approx \hat{g}_0 := G_0^0$ and $g_{2^m} \approx \hat{g}_{2^m} := G_{2^m}^0$
  - refinement

$$g_{(2k+1)2^{m-l}} = \underbrace{\frac{g_{k2^{m-(l-1)}} + g_{(k+1)2^{m-(l-1)}}}{2}}_{\text{Predictor}} + \underbrace{\lambda_k^l}_{\text{Update}}$$

Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$
- Method of dependent tests

$$G_k^l := \frac{1}{N_l} \sum_{i=0}^{N_l - 1} f(x_i, y_k) \text{ with } N_l := N \cdot \frac{2^m + 1}{2^l + 1} \cdot 2^{\alpha l} \cdot \frac{2^\alpha - 1}{2^{\alpha(m+1)} - 1}$$

- Compute approximation $g_i \approx \hat{g}_i$
  - boundary $g_0 \approx \hat{g}_0 := G_0^0$ and $g_{2^m} \approx \hat{g}_{2^m} := G_{2^m}^0$
  - refinement

$$g_{(2k+1)2^{m-l}} = \underbrace{\frac{g_{k2^{m-(l-1)}} + g_{(k+1)2^{m-(l-1)}}}{2}}_{\text{Predictor}} + \underbrace{\lambda_k^l}_{\text{Update}}$$

$$\approx \hat{g}_{(2k+1)2^{m-l}}$$

Multilevel Method of Dependent Tests

- Linear Lagrange interpolation of $g_k := g(y_k) = (P_m g)(y_k)$ in $y_k = \frac{k}{2^m}$
- Method of dependent tests

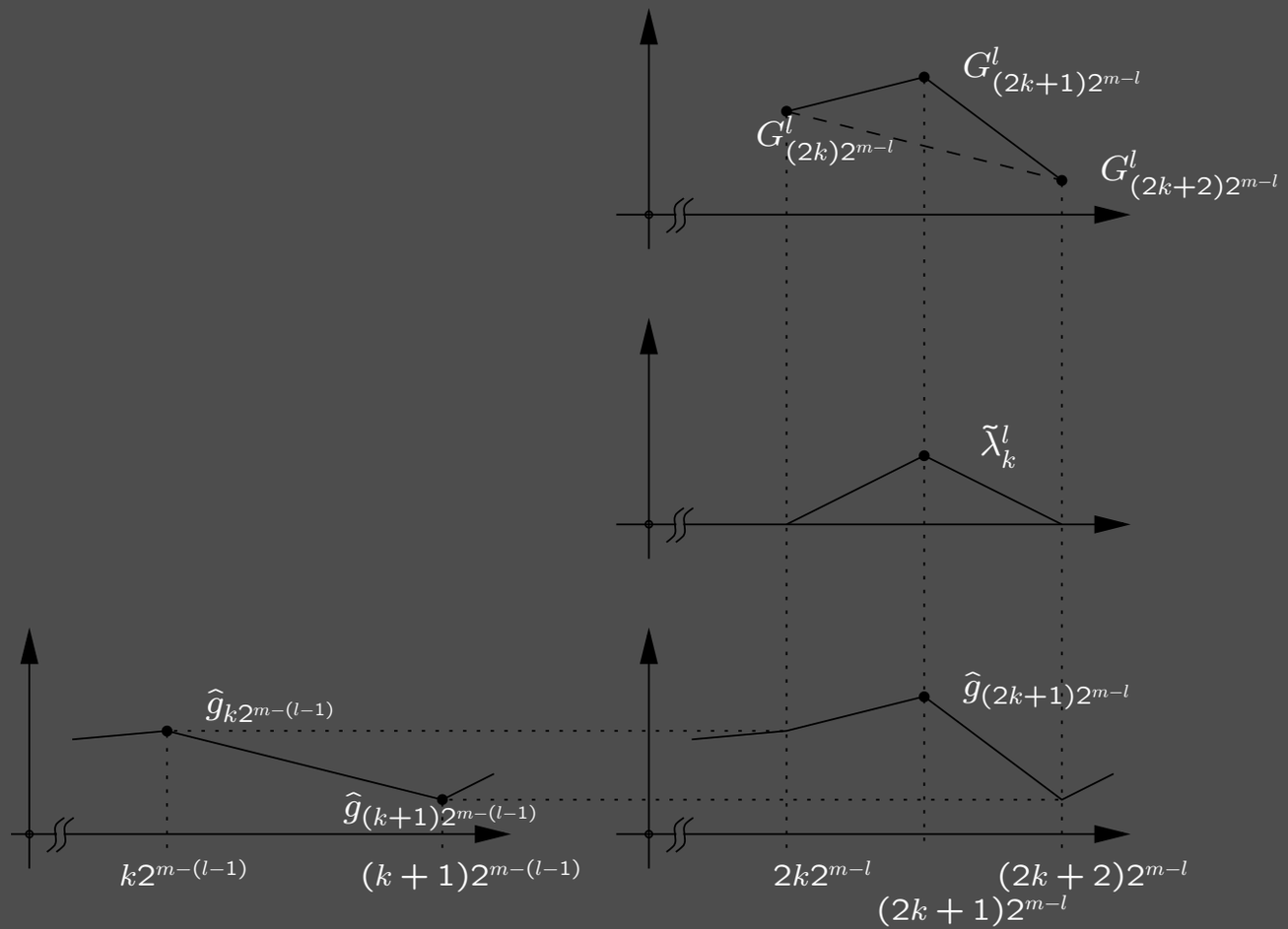
$$G_k^l := \frac{1}{N_l} \sum_{i=0}^{N_l - 1} f(x_i, y_k) \text{ with } N_l := N \cdot \frac{2^m + 1}{2^l + 1} \cdot 2^{\alpha l} \cdot \frac{2^\alpha - 1}{2^{\alpha(m+1)} - 1}$$

- Compute approximation $g_i \approx \hat{g}_i$
  - boundary $g_0 \approx \hat{g}_0 := G_0^0$ and $g_{2^m} \approx \hat{g}_{2^m} := G_{2^m}^0$
  - refinement

$$\begin{aligned} g_{(2k+1)2^{m-l}} &= \underbrace{\frac{g_{k2^{m-(l-1)}} + g_{(k+1)2^{m-(l-1)}}}{2}}_{\text{Predictor}} + \underbrace{\lambda_k^l}_{\text{Update}} \\ \approx \hat{g}_{(2k+1)2^{m-l}} &:= \frac{\hat{g}_{k2^{m-(l-1)}} + \hat{g}_{(k+1)2^{m-(l-1)}}}{2} \\ &\quad + \underbrace{G^l_{(2k+1)2^{m-l}} - \frac{G^l_{(2k)2^{m-l}} + G^l_{(2k+2)2^{m-l}}}{2}}_{=:\tilde{\lambda}_k^l} \end{aligned}$$

Implementation

- In-place reconstruction



Coarser level $l-1$ Finer level l

Efficiency Issues

- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples

Efficiency Issues

- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples
- One function
  - small detail contribution if correlated

$$\tilde{\lambda}_k^l = \frac{1}{N_l} \sum_{i=0}^{N_l-1} \left(f(x_i, y_{(2k+1)2^{m-l}}) - \frac{f(x_i, y_{(2k)2^{m-l}}) + f(x_i, y_{(2k+2)2^{m-l}})}{2} \right)$$

  - adapt sampling rate N_l to support size

  $\Rightarrow$ reduced computational cost by exploiting correlation

Efficiency Issues

- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples
- One function
  - small detail contribution if correlated

$$\tilde{\lambda}_k^l = \frac{1}{N_l} \sum_{i=0}^{N_l-1} \left(f(x_i, y_{(2k+1)2^{m-l}}) - \frac{f(x_i, y_{(2k)2^{m-l}}) + f(x_i, y_{(2k+2)2^{m-l}})}{2} \right)$$

  - adapt sampling rate N_l to support size

  $\Rightarrow$ reduced computational cost by exploiting correlation
- Localization heuristics
  - range check
  - predictor-corrector difference
  - relative error

Efficiency Issues

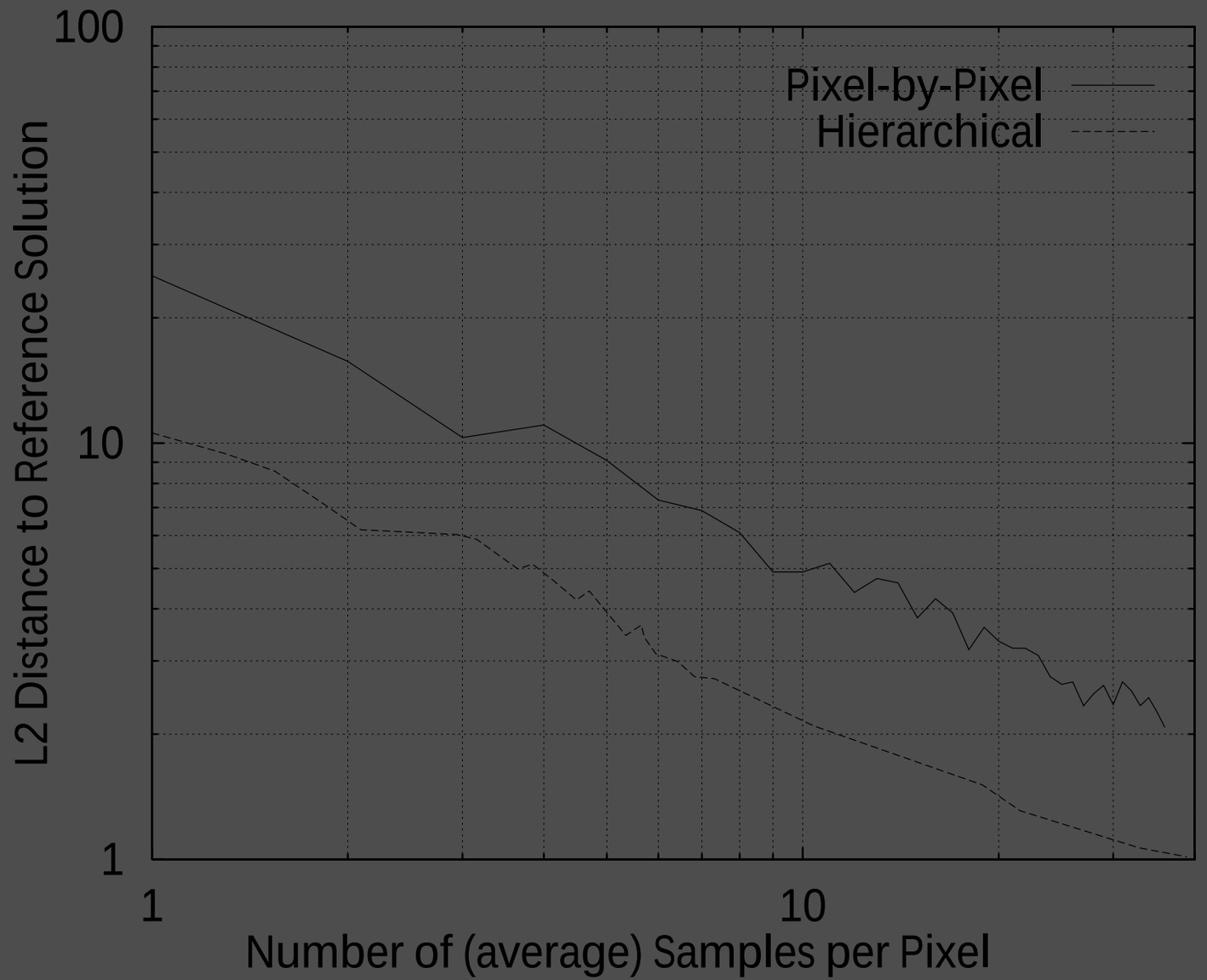
- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples
- One function
  - small detail contribution if correlated

$$\tilde{\lambda}_k^l = \frac{1}{N_l} \sum_{i=0}^{N_l-1} \left(f(x_i, y_{(2k+1)2^{m-l}}) - \frac{f(x_i, y_{(2k)2^{m-l}}) + f(x_i, y_{(2k+2)2^{m-l}})}{2} \right)$$

  - adapt sampling rate N_l to support size

  $\Rightarrow$ reduced computational cost by exploiting correlation
- Localization heuristics
  - range check
  - predictor-corrector difference
  - relative error
- With lifting scheme on arbitrary topology and boundaries

Numerical Results



Importance Sampling

- Integral transformation by introducing a probability density p

$$\int_{I^s} f(x)dx = \int_{I^s} f(x)\frac{p(x)}{p(x)}dx = \int_{I^s} \frac{f(y)}{p(y)}dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \qquad y_i \sim p$$

Importance Sampling

- Integral transformation by introducing a probability density p

$$\int_{I^s} f(x)dx = \int_{I^s} f(x)\frac{p(x)}{p(x)}dx = \int_{I^s} \frac{f(y)}{p(y)}dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \qquad y_i \sim p$$

- Variance

$$\sigma^2\left(\frac{f}{p}\right) = \int_{I^s} \frac{f^2(x)}{p(x)}dx - \left(\int_{I^s} f(x)dx\right)^2$$

Importance Sampling

- Integral transformation by introducing a probability density p

$$\int_{I^s} f(x)dx = \int_{I^s} f(x)\frac{p(x)}{p(x)}dx = \int_{I^s} \frac{f(y)}{p(y)}dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \qquad y_i \sim p$$

- Variance

$$\sigma^2\left(\frac{f}{p}\right) = \int_{I^s} \frac{f^2(x)}{p(x)}dx - \left(\int_{I^s} f(x)dx\right)^2$$

- Often $f(x) = g(x)p(x)$

$$\int_{I^s} f(x)dx = \int_{I^s} g(x)p(x)dx = \int_{I^s} g(y)dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} g(y_i) \qquad y_i \sim p$$

Importance Sampling

- Integral transformation by introducing a probability density p

$$\int_{I^s} f(x)dx = \int_{I^s} f(x)\frac{p(x)}{p(x)}dx = \int_{I^s} \frac{f(y)}{p(y)}dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \qquad y_i \sim p$$

- Variance

$$\sigma^2\left(\frac{f}{p}\right) = \int_{I^s} \frac{f^2(x)}{p(x)}dx - \left(\int_{I^s} f(x)dx\right)^2$$

- Often $f(x) = g(x)p(x)$

$$\int_{I^s} f(x)dx = \int_{I^s} g(x)p(x)dx = \int_{I^s} g(y)dP(y) \approx \frac{1}{N}\sum_{i=0}^{N-1} g(y_i) \qquad y_i \sim p$$

- Often separating the main part is more efficient than importance sampling

Replication: Independent and Dependent Sampling

- Replication heuristic

$$\left(w_j, R_j\right)_{j=0}^{M-1}$$

  - weight functions $w_j(x) : I^s \to \mathbb{R}$, and
  - mappings $R_j(x) : I^s \to I^s$ so that

$$\int_{I^s} f(x)dx = \int_{I^s} \sum_{j=0}^{M-1} w_j(x) f(R_j(x))dx = \sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx$$

Replication: Independent and Dependent Sampling

- Replication heuristic

$$\left(w_j, R_j\right)_{j=0}^{M-1}$$

  - weight functions $w_j(x) : I^s \to \mathbb{R}$, and
  - mappings $R_j(x) : I^s \to I^s$ so that

$$\int_{I^s} f(x)dx = \int_{I^s} \sum_{j=0}^{M-1} w_j(x) f(R_j(x))dx = \sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx$$

- Either independent integral estimation

$$\sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx \approx \sum_{j=0}^{M-1} \frac{1}{N_j} \sum_{i=0}^{N_j-1} w_j(x_{i,j}) f(R_j(x_{i,j})),$$

Replication: Independent and Dependent Sampling

- Replication heuristic

  $$\left(w_j, R_j\right)_{j=0}^{M-1}$$

  - weight functions $w_j(x) : I^s \to \mathbb{R}$, and
  - mappings $R_j(x) : I^s \to I^s$ so that

$$\int_{I^s} f(x)dx = \int_{I^s} \sum_{j=0}^{M-1} w_j(x) f(R_j(x))dx = \sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx$$

- Either independent integral estimation

$$\sum_{j=0}^{M-1} \int_{I^s} w_j(x) f(R_j(x))dx \approx \sum_{j=0}^{M-1} \frac{1}{N_j} \sum_{i=0}^{N_j-1} w_j(x_{i,j}) f(R_j(x_{i,j})),$$

  or dependent, i.e. correlated sampling

$$\int_{I^s} \sum_{j=0}^{M-1} w_j(x) f(R_j(x))dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} w_j(x_i) f(R_j(x_i)),$$

Replication Heuristics: Multiple importance sampling

- Simple importance sampling can cause infinite variance
- For a set of techniques p_j, i.e. $R_j := P_j^{-1}$, the weights are

Heuristic	independent sampling	dependent sampling
Power ($\beta \in \mathbb{R}^+$)	$w_j(x) := \frac{N_j^\beta p_j^\beta(x)}{\sum_{k=0}^{M-1} N_k^\beta p_k^\beta(x)} \cdot \frac{1}{p_j(x)}$	$w_j(x) = \frac{p_j^\beta(x)}{\sum_{k=0}^{M-1} p_k^\beta(x)} \cdot \frac{1}{p_j(x)}$
Balance ($\beta = 1$)	$w_j(x) := \frac{N_j}{\sum_{k=0}^{M-1} N_k p_k(x)}$	$w_j(x) = \frac{1}{\sum_{k=0}^{M-1} p_k(x)}$
Uniform ($\beta = 0$)	$w_j(x) := \frac{N_j}{p_j(x) \sum_{k=0}^{M-1} N_k}$	$w_j(x) = \frac{1}{M p_j(x)}$

Replication Heuristics: Multiple importance sampling

- Simple importance sampling can cause infinite variance
- For a set of techniques p_j, i.e. $R_j := P_j^{-1}$, the weights are

Heuristic	independent sampling	dependent sampling
Power ($\beta \in \mathbb{R}^+$)	$w_j(x) := \frac{N_j^\beta p_j^\beta(x)}{\sum_{k=0}^{M-1} N_k^\beta p_k^\beta(x)} \cdot \frac{1}{p_j(x)}$	$w_j(x) = \frac{p_j^\beta(x)}{\sum_{k=0}^{M-1} p_k^\beta(x)} \cdot \frac{1}{p_j(x)}$
Balance ($\beta = 1$)	$w_j(x) := \frac{N_j}{\sum_{k=0}^{M-1} N_k p_k(x)}$	$w_j(x) = \frac{1}{\sum_{k=0}^{M-1} p_k(x)}$
Uniform ($\beta = 0$)	$w_j(x) := \frac{N_j}{p_j(x) \sum_{k=0}^{M-1} N_k}$	$w_j(x) = \frac{1}{M p_j(x)}$

- Problem of insufficient techniques

Stratification

- Partition of integration domain $I^s = \cup_{k=1}^{K} A_k$
- Monte Carlo integration on each of the disjoint strata A_k

$$\int_{I^s} f(x)dx = \sum_{k=1}^{K} \int_{A_k} f(x)dx$$

Stratification

- Partition of integration domain $I^s = \cup_{k=1}^{K} A_k$
- Monte Carlo integration on each of the disjoint strata A_k

$$\int_{I^s} f(x)dx = \sum_{k=1}^{K} \int_{A_k} f(x)dx \approx \sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \sum_{i=0}^{N_k-1} f(x_{k,i})$$

Stratification

- Partition of integration domain $I^s = \cup_{k=1}^{K} A_k$
- Monte Carlo integration on each of the disjoint strata A_k

$$\int_{I^s} f(x)dx = \sum_{k=1}^{K} \int_{A_k} f(x)dx \approx \sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \sum_{i=0}^{N_k-1} f(x_{k,i})$$

- Variance reduction for standard choice $N_k = \lambda_s(A_k)N$

$$\sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \int_{A_k} \left(f(y) - \frac{1}{\lambda_s(A_k)} \int_{A_k} f(x)dx \right)^2 dy \leq \frac{\sigma^2(f)}{N}$$

$\Rightarrow$ at least as good as uniform random sampling

Stratification

- Partition of integration domain $I^s = \cup_{k=1}^{K} A_k$
- Monte Carlo integration on each of the disjoint strata A_k

$$\int_{I^s} f(x)dx = \sum_{k=1}^{K} \int_{A_k} f(x)dx \approx \sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \sum_{i=0}^{N_k-1} f(x_{k,i})$$

- Variance reduction for standard choice $N_k = \lambda_s(A_k)N$

$$\sum_{k=1}^{K} \frac{\lambda_s(A_k)}{N_k} \int_{A_k} \left(f(y) - \frac{1}{\lambda_s(A_k)} \int_{A_k} f(x)dx \right)^2 dy \leq \frac{\sigma^2(f)}{N}$$

  $\Rightarrow$ at least as good as uniform random sampling

- $\lambda_s(A_k) = \frac{1}{N}$ yields

$$\int_{I^s} f(x)dx \approx \frac{1}{N} \sum_{k=0}^{N-1} f(x_k|_{A_k})$$

  - Lloyd-relaxation
  - jittered sampling

Stratification by Lloyd-Relaxation

- Algorithm (similar to vector quantization)
  - Take N random initial points
  - Loop: Move each point into the center of gravity of its Voronoi-cell
- Periodic boundary conditions

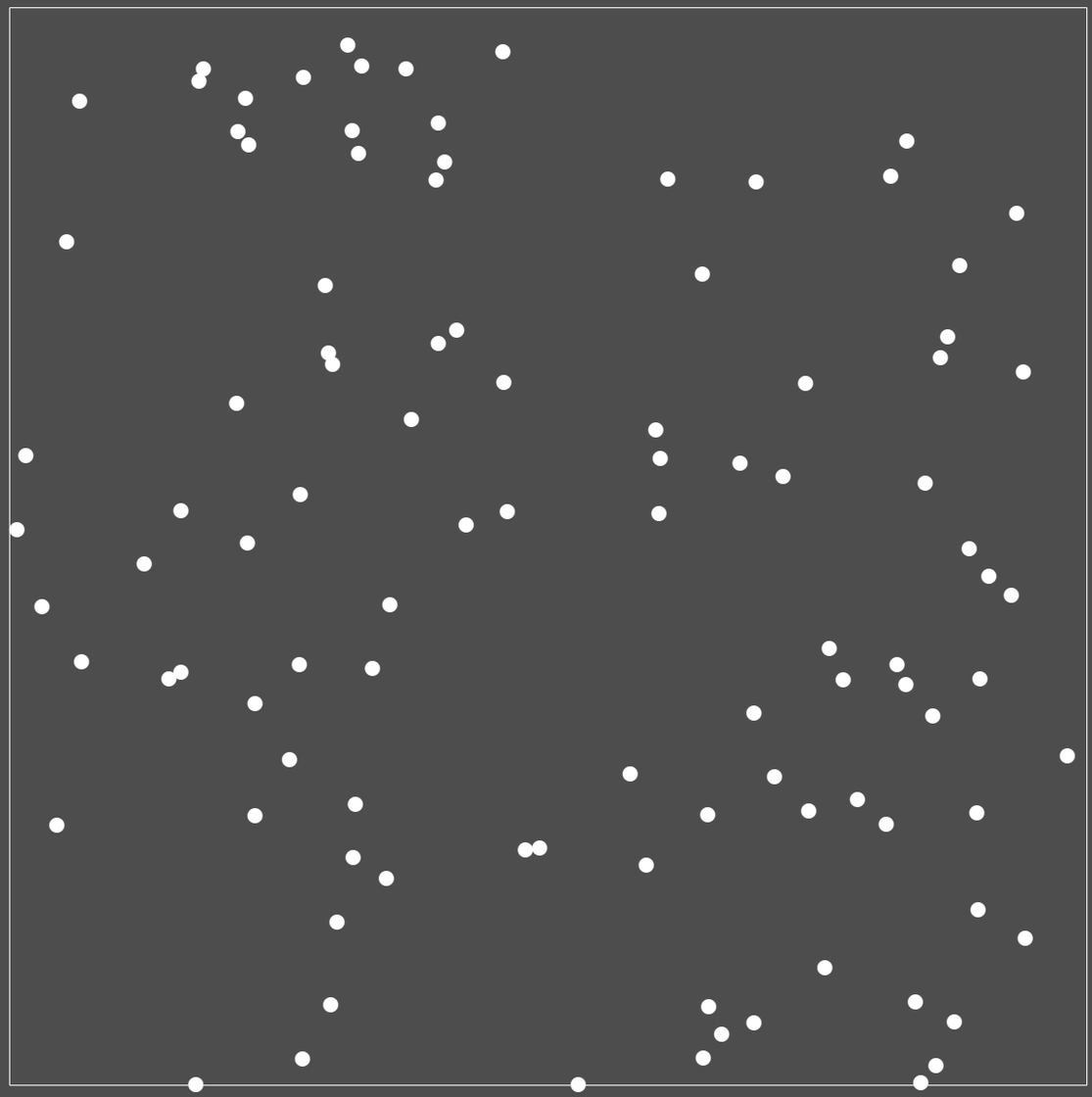
\+ Fast convergence to regular patterns

$\Rightarrow$ *Small* number of relaxation steps yields blue-noise-samples

\- Expensive iteration step

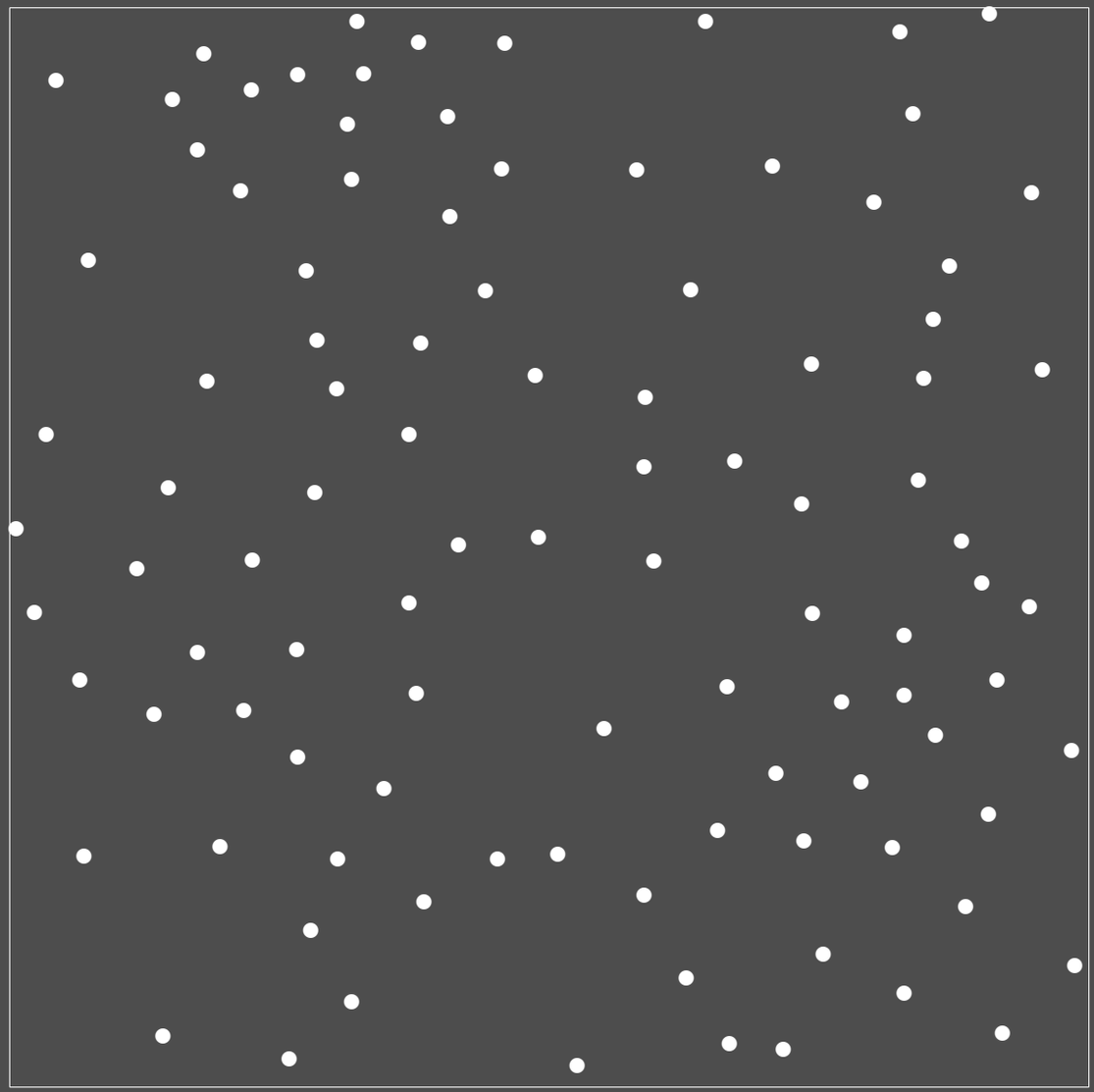
\- No incremental sampling

Stratification by Lloyd-Relaxation



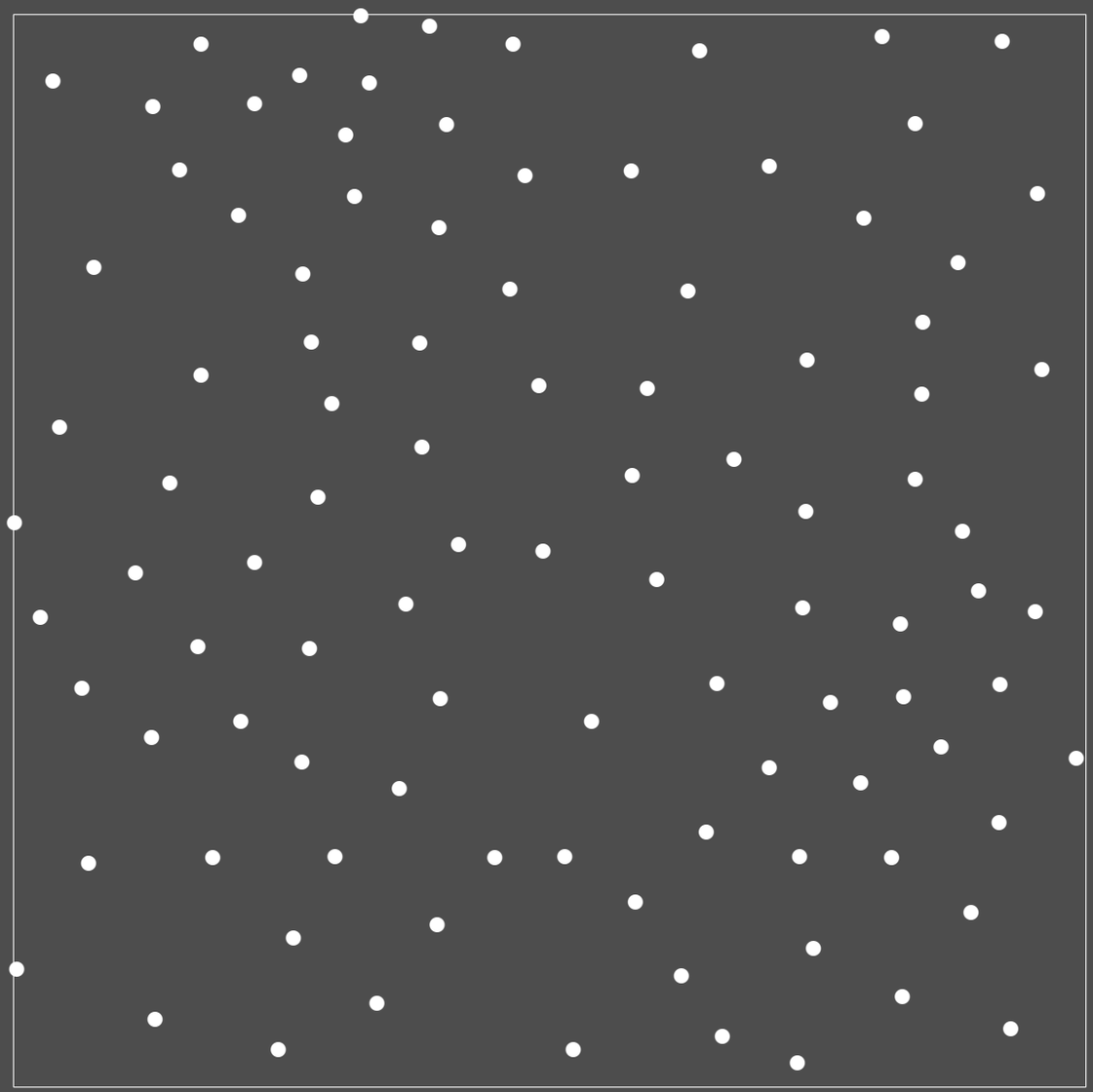
- Iteration 0

Stratification by Lloyd-Relaxation



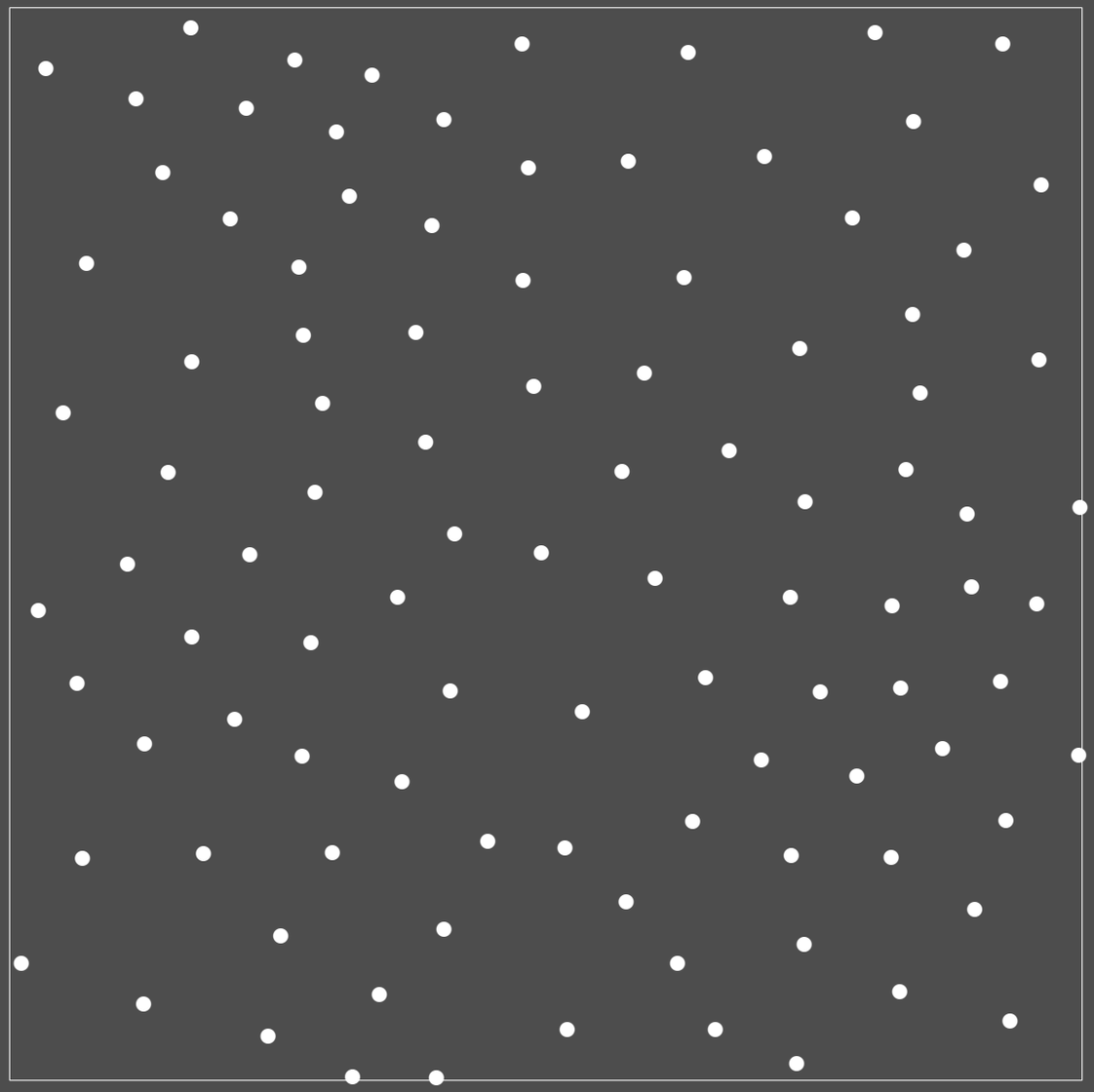
- Iteration 1

Stratification by Lloyd-Relaxation



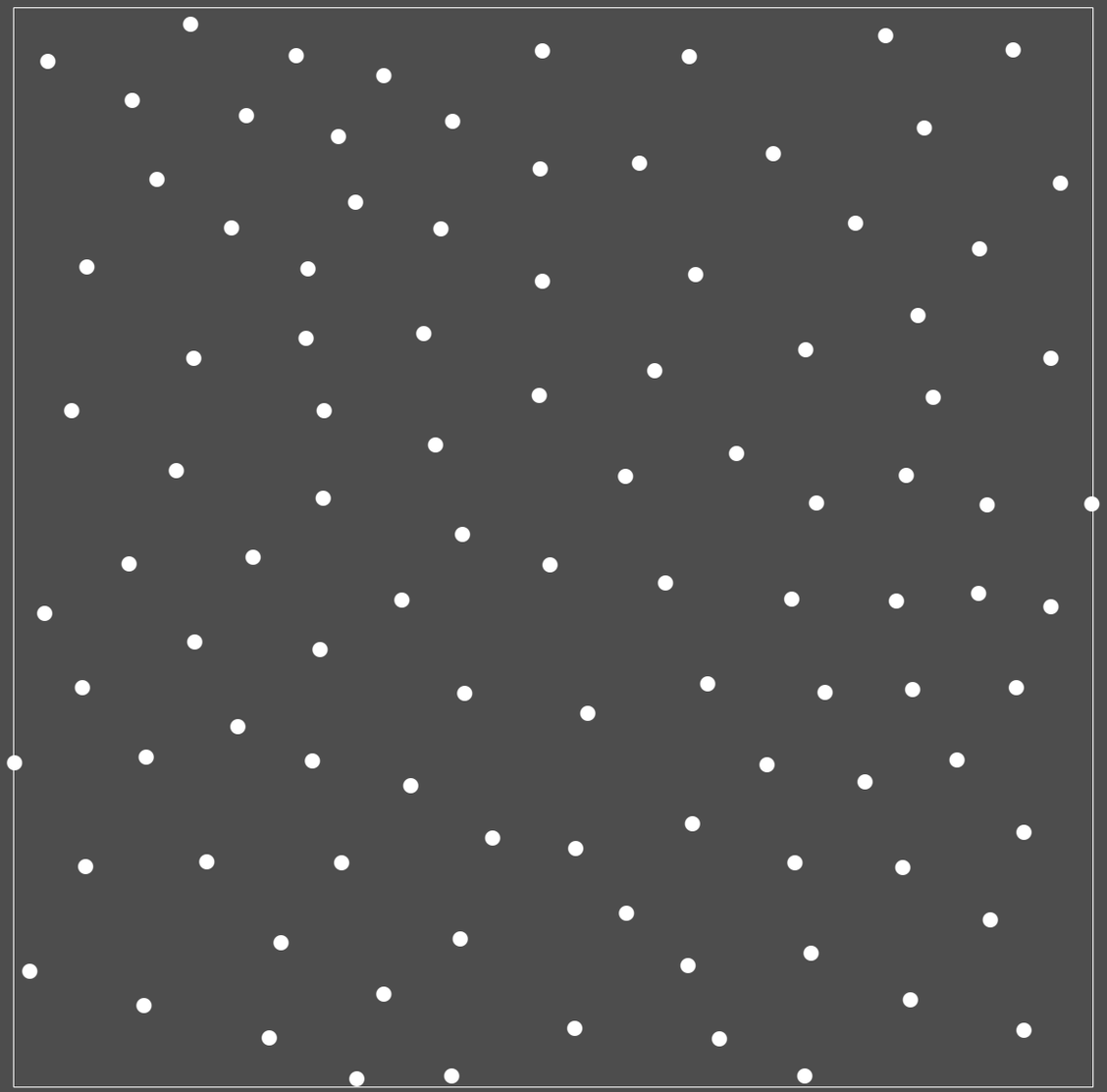
- Iteration 2

Stratification by Lloyd-Relaxation



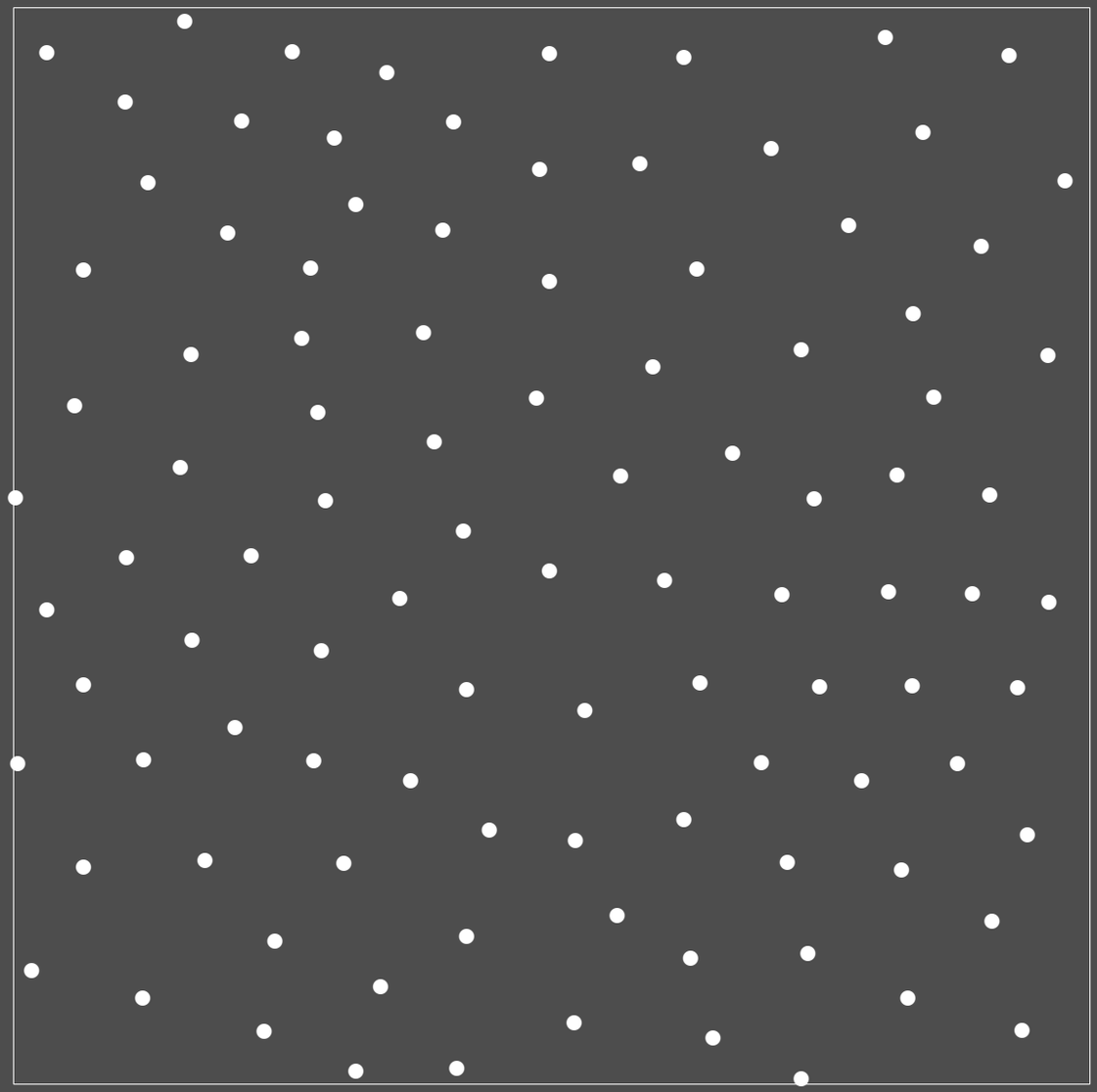
- Iteration 3

Stratification by Lloyd-Relaxation



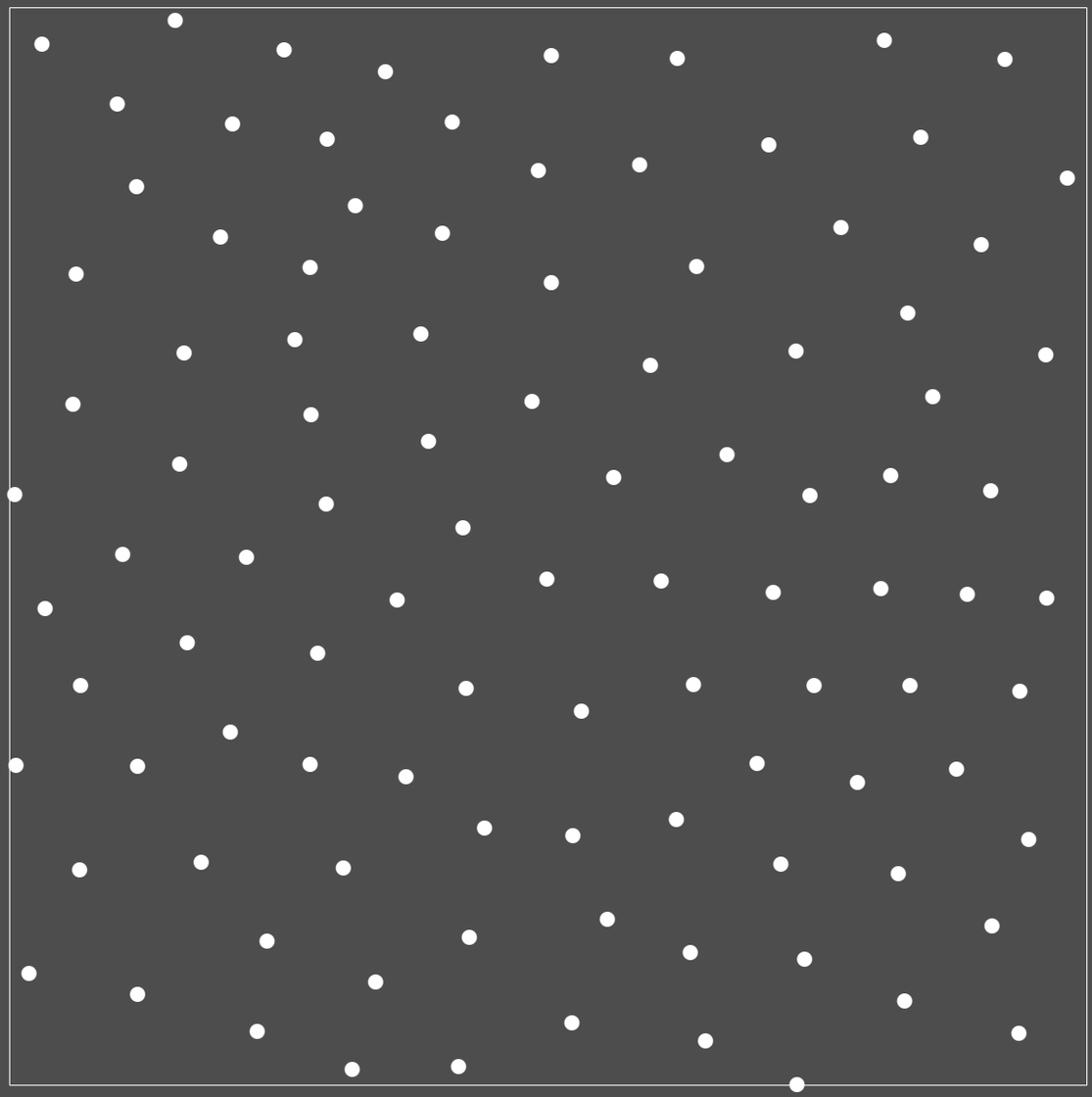
- Iteration 4

Stratification by Lloyd-Relaxation



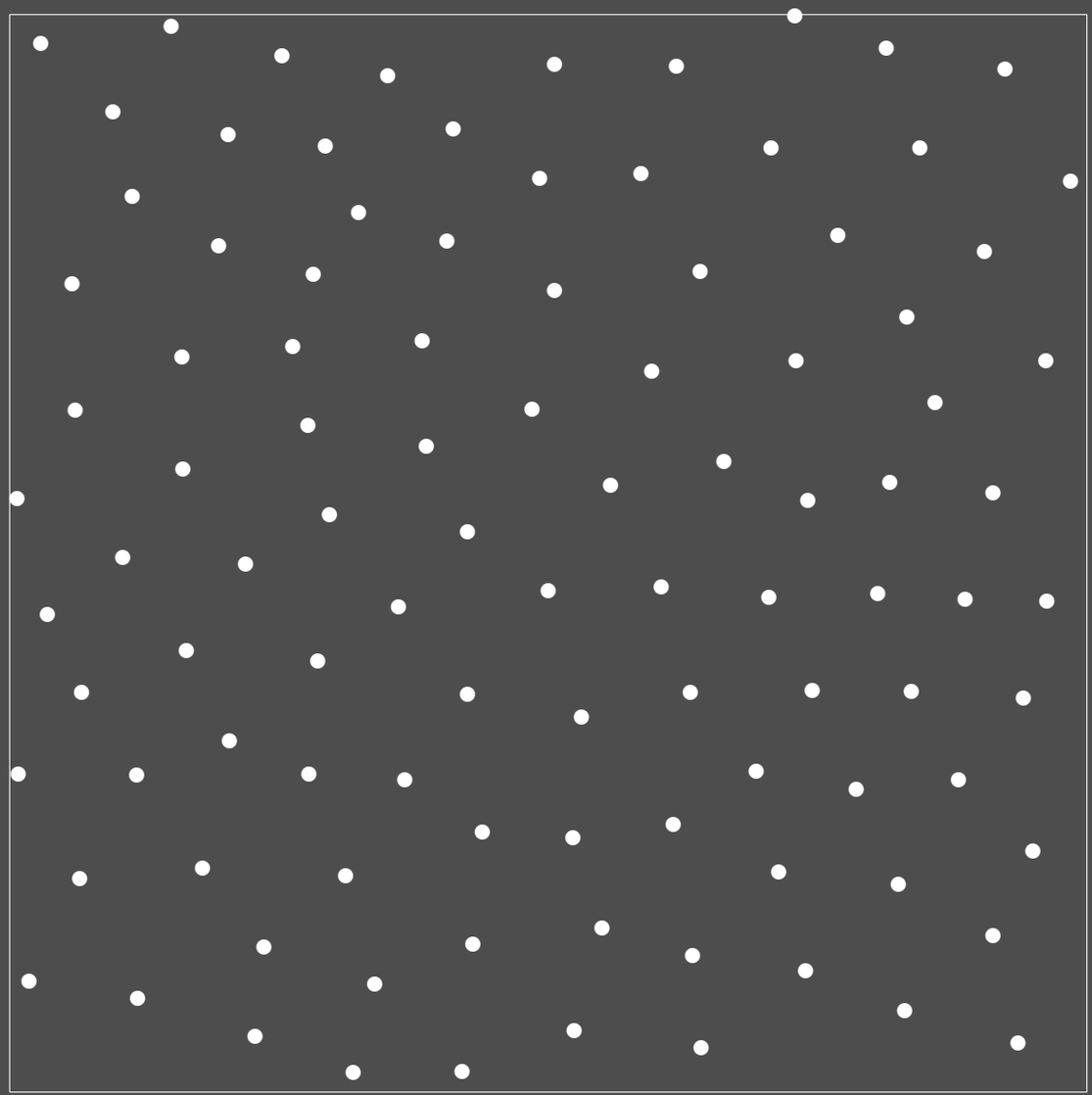
- Iteration 5

Stratification by Lloyd-Relaxation



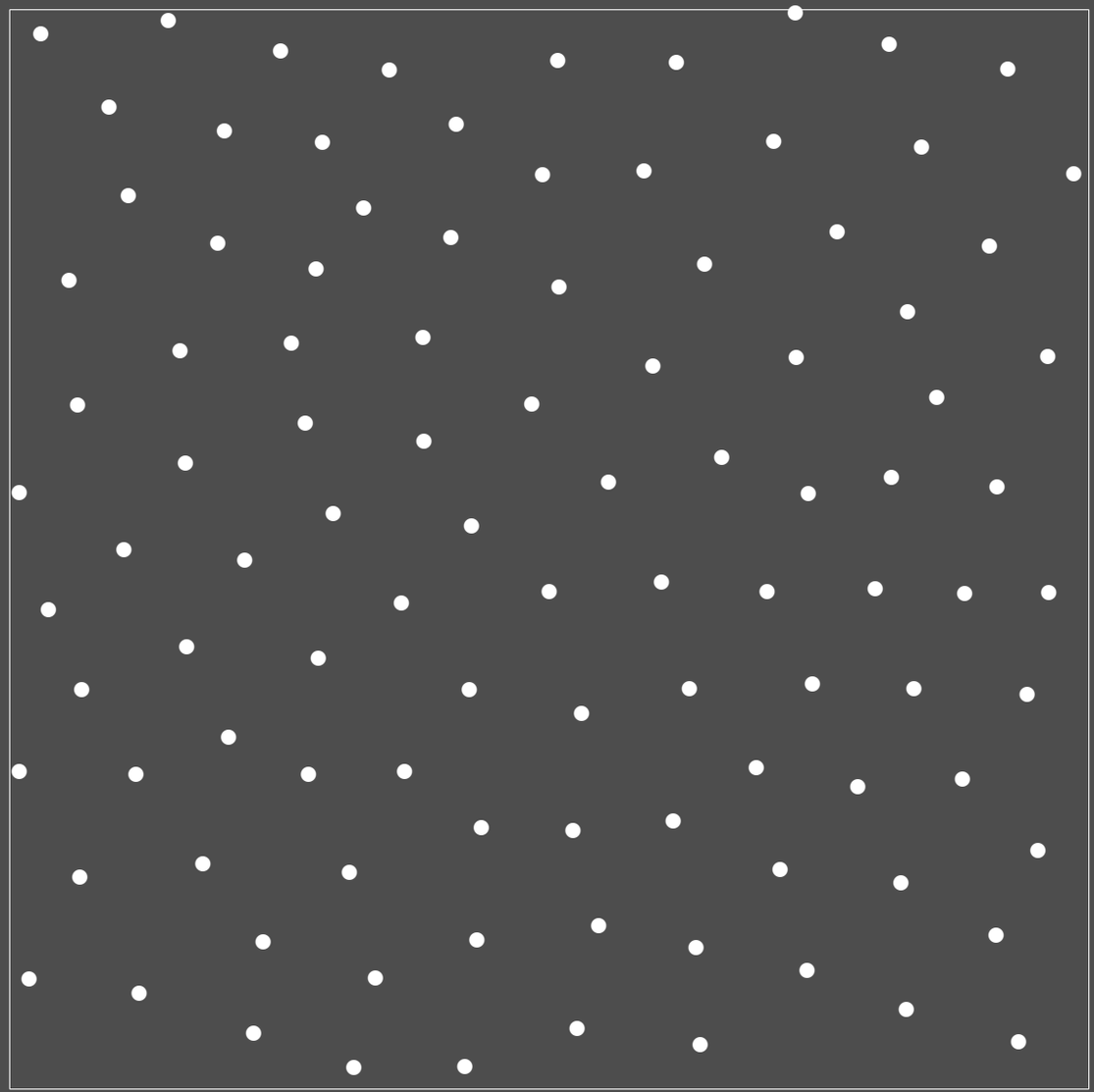
- Iteration 6

Stratification by Lloyd-Relaxation



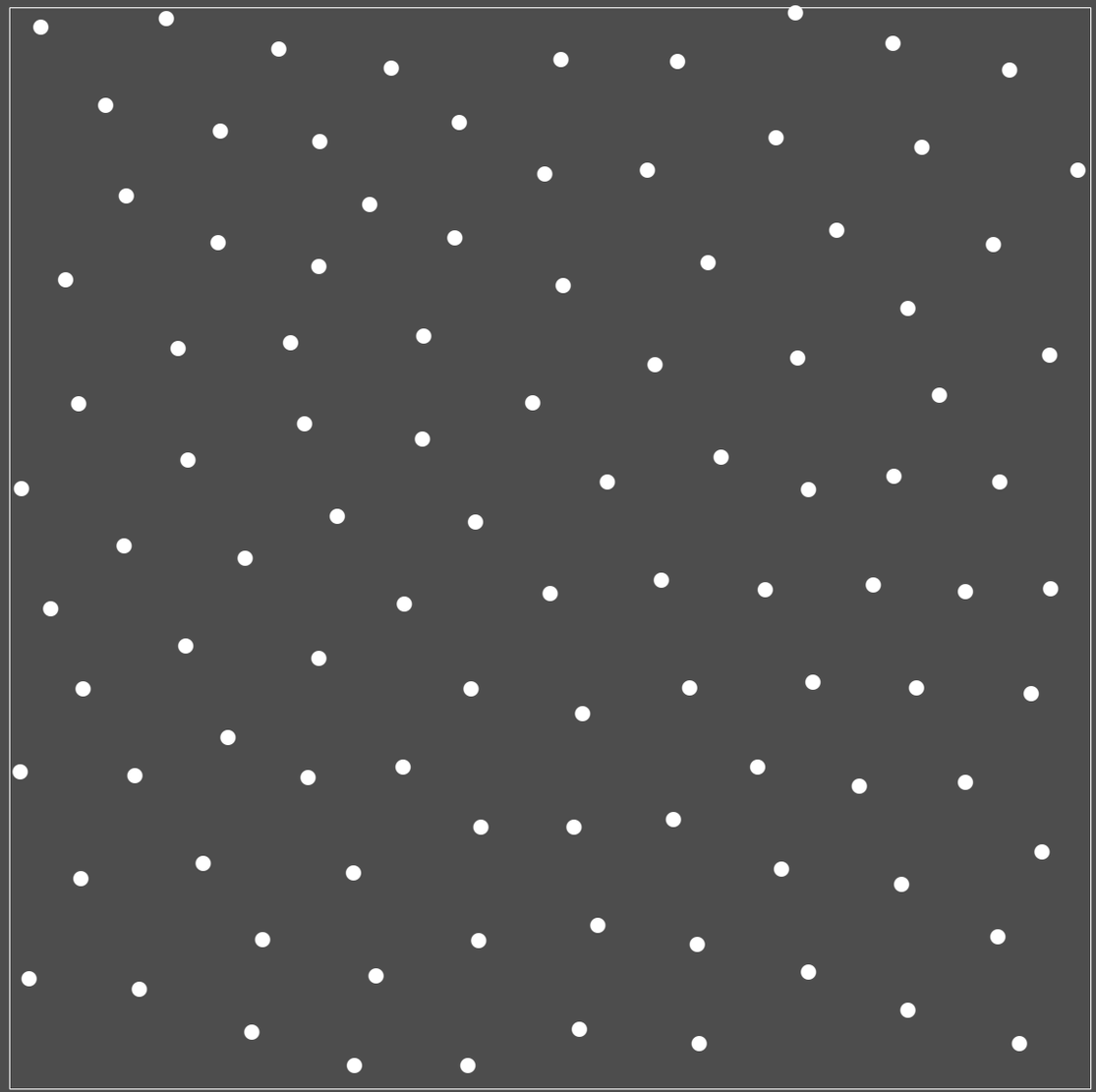
- Iteration 7

Stratification by Lloyd-Relaxation



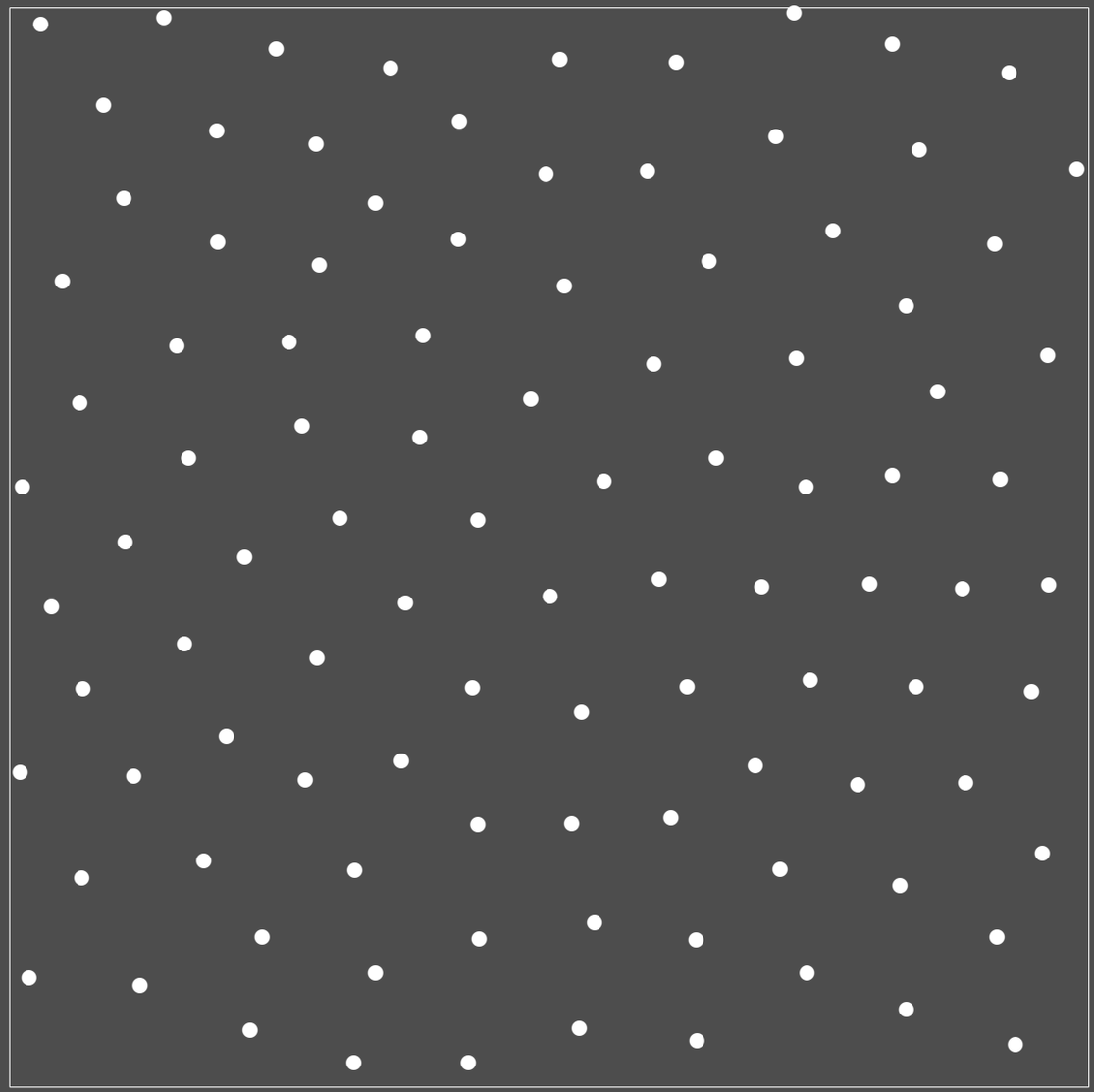
- Iteration 8

Stratification by Lloyd-Relaxation



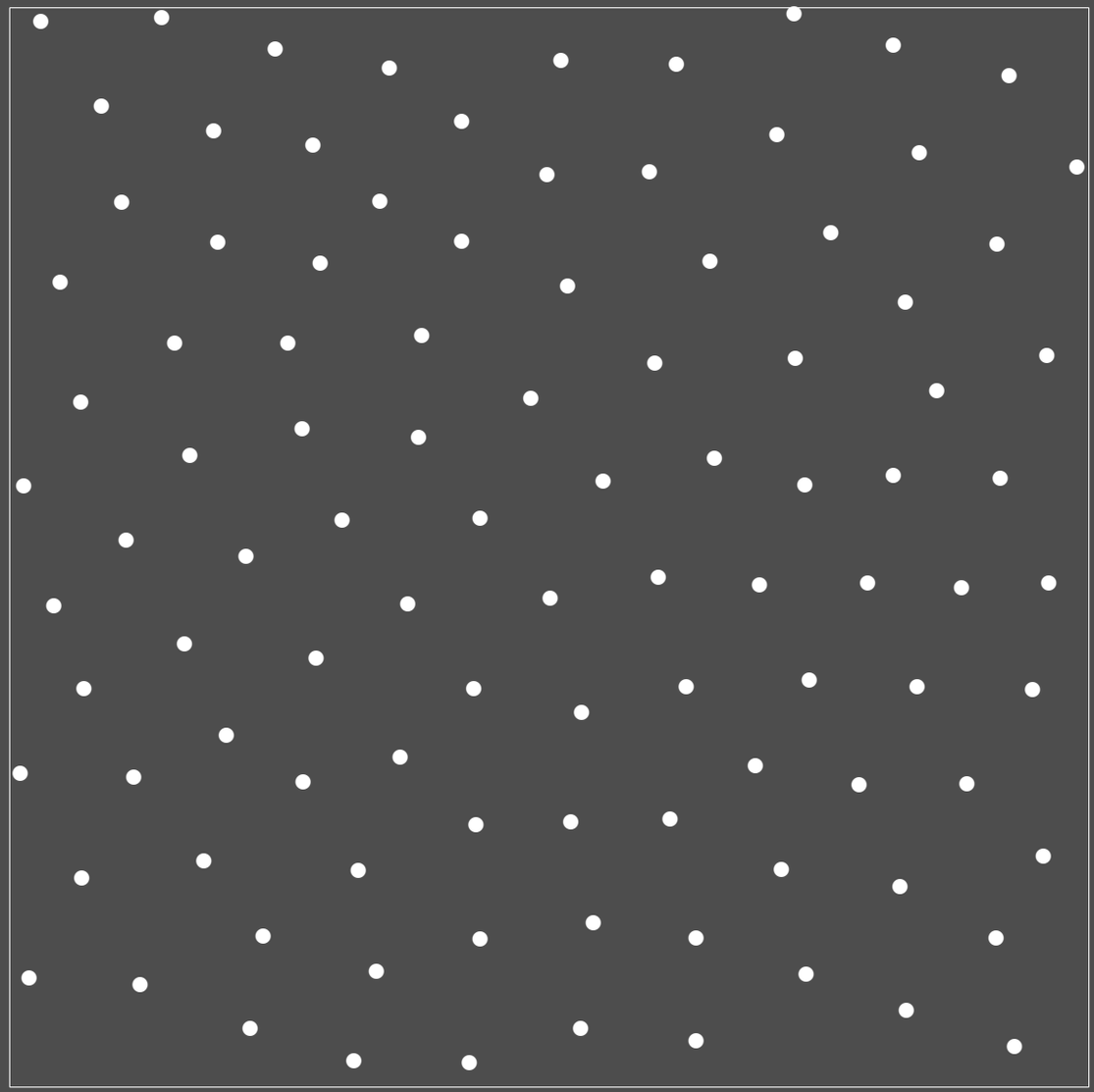
- Iteration 9

Stratification by Lloyd-Relaxation



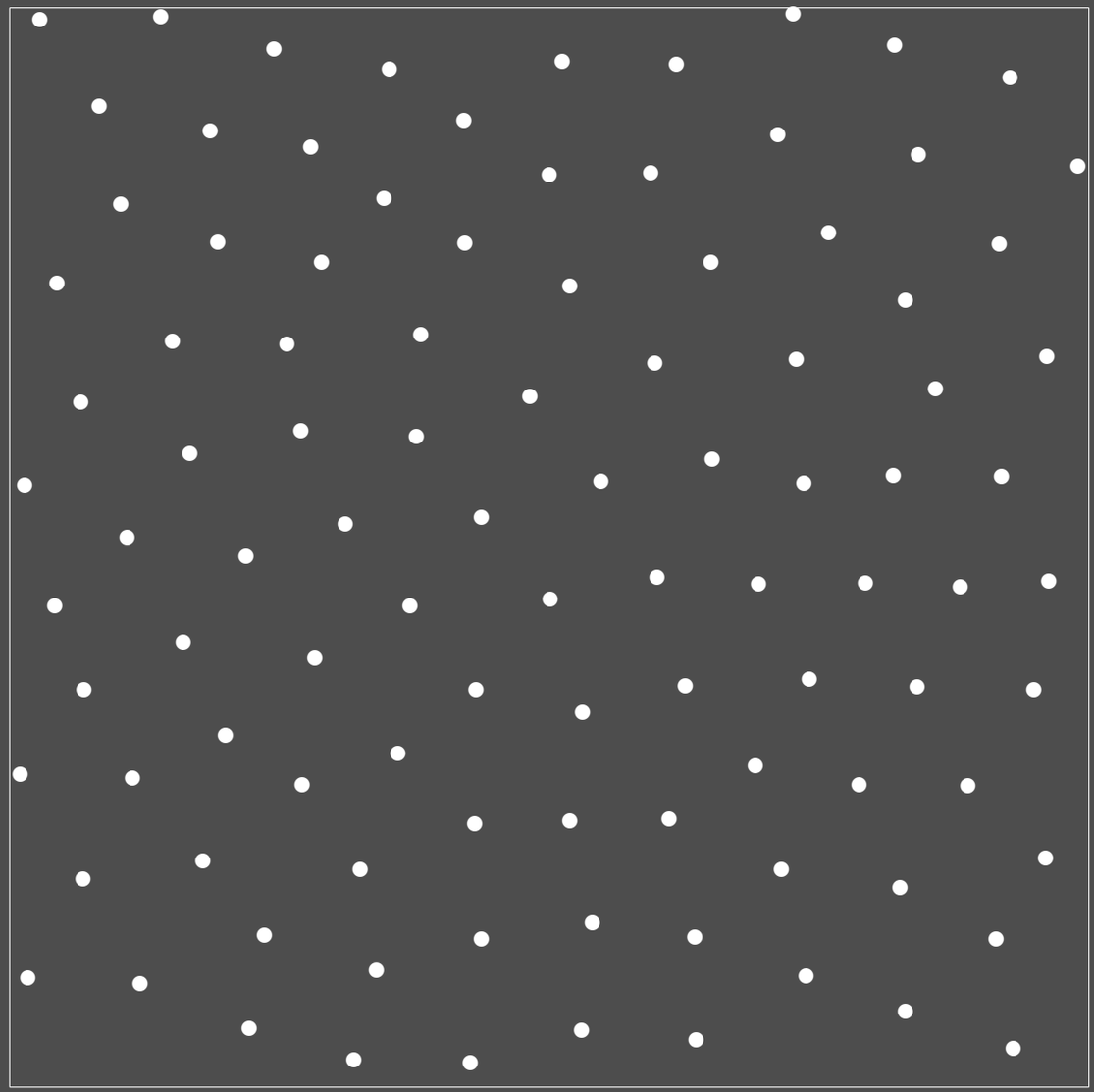
- Iteration 10

Stratification by Lloyd-Relaxation



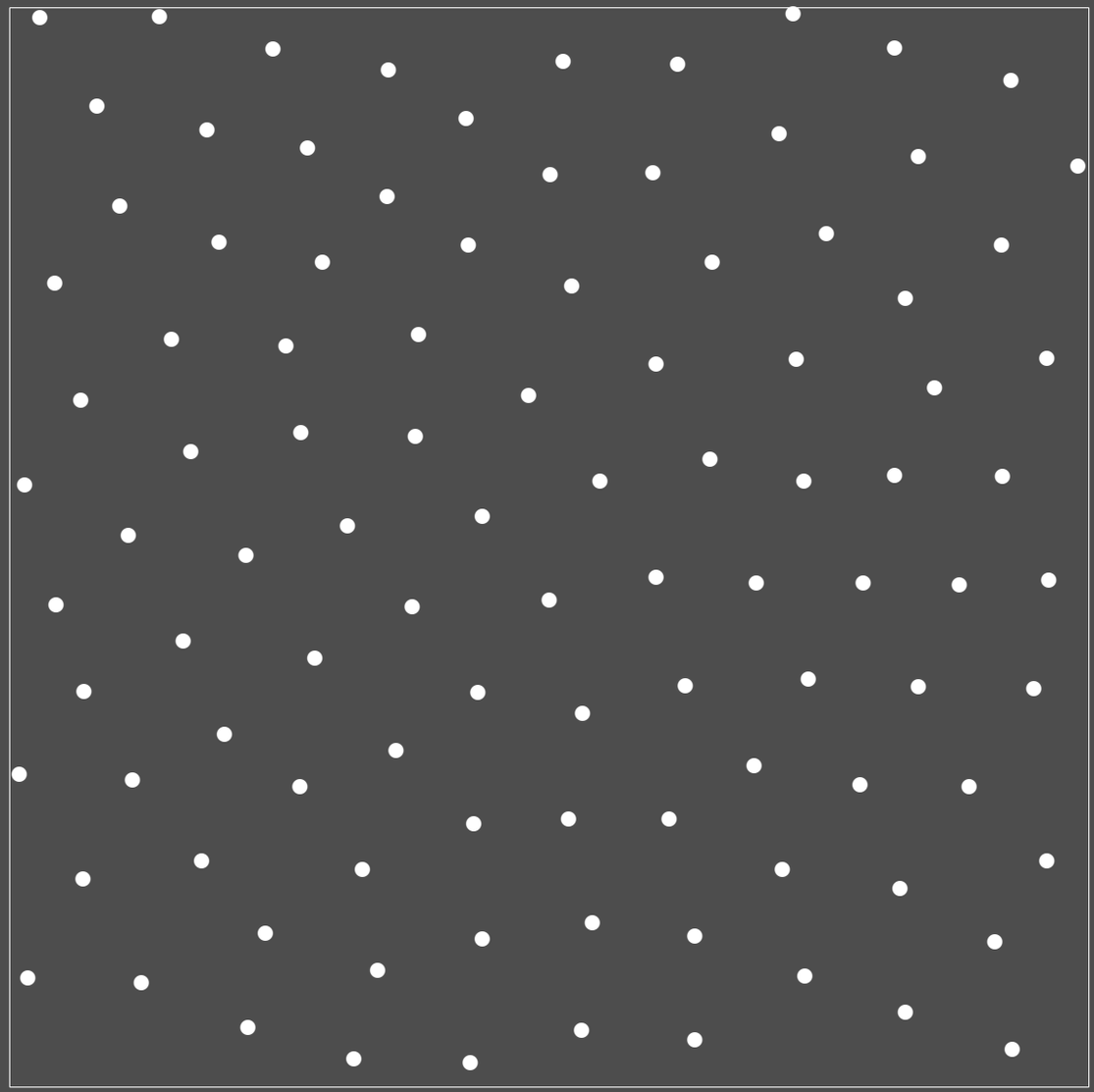
- Iteration 11

Stratification by Lloyd-Relaxation



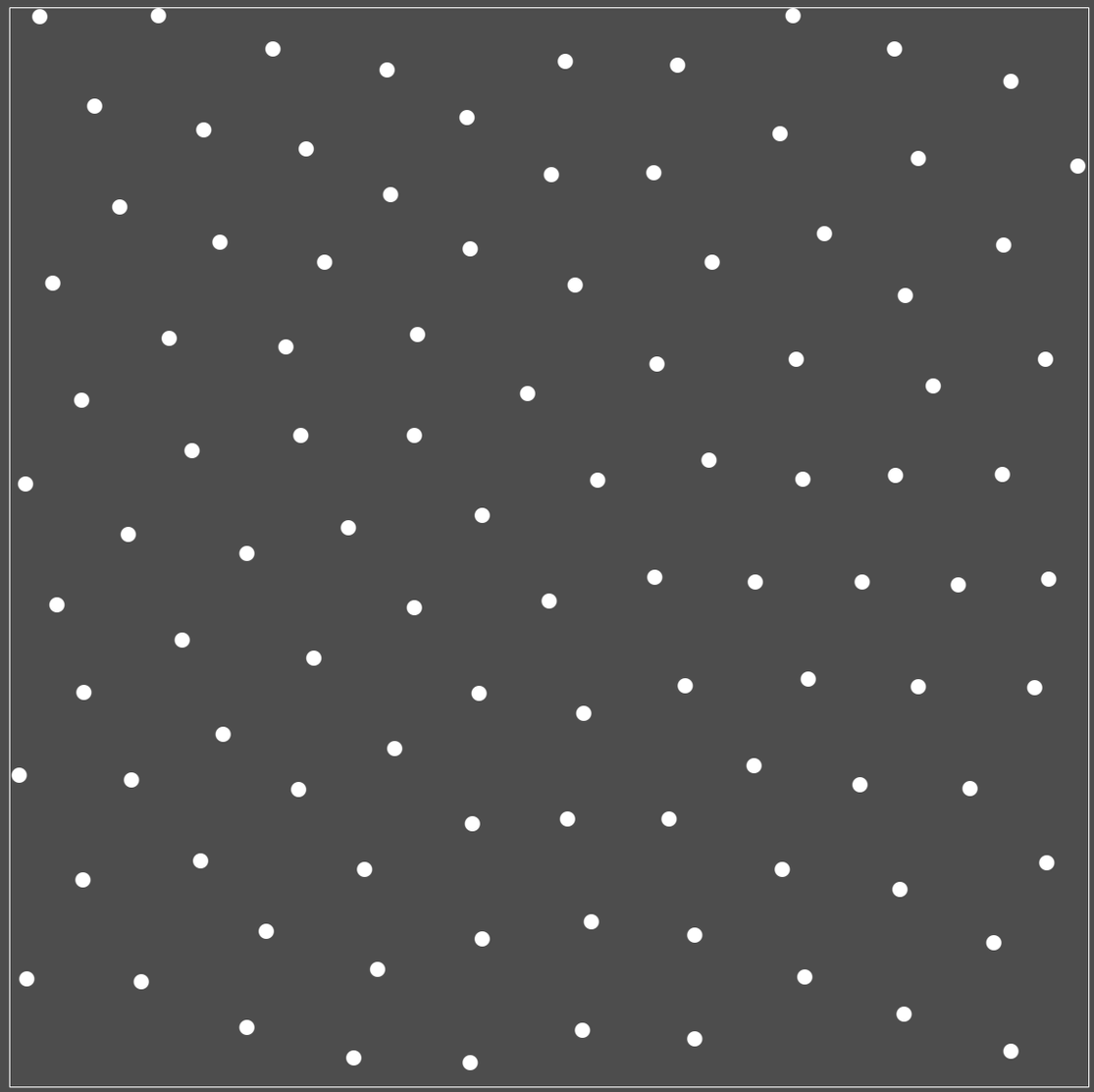
- Iteration 12

Stratification by Lloyd-Relaxation



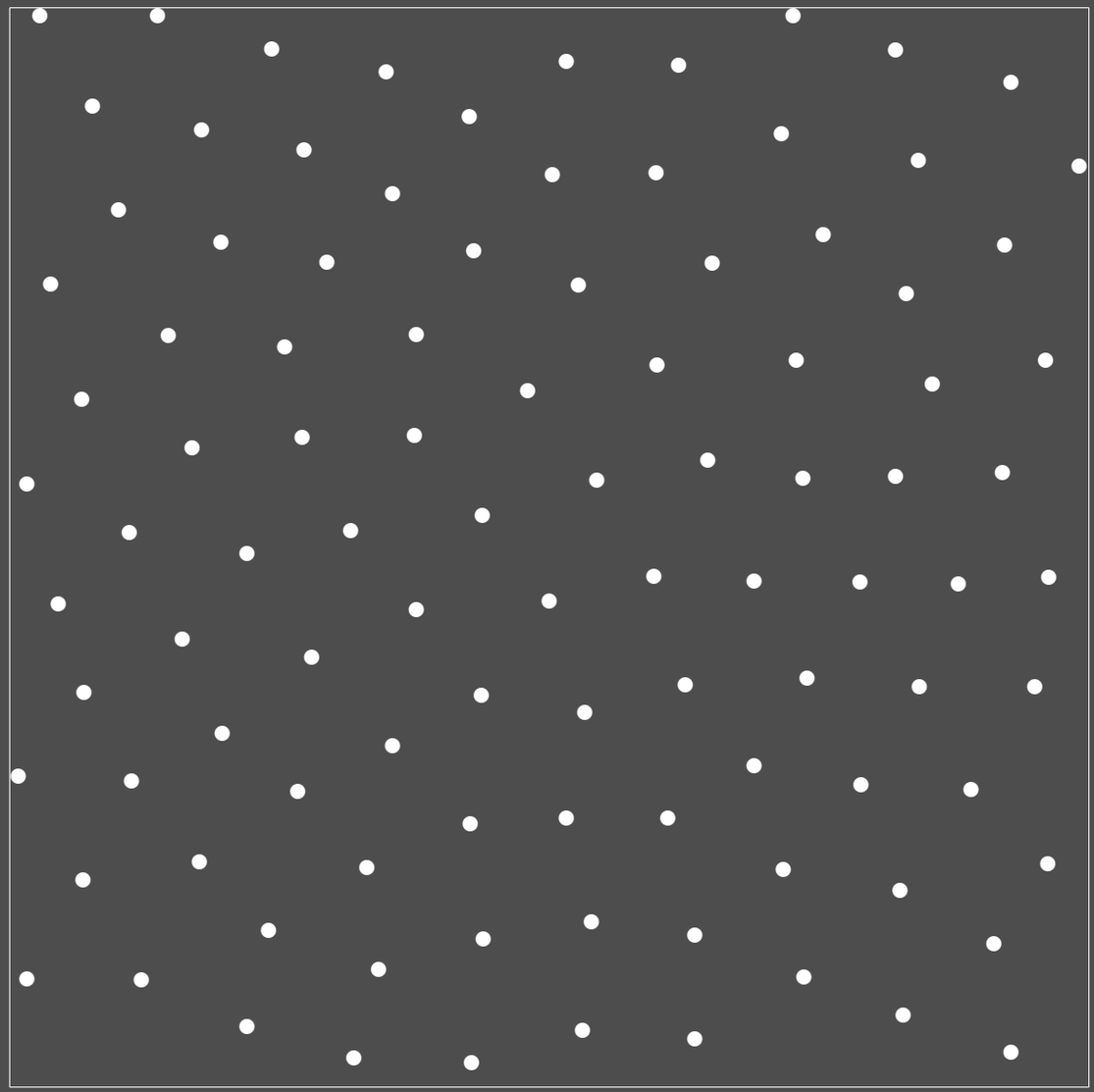
- Iteration 13

Stratification by Lloyd-Relaxation



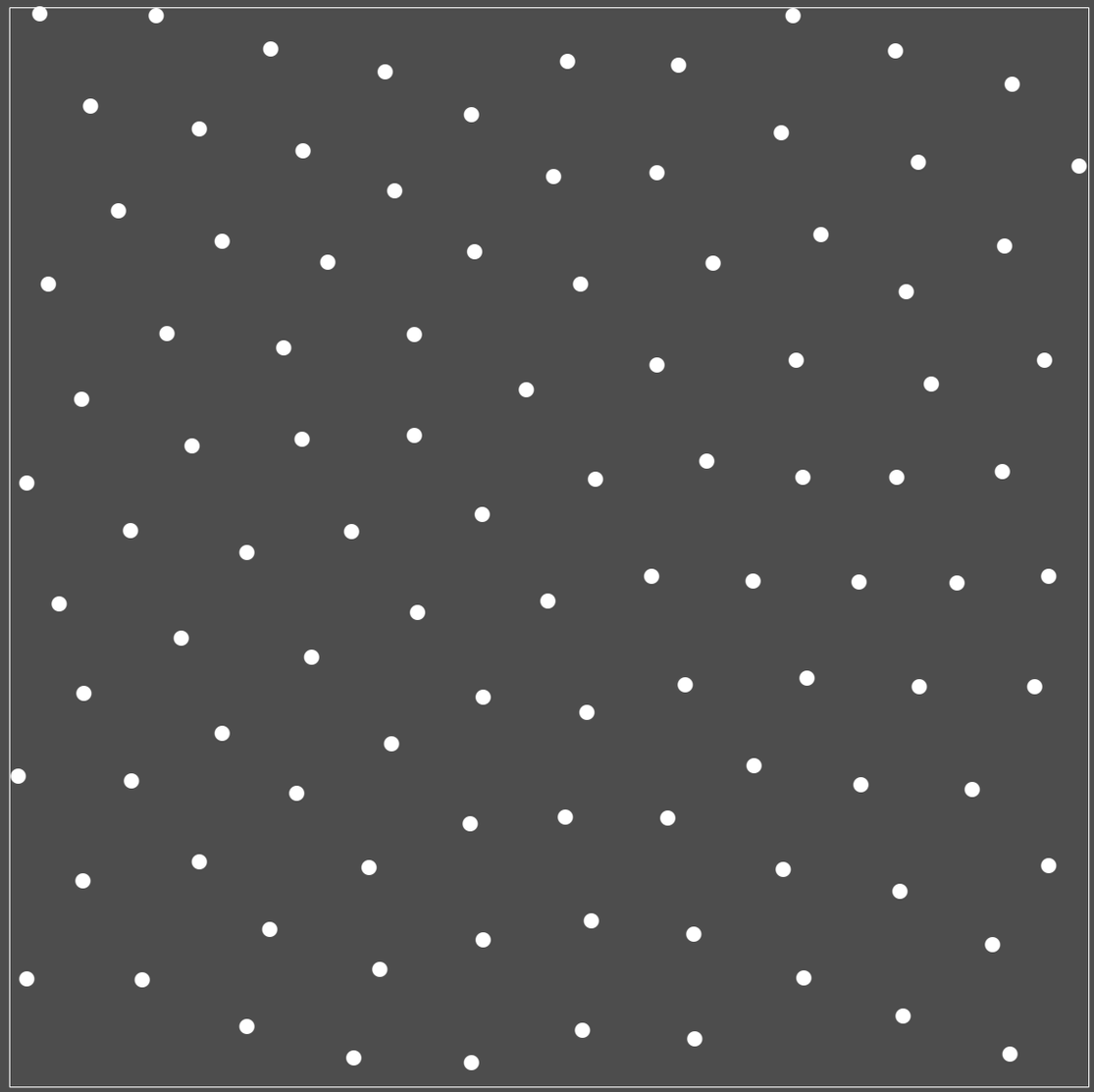
- Iteration 14

Stratification by Lloyd-Relaxation



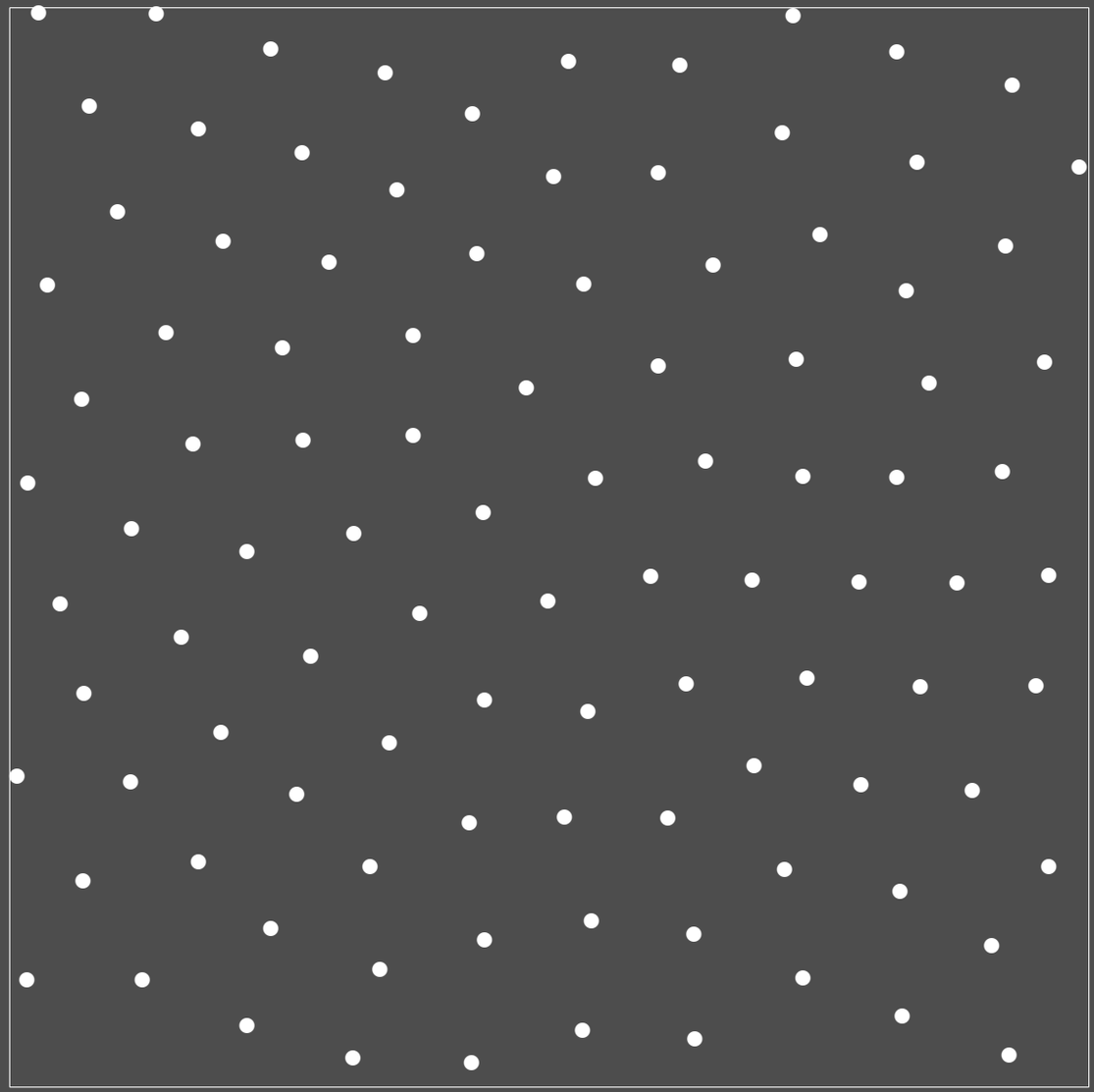
- Iteration 15

Stratification by Lloyd-Relaxation



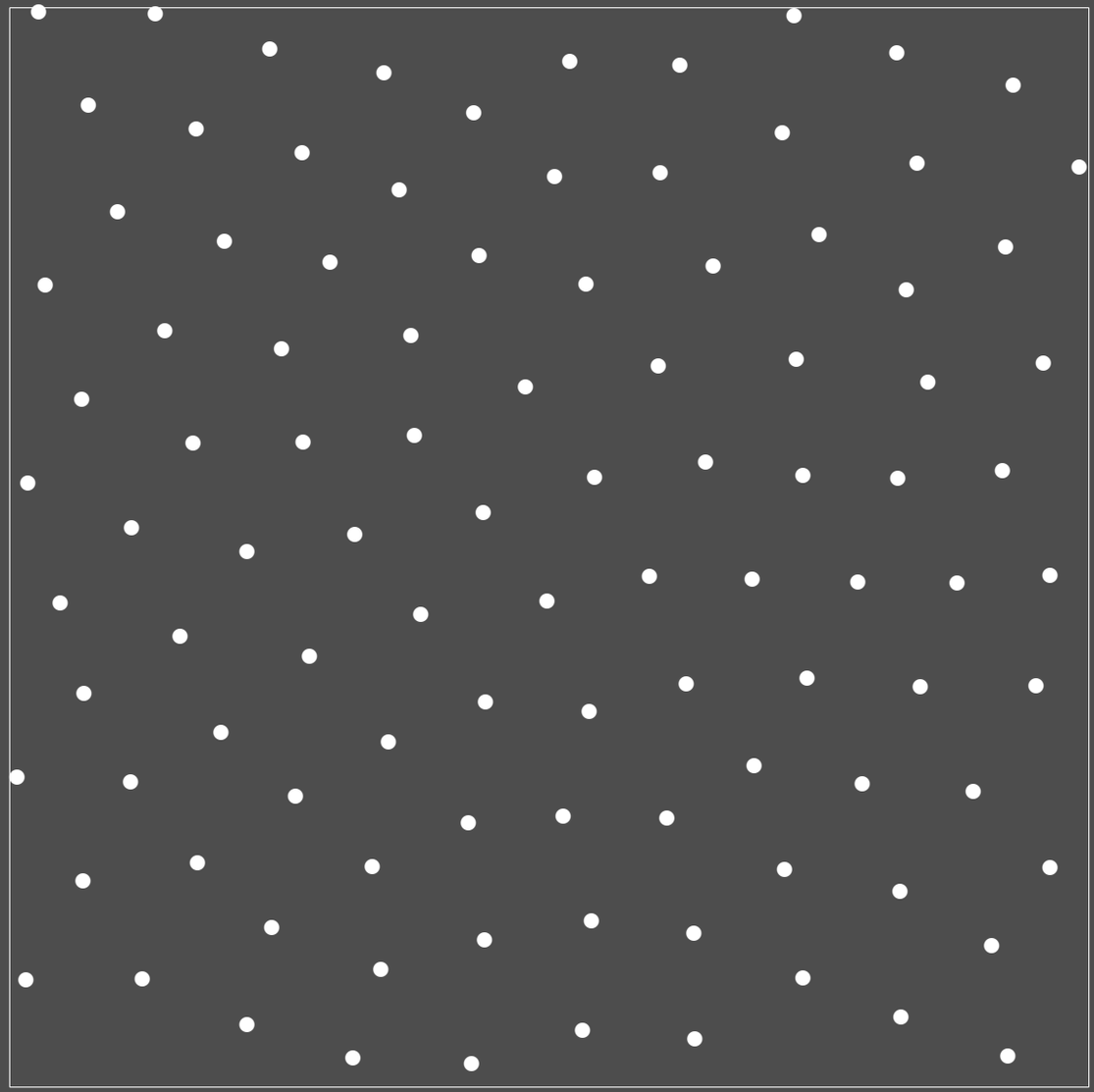
- Iteration 16

Stratification by Lloyd-Relaxation



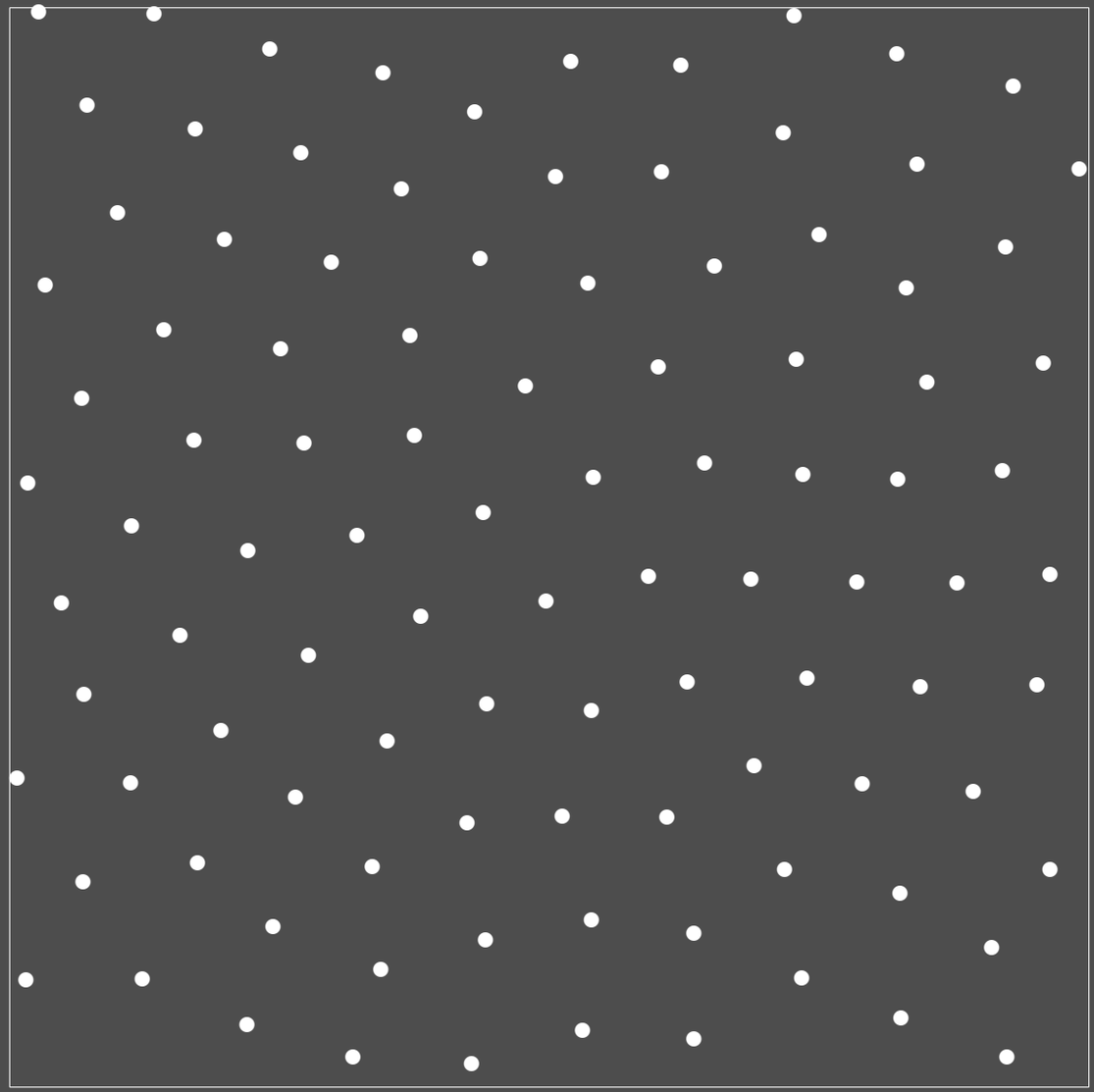
- Iteration 17

Stratification by Lloyd-Relaxation



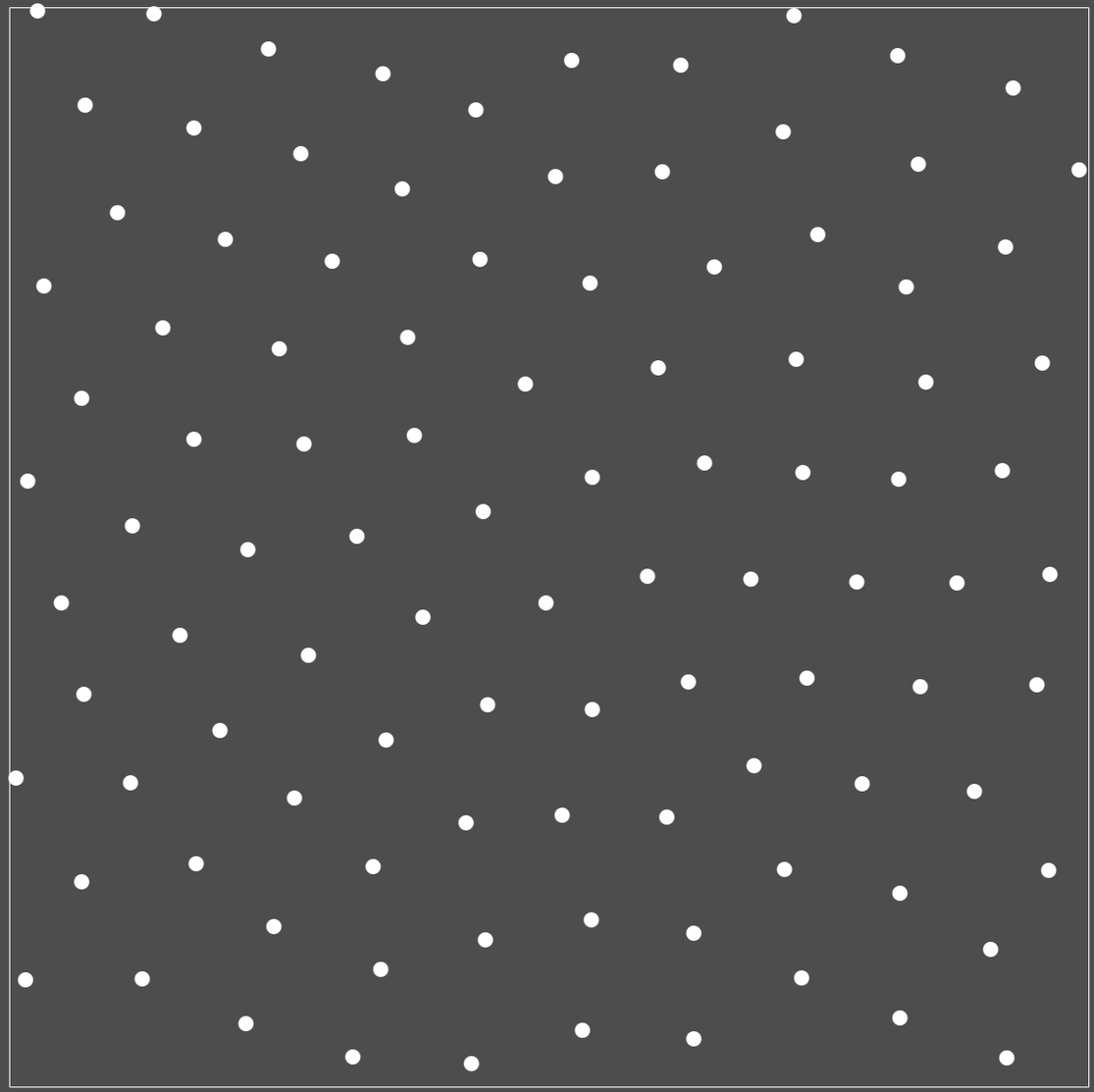
- Iteration 18

Stratification by Lloyd-Relaxation



- Iteration 19

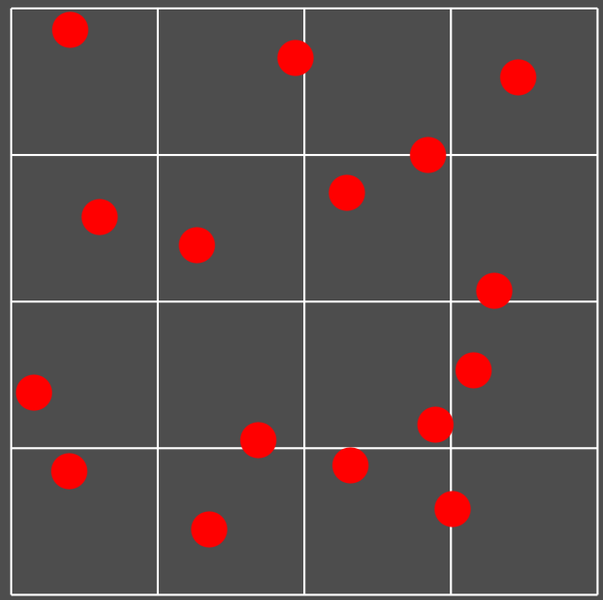
Stratification by Lloyd-Relaxation



- Iteration 20

Stratification: Jittered Sampling

- Division of each axis into N_j intervals for $N = \prod_{j=1}^{s} N_j$



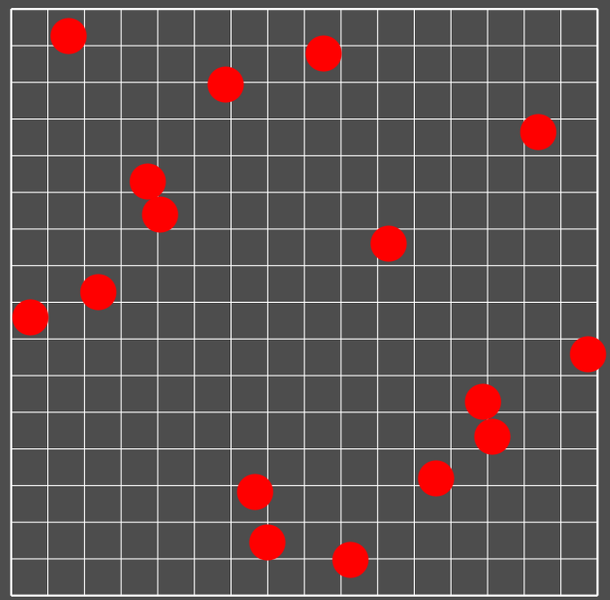
- Increased efficiency by increased uniformity of distribution
- Problem: N must be factorized

Latin Hypercube Sampling (N-Rooks Sampling)

- Using s uniform random permutations $\sigma_N^{(j)}$ of size N yields

$$x_i = \left(\frac{\sigma_N^{(1)}(i) + \xi_i^{(1)}}{N}, \ldots, \frac{\sigma_N^{(s)}(i) + \xi_i^{(s)}}{N} \right)$$

where $\sigma_N^{(1)}$ can be chosen as identity

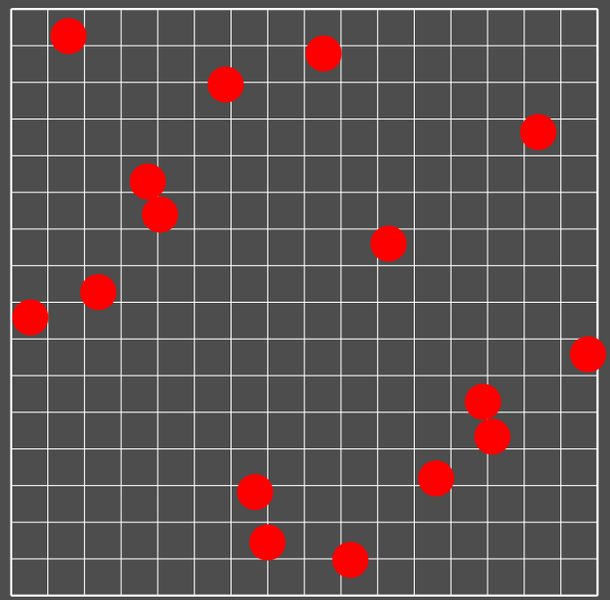


Latin Hypercube Sampling (N-Rooks Sampling)

- Using s uniform random permutations $\sigma_N^{(j)}$ of size N yields

$$x_i = \left(\frac{\sigma_N^{(1)}(i) + \xi_i^{(1)}}{N}, \ldots, \frac{\sigma_N^{(s)}(i) + \xi_i^{(s)}}{N} \right)$$

  where $\sigma_N^{(1)}$ can be chosen as identity



- Cannot be much worse than uniform random sampling

$$\sigma^2(f_{\text{LHS}}) \leq \frac{N}{N-1}\sigma^2(f_{\text{MC}})$$

Replication Heuristics: Stratification

- Heuristic with
  - weights $w_j = \lambda_s(A_j)$, and
  - mappings $R_j : I^s \to A_j$
- Independent sampling for $N_j = \lambda_s(A_j)N$

$$\int_{I^s} f(x)dx \approx \sum_{j=0}^{M-1} \frac{1}{N_j} \sum_{i=0}^{N_j-1} \lambda_s(A_j) f(R_j(x_{i,j})) = \frac{1}{N} \sum_{j=0}^{M-1} \sum_{i=0}^{N_j-1} f(R_j(x_{i,j}))$$

- Dependent sampling

$$\int_{I^s} f(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \lambda_s(A_j) f(R_j(x_i))$$

Replication Heuristics: Regularization

- Antithetic variables

$$\int_I f(x)dx = \int_I \frac{1}{2}f(x) + \frac{1}{2}f(1-x)dx \approx \frac{1}{2N}\sum_{i=0}^{N-1}\left(f(x_i) + f(1-x_i)\right)$$

  - sample points doubled and symmetrized
  - more efficient if variance reduced to less than half of original variance
  - good for monotonic problems
  - effect killed by independent sampling !

Replication Heuristics: Regularization

- Antithetic variables

$$\int_I f(x)dx = \int_I \frac{1}{2}f(x) + \frac{1}{2}f(1-x)dx \approx \frac{1}{2N}\sum_{i=0}^{N-1}\left(f(x_i) + f(1-x_i)\right)$$

  - sample points doubled and symmetrized
  - more efficient if variance reduced to less than half of original variance
  - good for monotonic problems
  - effect killed by independent sampling !

- Combining stratification

$$f_{\text{strat}}(x) = \frac{1}{2}\left(f\left(\frac{x}{2}\right) + f\left(1-\frac{x}{2}\right)\right)$$

  and antithetic variables

$$\int_I f_{\text{strat, anti}}(x)dx \approx \frac{1}{4N}\sum_{i=0}^{N-1}\left(f\left(\frac{x_i}{2}\right) + f\left(1-\frac{x_i}{2}\right) + f\left(\frac{1}{2}+\frac{x_i}{2}\right) + f\left(\frac{1}{2}-\frac{x_i}{2}\right)\right)$$

Splitting

- Instead of

$$\int_{I^{s1}} \int_{I^{s2}} f(x,y)dydx \approx \frac{1}{N} \sum_{i=0}^{N-1} f(x_i, y_i)$$

  computational complexity can be improved by

$$\int_{I^{s1}} \int_{I^{s2}} f(x,y)dydx \approx \frac{1}{NM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f(x_i, y_{i,j})$$

- Low pass filtering of problematic dimensions of the integrand
  - e.g. splitting for shadow rays

Replication Heuristics: Dependent Splitting

- Splitting considered as a replication heuristic restricted to selected dimensions

$$\int_{I^{s1}}\int_{I^{s2}} f(x,y)dydx = \int_{I^{s1}}\int_{I^{s2}} \sum_{j=0}^{M-1} w_j(x,y) f(x, R_j(x,y))dydx$$

$$\approx \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} w_j(x_i,y_i) f(x_i, R_j(x_i,y_i))dydx$$

- Realize splitting much more efficiently by e.g.
  - stratification heuristic (independent sampling)
  - randomized quadratures (dependent sampling)

Summary

- Simulation of random variables and fields
- Monte Carlo integration
- Method of dependent tests
- Efficiency and time complexity
- Dependent sampling
- Replication

Summary

- Simulation of random variables and fields
- Monte Carlo integration
- Method of dependent tests
- Efficiency and time complexity
- Dependent sampling
- Replication

$\Rightarrow$ **Use as few random numbers as possible**

Monte Carlo and Beyond

- Principles of rendering algorithms
- Monte Carlo integration
- **Quasi-Monte Carlo points**
  - Discrepancy
  - Deterministic low discrepancy
    - Halton and Hammersley points
    - Scrambling
    - (t, m, s)-nets and (t, s)-sequences
    - Digital constructions
    - Good lattice points
- Quasi-Monte Carlo integration
- Monte Carlo extensions of quasi-Monte Carlo
- Application to computer graphics

Discrepancy

- **Definition:** The ***discrepancy***

$$D(P_N, \mathcal{A}) := \sup_{A \in \mathcal{A}} \left| \lambda_s(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

is a measure of the uniform distribution of a given point set $P_N = \{x_0, \ldots, x_{N-1}\}$ with respect to non-empty families $\mathcal{A}$ of Lebesgue-measurable subsets of I^s. χ_A is the characteristic function of the set A.

Discrepancy

- **Definition:** The ***discrepancy***

$$D(P_N, \mathcal{A}) := \sup_{A \in \mathcal{A}} \left| \lambda_s(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

  is a measure of the uniform distribution of a given point set $P_N = \{x_0, \ldots, x_{N-1}\}$ with respect to non-empty families $\mathcal{A}$ of Lebesgue-measurable subsets of I^s. χ_A is the characteristic function of the set A.

- $D(P_N, \mathcal{A}) \sim$ worst case integration error

Discrepancy

- **Definition:** The ***discrepancy***

$$D(P_N, \mathcal{A}) := \sup_{A \in \mathcal{A}} \left| \lambda_s(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

  is a measure of the uniform distribution of a given point set $P_N = \{x_0, \ldots, x_{N-1}\}$ with respect to non-empty families $\mathcal{A}$ of Lebesgue-measurable subsets of I^s. χ_A is the characteristic function of the set A.

- $D(P_N, \mathcal{A}) \sim$ worst case integration error

- (Star-) discrepancy

$$D^*(P_N) := D\left(P_N, \left\{A | A = \prod_{j=1}^{s} [0, a_j) \subset I^s\right\}\right)$$

- Extreme discrepancy

$$D(P_N) := D\left(P_N, \left\{A | A = \prod_{j=1}^{s} [a_j, b_j) \subset I^s\right\}\right)$$

Discrepancy

- **Definition:** The ***discrepancy***

$$D(P_N, \mathcal{A}) := \sup_{A \in \mathcal{A}} \left| \lambda_s(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

  is a measure of the uniform distribution of a given point set $P_N = \{x_0, \ldots, x_{N-1}\}$ with respect to non-empty families $\mathcal{A}$ of Lebesgue-measurable subsets of I^s. χ_A is the characteristic function of the set A.

- $D(P_N, \mathcal{A}) \sim$ worst case integration error

- (Star-) discrepancy

$$D^*(P_N) := D\left(P_N, \left\{A | A = \prod_{j=1}^{s} [0, a_j) \subset I^s\right\}\right)$$

- Extreme discrepancy

$$D(P_N) := D\left(P_N, \left\{A | A = \prod_{j=1}^{s} [a_j, b_j) \subset I^s\right\}\right)$$

- The (Star-) discrepancy and extreme discrepancy are anisotropic measures

Discrepancy Bounds

- Case $s = 1$: Discrepancy is size of largest gap

$$D^*(P_N) \geq \frac{1}{2N}$$

$$D(P_N) \geq \frac{1}{N}$$

- General case

$$D^*(P_N) \geq B_s \frac{\log^{\frac{s-1}{2}} N}{N}$$

Discrepancy Bounds

- Case $s = 1$: Discrepancy is size of largest gap

$$D^*(P_N) \geq \frac{1}{2N}$$

$$D(P_N) \geq \frac{1}{N}$$

- General case

$$D^*(P_N) \geq B_s \frac{\log^{\frac{s-1}{2}} N}{N}$$

- Discrepancy of random points

$$D^*(P_N^{\text{random}}) \in \mathcal{O}\left(\sqrt{\frac{\log\log N}{N}}\right)$$

- Discrepancy of regular grids

$$D^*(P_N) \in \mathcal{O}\left(\frac{1}{\sqrt[s]{N}}\right)$$

Discrepancy Bounds

- Case $s = 1$: Discrepancy is size of largest gap

$$D^*(P_N) \geq \frac{1}{2N}$$

$$D(P_N) \geq \frac{1}{N}$$

- General case

$$D^*(P_N) \geq B_s \frac{\log^{\frac{s-1}{2}} N}{N}$$

- Discrepancy of random points

$$D^*(P_N^{\text{random}}) \in \mathcal{O}\left(\sqrt{\frac{\log\log N}{N}}\right)$$

- Discrepancy of regular grids

$$D^*(P_N) \in \mathcal{O}\left(\frac{1}{\sqrt[s]{N}}\right)$$

  - includes points taken from space filling curves like e.g. the Hilbert curve

Uniform and Completely Uniform Distribution

- By the theory of uniform distribution

  (x_i) is uniformly distributed in I^s

  $\Leftrightarrow \lim_{N\to\infty} D(P_N) = 0$

  $\Leftrightarrow \lim_{N\to\infty} D^*(P_N) = 0$

Uniform and Completely Uniform Distribution

- By the theory of uniform distribution

  (x_i) is uniformly distributed in I^s

  $\Leftrightarrow \lim_{N\to\infty} D(P_N) = 0$

  $\Leftrightarrow \lim_{N\to\infty} D^*(P_N) = 0$

- **Definition:** A sequence (x_i) of numbers in I is **completely uniformly distributed** if for every $s \in \mathbb{N}$ the sequence of points $(x_n, x_{n+1}, \ldots, x_{n+s-1})$ is uniformly distributed in I^s for $n \in \mathbb{N}_0$.
- Formalization of independence

Quasi-Monte Carlo Point Sets

- Low discrepancy means

$$D^*(P_N) \in \mathcal{O}\left(\frac{\log^s N}{N}\right)$$

Quasi-Monte Carlo Point Sets

- Low discrepancy means

$$D^*(P_N) \in \mathcal{O}\left(\frac{\log^s N}{N}\right)$$

- Low discrepancy sequences cannot be completely uniformly distributed

Quasi-Monte Carlo Point Sets

- Low discrepancy means

$$D^*(P_N) \in \mathcal{O}\left(\frac{\log^s N}{N}\right)$$

- Low discrepancy sequences cannot be completely uniformly distributed

- Quasi-Monte Carlo points means
  - low discrepancy and
  - deterministic points

  $\Rightarrow$ Discrete density approximation of uniform distribution $\mathcal{U}$

Halton Sequence and Hammersley Points

- Radical inverse (van der Corput sequence) in base b

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1}$$

Halton Sequence and Hammersley Points

- Radical inverse (van der Corput sequence) in base b

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1}$$

Note: The radical inverses are not completely uniform distributed !!!

Halton Sequence and Hammersley Points

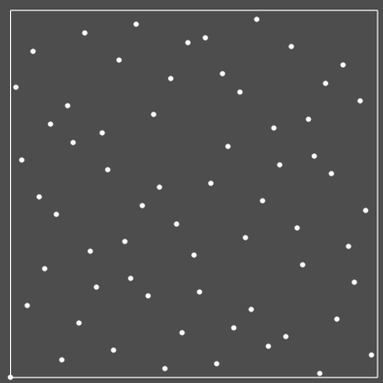
- Radical inverse (van der Corput sequence) in base b

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1}$$

Note: The radical inverses are not completely uniform distributed !!!

- Halton sequence $x_i := \left(\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i)\right)$ where b_i is the i-th prime number

$$D^*(P_N^{\text{Halton}}) < \frac{s}{N} + \frac{1}{N} \prod_{j=1}^{s} \left(\frac{b_j - 1}{2 \log b_j} \log N + \frac{b_j + 1}{2} \right)$$



Halton Sequence and Hammersley Points

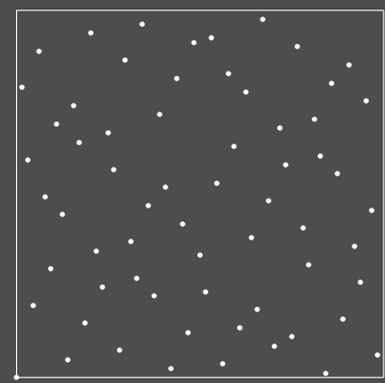
- Radical inverse (van der Corput sequence) in base b

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1}$$

Note: The radical inverses are not completely uniform distributed !!!

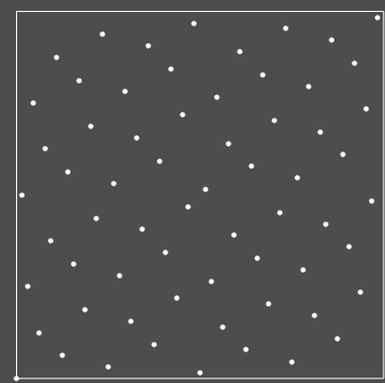
- Halton sequence $x_i := \left(\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i)\right)$ where b_i is the i-th prime number

$$D^*(P_N^{\text{Halton}}) < \frac{s}{N} + \frac{1}{N} \prod_{j=1}^{s} \left(\frac{b_j - 1}{2 \log b_j} \log N + \frac{b_j + 1}{2} \right)$$



- Hammersley point set $x_i := \left(\frac{i}{N}, \Phi_{b_1}(i), \ldots, \Phi_{b_{s-1}}(i)\right)$

$$D^*(P_N^{\text{Hammersley}}) < \frac{s}{N} + \frac{1}{N} \prod_{j=1}^{s-1} \left(\frac{b_j - 1}{2 \log b_j} \log N + \frac{b_j + 1}{2} \right)$$



Algorithm: Radical Inversion

```
double RadicalInverse(const int Base, int i)
{
  double Digit, Radical, Inverse;

  Digit = Radical = 1.0 / (double) Base;
  Inverse = 0.0;

  while(i)
  {
    Inverse += Digit * (double) (i % Base);
    Digit *= Radical;
    i /= Base;
  }

  return Inverse;
}
```

Algorithm: Incremental Radical Inversion

```
double NextRadicalInverse(const double Radical, double Inverse)
// Radical = 1.0 / Base
{
   const double AlmostOne = 1.0 - 1e-10;
   double NextInverse, Digit1, Digit2;

   NextInverse = Inverse + Radical;

   if(NextInverse < AlmostOne)
      return NextInverse;
   else
   {
      Digit1 = Radical;
      Digit2 = Radical * Radical;

      while(Inverse + Digit2 >= AlmostOne)
      {
         Digit1 = Digit2;
         Digit2 *= Radical;
      }

      return Inverse + (Digit1 - 1.0) + Digit2;
   }
}
```

Other Discrepancies

- Isotropic discrepancy $J(P_N)$
  - $\mathcal{A}$ is family of all convex subsets of I^s

Other Discrepancies

- Isotropic discrepancy $J(P_N)$
  - $\mathcal{A}$ is family of all convex subsets of I^s
  - by

$$D^*(P_N) \leq D(P_N) \leq 2^s D^*(P_N)$$
$$D(P_N) \leq J(P_N) \leq 4sD(P_N)^{1/s}$$

    - upper bound

$$J(P_N) \leq 4sD(P_N)^{1/s} \leq 4s(2^s D^*(P_N))^{1/s} = 8sD^*(P_N)^{1/s}$$

    - lower bound

$$J(P_N) \geq D(P_N) \geq D^*(P_N)$$

Other Discrepancies

- Isotropic discrepancy $J(P_N)$
  - $\mathcal{A}$ is family of all convex subsets of I^s
  - by

$$D^*(P_N) \leq D(P_N) \leq 2^s D^*(P_N)$$
$$D(P_N) \leq J(P_N) \leq 4sD(P_N)^{1/s}$$

    * upper bound

$$J(P_N) \leq 4sD(P_N)^{1/s} \leq 4s(2^s D^*(P_N))^{1/s} = 8sD^*(P_N)^{1/s}$$

    * lower bound

$$J(P_N) \geq D(P_N) \geq D^*(P_N)$$

- Triangle discrepancy
- Edge discrepancy

Computing Discrepancies

- L_2-norm based discrepancy

$$D_2^*(P_N) := \sqrt{\int_{I^s} \left(\lambda_s(A(x)) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_{A(x)}(x_i) \right)^2 dx}$$

  where $A(x) = \prod_{j=1}^{s} [0, x^{(j)})$

- Can be efficiently computed in contrast to L_∞-norm based discrepancies

Computing Discrepancies

- L_2-norm based discrepancy

$$D_2^*(P_N) := \sqrt{\int_{I^s} \left(\lambda_s(A(x)) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_{A(x)}(x_i) \right)^2 dx}$$

  where $A(x) = \prod_{j=1}^{s} [0, x^{(j)})$

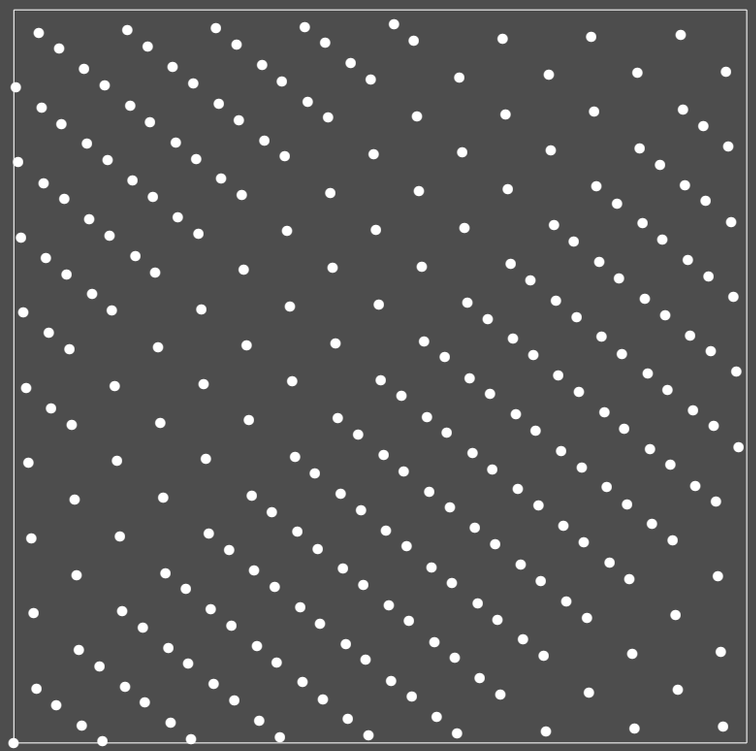
- Can be efficiently computed in contrast to L_∞-norm based discrepancies
- Numerical example: Triangular discrepancy

$$D(P_N, \mathcal{T}) \leq J(P_N) \leq 16\sqrt{D^*(P_N)}$$

N	10000 random triangles	100000 random triangles	theoretical bound
4	0.539712	0.591708	16.971
16	0.18326	0.230355	9.381
64	0.0660696	0.0777368	5.099
256	0.032454	0.0364673	2.739
1024	0.0118695	0.0178952	1.458
4096	0.00521621	0.00715305	0.771

Correlation Problems of Projections

- Dimensions 7 and 8 of the Halton sequence



Scrambling Permutations by Faure

- Scrambled radical inverse

$$i = \sum_{j=0}^{\infty} a_j(i) b^j \mapsto \sum_{j=0}^{\infty} \sigma_b(a_j(i)) b^{-j-1},$$

  using permutations σ_b by Faure

$$\begin{aligned}
\sigma_2 &= (0,1) \\
\sigma_3 &= (0,1,2) \\
\sigma_4 &= (0,2,1,3) \\
\sigma_5 &= (0,3,2,1,4) \\
\sigma_6 &= (0,2,4,1,3,5) \\
\sigma_7 &= (0,2,5,3,1,4,6) \\
\sigma_8 &= (0,4,2,6,1,5,3,7) \\
&\vdots
\end{aligned}$$

- Construction rule
  - b is even: Take $2\sigma_{\frac{b}{2}}$ and append $2\sigma_{\frac{b}{2}} + 1$
  - b is odd: Take σ_{b-1}, increment each value $\geq \frac{b-1}{2}$ and insert $\frac{b-1}{2}$ in the middle

Scrambled Halton Sequence and Hammersley Points

- Scrambled Halton sequence

$$x_i := \left(\Phi_{b_1}(i, \sigma_{b_1}), \ldots, \Phi_{b_s}(i, \sigma_{b_s})\right)$$

- Scrambled Hammersley point set

$$x_i := \left(\frac{i}{N}, \Phi_{b_1}(i, \sigma_{b_1}), \ldots, \Phi_{b_{s-1}}(i, \sigma_{b_{s-1}})\right)$$

Scrambled Halton Sequence and Hammersley Points

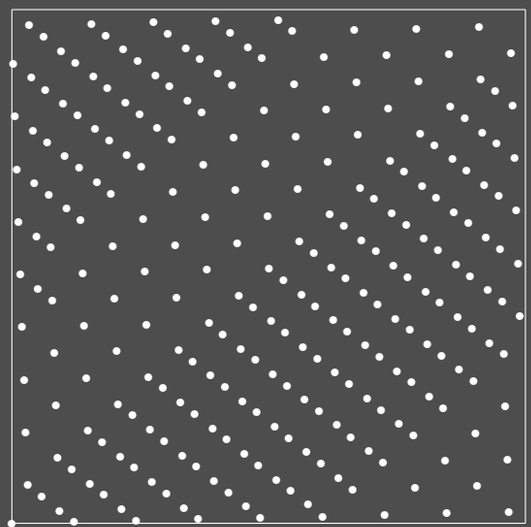
- Scrambled Halton sequence

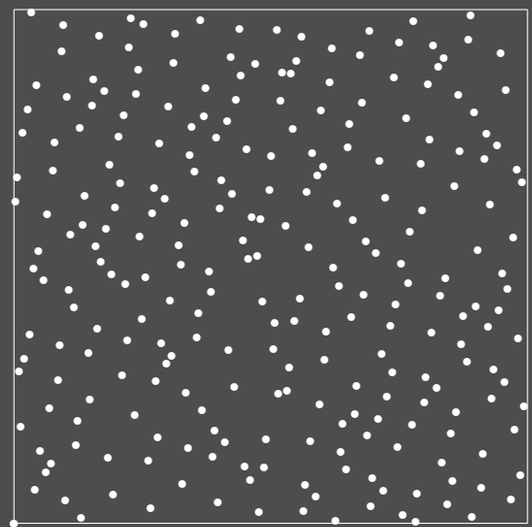
$$x_i := \left(\Phi_{b_1}(i, \sigma_{b_1}), \ldots, \Phi_{b_s}(i, \sigma_{b_s})\right)$$

- Scrambled Hammersley point set

$$x_i := \left(\frac{i}{N}, \Phi_{b_1}(i, \sigma_{b_1}), \ldots, \Phi_{b_{s-1}}(i, \sigma_{b_{s-1}})\right)$$

- Improvement by scrambling (scrambled Halton sequence dimensions 7 and 8)





(t, m, s)-*Nets in Base* b

- Elementary interval

$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq I^s \text{ for integers } l_j \geq 0 \text{ and } 0 \leq a_j < b^{l_j}$$

- Consequently its volume is

$$\lambda_s(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = \frac{1}{b^{\sum_{j=1}^{s} l_j}}$$

(t, m, s)-Nets in Base b

- Elementary interval

$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq I^s \text{ for integers } l_j \geq 0 \text{ and } 0 \leq a_j < b^{l_j}$$

- Consequently its volume is

$$\lambda_s(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = \frac{1}{b^{\sum_{j=1}^{s} l_j}}$$

- **Definition:** For two integers $0 \leq t \leq m$, a finite point set of b^m points in s dimensions is called a (t, m, s)-*net in base* b, if every elementary interval of volume $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.

(t, m, s)-*Nets in Base* b

- Elementary interval

$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq I^s \text{ for integers } l_j \geq 0 \text{ and } 0 \leq a_j < b^{l_j}$$

- Consequently its volume is

$$\lambda_s(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = \frac{1}{b^{\sum_{j=1}^{s} l_j}}$$

- **Definition:** For two integers $0 \leq t \leq m$, a finite point set of b^m points in s dimensions is called a (t, m, s)-*net in base* b, if every elementary interval of volume $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.

- For (t, m, s)-nets in base b we have

$$D^*(P_N) \leq B(s, b) b^t \frac{\log^{s-1} N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-2} N}{N} \right)$$

  - t is the quality parameter

(t, m, s)-*Nets in Base* b

- Elementary interval

$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq I^s \text{ for integers } l_j \geq 0 \text{ and } 0 \leq a_j < b^{l_j}$$

- Consequently its volume is

$$\lambda_s(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = \frac{1}{b^{\sum_{j=1}^{s} l_j}}$$

- **Definition:** For two integers $0 \leq t \leq m$, a finite point set of b^m points in s dimensions is called a (t, m, s)-*net in base* b, if every elementary interval of volume $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.

- For (t, m, s)-nets in base b we have

$$D^*(P_N) \leq B(s, b) b^t \frac{\log^{s-1} N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-2} N}{N} \right)$$

  - t is the quality parameter

- **Note:** So far the concept applies to random and deterministic points

Structure of $(0, m, 2)$-Nets in Base $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points

Structure of $(0, m, 2)$-Nets in Base $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points

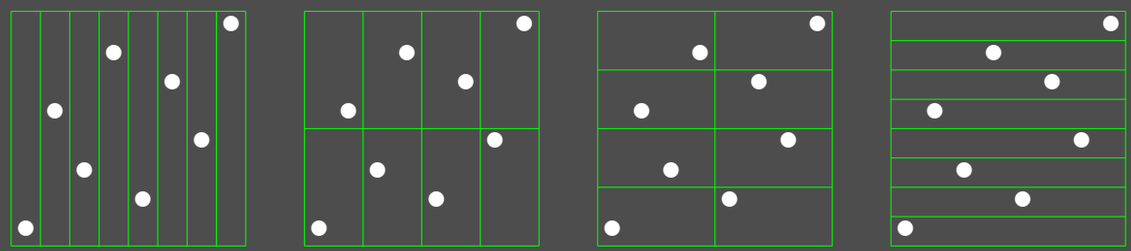
- $(0, m, 2)$-net in base $b = 2$
  - Set P_N of $N = 2^m$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-m} = \frac{1}{N}$ contains exactly 1 point

Structure of $(0, m, 2)$*-Nets in Base* $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points

- $(0, m, 2)$-net in base $b = 2$
  - Set P_N of $N = 2^m$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-m} = \frac{1}{N}$ contains exactly 1 point

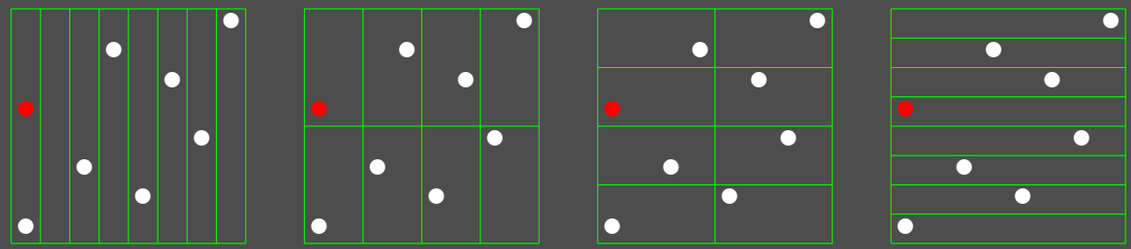
- Example: All elementary volumes of a $(0, 3, 2)$-net in base $b = 2$:



  - more general than stratification and Latin hypercube sampling

Example of a $(1, 3, 2)$*-Net in Base* $b = 2$

- All elementary volumes of a $(0, 3, 2)$-net in base $b = 2$:

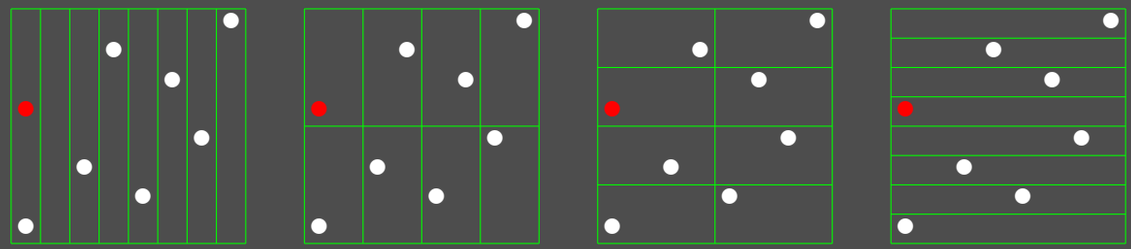


$\lambda_s(E) = b^{t-m} = 2^{0-3} = \frac{1}{8}$ with exactly $b^t = 2^0 = 1$ point

$\Rightarrow$ it cannot be a $(0, 3, 2)$-net !

Example of a $(1, 3, 2)$*-Net in Base* $b = 2$

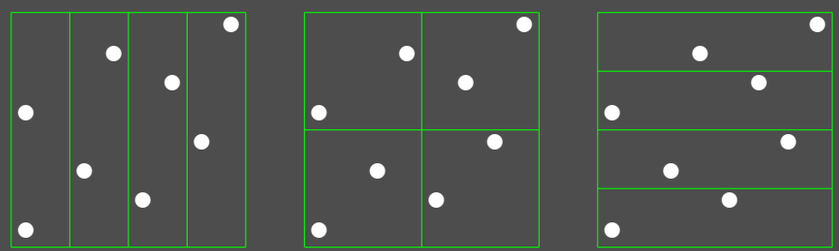
- All elementary volumes of a $(0, 3, 2)$-net in base $b = 2$:



$\lambda_s(E) = b^{t-m} = 2^{0-3} = \frac{1}{8}$ with exactly $b^t = 2^0 = 1$ point

$\Rightarrow$ it cannot be a $(0, 3, 2)$-net !

- All elementary volumes of a $(1, 3, 2)$-net in base $b = 2$:



$\lambda_s(E) = b^{t-m} = 2^{1-3} = \frac{1}{4}$ with exactly $b^t = 2^1 = 2$ points

$\Rightarrow$ it is only a $(1, 3, 2)$-net...

Structure of $(0, 2n, 2)$-Nets in Base $b = 2$

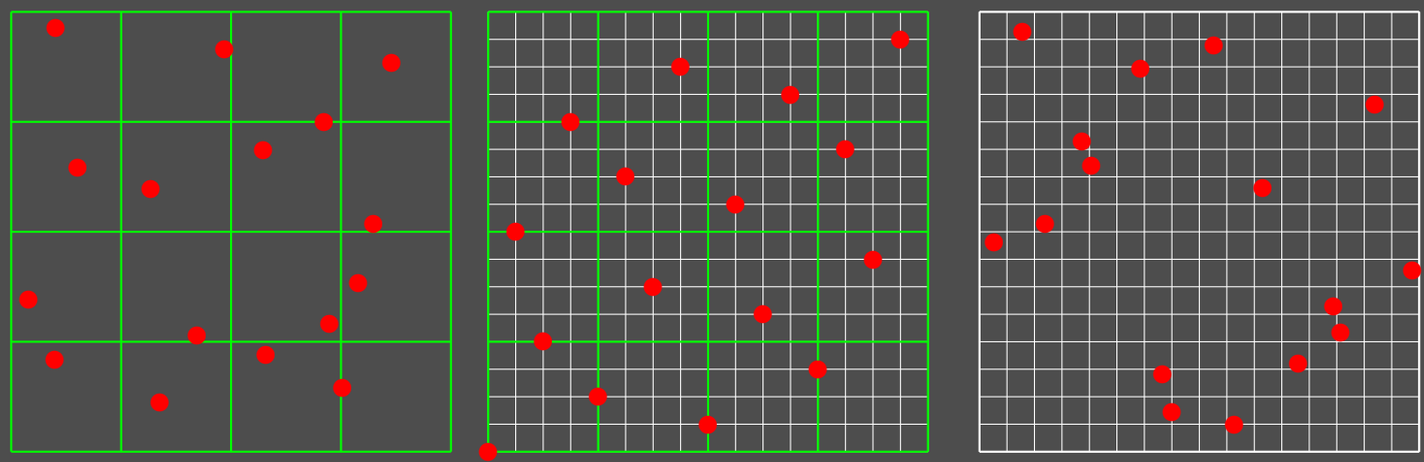
- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points

Structure of $(0, 2n, 2)$*-Nets in Base* $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points
- $(0, 2n, 2)$-net in base $b = 2$
  - Set P_N of $N = (2^n)^2$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-2n} = \frac{1}{N}$ contains exactly 1 point

Structure of $(0, 2n, 2)$*-Nets in Base* $b = 2$

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points
- $(0, 2n, 2)$-net in base $b = 2$
  - Set P_N of $N = (2^n)^2$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-2n} = \frac{1}{N}$ contains exactly 1 point



jittered **and** LHS (N -rooks)

- (t, m, s)-nets: Much more general concept of stratification

(t, s)-*Sequences in Base* b

- **Definition:** For $t \geq 0$, an infinite point sequence is called a (t, s)-*sequence* in base b, if for all $k \geq 0$ and $m \geq t$, the vectors $x_{kb^m+1}, \ldots, x_{(k+1)b^m} \in I^s$ form a (t, m, s)-net.

(t, s)-*Sequences in Base* b

- **Definition:** For $t \geq 0$, an infinite point sequence is called a (t, s)-*sequence* in base b, if for all $k \geq 0$ and $m \geq t$, the vectors $x_{kb^m+1}, \ldots, x_{(k+1)b^m} \in I^s$ form a (t, m, s)-net.

- For (t, s)-sequence in base b we have

$$D^*(P_N) \leq C(s, b) b^t \frac{\log^s N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-1} N}{N}\right)$$

(t, s)-*Sequences in Base* b

- **Definition:** For $t \geq 0$, an infinite point sequence is called a (t, s)-*sequence* in base b, if for all $k \geq 0$ and $m \geq t$, the vectors $x_{kb^m+1}, \ldots, x_{(k+1)b^m} \in I^s$ form a (t, m, s)-net.

- For (t, s)-sequence in base b we have

$$D^*(P_N) \leq C(s, b)b^t \frac{\log^s N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-1} N}{N}\right)$$

- Adding the component $\frac{i}{N} = \frac{i}{b^m}$ to a (t, s)-sequence yields a $(t, m, s + 1)$-net
- $(0, s)$-sequences can only exist for $b \geq s$

(t, s)-*Sequences in Base* b

- **Definition:** For $t \geq 0$, an infinite point sequence is called a (t, s)-*sequence* in base b, if for all $k \geq 0$ and $m \geq t$, the vectors $x_{kb^m+1}, \ldots, x_{(k+1)b^m} \in I^s$ form a (t, m, s)-net.

- For (t, s)-sequence in base b we have

$$D^*(P_N) \leq C(s, b) b^t \frac{\log^s N}{N} + \mathcal{O}\left(b^t \frac{\log^{s-1} N}{N}\right)$$

- Adding the component $\frac{i}{N} = \frac{i}{b^m}$ to a (t, s)-sequence yields a $(t, m, s + 1)$-net
- $(0, s)$-sequences can only exist for $b \geq s$

- Examples
  - Van der Corput sequences are $(0, 1)$-sequences in base b
  - adding the component $\frac{i}{N}$ with $N = b^m$ yields a $(0, m, 2)$-net
    - e.g. Hammersley point set for $s = 2$ and $N = 2^m$ points
    - many applications in finance and particle transport problems

Digital (t, m, s)-Nets and (t, s)-Sequences

- Fixed-point numbers with M digits in base b

$$[0, 1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0, 1)$$

Digital (t, m, s)-Nets and (t, s)-Sequences

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} \quad = \quad \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k}$$

Digital (t, m, s)-Nets and (t, s)-Sequences

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} \quad = \quad \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)} \in [0,1)_{b,M}$$

Digital (t, m, s)*-Nets and* (t, s)*-Sequences*

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)} \in [0,1)_{b,M} \text{ where}$$

$$a_{i,k}^{(j)} := \eta_k^{(j)} \left(\sum_{l=0}^{M-1} c_{k,l}^{(j)} \cdot \psi_l\left(d_{i,l}\right) \right)$$

for $1 \leq j \leq s$ and

$$i =: \sum_{l=0}^{M-1} d_{i,l} \cdot b^l \qquad d_{i,l} \in \mathbb{Z}_b := \{0, \ldots, b-1\}$$

Digital (t, m, s)*-Nets and* (t, s)*-Sequences*

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)} \in [0,1)_{b,M} \text{ where}$$

$$a_{i,k}^{(j)} := \eta_k^{(j)} \left(\sum_{l=0}^{M-1} c_{k,l}^{(j)} \cdot \psi_l\left(d_{i,l}\right) \right)$$

for $1 \leq j \leq s$ and

$$i =: \sum_{l=0}^{M-1} d_{i,l} \cdot b^l \qquad d_{i,l} \in \mathbb{Z}_b := \{0, \ldots, b-1\}$$

- Arithmetic in commutative ring $(R, +, \cdot)$ with $|R| = b$ elements
- Bijections $\eta_k^{(j)} : R \to \mathbb{Z}_b$ and $\psi_l : \mathbb{Z}_b \to R$

Digital (t, m, s)*-Nets and* (t, s)*-Sequences*

- Fixed-point numbers with M digits in base b

$$[0,1)_{b,M} := \left\{ kb^{-M} \mid k = 0, \ldots, b^M - 1 \right\} \subset [0,1)$$

- Components $A_i^{(j)}$ of a point set $A = \{A_0, \ldots, A_{N-1}\}$

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)} \in [0,1)_{b,M} \text{ where}$$

$$a_{i,k}^{(j)} := \eta_k^{(j)} \left(\sum_{l=0}^{M-1} c_{k,l}^{(j)} \cdot \psi_l\left(d_{i,l}\right) \right)$$

for $1 \leq j \leq s$ and

$$i =: \sum_{l=0}^{M-1} d_{i,l} \cdot b^l \qquad d_{i,l} \in \mathbb{Z}_b := \{0, \ldots, b-1\}$$

- Arithmetic in commutative ring $(R, +, \cdot)$ with $|R| = b$ elements
- Bijections $\eta_k^{(j)} : R \to \mathbb{Z}_b$ and $\psi_l : \mathbb{Z}_b \to R$

$\Rightarrow$ If now A is a (t, m, s)-net, it is called a ***digital*** (t, m, s)***-net***

$\Rightarrow$ If now A is a (t, s)-sequence, it is called a ***digital*** (t, s)***-sequence***

Deterministic Constructions of Digital Point Sets

- Generator matrix

$$C^{(j)} := \left(c_{k,l}^{(j)} \right)_{k=1,l=0}^{M,M-1} \in R^{M \times M}$$

- van der Corput, Sobol', Faure, Niederreiter, and Niederreiter-Xing
  - increased quality by decreased parameter t
  - difficult computation of the generator matrices

Deterministic Constructions of Digital Point Sets

- Generator matrix

$$C^{(j)} := \left(c_{k,l}^{(j)} \right)_{k=1,l=0}^{M,M-1} \in R^{M \times M}$$

- van der Corput, Sobol', Faure, Niederreiter, and Niederreiter-Xing
  - increased quality by decreased parameter t
  - difficult computation of the generator matrices
- Fast evaluation by
  - Gray codes
  - vectorization
  - buffering of invariants
  - rings implemented as lookup tables
- Very often

$$\mathbf{a}_i^{(j)} = C^{(j)} \mathbf{d}_i$$

Vectorization Example for Base $b = 2$

- Ring $R = (\{0, 1\}, +, \cdot) = \mathbb{Z}_2$ by bit vector operations
- One component at M bits precision

$$x_i = \left(\frac{1}{2} \cdots \frac{1}{2^M}\right) \cdot C \cdot \begin{pmatrix} d_0(i) \\ \vdots \\ d_{M-1}(i) \end{pmatrix} \qquad \text{where } i = \sum_{k=0}^{m-1} d_k(i) 2^k$$

Vectorization Example for Base $b = 2$

- Ring $R = (\{0, 1\}, +, \cdot) = \mathbb{Z}_2$ by bit vector operations
- One component at M bits precision

$$x_i = \left(\frac{1}{2} \cdots \frac{1}{2^M}\right) \cdot C \cdot \begin{pmatrix} d_0(i) \\ \vdots \\ d_{M-1}(i) \end{pmatrix} \qquad \text{where } i = \sum_{k=0}^{m-1} d_k(i) 2^k$$

- Basic vectorized algorithm

```
double x(int i)
{
  for(int y = 0, int k = 0; i; i /= 2, k++)
    if(i & 1)
      y ^= C[k];

  return (double) y / (double) (1 << (M + 1));
}
```

Examples Matrices for Base $b = 2$

- $(0, m, 1)$-nets at $N = 2^m$

$$C_1 = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ & & \ddots & & \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

implements $x = \frac{i}{N}$

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Bit reversal, or $\phi_2(i)$ by van der Corput

  $$C_2 = I$$

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Bit reversal, or $\phi_2(i)$ by van der Corput

$$C_2 = I$$

- Algorithm

```
double RadicalInverse(unsigned int bits) // M=32 bits version
{
  bits = ( bits               <<  16) | ( bits               >>  16);
  bits = ((bits & 0x00ff00ff) <<   8) | ((bits & 0xff00ff00) >>   8);
  bits = ((bits & 0x0f0f0f0f) <<   4) | ((bits & 0xf0f0f0f0) >>   4);
  bits = ((bits & 0x33333333) <<   2) | ((bits & 0xcccccccc) >>   2);
  bits = ((bits & 0x55555555) <<   1) | ((bits & 0xaaaaaaaa) >>   1);

  return (double) bits / (double) 0x100000000L;
}
```

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Sobol' scrambled radical inverse

$$C_3 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ 1 & 1 & 1 & \ddots & 0 & 0 \end{pmatrix} = \binom{k-1}{l-1} \bmod 2$$

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Sobol' scrambled radical inverse

$$C_3 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ 1 & 1 & 1 & \ddots & 0 & 0 \end{pmatrix} = \binom{k-1}{l-1} \bmod 2$$

- Algorithm

```
double SobolRadicalInverse(int i)
{
  int r, v;

  v = 1 << M;

  for(r = 0; i; i >>= 1)
  {
    if(i & 1)
      r ^= v;

    v ^= v >> 1;
  }

  return (double) r / (double) (1 << (M + 1));
}
```

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Larcher-Pillichshammer scrambled radical inverse

$$C_4 = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ & & \ddots & & \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix}$$

Examples Matrices for Base $b = 2$

- $(0, 1)$-sequences: Larcher-Pillichshammer scrambled radical inverse

$$C_4 = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ & & \ddots & & \\ 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix}$$

- Algorithm

```
double LarcherPillichshammerRadicalInverse(int i)
{
  int r, v;

  v = 1 << M;

  for(r = 0; i; i >>= 1)
  {
    if(i & 1)
      r ^= v;

    v |= v >> 1;
  }

  return (double) r / (double) (1 << (M + 1));
}
```

Digital $(0, m, s)$***-Nets and*** $(0, s)$***-Sequences in Base*** $b = 2$

- $(0, m, 2)$-nets at $N = 2^m$
  - Hammersley points (worst constant)

    (C_1, C_2)
  - Larcher-Pillichshammer points (best constant)

    (C_1, C_4)

Digital $(0, m, s)$*-Nets and* $(0, s)$*-Sequences in Base* $b = 2$

- $(0, m, 2)$-nets at $N = 2^m$
  - Hammersley points (worst constant)

    (C_1, C_2)
  - Larcher-Pillichshammer points (best constant)

    (C_1, C_4)
- $(0, 2)$-sequence: Sobol' LP_0-sequence

  (C_2, C_3)

Digital $(0, m, s)$*-Nets and* $(0, s)$*-Sequences in Base* $b = 2$

- $(0, m, 2)$-nets at $N = 2^m$
  - Hammersley points (worst constant)

    (C_1, C_2)
  - Larcher-Pillichshammer points (best constant)

    (C_1, C_4)
- $(0, 2)$-sequence: Sobol' LP_0-sequence

  (C_2, C_3)
- $(0, m, 3)$-net at $N = 2^m$: Sobol' LP_0-net

  (C_1, C_2, C_3)
- Very useful in particle transport, especially computer graphics

Software

- **http://www.uni-kl.de/AG-Heinrich/SamplePack.html**
  - Sobol’ sequence
  - Niederreiter sequence
  - Niederreiter-Xing sequence
- **http://www.dismat.oeaw.ac.at/pirs/niedxing.html**
  - generator matrices for the Niederreiter-Xing sequence
- **http://www.multires.caltech.edu/software/libseq/index.html**
  - general package
  - several sequences (Halton, Niederreiter, ...)
- **Numerical Recipes**
  - Sobol’ sequence

Good Lattice Points: Rank-1 Lattices

- **Definition:** A discrete subset

$$L := P_N + \mathbb{Z}^s \subset \mathbb{R}^s$$

that is closed under addition and subtraction is called a **lattice**.

Good Lattice Points: Rank-1 Lattices

- **Definition:** A discrete subset

  $$L := P_N + \mathbb{Z}^s \subset \mathbb{R}^s$$

  that is closed under addition and subtraction is called a **lattice**.

- Rank-1 lattice

  $$\mathbf{x}_i := \frac{i}{N}\mathbf{g}$$

  by suitable generating vector $\mathbf{g} \in \mathbb{N}^s$

- Low discrepancy constructions
  - Fibonacci lattices for $s = 2$
  - lattices with generator vector of Korobov-form $\mathbf{g} = (1, l, l^2, \ldots)$

Good Lattice Points: Rank-1 Lattices

- **Definition:** A discrete subset

  $$L := P_N + \mathbb{Z}^s \subset \mathbb{R}^s$$

  that is closed under addition and subtraction is called a **lattice**.

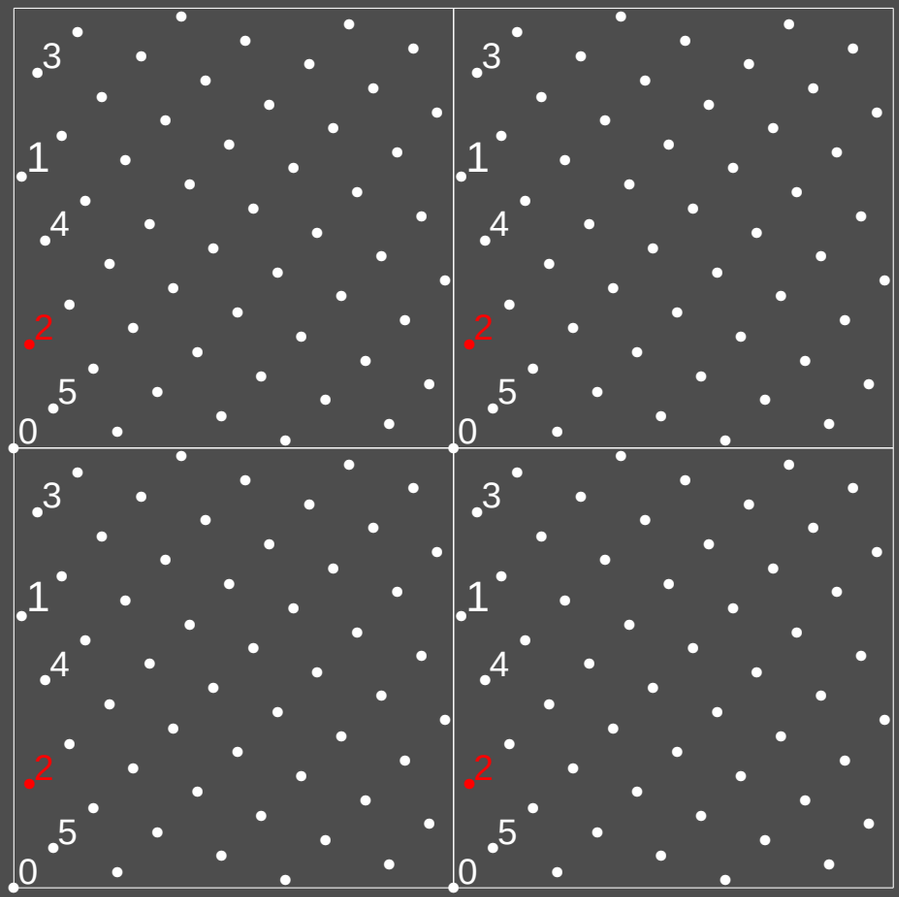
- Rank-1 lattice

  $$\mathbf{x}_i := \frac{i}{N}\mathbf{g}$$

  by suitable generating vector $\mathbf{g} \in \mathbb{N}^s$

- Low discrepancy constructions
  - Fibonacci lattices for $s = 2$
  - lattices with generator vector of Korobov-form $\mathbf{g} = (1, l, l^2, \ldots)$
- No explicit construction - only tables

- One-periodic pattern $L \cap [0,1)^s$



- Low discrepancy
- Much better discrepancy than regular grids

Example: Fibonacci Rank-1 *Lattice*

- Fibonacci numbers: $F_1 = F_2 = 1$, $F_k = F_{k-1} + F_{k-2}$ for $k > 2$
- **Fibonacci lattice** by generator vector $\mathrm{g} = (1, F_{k-1})$ at $N = F_k$ points

$$\mathbf{x}_i := \frac{i}{F_k}(1, F_{k-1})$$

  - Low discrepancy

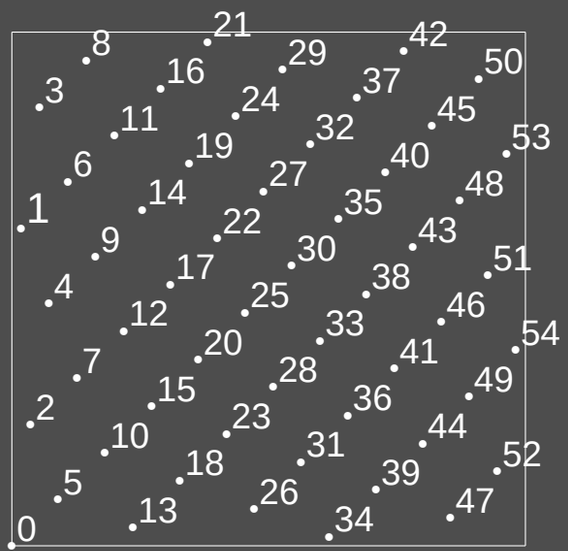
*Example: Fibonacci Rank-*1 *Lattice*

- Fibonacci numbers: $F_1 = F_2 = 1$, $F_k = F_{k-1} + F_{k-2}$ for $k > 2$
- **Fibonacci lattice** by generator vector $\mathrm{g} = (1, F_{k-1})$ at $N = F_k$ points

$$\mathbf{x}_i := \frac{i}{F_k}(1, F_{k-1})$$

  - Low discrepancy

- Example: $N = F_{10} = 55$, $\mathbf{x}_i := \frac{i}{55}(1, 34)$



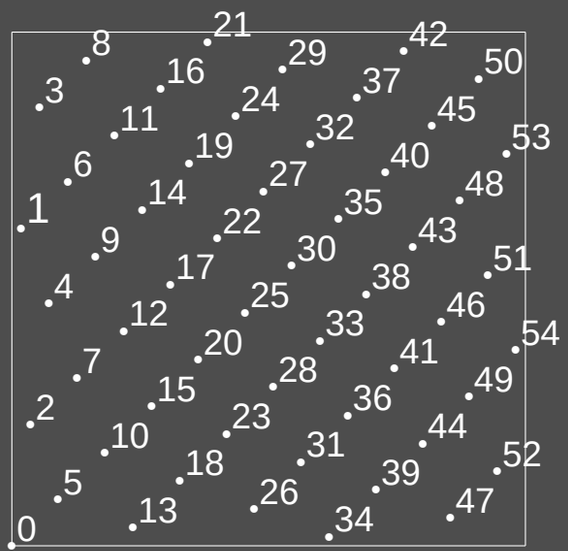
*Example: Fibonacci Rank-*1 *Lattice*

- Fibonacci numbers: $F_1 = F_2 = 1$, $F_k = F_{k-1} + F_{k-2}$ for $k > 2$
- **Fibonacci lattice** by generator vector $\mathrm{g} = (1, F_{k-1})$ at $N = F_k$ points

$$\mathbf{x}_i := \frac{i}{F_k}(1, F_{k-1})$$

  - Low discrepancy

- Example: $N = F_{10} = 55$, $\mathbf{x}_i := \frac{i}{55}(1, 34)$



- **Note:** N grows exponentially for Fibonacci lattices

Lattice Sequences

- Rank-1 lattice

$$\mathbf{x}_i = \frac{i}{N} \cdot \mathbf{g}$$

- Hide N by choosing $N = b^m$ and

$$\mathbf{x}_i = \phi_b(i) \cdot \mathbf{g}$$

Lattice Sequences

- Rank-1 lattice

$$\mathbf{x}_i = \frac{i}{N} \cdot \mathbf{g}$$

- Hide N by choosing $N = b^m$ and

$$\mathbf{x}_i = \phi_b(i) \cdot \mathbf{g}$$

- Similar to (t, s)-sequences: $\mathbf{x}_{kb^m}, \ldots, \mathbf{x}_{(k+1)b^m-1}$ form a shifted lattice

Lattice Sequences

- Rank-1 lattice

$$\mathbf{x}_i = \frac{i}{N} \cdot \mathbf{g}$$

- Hide N by choosing $N = b^m$ and

$$\mathbf{x}_i = \phi_b(i) \cdot \mathbf{g}$$

- Similar to (t, s)-sequences: $\mathbf{x}_{kb^m}, \ldots, \mathbf{x}_{(k+1)b^m-1}$ form a shifted lattice

- Shift Δ in the $k+1$st run for $N = b^m$

$$\begin{aligned} \phi_b(i + kb^m) \cdot \mathbf{g} &= (\phi_b(i) + \phi_b(kb^m)) \cdot \mathbf{g} \\ &= \phi_b(i) \cdot \mathbf{g} + \underbrace{\phi_b(k) b^{-m-1} \mathbf{g}}_{=:\Delta} \end{aligned}$$

Summary

- Quasi-Monte Carlo Points
  - low discrepancy
  - deterministic
  - intrinsic stratification (Latin hypercube, symmetrized, regularized, antithetic)
    - no extra programming

Summary

- Quasi-Monte Carlo Points
  - low discrepancy
  - deterministic
  - intrinsic stratification (Latin hypercube, symmetrized, regularized, antithetic)
    - no extra programming
  - no completely uniform distribution due to correlation

Monte Carlo and Beyond

- Principles of rendering algorithms
- Monte Carlo integration
- Quasi-Monte Carlo points
- **Quasi-Monte Carlo integration**
  - Koksma-Hlawka inequality and variation in the sense of Hardy and Krause
  - Discrete density approximation
  - Error control
  - Transferring Monte Carlo techniques to quasi-Monte Carlo
  - Integrands of infinite variation
  - Discrete Fourier transform on good lattice points
- Monte Carlo extensions of quasi-Monte Carlo
- Application to computer graphics

Quasi-Monte Carlo Integration

- Numerical integration by **Quasi-Monte Carlo points**

$$\left| \int_{I^s} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| \leq V(f) D^*(P_N)$$

with variation $V(f)$ in the sense of Hardy and Krause and star-discrepancy

$$D^*(P_N) := \sup_{A=\prod_{j=1}^{s}[0,a_j)\subseteq I^s} \left| \underbrace{\int_{I^s} \chi_A(x)dx}_{=\lambda_s(A)} - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

Quasi-Monte Carlo Integration

- Numerical integration by **Quasi-Monte Carlo points**

$$\left| \int_{I^s} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| \leq V(f) D^*(P_N)$$

with variation $V(f)$ in the sense of Hardy and Krause and star-discrepancy

$$D^*(P_N) := \sup_{A=\prod_{j=1}^{s}[0,a_j)\subseteq I^s} \left| \underbrace{\int_{I^s} \chi_A(x)dx}_{=\lambda_s(A)} - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right|$$

- Deterministic error bound by the Koksma-Hlawka inequality

- Independent of dimension by using quasi-Monte Carlo points
  - roughly quadratically faster as compared to random sampling

Theorem: The Koksma-Hlawka Inequality

$$\left| \int_I f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| \leq V(f) D^*(P_N)$$

- Proof for $s = 1$: Decompose

  $$f(x) = f(1) - \int_x^1 f'(u)du = f(1) - \int_I \chi_{[0,u]}(x) f'(u)du$$

  and define

  $$V(f) := \int_I \left| \frac{\partial f(u)}{\partial u} \right| du$$

- **Note:**

  $$\chi_{[0,u]}(x) = \begin{cases} 1 & x \in [0,u) \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & x < u \\ 0 & \text{else} \end{cases}$$

Theorem: The Koksma-Hlawka Inequality

$$\left| \int_I f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| \leq V(f) D^*(P_N)$$

- Proof for $s = 1$: Decompose

  $$f(x) = f(1) - \int_x^1 f'(u)du = f(1) - \int_I \chi_{[0,u]}(x) f'(u)du$$

  and define

  $$V(f) := \int_I \left| \frac{\partial f(u)}{\partial u} \right| du$$

- **Note:**

  $$\chi_{[0,u]}(x) = \begin{cases} 1 & x \in [0,u) \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & x < u \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & u > x \\ 0 & \text{else} \end{cases}$$

$$\left| \int_I f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right|$$

$$= \left| \int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u) du dx - \frac{1}{N} \sum_{i=0}^{N-1} \left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u) du \right) \right|$$

$$
\begin{aligned}
&\left| \int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \right| \\
&= \left| \int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u)dudx - \frac{1}{N}\sum_{i=0}^{N-1} \left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u)du \right) \right| \\
&= \left| f(1) - \int_I \int_I \chi_{[0,u]}(x) f'(u)dudx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u)du \right|
\end{aligned}
$$

$$
\begin{aligned}
&\left| \int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \right| \\
&= \left| \int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u)dudx - \frac{1}{N}\sum_{i=0}^{N-1} \left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u)du \right) \right| \\
&= \left| f(1) - \int_I \int_I \chi_{[0,u]}(x) f'(u)dudx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u)du \right| \\
&= \left| \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u)du - \int_I \int_I \chi_{[0,u]}(x)dx f'(u)du \right|
\end{aligned}
$$

$$
\begin{aligned}
&\left| \int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \right| \\
&= \left| \int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u) du dx - \frac{1}{N}\sum_{i=0}^{N-1} \left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u) du \right) \right| \\
&= \left| f(1) - \int_I \int_I \chi_{[0,u]}(x) f'(u) du dx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u) du \right| \\
&= \left| \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u) du - \int_I \int_I \chi_{[0,u]}(x) dx f'(u) du \right| \\
&= \left| \int_I f'(u) \left[\frac{1}{N}\sum_{i=0}^{N-1} \chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x) dx \right] du \right|
\end{aligned}
$$

$$
\begin{aligned}
&\left| \int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \right| \\
&= \left| \int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u) du dx - \frac{1}{N}\sum_{i=0}^{N-1} \left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u) du \right) \right| \\
&= \left| f(1) - \int_I \int_I \chi_{[0,u]}(x) f'(u) du dx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u) du \right| \\
&= \left| \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u) du - \int_I \int_I \chi_{[0,u]}(x) dx f'(u) du \right| \\
&= \left| \int_I f'(u) \left[\frac{1}{N}\sum_{i=0}^{N-1} \chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x) dx \right] du \right| \\
&\leq \int_I \left| f'(u) \right| \left| \frac{1}{N}\sum_{i=0}^{N-1} \chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x) dx \right| du
\end{aligned}
$$

$$
\begin{aligned}
&\left|\int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| \\
&= \left|\int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u)dudx - \frac{1}{N}\sum_{i=0}^{N-1}\left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u)du\right)\right| \\
&= \left|f(1) - \int_I\int_I \chi_{[0,u]}(x) f'(u)dudx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1}\int_I \chi_{[0,u]}(x_i) f'(u)du\right| \\
&= \left|\frac{1}{N}\sum_{i=0}^{N-1}\int_I \chi_{[0,u]}(x_i) f'(u)du - \int_I\int_I \chi_{[0,u]}(x)dx f'(u)du\right| \\
&= \left|\int_I f'(u)\left[\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right]du\right| \\
&\leq \int_I \left|f'(u)\right|\left|\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right|du \\
&\leq \int_I \left|f'(u)\right|du \cdot \sup_{u\in I}\left|\frac{1}{N}\sum_{i=0}^{N-1}\chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x)dx\right|
\end{aligned}
$$

$$
\begin{aligned}
&\left| \int_I f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \right| \\
&= \left| \int_I f(1) - \int_I \chi_{[0,u]}(x) f'(u) du dx - \frac{1}{N}\sum_{i=0}^{N-1} \left(f(1) - \int_I \chi_{[0,u]}(x_i) f'(u) du \right) \right| \\
&= \left| f(1) - \int_I \int_I \chi_{[0,u]}(x) f'(u) du dx - f(1) + \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u) du \right| \\
&= \left| \frac{1}{N}\sum_{i=0}^{N-1} \int_I \chi_{[0,u]}(x_i) f'(u) du - \int_I \int_I \chi_{[0,u]}(x) dx f'(u) du \right| \\
&= \left| \int_I f'(u) \left[\frac{1}{N}\sum_{i=0}^{N-1} \chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x) dx \right] du \right| \\
&\leq \int_I \left| f'(u) \right| \left| \frac{1}{N}\sum_{i=0}^{N-1} \chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x) dx \right| du \\
&\leq \int_I \left| f'(u) \right| du \cdot \sup_{u \in I} \left| \frac{1}{N}\sum_{i=0}^{N-1} \chi_{[0,u]}(x_i) - \int_I \chi_{[0,u]}(x) dx \right| \\
&= V(f) D^*(P_N) \qquad \textit{q.e.d.}
\end{aligned}
$$

Variation in the Sense of Vitali

- Difference operator for intervals of the form $A = \prod_{i=1}^{s} [a_i, b_i) \subseteq I^s$

$$\Delta(f, A) := \sum_{j_1=0}^{1} \cdots \sum_{j_s=0}^{1} (-1)^{\sum_{k=1}^{s} j_k} f(j_1 a_1 + (1 - j_1) b_1, \ldots, j_s a_s + (1 - j_s) b_s)$$

Variation in the Sense of Vitali

- Difference operator for intervals of the form $A = \prod_{i=1}^{s} [a_i, b_i) \subseteq I^s$

$$\Delta(f, A) := \sum_{j_1=0}^{1} \cdots \sum_{j_s=0}^{1} (-1)^{\sum_{k=1}^{s} j_k} f(j_1 a_1 + (1-j_1) b_1, \ldots, j_s a_s + (1-j_s) b_s)$$

- Variation in the sense of Vitali

$$V^{(s)}(f) := \sup_{\mathcal{P}} \sum_{A \in \mathcal{P}} |\Delta(f, A)|$$

where $\mathcal{P}$ is the set of partitions of I^s into subintervals A as above

Variation in the Sense of Vitali

- Difference operator for intervals of the form $A = \prod_{i=1}^{s} [a_i, b_i) \subseteq I^s$

$$\Delta(f, A) := \sum_{j_1=0}^{1} \cdots \sum_{j_s=0}^{1} (-1)^{\sum_{k=1}^{s} j_k} f(j_1 a_1 + (1-j_1) b_1, \ldots, j_s a_s + (1-j_s) b_s)$$

- Variation in the sense of Vitali

$$V^{(s)}(f) := \sup_{\mathcal{P}} \sum_{A \in \mathcal{P}} |\Delta(f, A)|$$

  where $\mathcal{P}$ is the set of partitions of I^s into subintervals A as above

- If f has a continuous derivative

$$V^{(s)}(f) = \int_{I^s} \left| \frac{\partial^s f(u_1, \ldots, u_s)}{\partial u_1 \cdots \partial u_s} \right| du$$

Variation in the Sense of Vitali

- Difference operator for intervals of the form $A = \prod_{i=1}^{s} [a_i, b_i) \subseteq I^s$

$$\Delta(f, A) := \sum_{j_1=0}^{1} \cdots \sum_{j_s=0}^{1} (-1)^{\sum_{k=1}^{s} j_k} f(j_1 a_1 + (1-j_1) b_1, \ldots, j_s a_s + (1-j_s) b_s)$$

- Variation in the sense of Vitali

$$V^{(s)}(f) := \sup_{\mathcal{P}} \sum_{A \in \mathcal{P}} |\Delta(f, A)|$$

  where $\mathcal{P}$ is the set of partitions of I^s into subintervals A as above

- If f has a continuous derivative

$$V^{(s)}(f) = \int_{I^s} \left| \frac{\partial^s f(u_1, \ldots, u_s)}{\partial u_1 \cdots \partial u_s} \right| du$$

- Problem if f constant in only some of the variables $u_1, \ldots, u_s$

$$\Rightarrow \Delta(f, A) = 0 \qquad \Rightarrow V^{(s)}(f) = 0$$

Variation in the Sense of Hardy and Krause

- Restrict variation in the sense of Vitali

  $$V^{(k)}(f; i_1, \ldots, i_k)$$

  to the k-dimensional face $\{(u_1, \ldots, u_s) \in [0,1]^s | u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$

Variation in the Sense of Hardy and Krause

- Restrict variation in the sense of Vitali

  $$V^{(k)}(f; i_1, \ldots, i_k)$$

  to the k-dimensional face $\{(u_1, \ldots, u_s) \in [0,1]^s | u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$

- Variation in the sense of Hardy and Krause

  $$V(f) := \sum_{k=1}^{s} \sum_{1 \leq i_1 < \cdots < i_k \leq s} V^{(k)}(f; i_1, \ldots, i_k)$$

Variation in the Sense of Hardy and Krause

- Restrict variation in the sense of Vitali

  $$V^{(k)}(f; i_1, \ldots, i_k)$$

  to the k-dimensional face $\{(u_1, \ldots, u_s) \in [0,1]^s | u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$

- Variation in the sense of Hardy and Krause

  $$V(f) := \sum_{k=1}^{s} \sum_{1 \leq i_1 < \cdots < i_k \leq s} V^{(k)}(f; i_1, \ldots, i_k)$$

- **Definition:**

  f is of bounded variation in the sense of Hardy and Krause, if $V(f)$ is finite.

Variation in the Sense of Hardy and Krause

- Restrict variation in the sense of Vitali

  $$V^{(k)}(f; i_1, \ldots, i_k)$$

  to the k-dimensional face $\{(u_1, \ldots, u_s) \in [0,1]^s | u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$

- Variation in the sense of Hardy and Krause

  $$V(f) := \sum_{k=1}^{s} \sum_{1 \leq i_1 < \cdots < i_k \leq s} V^{(k)}(f; i_1, \ldots, i_k)$$

- **Definition:**

  f is of bounded variation in the sense of Hardy and Krause, if $V(f)$ is finite.

- Estimating the variation in the sense of Hardy and Krause
  - use regular grid at $N = n^s$ samples
  - compute difference operator $\triangle$ on the grid
  - sum up the approximations of the single Vitali variations
  - $n \to \infty$

Variation Reduction

- Transfer Monte Carlo variance reduction techniques to quasi-Monte Carlo
  - separation of the main part
  - multilevel method of dependent tests
  - importance sampling
  - replication heuristics (presmoothing the integrand)

Variation Reduction

- Transfer Monte Carlo variance reduction techniques to quasi-Monte Carlo
  - separation of the main part
  - multilevel method of dependent tests
  - importance sampling
  - replication heuristics (presmoothing the integrand)

- Quasi-Monte Carlo importance sampling

$$\left| \int_{I^s} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(y_i)}{p(y_i)} \right| \leq V\left(\frac{f}{p}\right) D^*(P_N)$$

  where $y_i \sim p$ by the multidimensional inversion method

  - Similar to the Monte Carlo case, the variation is not changed

  - For low discrepancy points P_N quadratically faster than random sampling

Approximating Continuous by Discrete Measures

- Often integrands of the form $f = gp$
  - p can be modeled using the multidimensional inversion method
  - g is hard to handle (e.g. discontinuous, expensive)

Approximating Continuous by Discrete Measures

- Often integrands of the form $f = gp$
  - p can be modeled using the multidimensional inversion method
  - g is hard to handle (e.g. discontinuous, expensive)
- Avoid weighting by small probabilities

$$\int_{I^s} f(x)dx = \int_{I^s} g(x)p(x)dx = \int_{I^s} g(y)dP(y)$$

- Approximate measure P by discrete measure

$$P_N := \frac{1}{N}\sum_{i=0}^{N-1} \delta_{y_i}$$

  modeled by $y_i = P^{-1}(x_i)$ from $x_i \sim \mathcal{U}$

- Then

$$\int_{I^s} g(y)dP(y) \approx \int_{I^s} g(y)dP_N(y) := \frac{1}{N}\sum_{i=0}^{N-1} g(y_i)$$

Discrepancy Bounds for Transformed Points

- **Definition:** The **discrepancy with respect to the density** p is

$$D^*(p, C_N) := \sup_{A \in \mathcal{J}^*} \left| \int_{I^s} \chi_A(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(y_i) \right|$$

where $C_N = \{y_0, \ldots, y_{N-1}\}$

Discrepancy Bounds for Transformed Points

- **Definition:** The **discrepancy with respect to the density** p is

$$D^*(p, C_N) := \sup_{A \in \mathcal{J}^*} \left| \int_{I^s} \chi_A(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(y_i) \right|$$

  where $C_N = \{y_0, \ldots, y_{N-1}\}$

- Multidimensional inversion method: If p is separable, i.e. $p(x) = \prod_{j=1}^{s} p^{(j)}(x^{(j)})$

$$D^*(p, C_N) = D^*(P_N)$$

  otherwise

$$D^*(p, C_N) \leq c \left(D^*(P_N)\right)^{\frac{1}{s}} \qquad c \in \mathbb{R}^+$$

Discrepancy Bounds for Transformed Points

- **Definition:** The **discrepancy with respect to the density** p is

$$D^*(p, C_N) := \sup_{A \in \mathcal{J}^*} \left| \int_{I^s} \chi_A(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(y_i) \right|$$

  where $C_N = \{y_0, \ldots, y_{N-1}\}$

- Multidimensional inversion method: If p is separable, i.e. $p(x) = \prod_{j=1}^{s} p^{(j)}(x^{(j)})$

$$D^*(p, C_N) = D^*(P_N)$$

  otherwise

$$D^*(p, C_N) \leq c \, (D^*(P_N))^{\frac{1}{s}} \qquad c \in \mathbb{R}^+$$

  Discrete density approximation by elements of low discrepancy outperforms random sampling !!!

Discrepancy Bounds for Transformed Points

- **Definition:** The **discrepancy with respect to the density** p is

$$D^*(p, C_N) := \sup_{A \in \mathcal{J}^*} \left| \int_{I^s} \chi_A(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(y_i) \right|$$

  where $C_N = \{y_0, \ldots, y_{N-1}\}$

- Multidimensional inversion method: If p is separable, i.e. $p(x) = \prod_{j=1}^{s} p^{(j)}(x^{(j)})$

$$D^*(p, C_N) = D^*(P_N)$$

  otherwise

$$D^*(p, C_N) \leq c \left(D^*(P_N)\right)^{\frac{1}{s}} \qquad c \in \mathbb{R}^+$$

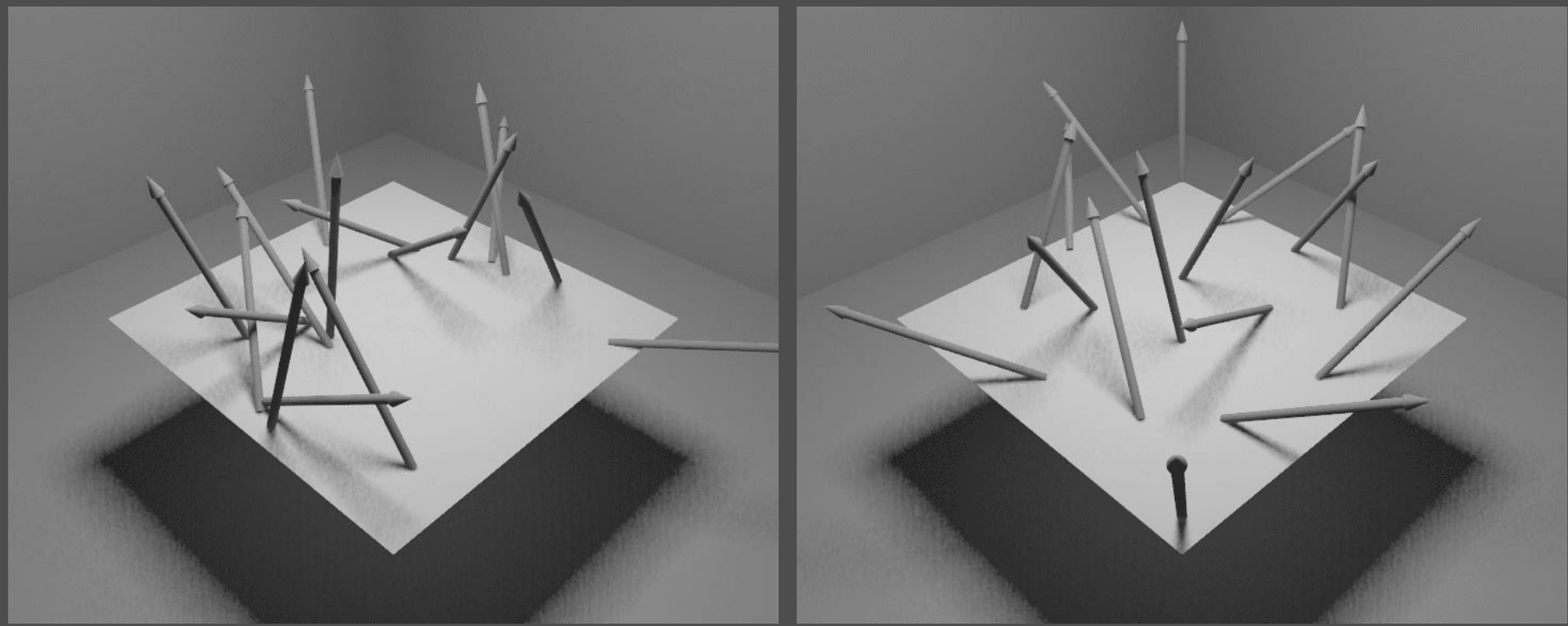
  Discrete density approximation by elements of low discrepancy outperforms random sampling !!!

- Generalized Koksma-Hlawka inequality

$$\left| \int_{I^s} g(x) p(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} g(y_i) \right| \leq V(g) D^*(p, C_N)$$

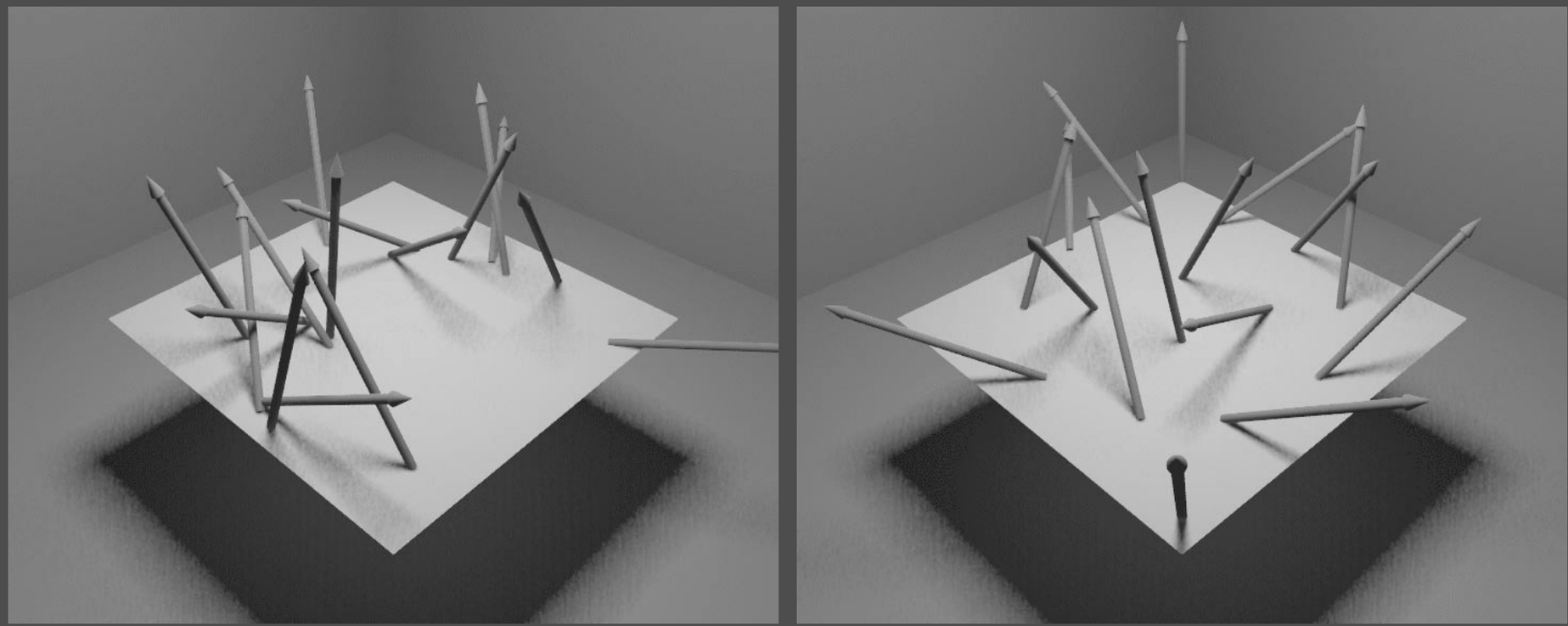
Discrete Density Approximation

- Example: Particle emission (jittered sampling and Hammersley points at $N = 16$)



Discrete Density Approximation

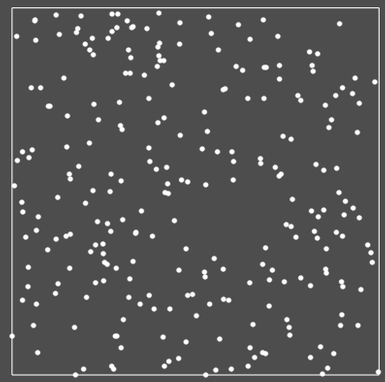
- Example: Particle emission (jittered sampling and Hammersley points at $N = 16$)

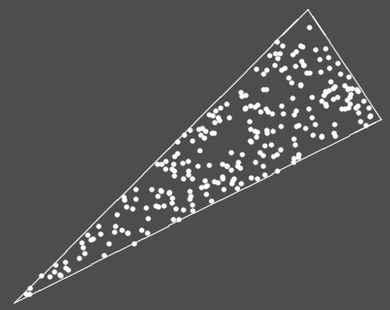


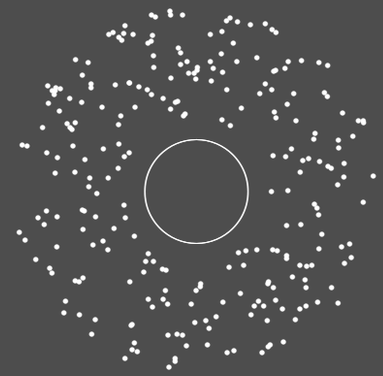
- **Note:** Assigning dimensions is crucial

Discrete Density Approximation

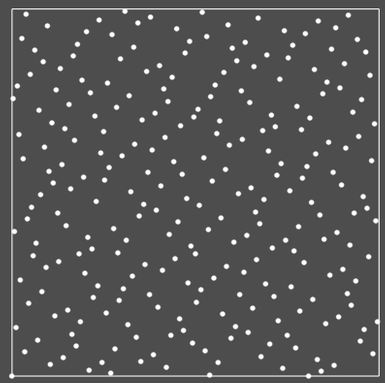
Random

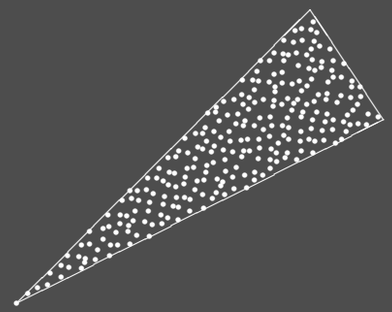


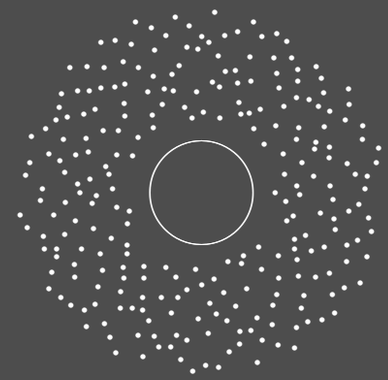




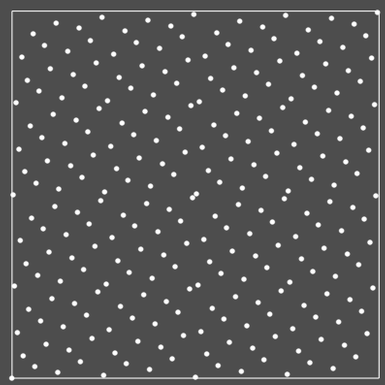
Halton

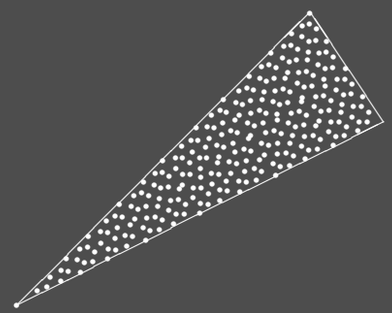


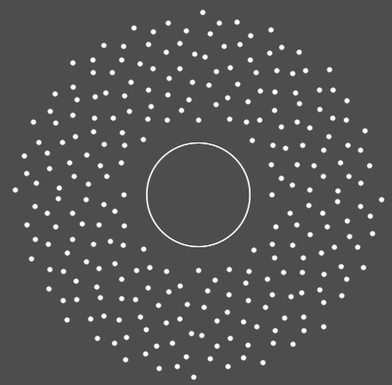




Hammersley





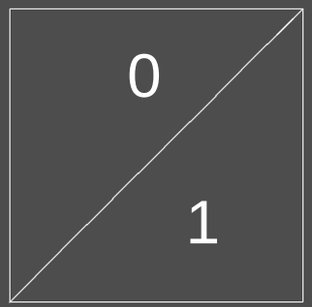


Infinite Variation

- Quasi-Monte Carlo is roughly quadratically faster than random sampling
- Case $s = 1$: $V(f) < \infty$ for piecewise continuous functions
- General case: Usually infinite variation for piecewise continuous functions

Infinite Variation

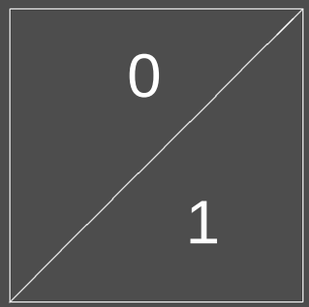
- Quasi-Monte Carlo is roughly quadratically faster than random sampling
- Case $s = 1$: $V(f) < \infty$ for piecewise continuous functions
- General case: Usually infinite variation for piecewise continuous functions
- In computer graphics: Triangles and edges



$$V(f) = \infty \qquad \sigma^2(f) = \frac{1}{4}$$

Infinite Variation

- Quasi-Monte Carlo is roughly quadratically faster than random sampling
- Case $s = 1$: $V(f) < \infty$ for piecewise continuous functions
- General case: Usually infinite variation for piecewise continuous functions
- In computer graphics: Triangles and edges



$$V(f) = \infty \qquad \sigma^2(f) = \frac{1}{4}$$

- Proof for the Hammersley points at $N = 2^l$

$$\left| \int_{I^2} f(x)dx - \frac{1}{N} \sum_{i=0}^{N-1} f(x_i) \right| = \begin{cases} \frac{1}{2\sqrt{N}} & l \text{ even} \\ \frac{1}{\sqrt{2N}} & \text{else} \end{cases}$$

Far Too Pessimistic Bounds by Isotropic Discrepancy

- Restrict f to convex domains C, where $f|_C$ is of bounded variation

$$\begin{aligned}\left|\int_C f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1}\chi_C(x_i)f(x_i)\right| &\leq (V(f) + |f(1,\ldots,1)|)\, J(P_N) \\ &\leq (V(f) + |f(1,\ldots,1)|)\, 8sD^*(P_N)^{\frac{1}{s}}\end{aligned}$$

- Bound worse than the Monte Carlo rate for $s > 2$

Far Too Pessimistic Bounds by Isotropic Discrepancy

- Restrict f to convex domains C, where $f|_C$ is of bounded variation

$$\begin{aligned}\left|\int_C f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1}\chi_C(x_i)f(x_i)\right| &\leq (V(f) + |f(1,\ldots,1)|)\, J(P_N) \\ &\leq (V(f) + |f(1,\ldots,1)|)\, 8sD^*(P_N)^{\frac{1}{s}}\end{aligned}$$

- Bound worse than the Monte Carlo rate for $s > 2$

- Numerical experiments tell a different story...
  - see e.g. the experiments on the triangle discrepancy

- Justification by discrete density approximation
  - using low discrepancy sequences always is better

Far Too Pessimistic Bounds by Isotropic Discrepancy

- Restrict f to convex domains C, where $f|_C$ is of bounded variation

$$\begin{aligned}\left|\int_C f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1}\chi_C(x_i)f(x_i)\right| &\leq (V(f) + |f(1,\ldots,1)|)\, J(P_N) \\ &\leq (V(f) + |f(1,\ldots,1)|)\, 8sD^*(P_N)^{\frac{1}{s}}\end{aligned}$$

- Bound worse than the Monte Carlo rate for $s > 2$

- Numerical experiments tell a different story...
  - see e.g. the experiments on the triangle discrepancy

- Justification by discrete density approximation
  - using low discrepancy sequences always is better

- Which function class other than bounded variation ?

Convergence

- Quasi-Monte Carlo integration converges for Riemann-integrable functions

Convergence

- Quasi-Monte Carlo integration converges for Riemann-integrable functions
- Observed rate for discontinuous functions $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$

Convergence

- Quasi-Monte Carlo integration converges for Riemann-integrable functions
- Observed rate for discontinuous functions $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$
- Argument in ”Numerical Recipes”
  - **Weak assumption:**

    The behavior of low discrepancy samples at the border of characteristic sets is uncorrelated.
  - in fact true for jittered sampling [Mitchell]
  - generalized by Szirmay-Kalos

Convergence

- Quasi-Monte Carlo integration converges for Riemann-integrable functions
- Observed rate for discontinuous functions $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$
- Argument in ”Numerical Recipes”
  - **Weak assumption:**

    The behavior of low discrepancy samples at the border of characteristic sets is uncorrelated.
  - in fact true for jittered sampling [Mitchell]
  - generalized by Szirmay-Kalos
- Argument by [MC95]
  - **Weak assumption:**

    Rate of random sampling used as upper bound for low discrepancy sampling, i.e. it is assumed, that low discrepancy sampling deterministically (!) does not behave worse than random sampling.
  - there exist proofs for some special cases for $s = 2$

The Spirit of the Numerical Recipes' Argument

Proposition: *Using stratified sampling to integrate the characteristic function* χ_A *for some subset* $A \subset I^s$, $\lambda_s(A) > 0$, *for* $N = \prod_{j=1}^{s} N_j$ *and the axial subdivision into* N_j *equally spaced intervals, results in the convergence rate of* $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$.

The Spirit of the Numerical Recipes' Argument

Proposition: *Using stratified sampling to integrate the characteristic function χ_A for some subset $A \subset I^s$, $\lambda_s(A) > 0$, for $N = \prod_{j=1}^{s} N_j$ and the axial subdivision into N_j equally spaced intervals, results in the convergence rate of $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$.*

Proof:

- I^s partitioned into $N = \prod_{j=1}^{s} N_j$ voxels v_i, $\lambda_s(v_i) = \frac{1}{N}$, $1 \leq i \leq N$
- Jittered sampling for

$$\int_{I^s} \chi_A(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i|v_i)$$

The Spirit of the Numerical Recipes' Argument

Proposition: *Using stratified sampling to integrate the characteristic function* χ_A *for some subset* $A \subset I^s$, $\lambda_s(A) > 0$, *for* $N = \prod_{j=1}^{s} N_j$ *and the axial subdivision into* N_j *equally spaced intervals, results in the convergence rate of* $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$.

Proof:

- I^s partitioned into $N = \prod_{j=1}^{s} N_j$ voxels v_i, $\lambda_s(v_i) = \frac{1}{N}$, $1 \leq i \leq N$
- Jittered sampling for

$$\int_{I^s} \chi_A(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i|v_i)$$

- Three sets of voxel indices

$$\begin{aligned} V_i &= \{v_i | v_i \cap A = v_i\} \\ V_b &= \{v_i | \emptyset \neq v_i \cap A \neq v_i\} \\ V_o &= \{v_i | v_i \cap A = \emptyset\} \end{aligned}$$

The Spirit of the Numerical Recipes' Argument

Proposition: *Using stratified sampling to integrate the characteristic function χ_A for some subset $A \subset I^s$, $\lambda_s(A) > 0$, for $N = \prod_{j=1}^{s} N_j$ and the axial subdivision into N_j equally spaced intervals, results in the convergence rate of $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$.*

Proof:

- I^s partitioned into $N = \prod_{j=1}^{s} N_j$ voxels v_i, $\lambda_s(v_i) = \frac{1}{N}$, $1 \leq i \leq N$
- Jittered sampling for

$$\int_{I^s} \chi_A(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i|v_i)$$

- Three sets of voxel indices

$$\begin{aligned} V_i &= \{v_i | v_i \cap A = v_i\} \\ V_b &= \{v_i | \emptyset \neq v_i \cap A \neq v_i\} \\ V_o &= \{v_i | v_i \cap A = \emptyset\} \end{aligned}$$

- ***Assumption:*** $|V_i| \in \mathcal{O}(N)$
- ***Assumption:*** Dimension of the boundary $s-1 \Rightarrow |V_b| \in \mathcal{O}\left(N^{\frac{s-1}{s}}\right)$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1} \chi_A(x_i|v_i)\right) = \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1} \chi_{A \cap v_i}(x_i)\right)$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) = \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right)$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) = \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}|V_i| + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + 0\right)$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) = \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}|V_i| + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + 0\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i)\right)$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) = \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right)$$

$$= \sigma^2\left(\frac{1}{N}|V_i| + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + 0\right)$$

$$= \sigma^2\left(\frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i)\right) = \sum_{i\in V_b}\frac{\sigma^2(\chi_{A\cap v_i}(x_i))}{N^2}$$

- Random sample $x_i \in v_i \in V_b$ is Bernoulli random variable with

$$p_i = \frac{\lambda_s(A \cap v_i)}{\lambda_s(v_i)} \qquad \text{and} \qquad \sigma^2(\chi_{A \cap v_i}) \leq \frac{1}{4}$$

- Then

$$\begin{aligned}
\sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_A(x_i|v_i)\right) &= \sigma^2\left(\frac{1}{N}\sum_{i=0}^{N-1}\chi_{A\cap v_i}(x_i)\right) \\
&= \sigma^2\left(\frac{1}{N}\sum_{i\in V_i}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + \frac{1}{N}\sum_{i\in V_o}\chi_{A\cap v_i}(x_i)\right) \\
&= \sigma^2\left(\frac{1}{N}|V_i| + \frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i) + 0\right) \\
&= \sigma^2\left(\frac{1}{N}\sum_{i\in V_b}\chi_{A\cap v_i}(x_i)\right) = \sum_{i\in V_b}\frac{\sigma^2(\chi_{A\cap v_i}(x_i))}{N^2} \\
&\leq |V_b|\frac{\frac{1}{4}}{N^2} = cN^{\frac{s-1}{s}}N^{-2} = cN^{-\frac{s+1}{s}}
\end{aligned}$$

– By the Hölder inequality the error is expected to be

$$\left| \int_{I^s} \chi_A(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right| \leq \sqrt{c N^{-\frac{s+1}{s}}} \in \mathcal{O}(N^{-\frac{s+1}{2s}}) \qquad \textit{q.e.d.}$$

- By the Hölder inequality the error is expected to be

$$\left| \int_{I^s} \chi_A(x) dx - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(x_i) \right| \leq \sqrt{c N^{-\frac{s+1}{s}}} \in \mathcal{O}(N^{-\frac{s+1}{2s}}) \qquad \textit{q.e.d.}$$

- **Note:**

$$\lim_{s \to \infty} N^{-\frac{s+1}{2s}} = N^{-\frac{1}{2}}$$

Error Control

- Determinism: Variance of estimate is zero !
  - no cheap error estimate from samples
  - no efficiency - complex analysis by information based complexity theory
  - quasi-Monte Carlo integration is ”biased” but ”consistent”

Error Control

- Determinism: Variance of estimate is zero !
  - no cheap error estimate from samples
  - no efficiency - complex analysis by information based complexity theory
  - quasi-Monte Carlo integration is ”biased” but ”consistent”

- Adaptive sampling by using low discrepancy sequences
  - convergence is rather smooth due to intrinsic stratification properties
  - choose fixed distance $\triangle N$ of samples
  - compare difference of averages all $\triangle N$ to a threshold
  - must be below the threshold T times

Error Control

- Determinism: Variance of estimate is zero !
  - no cheap error estimate from samples
  - no efficiency - complex analysis by information based complexity theory
  - quasi-Monte Carlo integration is ”biased” but ”consistent”

- Adaptive sampling by using low discrepancy sequences
  - convergence is rather smooth due to intrinsic stratification properties
  - choose fixed distance $\triangle N$ of samples
  - compare difference of averages all $\triangle N$ to a threshold
  - must be below the threshold T times

- The points ”know” where to fall

Error Control

- Determinism: Variance of estimate is zero !
  - no cheap error estimate from samples
  - no efficiency - complex analysis by information based complexity theory
  - quasi-Monte Carlo integration is ”biased” but ”consistent”

- Adaptive sampling by using low discrepancy sequences
  - convergence is rather smooth due to intrinsic stratification properties
  - choose fixed distance $\triangle N$ of samples
  - compare difference of averages all $\triangle N$ to a threshold
  - must be below the threshold T times

- The points ”know” where to fall

- **Consider local minima for $\triangle N$!**
  - e.g. (t, s)-sequences at $\triangle N = b^m$
  - e.g. Hammersley in $s = 2$

From Monte Carlo to Quasi-Monte Carlo Integration

- The basic algorithms transfer
  - integration
  - integro-approximation
  - Separation of main part and multilevel method of dependent tests

From Monte Carlo to Quasi-Monte Carlo Integration

- The basic algorithms transfer
  - integration
  - integro-approximation
  - Separation of main part and multilevel method of dependent tests

- Faster convergence by deterministic low discrepancy sampling
  - intrinsically stratified, Latin hypercube, regularized, antithetic, ...

From Monte Carlo to Quasi-Monte Carlo Integration

- The basic algorithms transfer
  - integration
  - integro-approximation
  - Separation of main part and multilevel method of dependent tests

- Faster convergence by deterministic low discrepancy sampling
  - intrinsically stratified, Latin hypercube, regularized, antithetic, ...

- The simulation of random variables becomes discrete density approximation
  - no independence required due to averaging
  - importance sampling carries over
  - rejection modeling impossible

From Monte Carlo to Quasi-Monte Carlo Integration

- The basic algorithms transfer
  - integration
  - integro-approximation
  - Separation of main part and multilevel method of dependent tests
- Faster convergence by deterministic low discrepancy sampling
  - intrinsically stratified, Latin hypercube, regularized, antithetic, ...
- The simulation of random variables becomes discrete density approximation
  - no independence required due to averaging
  - importance sampling carries over
  - rejection modeling impossible
- Adaptive sampling by difference comparison
- What about splitting ?

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s
- Separate the main part

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s
- Separate the main part
- Apply (multiple) importance sampling

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s
- Separate the main part
- Apply (multiple) importance sampling
- Use quasi-Monte Carlo points
  - sample size N
  - assigning dimensions

Efficient Design of Quasi-Monte Carlo Algorithms

- Write down the integral
- Transform onto unit cube I^s
- Separate the main part
- Apply (multiple) importance sampling
- Use quasi-Monte Carlo points
  - sample size N
  - assigning dimensions
- Use dependent splitting

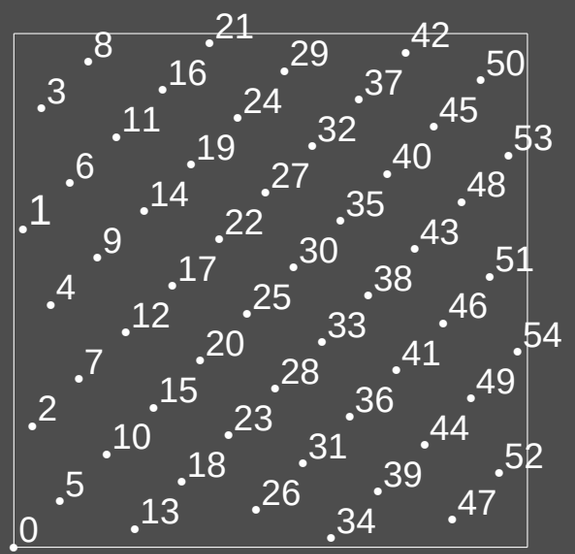
Quasi-Monte Carlo Integration using Lattice Points

- Originally developed for the class $E_\alpha(c)$ with $c > 0, \alpha > 1$, where

$$f \in E_\alpha(c) \Leftrightarrow |\widehat{f}(h)| \leq \frac{c}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha} \qquad \bar{h}_j := \max\{1, |h_j|\}, \mathbf{h} \in \mathbb{Z}^s$$

- Error bound

$$\left| \frac{1}{N} \sum_{i=0}^{N-1} f\left(\frac{i}{N}\mathbf{g}\right) - \int_{I^s} f(x)dx \right| \leq \sum_{\mathbf{h}\cdot\mathbf{g} \equiv 0 (\bmod N), \mathbf{h} \neq 0} \frac{1}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha}$$



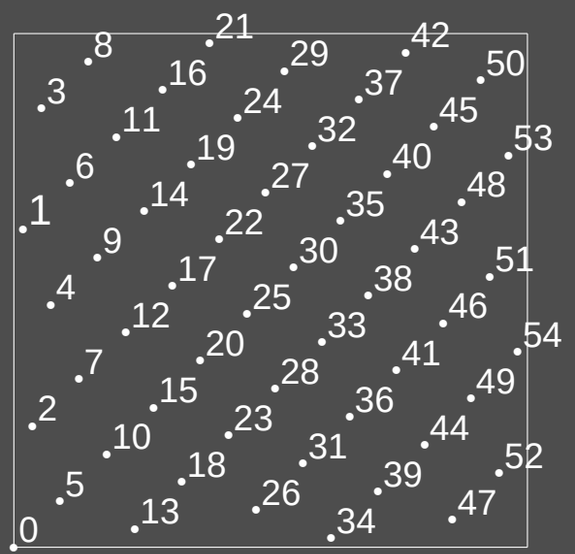
Quasi-Monte Carlo Integration using Lattice Points

- Originally developed for the class $E_\alpha(c)$ with $c > 0, \alpha > 1$, where

$$f \in E_\alpha(c) \Leftrightarrow |\hat{f}(h)| \leq \frac{c}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha} \qquad \bar{h}_j := \max\{1, |h_j|\}, \mathbf{h} \in \mathbb{Z}^s$$

- Error bound

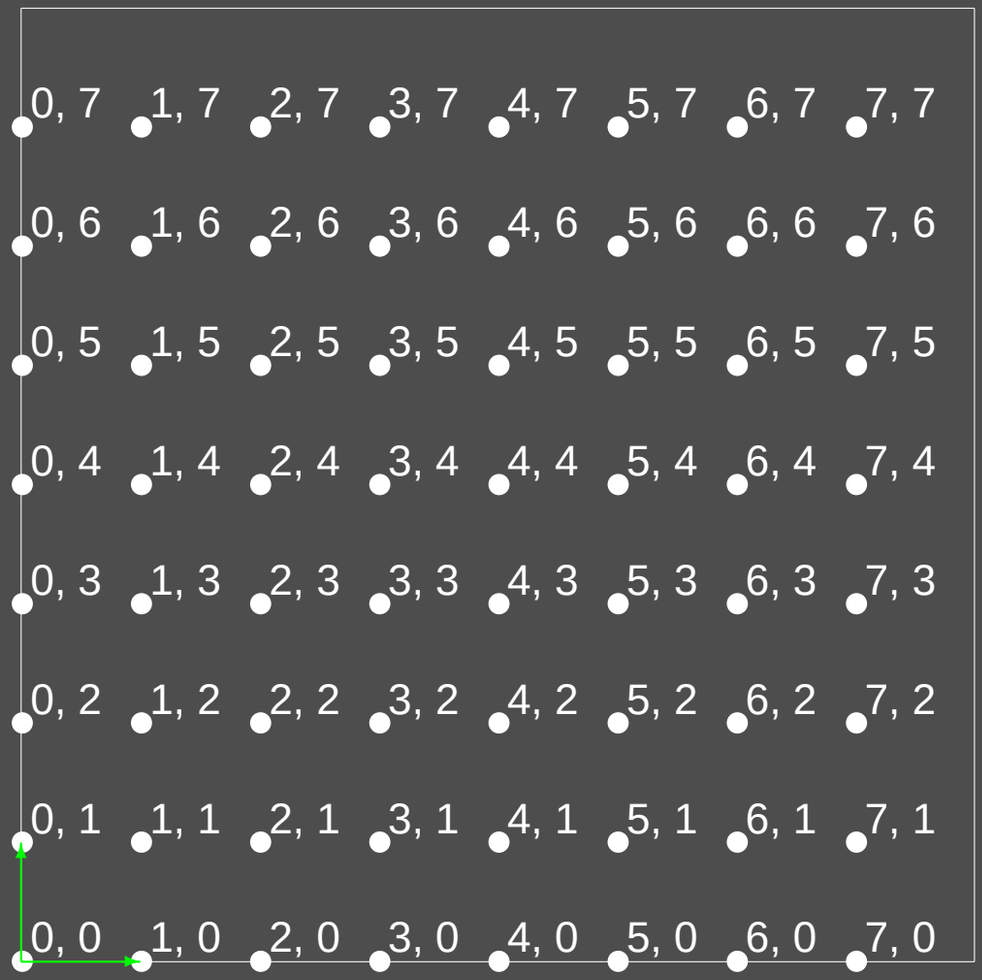
$$\left| \frac{1}{N} \sum_{i=0}^{N-1} f\left(\frac{i}{N}\mathbf{g}\right) - \int_{I^s} f(x)dx \right| \leq \sum_{\mathbf{h}\cdot\mathbf{g} \equiv 0 (\bmod N), \mathbf{h} \neq 0} \frac{1}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha}$$



- Generalized to class of bounded variation

Curse of Dimension from Regular Grids

- Lattices of rank s with $N = n^s$ points from tensor product approach



- $\mathcal{O}\left(n^s \log n\right)$ for s fast Fourier transforms

Fourier Transform on Rank-1 *Lattices*

- Choice of wave vectors

$$K_N := \{\mathbf{k}_0, \ldots, \mathbf{k}_{N-1}\} \subset \mathbb{Z}^s$$

such that

$$\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod{N}\}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors

$$K_N := \{\mathbf{k}_0, \ldots, \mathbf{k}_{N-1}\} \subset \mathbb{Z}^s$$

such that

$$\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod{N}\}$$

since then

$$\mathbf{k}_m^T \cdot \mathbf{x}_n = \mathbf{k}_m^T \cdot \frac{n}{N}\mathbf{g} = (m + l_m N)\frac{n}{N}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors

  $$K_N := \{\mathbf{k}_0, \ldots, \mathbf{k}_{N-1}\} \subset \mathbb{Z}^s$$

  such that

  $$\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod N\}$$

  since then

  $$\mathbf{k}_m^T \cdot \mathbf{x}_n = \mathbf{k}_m^T \cdot \frac{n}{N}\mathbf{g} = (m + l_m N)\frac{n}{N}$$

- Evaluate

  $$\mathbf{f}(\mathbf{x}_n) = \sum_{\mathbf{k} \in K_N} \hat{\mathbf{f}}(\mathbf{k}) e^{2\pi i \mathbf{k}^T \cdot \mathbf{x}_n} = \sum_{m=0}^{N-1} \hat{\mathbf{f}}(\mathbf{k}_m) e^{2\pi i \mathbf{k}_m^T \cdot \mathbf{x}_n}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors

$$K_N := \{\mathbf{k}_0, \ldots, \mathbf{k}_{N-1}\} \subset \mathbb{Z}^s$$

such that

$$\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod{N}\}$$

since then

$$\mathbf{k}_m^T \cdot \mathbf{x}_n = \mathbf{k}_m^T \cdot \frac{n}{N}\mathbf{g} = (m + l_m N)\frac{n}{N}$$

- Evaluate

$$\begin{aligned} \mathbf{f}(\mathbf{x}_n) &= \sum_{\mathbf{k} \in K_N} \hat{\mathbf{f}}(\mathbf{k}) e^{2\pi i \mathbf{k}^T \cdot \mathbf{x}_n} = \sum_{m=0}^{N-1} \hat{\mathbf{f}}(\mathbf{k}_m) e^{2\pi i \mathbf{k}_m^T \cdot \mathbf{x}_n} \\ &= \sum_{m=0}^{N-1} \hat{\mathbf{f}}(\mathbf{k}_m) e^{2\pi i (m\frac{n}{N} + l_m n)} \end{aligned}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors

  $$K_N := \{\mathbf{k}_0, \ldots, \mathbf{k}_{N-1}\} \subset \mathbb{Z}^s$$

  such that

  $$\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod N\}$$

  since then

  $$\mathbf{k}_m^T \cdot \mathbf{x}_n = \mathbf{k}_m^T \cdot \frac{n}{N}\mathbf{g} = (m + l_m N)\frac{n}{N}$$

- Evaluate

  $$\begin{aligned} \mathbf{f}(\mathbf{x}_n) &= \sum_{\mathbf{k} \in K_N} \hat{\mathbf{f}}(\mathbf{k}) e^{2\pi i \mathbf{k}^T \cdot \mathbf{x}_n} = \sum_{m=0}^{N-1} \hat{\mathbf{f}}(\mathbf{k}_m) e^{2\pi i \mathbf{k}_m^T \cdot \mathbf{x}_n} \\ &= \sum_{m=0}^{N-1} \hat{\mathbf{f}}(\mathbf{k}_m) e^{2\pi i (m\frac{n}{N} + l_m n)} \\ &= \sum_{m=0}^{N-1} \hat{\mathbf{f}}(\mathbf{k}_m) e^{2\pi i m \frac{n}{N}} \end{aligned}$$

  by **one-dimensional** Fourier transform $\Rightarrow$ **way to break curse of dimension !**

Determining the Wave Vectors

- Many possible choices for

$$\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod{N}\}$$

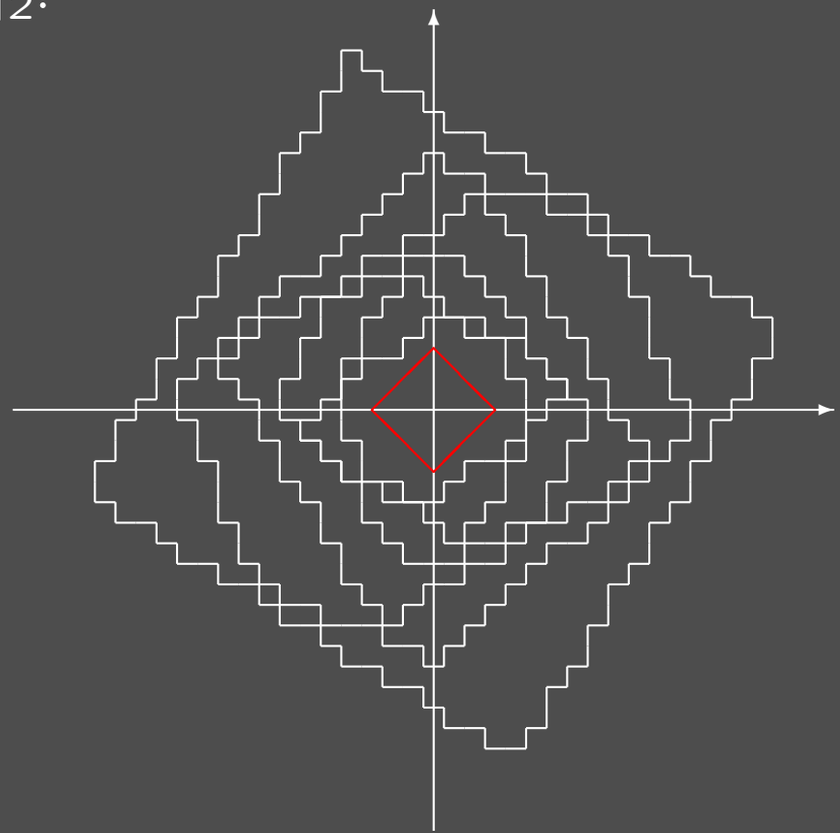
Determining the Wave Vectors

- Many possible choices for

$$\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod{N}\}$$

- Choose largest waves first

$$\|\mathbf{k}_m\|_2 = \min_{\mathbf{k} \in Z_m} \|\mathbf{k}\|_2.$$



- Enumerate along lines of constant $\|\cdot\|_1$-norm

Summary

- Quasi-Monte Carlo simpler and faster than Monte Carlo integration
- Most Monte Carlo techniques transfer
- However, no rejection sampling !
- Works fine on L^2, too
  - justification by discrete density approximation
- Breaks curse of dimension even for discrete Fourier transform

Summary

- Quasi-Monte Carlo simpler and faster than Monte Carlo integration
- Most Monte Carlo techniques transfer
- However, no rejection sampling !
- Works fine on L^2, too
  - justification by discrete density approximation
- Breaks curse of dimension even for discrete Fourier transform

- **Use whenever you can write the problem as an integral**

Monte Carlo and Beyond

- Principles of rendering algorithms
- Monte Carlo integration
- Quasi-Monte Carlo points
- Quasi-Monte Carlo integration
- **Monte Carlo extensions of quasi-Monte Carlo**
  - Random field synthesis on good lattice points
  - Randomized quasi-Monte Carlo integration
  - Randomized replications
  - Restricted randomized replications
- Application to computer graphics

Periodic Random Field Synthesis on Good Lattice Points

- Applications of Periodic Random Fields $\mathbf{f}_\omega(\mathbf{x}) = \mathbf{f}_\omega(\mathbf{x} + \mathbf{z})$ for $\mathbf{z} \in \mathbb{Z}^s$ (Period 1)
  - height fields: Waves, terrain
  - caustics
  - turbulent wind fields

Periodic Random Field Synthesis on Good Lattice Points

- Applications of Periodic Random Fields $\mathbf{f}_\omega(\mathbf{x}) = \mathbf{f}_\omega(\mathbf{x} + \mathbf{z})$ for $\mathbf{z} \in \mathbb{Z}^s$ (Period 1)
  - height fields: Waves, terrain
  - caustics
  - turbulent wind fields
- Typical procedure

1. Realize Gaussian noise

$$\mathbf{N}_\omega(\mathbf{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d$$

Periodic Random Field Synthesis on Good Lattice Points

- Applications of Periodic Random Fields $\mathbf{f}_\omega(\mathbf{x}) = \mathbf{f}_\omega(\mathbf{x} + \mathbf{z})$ for $\mathbf{z} \in \mathbb{Z}^s$ (Period 1)
  - height fields: Waves, terrain
  - caustics
  - turbulent wind fields
- Typical procedure

1. Realize Gaussian noise

$$\mathbf{N}_\omega(\mathbf{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d$$

2. Filter noise by spectrum S of phenomenon

$$\hat{\mathbf{f}}_\omega(\mathbf{k}) = S(\mathbf{k})\mathbf{N}_\omega(\mathbf{k})$$

Periodic Random Field Synthesis on Good Lattice Points

- Applications of Periodic Random Fields $\mathbf{f}_\omega(\mathbf{x}) = \mathbf{f}_\omega(\mathbf{x} + \mathbf{z})$ for $\mathbf{z} \in \mathbb{Z}^s$ (Period 1)
  - height fields: Waves, terrain
  - caustics
  - turbulent wind fields
- Typical procedure

1. Realize Gaussian noise

$$\mathbf{N}_\omega(\mathbf{k}) \sim (\mathcal{N}(0,1) \times i\mathcal{N}(0,1))^d$$

2. Filter noise by spectrum S of phenomenon

$$\hat{\mathbf{f}}_\omega(\mathbf{k}) = S(\mathbf{k})\mathbf{N}_\omega(\mathbf{k})$$

3. Band limited evaluation by fast Fourier transform

$$\mathbf{f}_\omega(\mathbf{x}) = \sum_{\mathbf{k} \in K_N} \hat{\mathbf{f}}_\omega(\mathbf{k}) e^{2\pi i \mathbf{k}^T \cdot \mathbf{x}}$$

Fourier Transform on Rank-1 Lattices

- Choice of wave vectors $K_N := \{\mathbf{k}_0, \ldots, \mathbf{k}_{N-1}\} \subset \mathbb{Z}^s$

  such that

  $$\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod N\}$$

  hence with $x_n = \frac{n}{N}\mathbf{g}$

  $$\mathbf{k}_m^T \cdot \mathbf{x}_n = \mathbf{k}_m^T \cdot \frac{n}{N}\mathbf{g} = (m + l_m N)\frac{n}{N}$$

- By **one-dimensional** Fourier transform evaluate

$$\begin{aligned}
\mathbf{f}(\mathbf{x}_n) &= \sum_{\mathbf{k}\in K_N} \hat{\mathbf{f}}_\omega(\mathbf{k}) e^{2\pi i \mathbf{k}^T \cdot \mathbf{x}_n} = \sum_{m=0}^{N-1} \hat{\mathbf{f}}_\omega(\mathbf{k}_m) e^{2\pi i \mathbf{k}_m^T \cdot \mathbf{x}_n} \\
&= \sum_{m=0}^{N-1} \hat{\mathbf{f}}_\omega(\mathbf{k}_m) e^{2\pi i (m\frac{n}{N} + l_m n)} \\
&= \sum_{m=0}^{N-1} \hat{\mathbf{f}}_\omega(\mathbf{k}_m) e^{2\pi i m \frac{n}{N}}
\end{aligned}$$

Application: Ocean Wave Simulation

- **Ocean height field synthesis**

1. Realize Gaussian noise random field $\xi_{r,m}, \xi_{i,m} \sim \mathcal{N}(0, 1)$

Application: Ocean Wave Simulation

- **Ocean height field synthesis**

1. Realize Gaussian noise random field $\xi_{r,m}, \xi_{i,m} \sim \mathcal{N}(0, 1)$

2. Fourier coefficients by filtering with Philipps spectrum $P_h(k_m)$

$$\hat{h}_\omega(\mathbf{k}_m, t) = \sqrt{\frac{P_h(k_m)}{2}} \left((\xi_{r,m} + i\xi_{i,m}) e^{i\omega(k_m)t} + (\xi_{r,m} - i\xi_{i,m}) e^{-i\omega(k_m)t} \right)$$

Application: Ocean Wave Simulation

- **Ocean height field synthesis**

1. Realize Gaussian noise random field $\xi_{r,m}, \xi_{i,m} \sim \mathcal{N}(0,1)$

2. Fourier coefficients by filtering with Philipps spectrum $P_h(k_m)$

$$\hat{h}_\omega(\mathbf{k}_m, t) = \sqrt{\frac{P_h(k_m)}{2}} \left((\xi_{r,m} + i\xi_{i,m}) e^{i\omega(k_m)t} + (\xi_{r,m} - i\xi_{i,m}) e^{-i\omega(k_m)t} \right)$$

3. Height field $h_\omega : \mathbb{R}^3 \to \mathbb{R}$ and normals by $\nabla h_\omega : \mathbb{R}^3 \to \mathbb{R}^3$

$$\begin{aligned} h_\omega(\mathbf{x}_n, t) &= \sum_{m=0}^{N-1} \hat{h}_\omega(\mathbf{k}_m, t) e^{2\pi i m \frac{n}{N}} \\ \nabla h_\omega(\mathbf{x}_n, t) &= \sum_{m=0}^{N-1} 2\pi i \mathbf{k}_m \hat{h}_\omega(\mathbf{k}_m, t) e^{2\pi i m \frac{n}{N}} \end{aligned}$$

$\Rightarrow \dim \mathbf{x}_n = 2$, but evaluation by **one-dimensional** fast Fourier transform

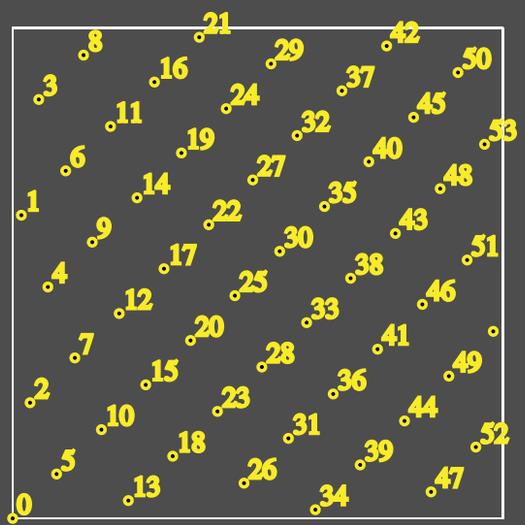
Example: Ocean Waves on Fibonacci Rank-1 Lattices

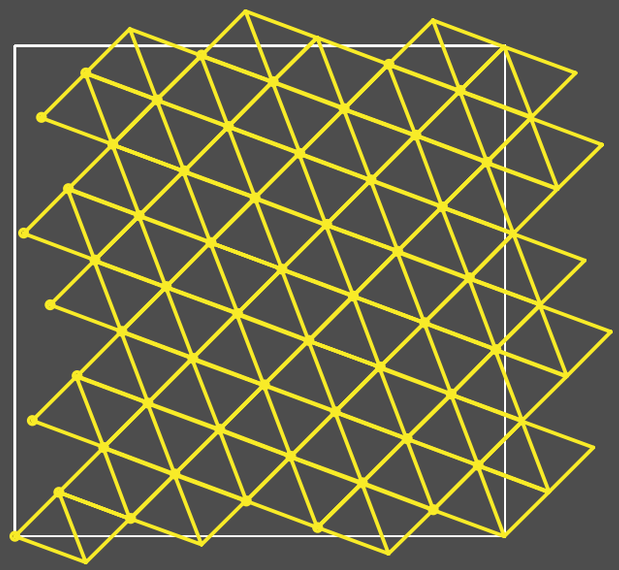
- Fibonacci numbers: $F_1 = F_2 = 1$, $F_k = F_{k-1} + F_{k-2}$ for $k > 2$
- **Fibonacci lattice** by generator vector $\mathrm{g} = (1, F_{k-1})$ at $N = F_k$ points

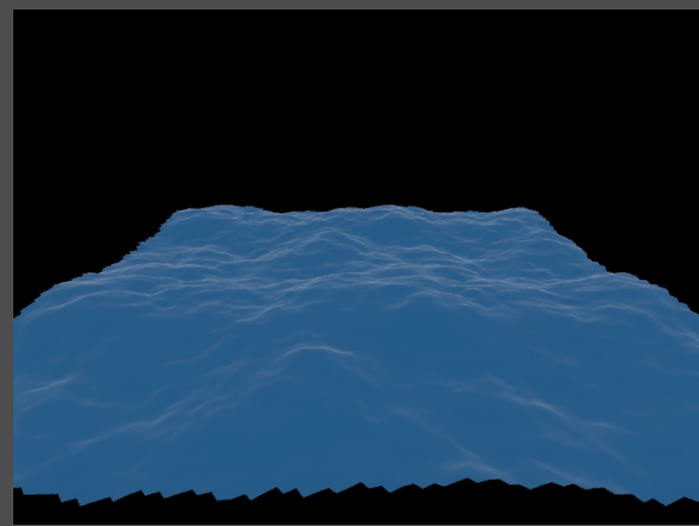
$$\mathbf{x}_n := \frac{n}{F_k}(1, F_{k-1})$$

  - Low discrepancy

- Example: $N = F_{10} = 55$, $\mathbf{x}_n := \frac{n}{55}(1, 34)$

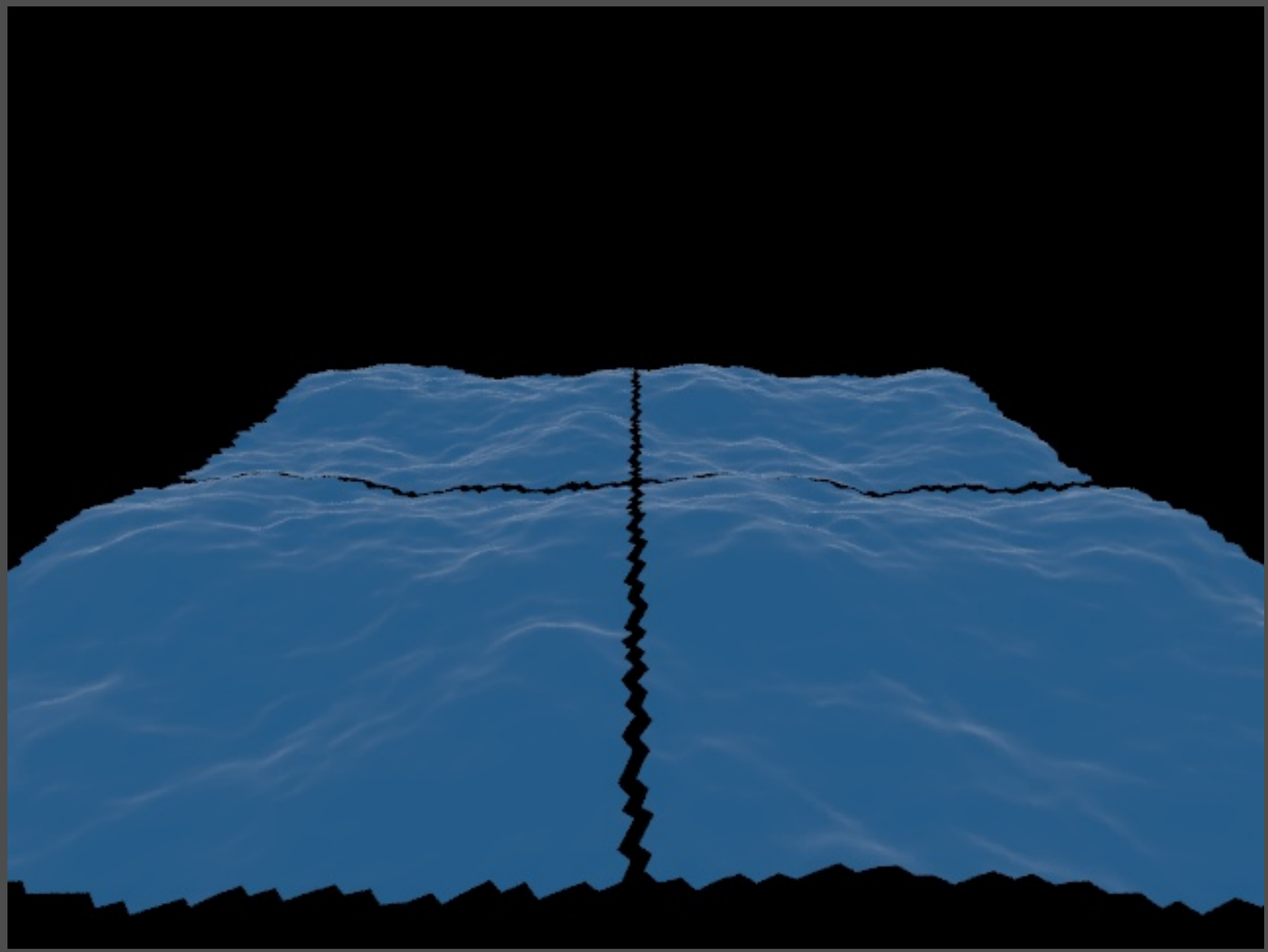




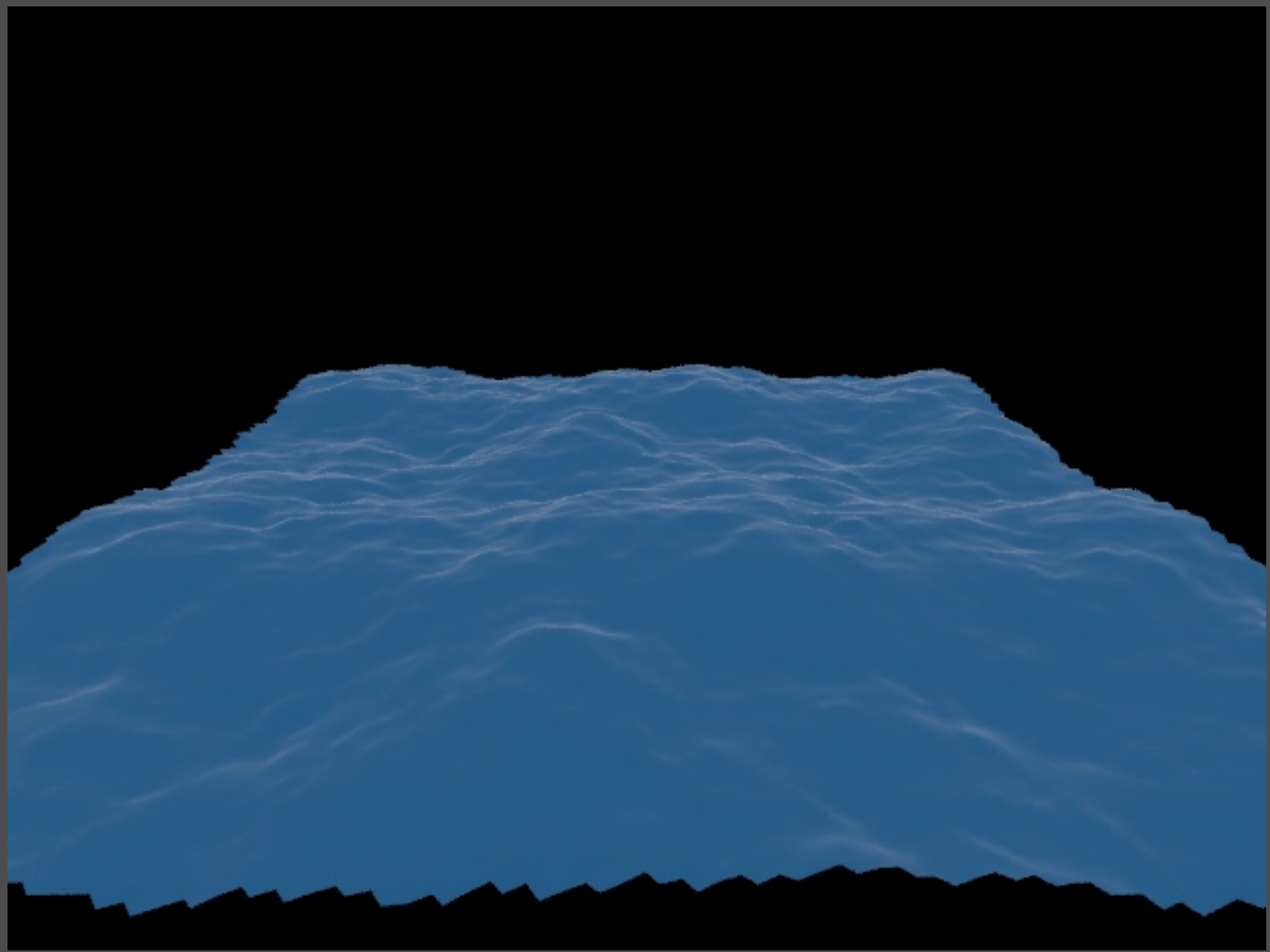


- Barycentric interpolation on periodic Delauney triangulation

Periodic Tiling



Periodic Tiling



Breaking the Curse of Dimension

- Point set $P_N = \{x_0, \ldots, x_{N-1}\}$
- **Monte Carlo Integration:** Random points P_N

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3}{\sqrt{N}}\sigma(f)\right\}\right) \approx 0.997$$

  - slow
  - **cheap error estimate**
  - **easy math for L^2**
- **Quasi-Monte Carlo Integration:** Quasi-Monte Carlo points P_N

$$\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < D^*(P_N)V(f)$$

  - **fast**
  - no error estimate
  - heavy math for BV

Breaking the Curse of Dimension

- Point set $P_N = \{x_0, \ldots, x_{N-1}\}$
- **Monte Carlo Integration:** Random points P_N

$$\textbf{Prob}\left(\left\{\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < \frac{3}{\sqrt{N}}\sigma(f)\right\}\right) \approx 0.997$$

  - slow
  - **cheap error estimate**
  - **easy math for L^2**
- **Quasi-Monte Carlo Integration:** Quasi-Monte Carlo points P_N

$$\left|\int_{I^s} f(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)\right| < D^*(P_N)V(f)$$

  - **fast**
  - no error estimate
  - heavy math for BV
- Combine and take the best !
- Price: A little bit of convergence, problems of random number generators

Randomized Quasi-Monte Carlo Integration

- Randomized replications of a QMC point set $A := \{A_0, \ldots, A_{n-1}\}$

  $X_k := \{X_{k,0}, \ldots, X_{k,n-1}\}$ for $1 \leq k \leq r$

  such that

  1. **Uniformity:** $X_{k,i} \sim U[0,1)^s$ for fixed i
  2. **Equidistribution:** $X_1, \ldots, X_r$ are low-discrepancy point sets with probability one

Randomized Quasi-Monte Carlo Integration

- Randomized replications of a QMC point set $A := \{A_0, \ldots, A_{n-1}\}$

  $X_k := \{X_{k,0}, \ldots, X_{k,n-1}\}$ for $1 \leq k \leq r$

  such that

  1. **Uniformity:** $X_{k,i} \sim U[0,1)^s$ for fixed i
  2. **Equidistribution:** $X_1, \ldots, X_r$ are low-discrepancy point sets with probability one

- Monte Carlo estimate

$$I_{r,n} f := \frac{1}{r} \sum_{k=1}^{r} \frac{1}{n} \sum_{i=0}^{n-1} f(X_{k,i})$$

Randomized Quasi-Monte Carlo Integration

- Randomized replications of a QMC point set $A := \{A_0, \ldots, A_{n-1}\}$

  $X_k := \{X_{k,0}, \ldots, X_{k,n-1}\}$ for $1 \leq k \leq r$

  such that

  1. **Uniformity:** $X_{k,i} \sim U[0,1)^s$ for fixed i
  2. **Equidistribution:** $X_1, \ldots, X_r$ are low-discrepancy point sets with probability one

- Monte Carlo estimate

  $$I_{r,n} f := \frac{1}{r} \sum_{k=1}^{r} \frac{1}{n} \sum_{i=0}^{n-1} f(X_{k,i})$$

  with error estimate

  $$\sigma^2(I_{r,n} f) \approx \frac{1}{r(r-1)} \sum_{k=1}^{r} \left(\frac{1}{n} \sum_{i=0}^{n-1} f(X_{k,i}) - I_{r,n} f \right)^2$$

Randomized Quasi-Monte Carlo Integration

- Randomized replications of a QMC point set $A := \{A_0, \ldots, A_{n-1}\}$

  $X_k := \{X_{k,0}, \ldots, X_{k,n-1}\}$ for $1 \le k \le r$

  such that

  1. **Uniformity:** $X_{k,i} \sim U[0,1)^s$ for fixed i
  2. **Equidistribution:** $X_1, \ldots, X_r$ are low-discrepancy point sets with probability one

- Monte Carlo estimate

  $$I_{r,n}f := \frac{1}{r}\sum_{k=1}^{r}\frac{1}{n}\sum_{i=0}^{n-1} f(X_{k,i})$$

  with error estimate

  $$\sigma^2(I_{r,n}f) \approx \frac{1}{r(r-1)}\sum_{k=1}^{r}\left(\frac{1}{n}\sum_{i=0}^{n-1} f(X_{k,i}) - I_{r,n}f\right)^2$$

- Presmoothing of the integrand by correlated sampling

Randomized Replications

- Random bijections

$$R_\omega : I^s \to I^s$$

  - in fact dependent sampling replication heuristics

Randomized Replications

- Random bijections

$$R_\omega : I^s \to I^s$$

  - in fact dependent sampling replication heuristics

- Cranley-Patterson rotations
  - originally designed for error estimation with lattice points
  - very simple

Randomized Replications

- Random bijections

  $$R_\omega : I^s \to I^s$$

  - in fact dependent sampling replication heuristics

- Cranley-Patterson rotations
  - originally designed for error estimation with lattice points
  - very simple

- Owen-Scrambling
  - designed for (t, m, s)-nets and (t, s)-sequences in base b
  - advanced

Randomized Replications by Cranley-Patterson Rotations

- Random shifts on the torus I^s applied to A

$$X^{(j)}_{k,i} := A^{(j)}_i + U^{(j)}_k \bmod 1 \text{ for } 1 \leq j \leq s$$

Randomized Replications by Cranley-Patterson Rotations

- Random shifts on the torus I^s applied to A

$$X_{k,i}^{(j)} := A_i^{(j)} + U_k^{(j)} \bmod 1 \text{ for } 1 \leq j \leq s$$

- Originally A was a lattice of low discrepancy

- **Note:** Cranley-Patterson rotations work with any arbitrary point set A
  - still unbiased Monte Carlo scheme

Randomized Replications by Cranley-Patterson Rotations

- Random shifts on the torus I^s applied to A

$$X_{k,i}^{(j)} := A_i^{(j)} + U_k^{(j)} \bmod 1 \text{ for } 1 \leq j \leq s$$

- Originally A was a lattice of low discrepancy

- **Note:** Cranley-Patterson rotations work with any arbitrary point set A
  - still unbiased Monte Carlo scheme
  - especially for (t, s)-sequences and (t, m, s)-nets
    - however discrepancy can be affected due to shifting

Randomized Replications by Cranley-Patterson Rotations

- Random shifts on the torus I^s applied to A

$$X_{k,i}^{(j)} := A_i^{(j)} + U_k^{(j)} \bmod 1 \text{ for } 1 \leq j \leq s$$

- Originally A was a lattice of low discrepancy

- **Note:** Cranley-Patterson rotations work with any arbitrary point set A
  - still unbiased Monte Carlo scheme
  - especially for (t, s)-sequences and (t, m, s)-nets
    - however discrepancy can be affected due to shifting
  - example: Padded replications sampling
    - pad A by low dimensional point sets, apply random shifts
    - exploit problem structure, e.g. in transport problems
    - cheaper point sets than quasi-Monte Carlo points in high dimensions

Randomized Replications by Owen-Scrambling

- Scramble (t, m, s)-nets and (t, s)-sequences in base b
- Algorithm: Start with $H = I^s$ and for each axis
  1. slice H into b equally sized volumes $H_1, H_2, \ldots, H_b$ along the axis
  2. randomly permute these volume
  3. for each H_h recursively repeat the procedure with $H = H_h$

Randomized Replications by Owen-Scrambling

- Scramble (t, m, s)-nets and (t, s)-sequences in base b
- Algorithm: Start with $H = I^s$ and for each axis
  1. slice H into b equally sized volumes $H_1, H_2, \ldots, H_b$ along the axis
  2. randomly permute these volume
  3. for each H_h recursively repeat the procedure with $H = H_h$
- Algorithm gets finite by finite precision of computation, i.e. digital constructions
- Net and sequence parameters remain untouched
  - contrary to random shifts by Cranley-Patterson

Randomized Replications by Owen-Scrambling

- Scramble (t, m, s)-nets and (t, s)-sequences in base b
- Algorithm: Start with $H = I^s$ and for each axis
  1. slice H into b equally sized volumes $H_1, H_2, \ldots, H_b$ along the axis
  2. randomly permute these volume
  3. for each H_h recursively repeat the procedure with $H = H_h$
- Algorithm gets finite by finite precision of computation, i.e. digital constructions
- Net and sequence parameters remain untouched
  - contrary to random shifts by Cranley-Patterson
- Much faster convergence for $N > s^s$

  $$\mathcal{O}\left(\frac{\log^{\frac{s-1}{2}} N}{N^{\frac{3}{2}}}\right)$$

  due to extinction effects by full stratification

Replication by Scrambling

- Unit square $[0, 1)^2$

Replication by Scrambling

- Bit 1 of x

Replication by Scrambling

- Bit 2 of x

Replication by Scrambling

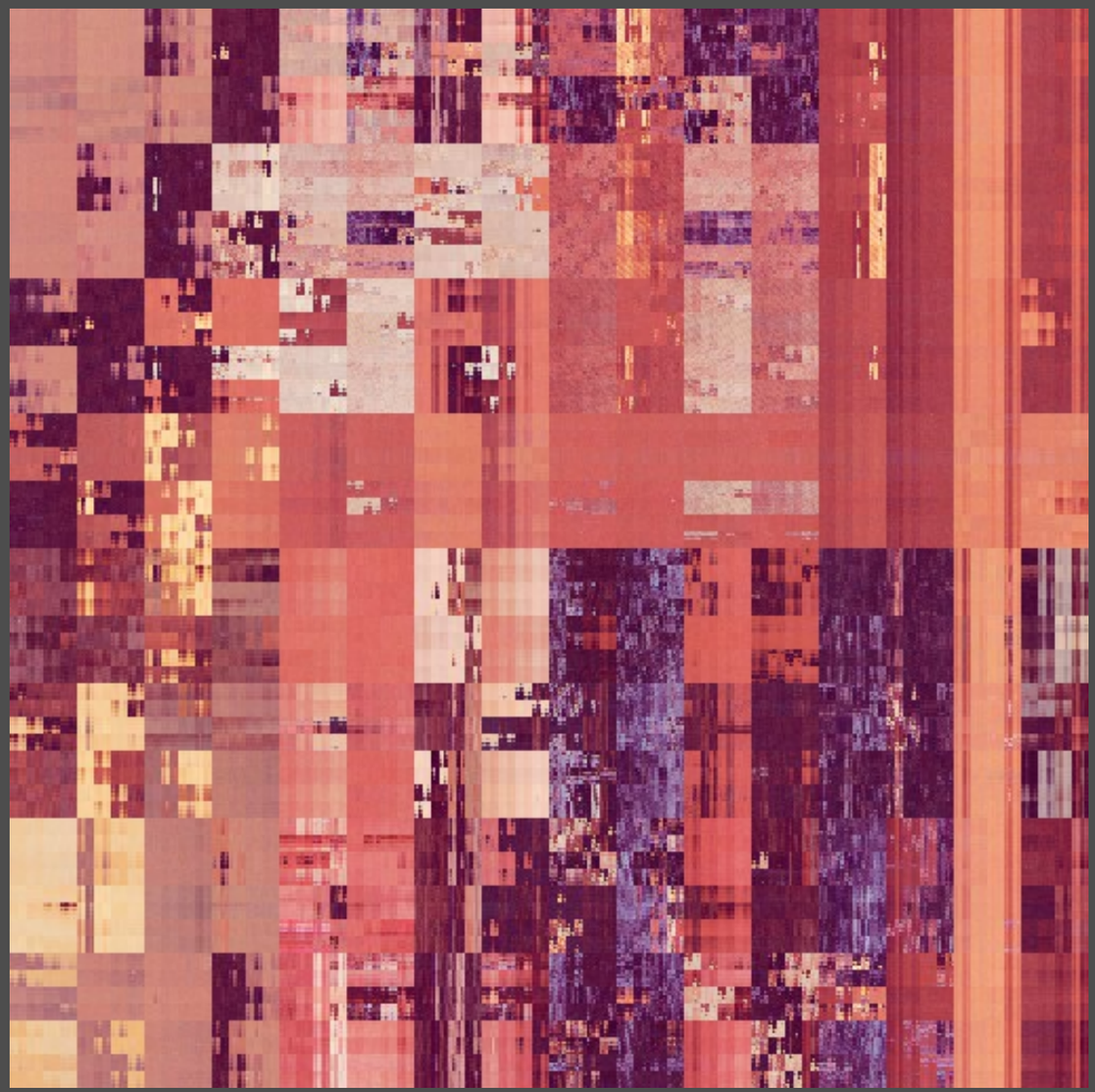
- Bit 3 of x

Replication by Scrambling

- All bits of x

Replication by Scrambling

- All bits of x and y



Formalization of Scrambling

- Given a digital (t, m, s)-net $A = \{A_0, \ldots, A_{N-1}\}$ in base b with components

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)}$$

Formalization of Scrambling

- Given a digital (t, m, s)-net $A = \{A_0, \ldots, A_{N-1}\}$ in base b with components

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)}$$

- A **scrambled replicate $\mathbf{X}$ of $\mathbf{A}$** is obtained by

$$X_i^{(j)} = \sum_{k=1}^{M} x_{i,k}^{(j)} \cdot b^{-k} =_b 0.x_{i,1}^{(j)} x_{i,2}^{(j)} x_{i,3}^{(j)} \cdots x_{i,M}^{(j)}$$

Formalization of Scrambling

- Given a digital (t, m, s)-net $A = \{A_0, \ldots, A_{N-1}\}$ in base b with components

$$A_i^{(j)} = \sum_{k=1}^{M} a_{i,k}^{(j)} \cdot b^{-k} =_b 0.a_{i,1}^{(j)} a_{i,2}^{(j)} \ldots a_{i,M}^{(j)}$$

- A **scrambled replicate $\mathbf{X}$ of $\mathbf{A}$** is obtained by

$$X_i^{(j)} = \sum_{k=1}^{M} x_{i,k}^{(j)} \cdot b^{-k} =_b 0.x_{i,1}^{(j)} x_{i,2}^{(j)} x_{i,3}^{(j)} \cdots x_{i,M}^{(j)}$$

where

$$\begin{aligned}
x_{i,1}^{(j)} &:= \pi^{(j)}\left(a_{i,1}^{(j)}\right) \\
x_{i,2}^{(j)} &:= \pi_{a_{i,1}^{(j)}}^{(j)}\left(a_{i,2}^{(j)}\right) \\
&\vdots \\
x_{i,M}^{(j)} &:= \pi_{a_{i,1}^{(j)}, a_{i,2}^{(j)}, \ldots, a_{i,M-1}^{(j)}}^{(j)}\left(a_{i,M}^{(j)}\right)
\end{aligned}$$

- Independent random permutations $\pi^{(j)} \in S_b$
- Permutation depends on the $k-1$ leading digits of $A_i^{(j)} \Rightarrow$ permutation tree

Efficient Implementation of Scrambling

- **Main ideas for efficient scrambling:**
  - keep only one path of the permutation tree in memory
  - traverse permutation tree paths that way, that each permutation is used only once

Efficient Implementation of Scrambling

- **Main ideas for efficient scrambling:**
  - keep only one path of the permutation tree in memory
  - traverse permutation tree paths that way, that each permutation is used only once

- Implies reordering of the points that should be scrambled
  - sorting the components

$$A^{(j)} = \{A_0^{(j)}, \ldots, A_{N-1}^{(j)}\} \rightarrow A_{\sigma_j(0)}^{(j)} \leq \cdots \leq A_{\sigma_j(N-1)}^{(j)}$$

  - in this order scramble the components

    $\Rightarrow$ each branch of the permutation tree is traversed at most once
  - undo the sorting using the inverse permutation σ_j^{-1}

Example: Scrambled $(0, m, 2)$*-Nets in Base* $b = 2$

- $N = 2^m$ points $A = \{A_0, \ldots, A_{N-1}\}$
- The components correspond to the inverse permutations $\sigma_j^{-1}(i) = N \cdot A_i^{(j)}$
  - e.g. Hammersley: $\sigma_0^{-1}(i) = 2^m \cdot \frac{i}{N}$ and $\sigma_1^{-1}(i) = 2^m \cdot \Phi_2(i)$
- Random permutations on $\mathbb{Z}_2$ are random bit flips and can be vectorized
  - i.e. applying a path of permutation means XORing the bit vector of bit permutations

Example: Scrambled $(0, m, 2)$*-Nets in Base* $b = 2$

- $N = 2^m$ points $A = \{A_0, \ldots, A_{N-1}\}$
- The components correspond to the inverse permutations $\sigma_j^{-1}(i) = N \cdot A_i^{(j)}$
  - e.g. Hammersley: $\sigma_0^{-1}(i) = 2^m \cdot \frac{i}{N}$ and $\sigma_1^{-1}(i) = 2^m \cdot \Phi_2(i)$
- Random permutations on $\mathbb{Z}_2$ are random bit flips and can be vectorized
  - i.e. applying a path of permutation means XORing the bit vector of bit permutations

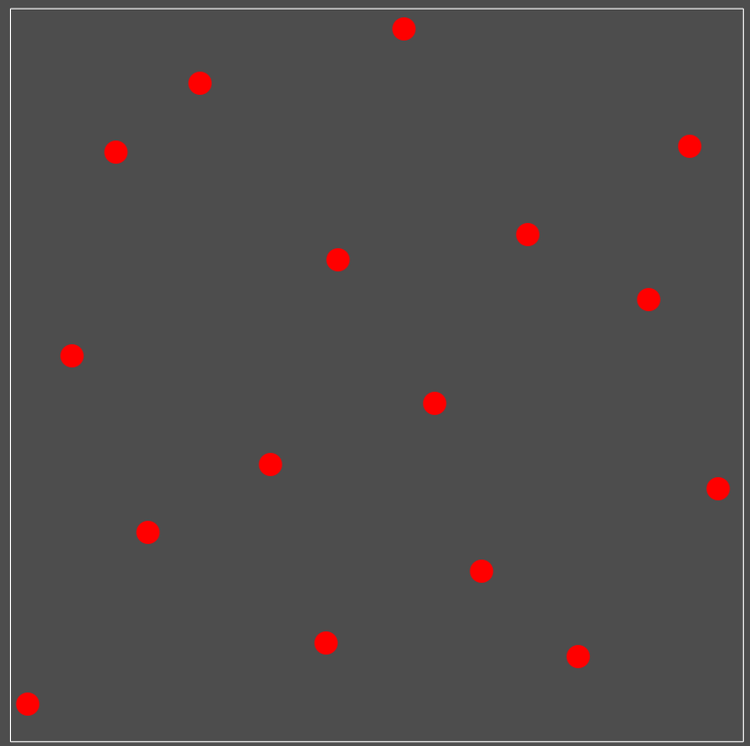
- Scrambling the component j:
  - start out with a random bit vector and save it in $X^{(j)}_{\sigma_j^{-1}(0)}$
  - permutation tree traversal by enumerating $i = 1, \ldots, 2^m - 1$
    - detect were tree ramifies: Number f of leading shared digits of $i - 1$ and i
    - XOR a bit vector with f leading zeros followed by a 1 filled by random bits
      $\equiv$ change the branch and choose new random permutations π
    - store result in $X^{(j)}_{\sigma_j^{-1}(i)}$

Implementation: Scrambled Hammersley Point Set

```
N = 1 << m;

Digits = get_32_random_bits();
P(0, 0) = (double) Digits / (double) 0x100000000L;

Digits2 = get_32_random_bits();
P(0, 1) = (double) Digits2 / (double) 0x100000000L;

for(i = 1; i < N; i++)
{
   Difference = (i - 1) ∧ i;

   for(Bits = 0; Difference; Bits++)
      Difference >>= 1;

   Shift = Log - Bits;

   Digits ∧= (0x80000000 | get_31_random_bits()) >> Shift;
   P(i, 0) = (double) Digits / (double) 0x100000000L;

   Digits2 ∧= (0x80000000 | get_31_random_bits()) >> Shift;
   P((int) ((double) N * Φ2(i)), 1) = (double) Digits2
            / (double) 0x100000000L;
}
```

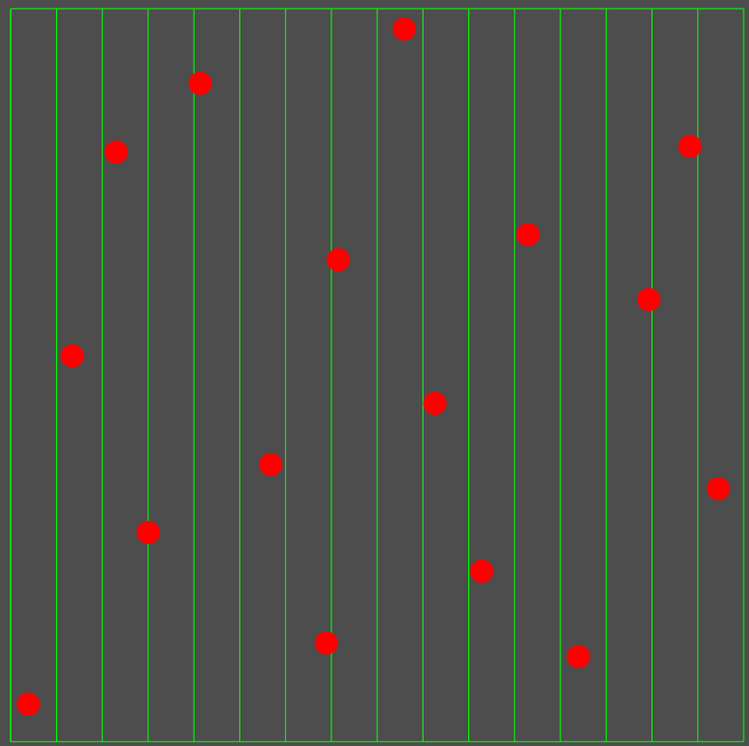
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



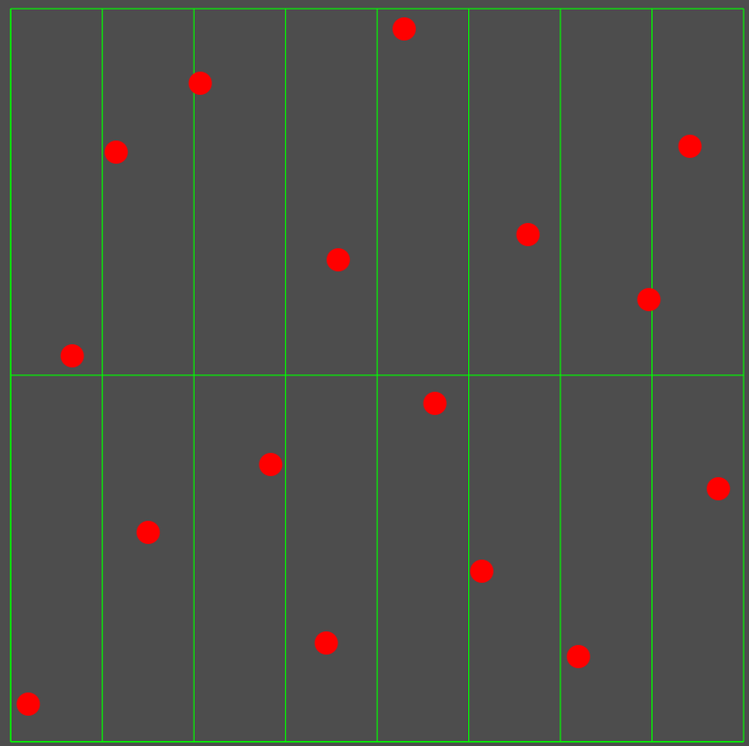
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



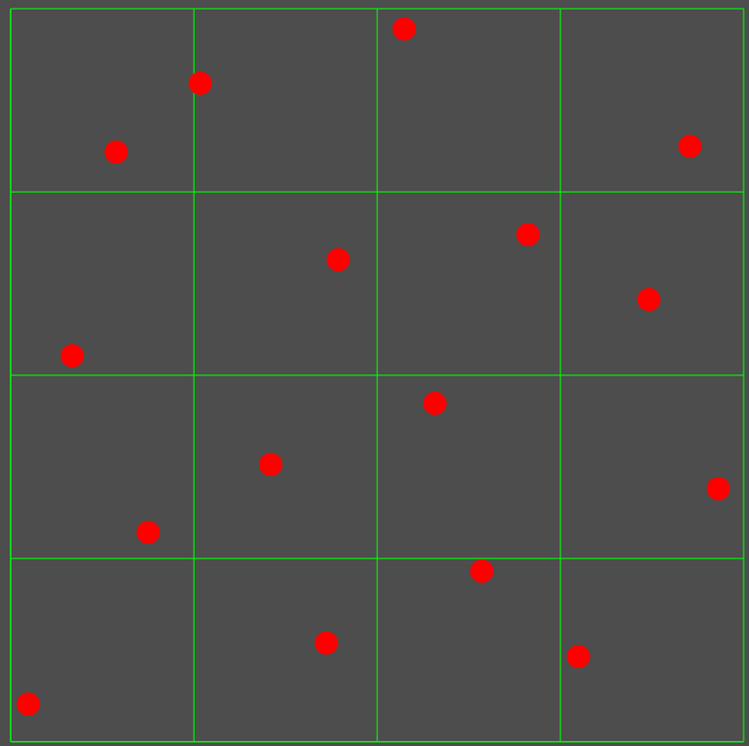
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



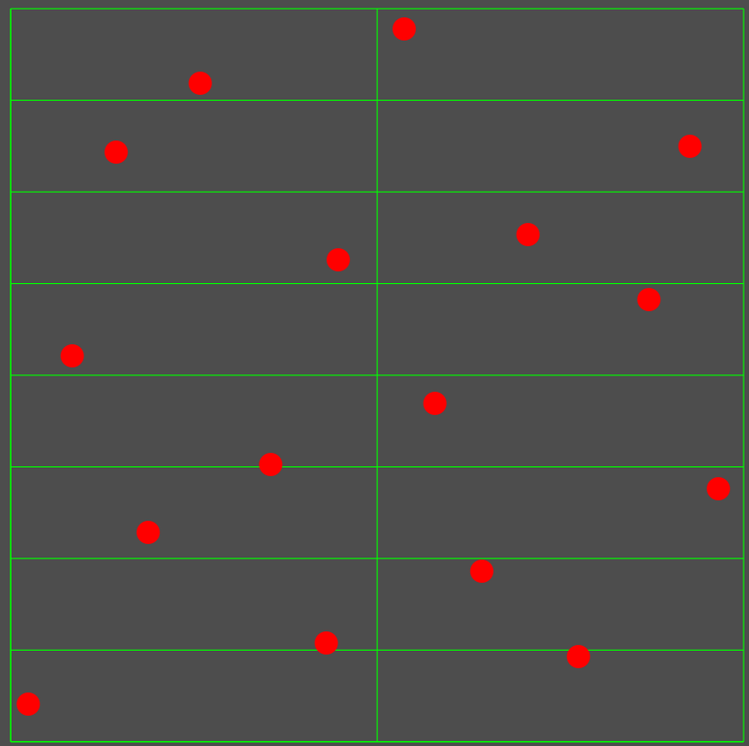
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



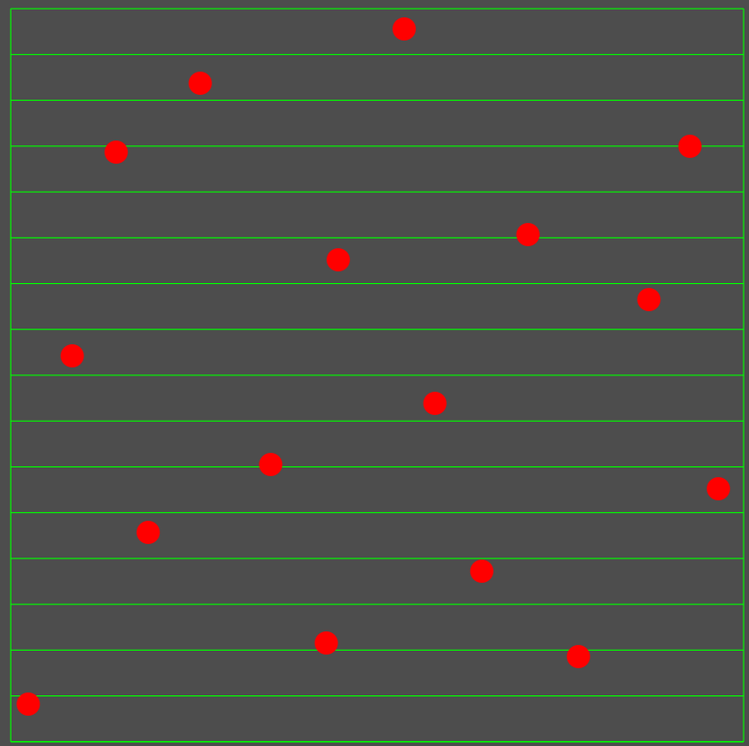
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



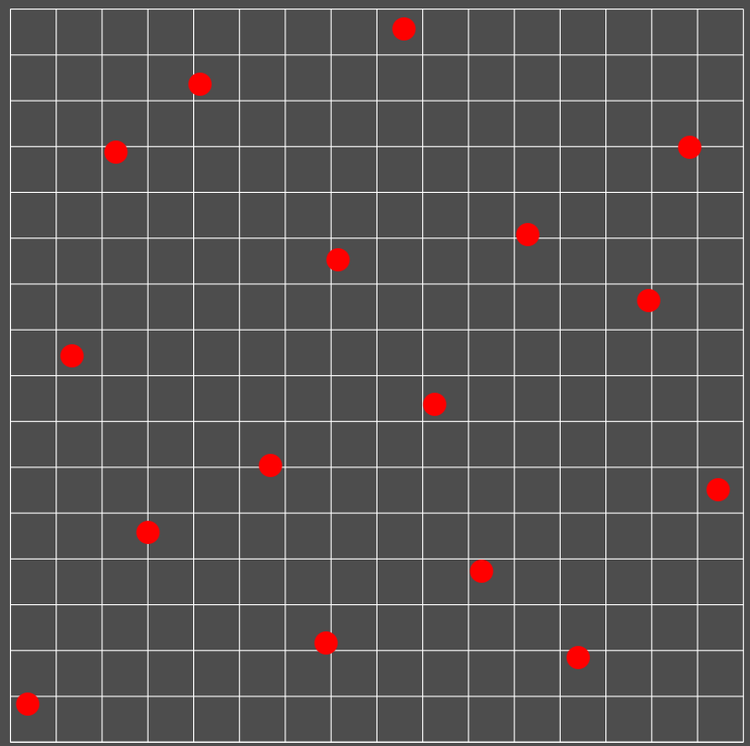
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



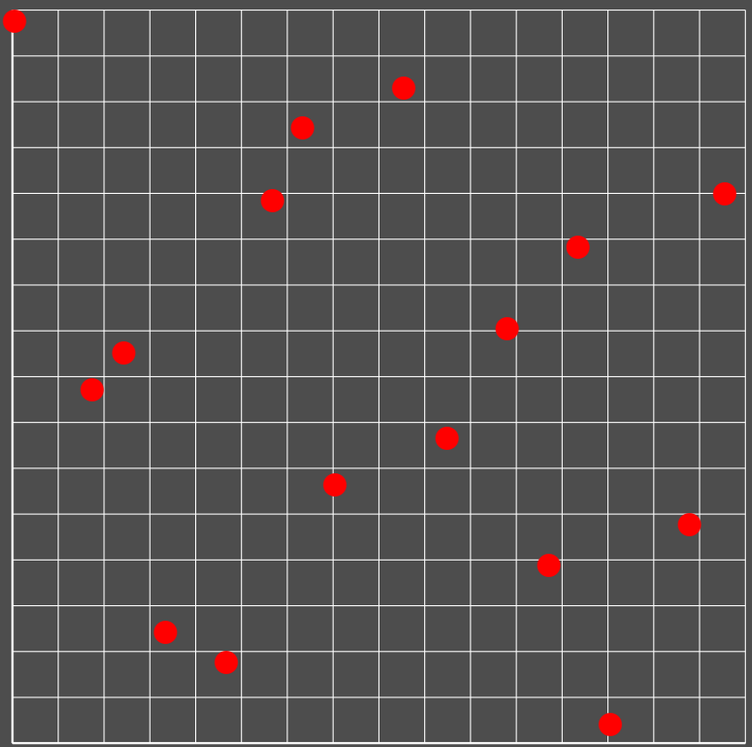
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- All instances are of low discrepancy
- Not all instances are equally good...



Another Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- All instances are of low discrepancy
- Not all instances are equally good...



Trajectory Splitting and Dependent Sampling

- Increase efficiency by splitting

$$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i, y_i) \approx \int_{I^{s_1}} \int_{I^{s_2}} f(x, y) dx dy$$

Trajectory Splitting and Dependent Sampling

- Increase efficiency by splitting

$$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i, y_i) \approx \int_{I^{s}1}\int_{I^{s}2} f(x,y)dxdy \approx \frac{1}{Ns}\sum_{i=0}^{N-1}\sum_{j=0}^{s-1} f(x_i, y_{i,j})$$

depending on the correlation coefficient of $f(\xi, \eta)$ and $f(\xi, \eta')$

Trajectory Splitting and Dependent Sampling

- Increase efficiency by splitting

$$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i, y_i) \approx \int_{I^{s_1}} \int_{I^{s_2}} f(x,y)dxdy \approx \frac{1}{Ns}\sum_{i=0}^{N-1}\sum_{j=0}^{s-1} f(x_i, y_{i,j})$$

  depending on the correlation coefficient of $f(\xi, \eta)$ and $f(\xi, \eta')$

- Exploit smoothness by correlated sampling

$$\sum_{j=1}^{M} \frac{1}{N_j} \sum_{i=0}^{N_j-1} f_j(x_{i,j}) \approx \sum_{j=1}^{M} \int_{I^s} f_j(x)dx$$

Trajectory Splitting and Dependent Sampling

- Increase efficiency by splitting

$$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i, y_i) \approx \int_{I^{s_1}} \int_{I^{s_2}} f(x,y)dxdy \approx \frac{1}{Ns}\sum_{i=0}^{N-1}\sum_{j=0}^{s-1} f(x_i, y_{i,j})$$

  depending on the correlation coefficient of $f(\xi, \eta)$ and $f(\xi, \eta')$

- Exploit smoothness by correlated sampling

$$\begin{aligned}\sum_{j=1}^{M} \frac{1}{N_j}\sum_{i=0}^{N_j-1} f_j(x_{i,j}) &\approx \sum_{j=1}^{M}\int_{I^s} f_j(x)dx \\ &= \int_{I^s}\sum_{j=1}^{M} f_j(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=1}^{M} f_j(x_i)\end{aligned}$$

  e.g. separation of the main part

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s2}$

$$\begin{array}{rcl} R_j : I^{s2} & \rightarrow & I^{s2} \\ y & \mapsto & (y + z_j) \bmod 1 \end{array} \quad \Rightarrow \quad \int_{I^{s2}} g(y)dy = \int_{I^{s2}} g(R_j(y))dy$$

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s_2}$

$$\begin{array}{rcl} R_j : I^{s_2} & \rightarrow & I^{s_2} \\ y & \mapsto & (y + z_j) \bmod 1 \end{array} \quad \Rightarrow \quad \int_{I^{s_2}} g(y)dy = \int_{I^{s_2}} g(R_j(y))dy$$

- Presmoothing of selected dimensions by replication

$$\int_{I^{s_1}} \int_{I^{s_2}} f(x,y)dydx \;=\; \int_{I^{s_1}} \int_{I^{s_2}} \frac{1}{M} \sum_{j=0}^{M-1} f(x, R_j(y))dydx$$

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s_2}$

$$\begin{array}{rcl} R_j : I^{s_2} & \rightarrow & I^{s_2} \\ y & \mapsto & (y + z_j) \bmod 1 \end{array} \quad \Rightarrow \quad \int_{I^{s_2}} g(y)dy = \int_{I^{s_2}} g(R_j(y))dy$$

- Presmoothing of selected dimensions by replication

$$\begin{aligned} \int_{I^{s_1}} \int_{I^{s_2}} f(x,y)dydx &= \int_{I^{s_1}} \int_{I^{s_2}} \frac{1}{M} \sum_{j=0}^{M-1} f(x, R_j(y))dydx \\ &\approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, R_j(y_i)) \\ &= \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i + z_j) \bmod 1) \end{aligned}$$

  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s_2}$

$$\begin{array}{rcl} R_j : I^{s_2} & \rightarrow & I^{s_2} \\ y & \mapsto & (y + z_j) \bmod 1 \end{array} \quad \Rightarrow \quad \int_{I^{s_2}} g(y)dy = \int_{I^{s_2}} g(R_j(y))dy$$

- Presmoothing of selected dimensions by replication

$$\begin{aligned} \int_{I^{s_1}} \int_{I^{s_2}} f(x,y)dydx &= \int_{I^{s_1}} \int_{I^{s_2}} \frac{1}{M} \sum_{j=0}^{M-1} f(x, R_j(y))dydx \\ &\approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, R_j(y_i)) \\ &= \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i + z_j) \bmod 1) \end{aligned}$$

  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$
  - local quadrature rule $P_{M,s_2} = (z_j)_{j=0}^{M-1}$

Trajectory Splitting by Dependent Sampling

- Integrals invariant under Cranley-Patterson rotation by $z_j \in I^{s_2}$

$$\begin{aligned} R_j : I^{s_2} &\to I^{s_2} \\ y &\mapsto (y + z_j) \bmod 1 \end{aligned} \quad \Rightarrow \quad \int_{I^{s_2}} g(y)dy = \int_{I^{s_2}} g(R_j(y))dy$$

- Presmoothing of selected dimensions by replication

$$\begin{aligned} \int_{I^{s_1}} \int_{I^{s_2}} f(x,y)dydx &= \int_{I^{s_1}} \int_{I^{s_2}} \frac{1}{M} \sum_{j=0}^{M-1} f(x, R_j(y))dydx \\ &\approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, R_j(y_i)) \\ &= \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i + z_j) \bmod 1) \end{aligned}$$

  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$
  - local quadrature rule $P_{M,s_2} = (z_j)_{j=0}^{M-1}$

$\Rightarrow$ Trajectories split by dependent sampling

Further Randomization Techniques

- Padding quasi-Monte Carlo points for high dimensions
  - by random numbers
  - by Latin hypercube samples

Further Randomization Techniques

- Padding quasi-Monte Carlo points for high dimensions
  - by random numbers
  - by Latin hypercube samples

- Jittered quasi-Monte Carlo point sets
  - Latin hypercube samples, however deterministic permutation

    Note: Rate of randomly permuted Latin hypercube samples does not apply !
  - e.g. $(0, m, 2)$-net with jitter of size b^{-m}

Further Randomization Techniques

- Padding quasi-Monte Carlo points for high dimensions
  - by random numbers
  - by Latin hypercube samples

- Jittered quasi-Monte Carlo point sets
  - Latin hypercube samples, however deterministic permutation

    Note: Rate of randomly permuted Latin hypercube samples does not apply !
  - e.g. $(0, m, 2)$-net with jitter of size b^{-m}

- Latin supercube sampling
  - biased
  - unbiased if used for decorrelating padded replications sampling

Summary

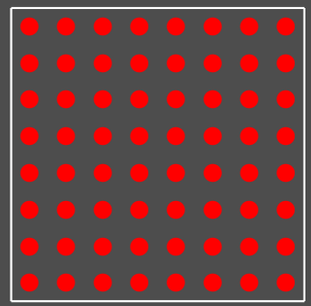
- Random field synthesis on good lattice points
- Randomized quasi-Monte Carlo integration
  - error estimate
  - L^2
  - almost as fast as pure quasi-Monte Carlo integration
  - concept of randomized replications
- Dependent splitting

Monte Carlo and Beyond

- Principles of rendering algorithms
- Monte Carlo integration
- Quasi-Monte Carlo points
- Quasi-Monte Carlo integration
- Monte Carlo extensions of quasi-Monte Carlo
- **Application to computer graphics: Discontinuous, high dimensional integrands**
  - Interleaved sampling
    - interleaved method of dependent tests
  - Volume rendering
    - dependent splitting by restricted Cranley-Patterson rotations
  - Bidirectional path tracing
    - padded replications sampling for cheap high-dimensional samples
  - Distribution ray tracing
    - strictly deterministic
    - dependent splitting by restricted Cranley-Patterson rotations

Sampling

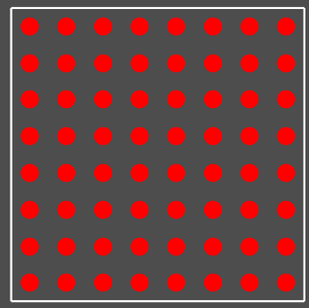
- Regular grids



+ fast rasterizers
- aliasing potential
- slow convergence

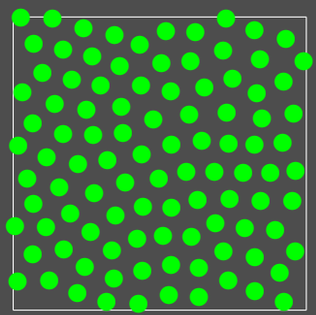
Sampling

- Regular grids



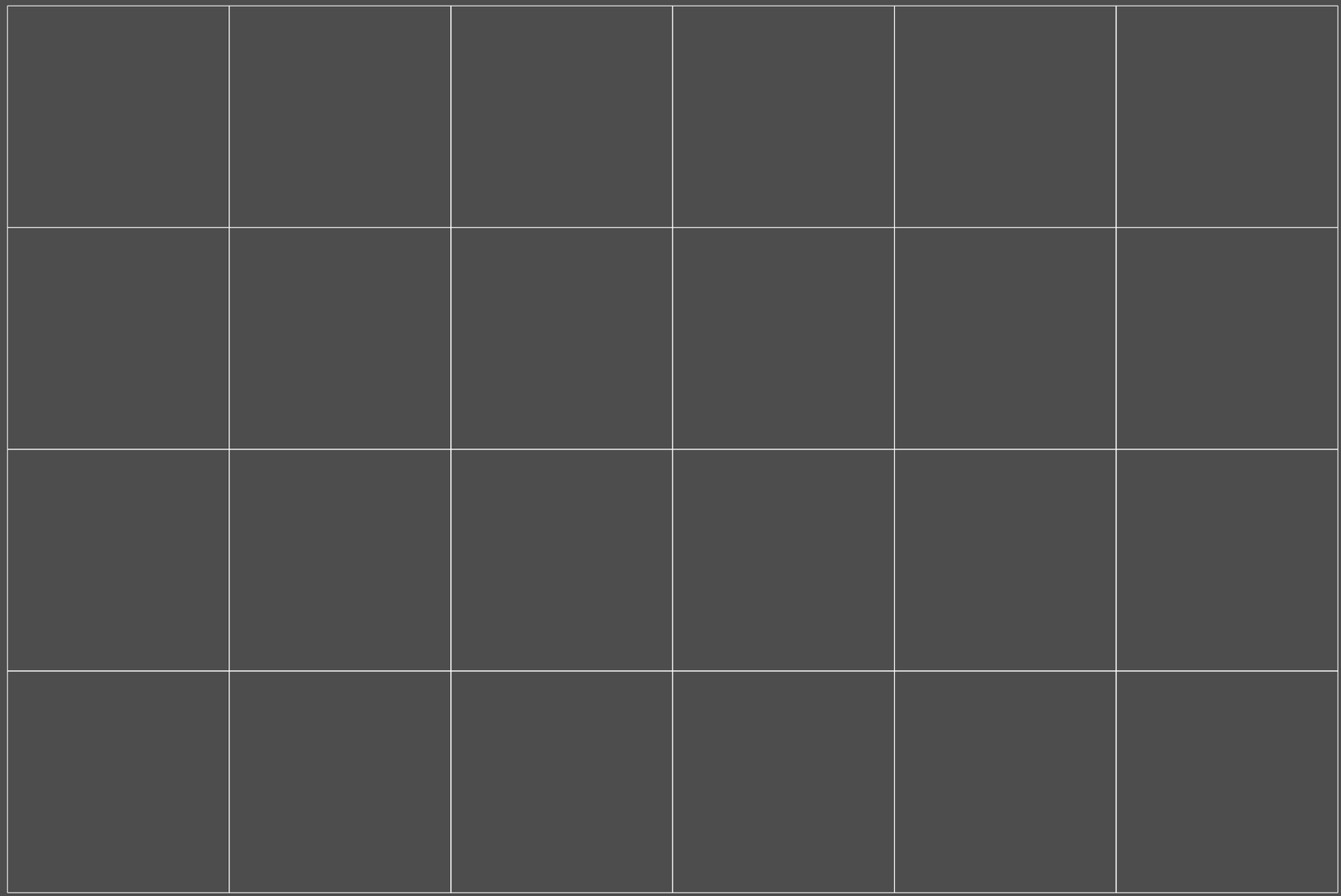
  + fast rasterizers
  - aliasing potential
  - slow convergence

- Irregular patterns

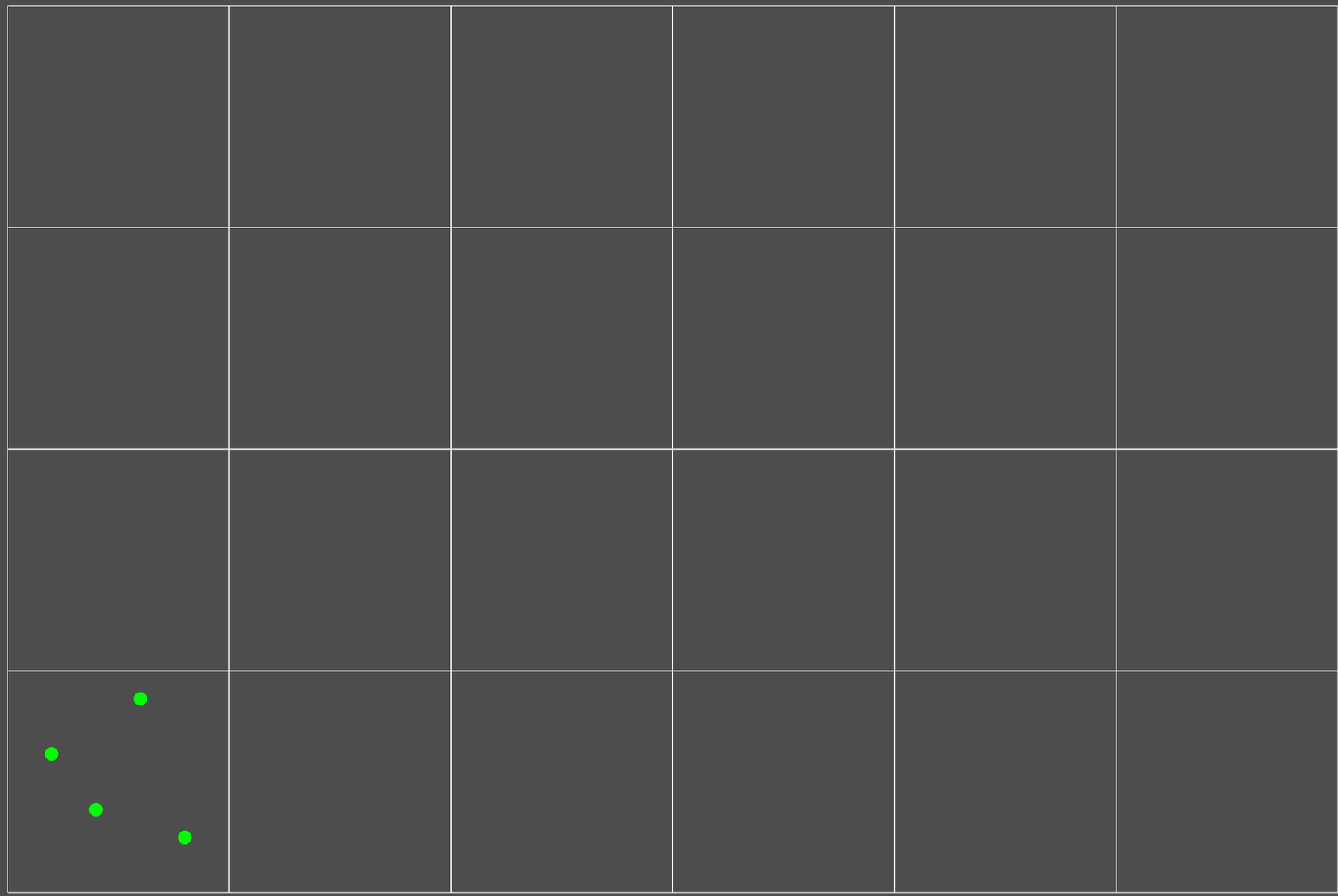


  - no rasterizers
  + optimal spectral properties
  + low discrepancy

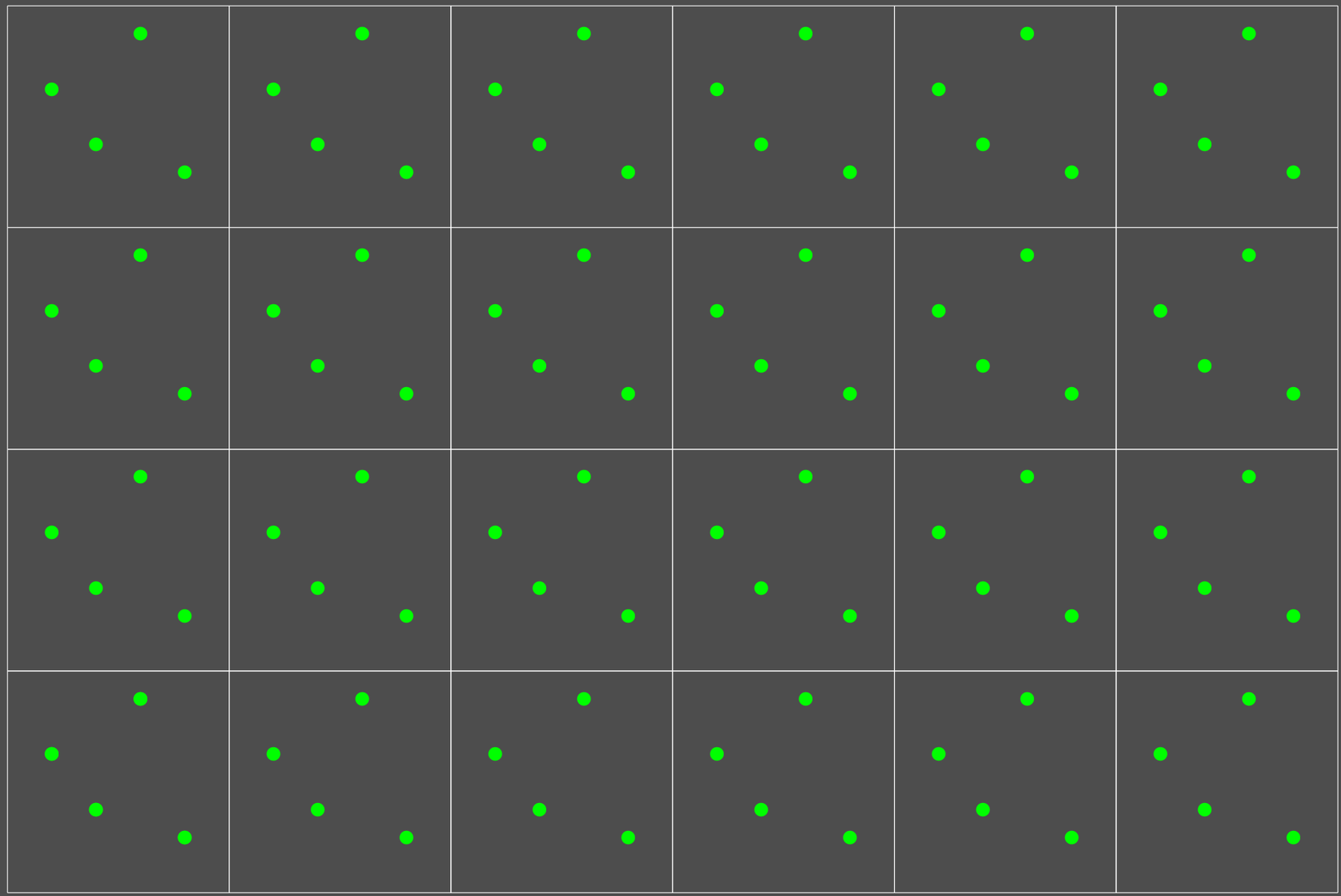
Accumulation Buffer



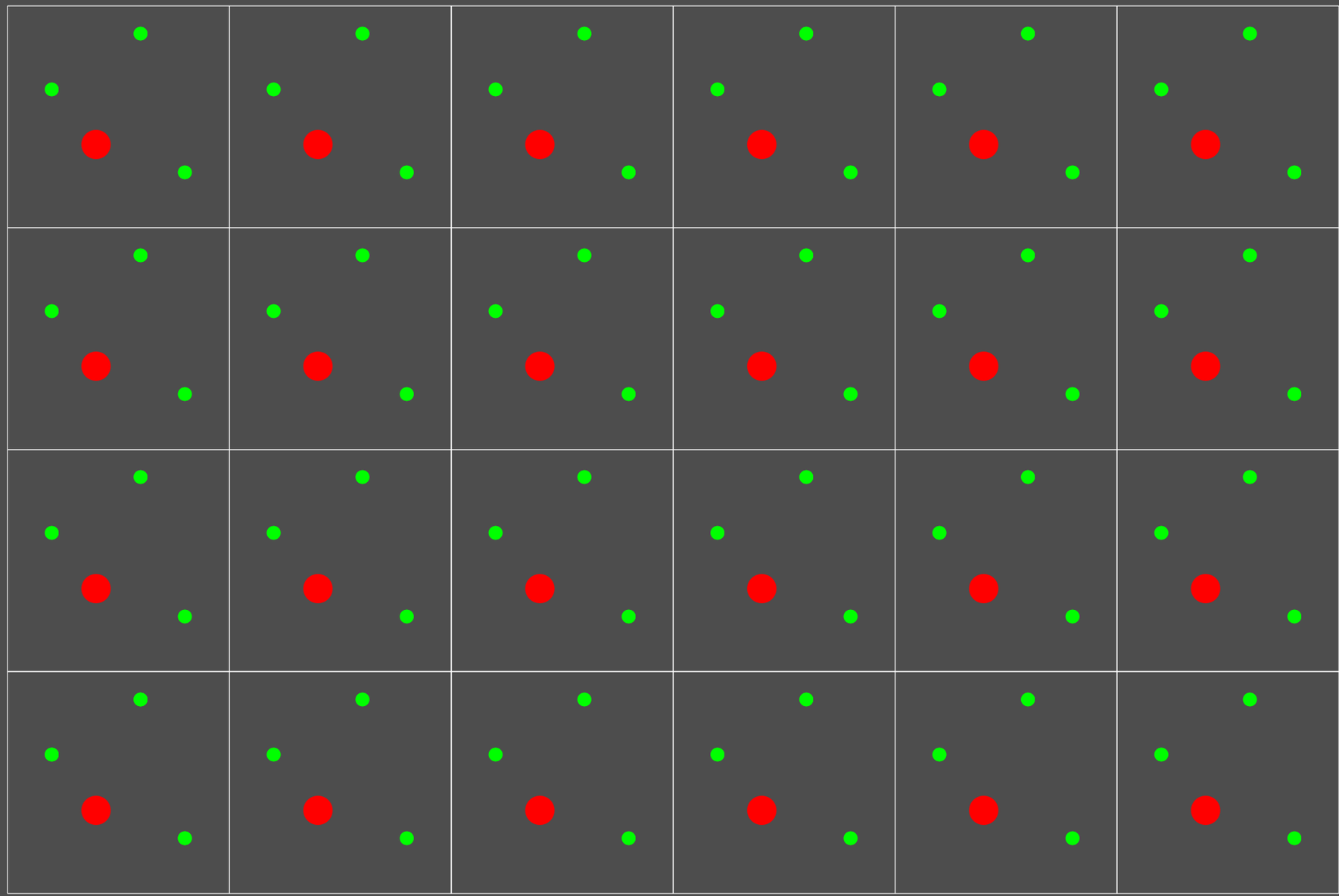
Accumulation Buffer



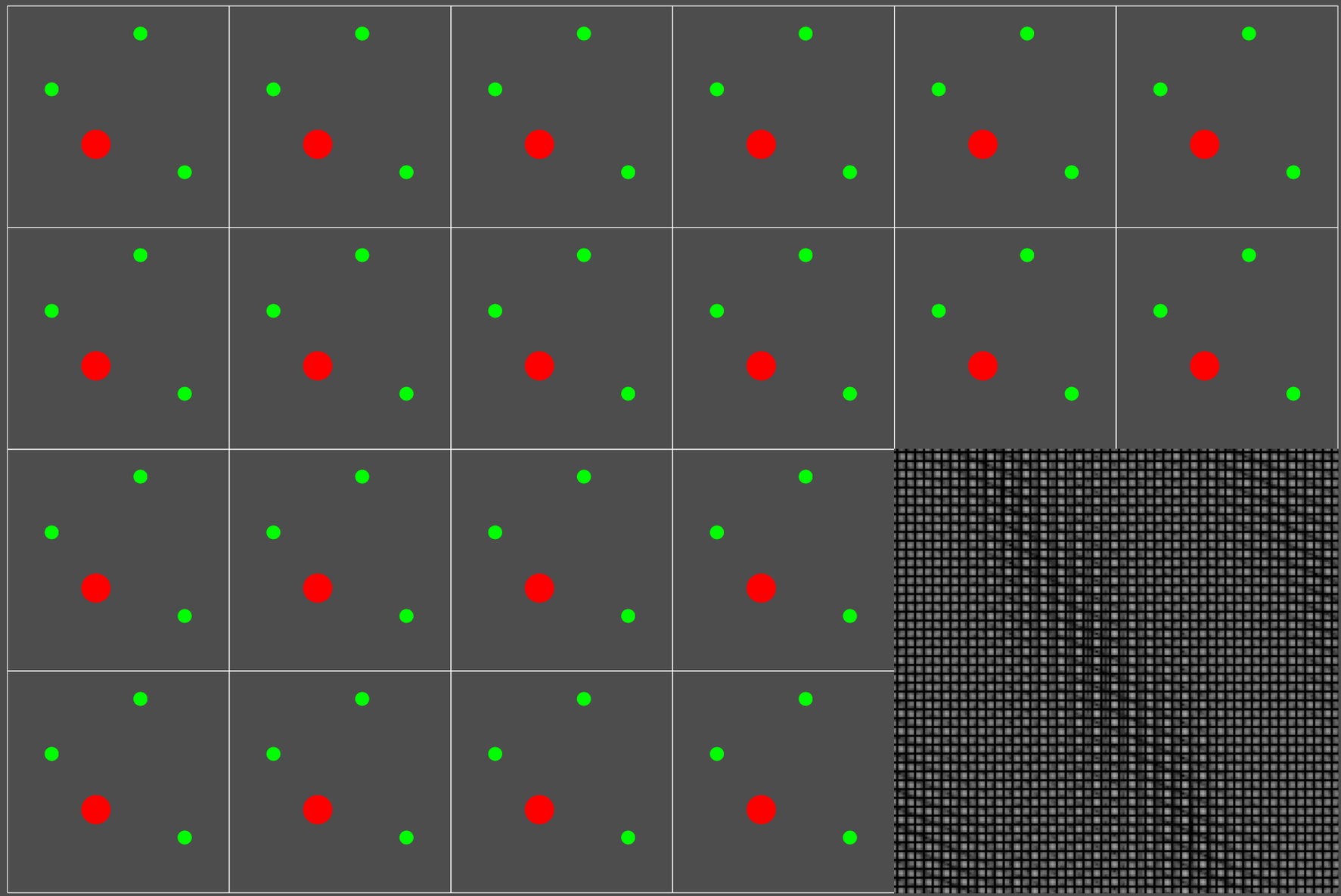
Accumulation Buffer



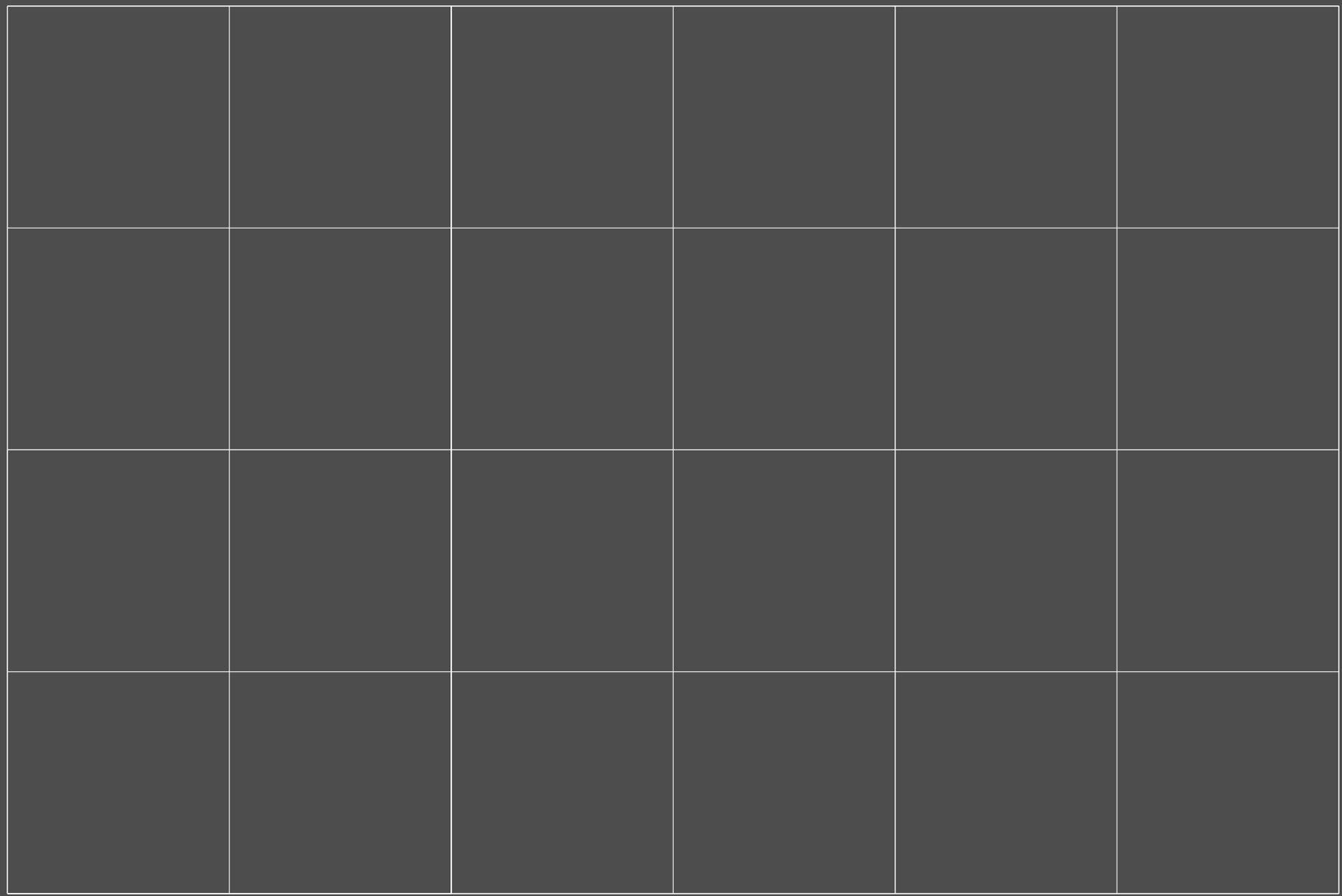
Accumulation Buffer



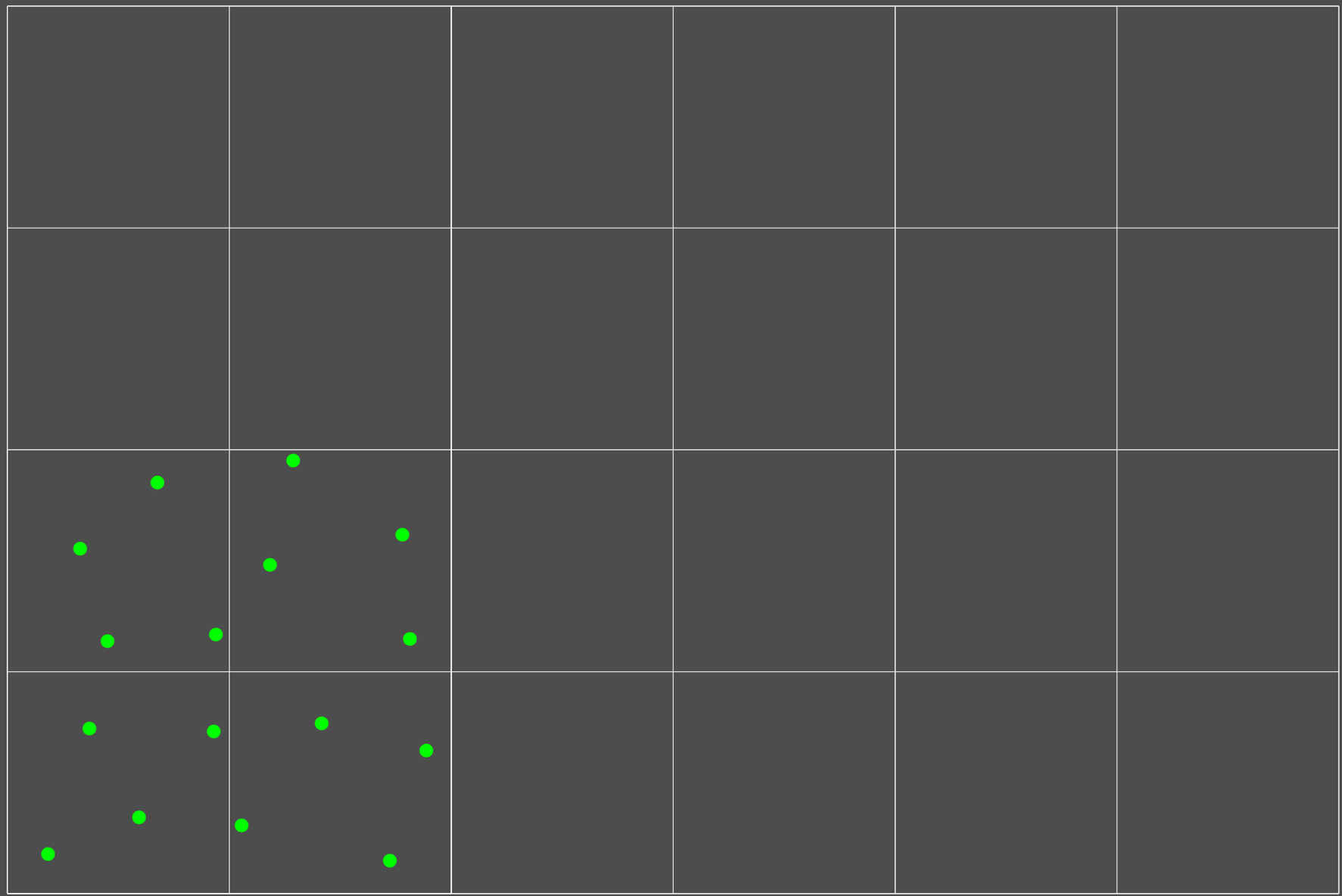
Accumulation Buffer



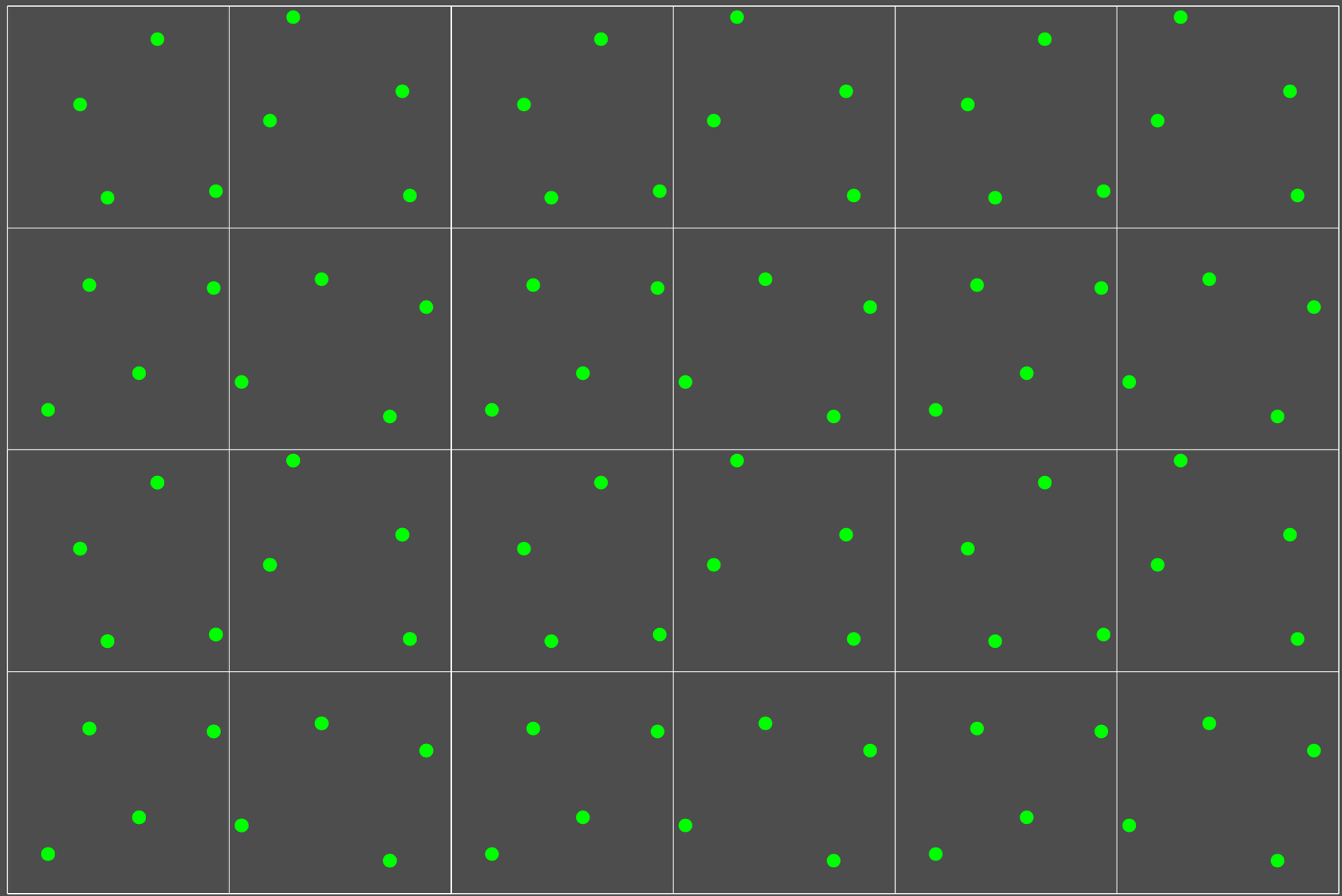
Interleaved Sampling



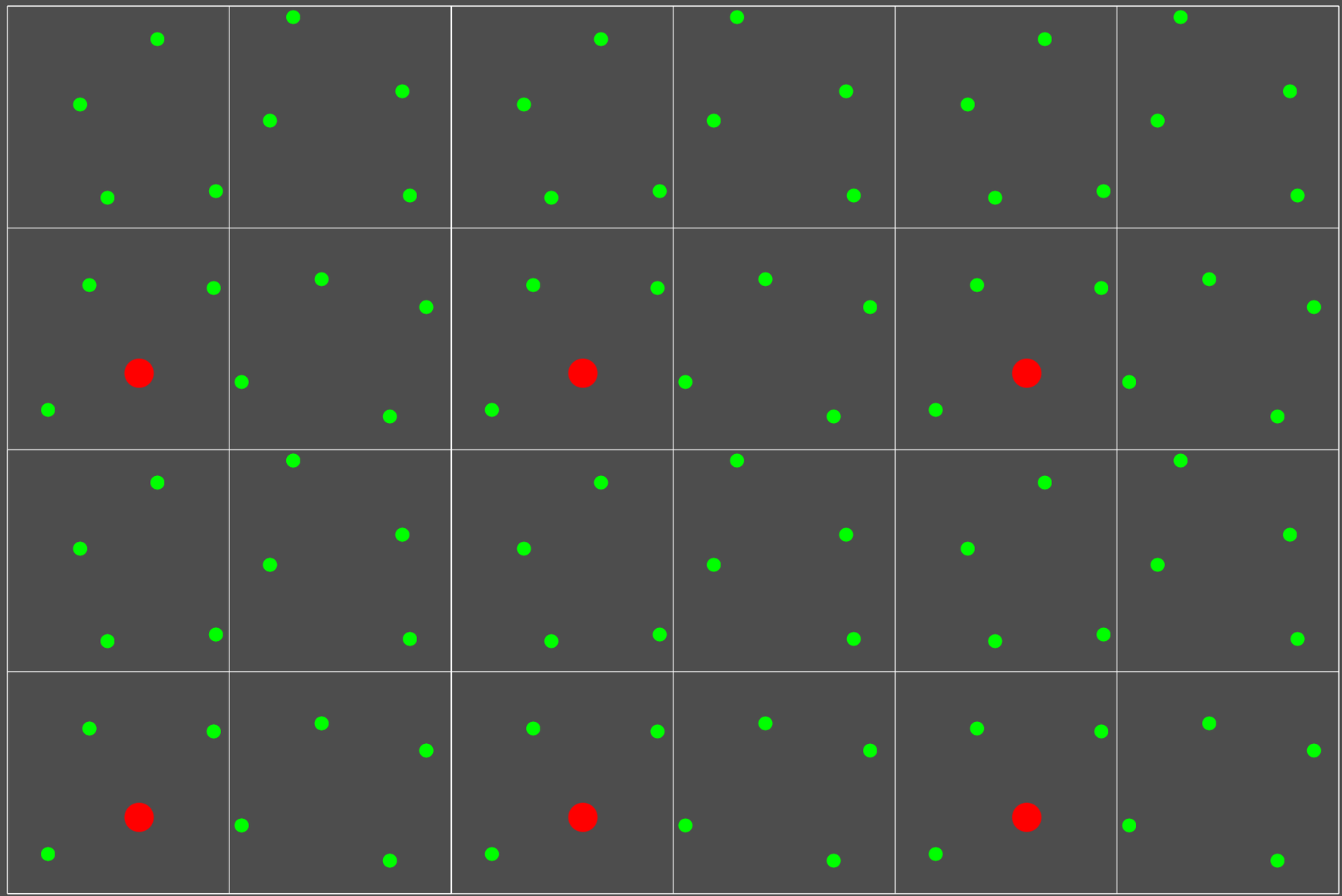
Interleaved Sampling



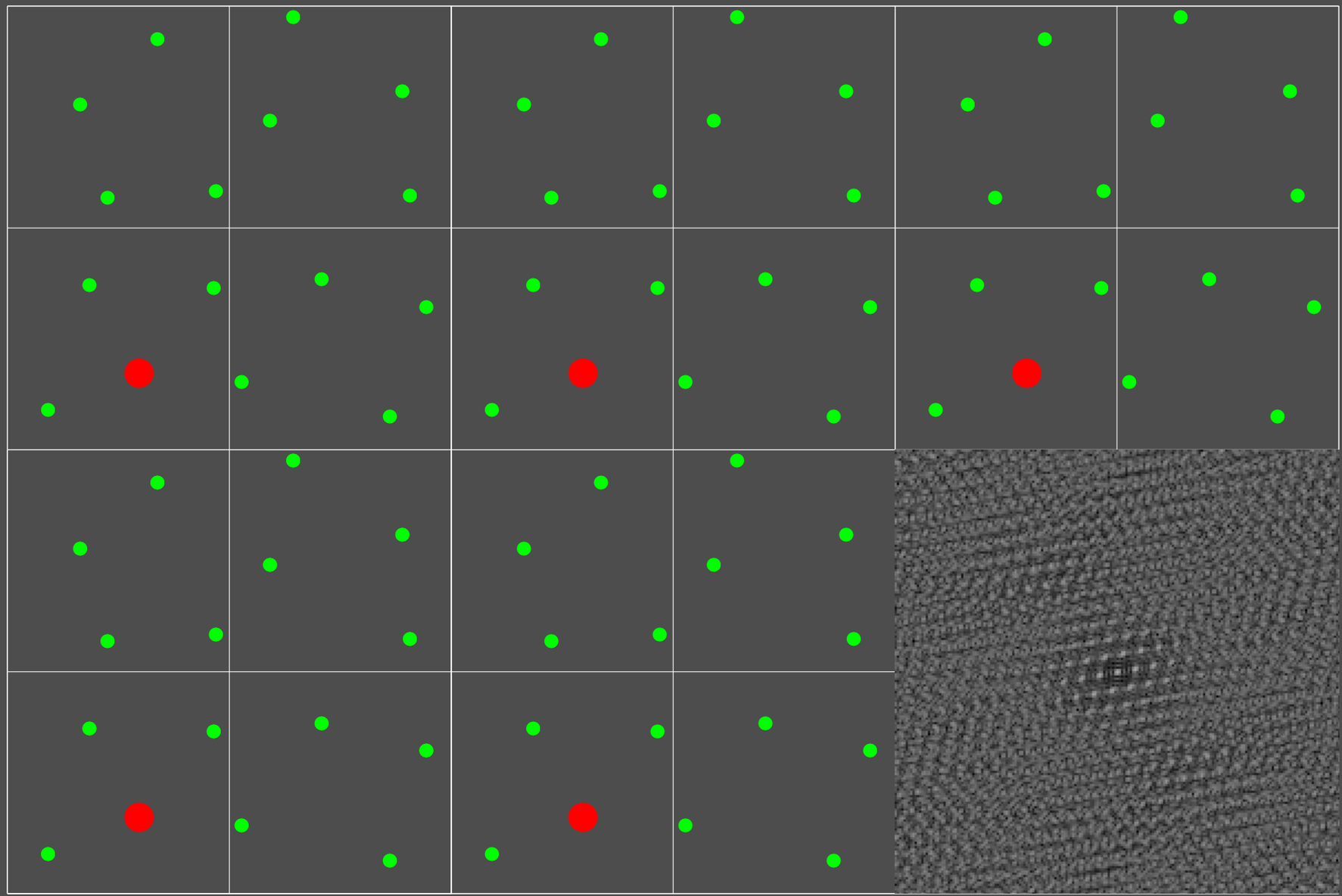
Interleaved Sampling



Interleaved Sampling



Interleaved Sampling



Consequences and Theoretical Considerations

- Aliasing by pattern repetition
  - spread out by larger-than-pixel-size patterns
  - arbitrary interleaving

Consequences and Theoretical Considerations

- Aliasing by pattern repetition
  - spread out by larger-than-pixel-size patterns
  - arbitrary interleaving
- Method of dependent tests (parametric Monte Carlo integration)
  - Accumulation buffer

$$g_P(y) = \int_{[0,1)^s} f(x, y)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} f(x_i, y)$$

Consequences and Theoretical Considerations

- Aliasing by pattern repetition
  - spread out by larger-than-pixel-size patterns
  - arbitrary interleaving
- Method of dependent tests (parametric Monte Carlo integration)
  - Accumulation buffer

$$g_P(y) = \int_{[0,1)^s} f(x, y)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} f(x_i, y)$$

  - Interleaved sampling

$$g_P(y) = \int_{[0,1)^s} \chi_P(x) f'(x, y)dx \approx \frac{1}{N'} \sum_{i=0}^{N'-1} \chi_P(x'_i) f'(x'_i, y)$$

Consequences and Theoretical Considerations

- Aliasing by pattern repetition
  - spread out by larger-than-pixel-size patterns
  - arbitrary interleaving
- Method of dependent tests (parametric Monte Carlo integration)
  - Accumulation buffer

$$g_P(y) = \int_{[0,1)^s} f(x, y)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} f(x_i, y)$$

  - Interleaved sampling

$$g_P(y) = \int_{[0,1)^s} \chi_P(x) f'(x, y)dx \approx \frac{1}{N'} \sum_{i=0}^{N'-1} \chi_P(x'_i) f'(x'_i, y)$$

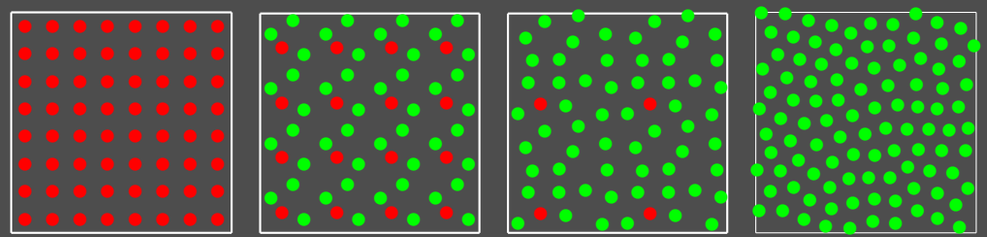
- Exploit intrinsic high coherence
  - new hardware
  - new software parallelization paradigm

Sampling Patterns for Interleaved Sampling

- Precomputed Max-Lloyd relaxation points as basis pattern
  - periodically tile seamlessly
  - blue noise spectral characteristics (minimum distance property)
  - low discrepancy (correlated)
  - for arbitrary problem dimension

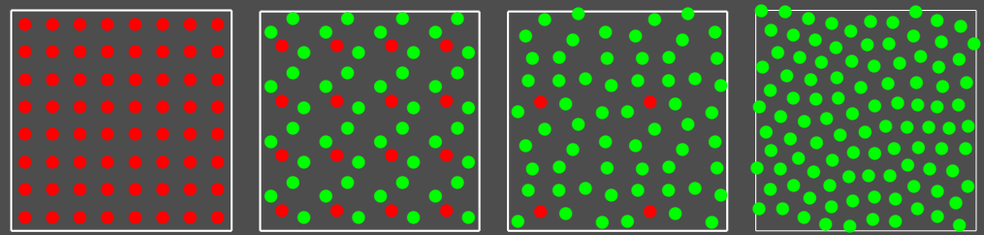
Sampling Patterns for Interleaved Sampling

- Precomputed Max-Lloyd relaxation points as basis pattern
  - periodically tile seamlessly
  - blue noise spectral characteristics (minimum distance property)
  - low discrepancy (correlated)
  - for arbitrary problem dimension
- Size N' of irregular basis pattern
  - blend between regular and irregular sampling

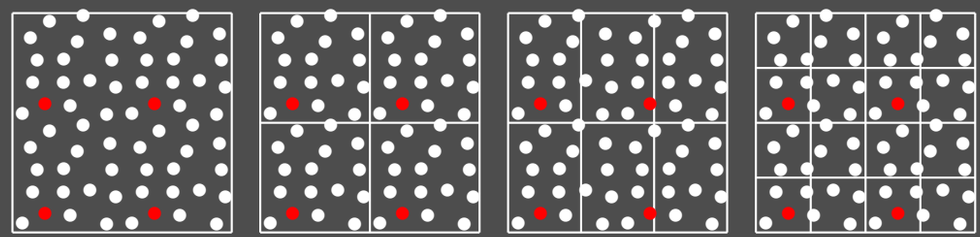


Sampling Patterns for Interleaved Sampling

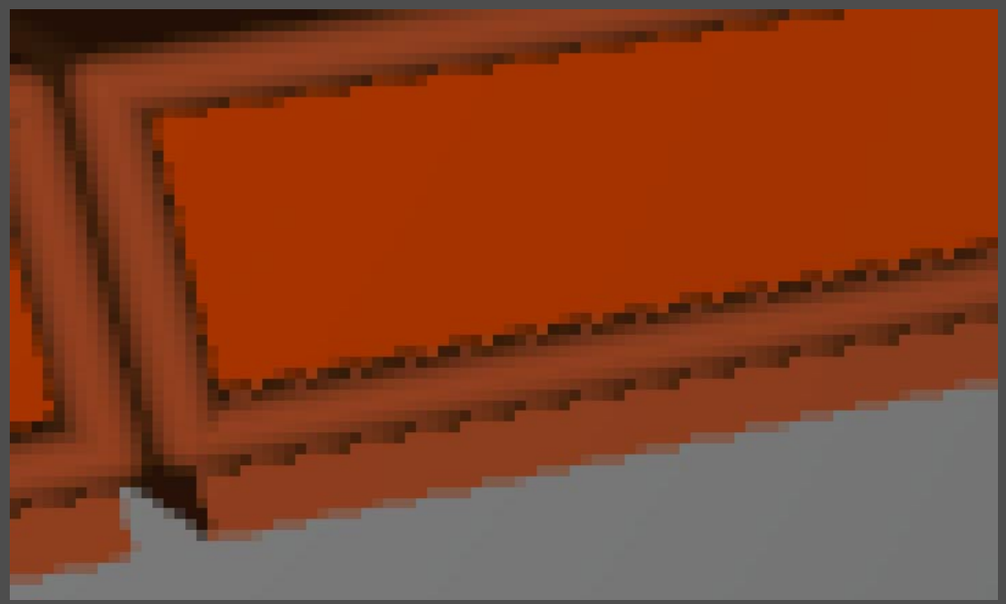
- Precomputed Max-Lloyd relaxation points as basis pattern
  - periodically tile seamlessly
  - blue noise spectral characteristics (minimum distance property)
  - low discrepancy (correlated)
  - for arbitrary problem dimension
- Size N' of irregular basis pattern
  - blend between regular and irregular sampling



- Choice of interleaving ratio by χ_P
  - spread out aliasing artifacts



Application: Antialiasing

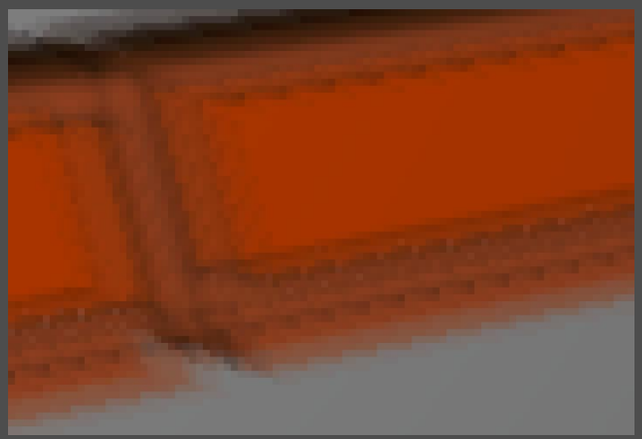


Accumulation Buffer

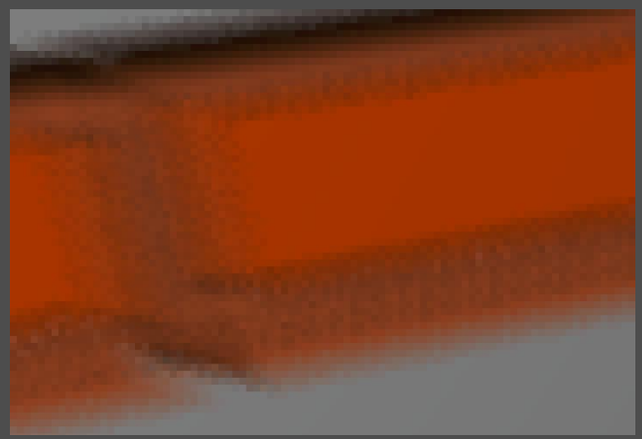
Interleaved Sampling

- Reduced aliasing at only 4 samples per pixel
  - artifacts spread out
  - artifacts from repetition, **not** from deterministic sampling
- Simple to implement by hardware (current and future)

Application: Motion Blur



Accumulation Buffer

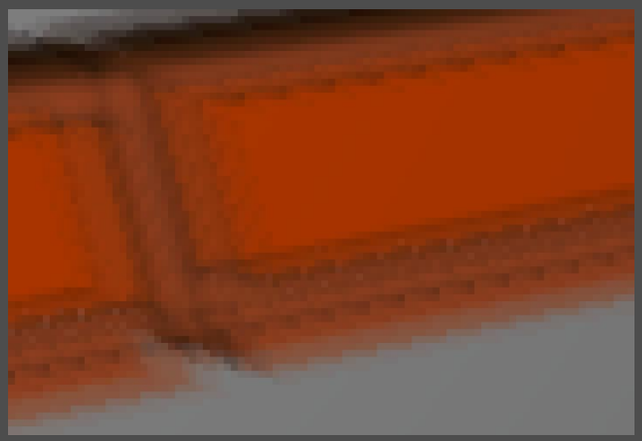


Interleaved Sampling

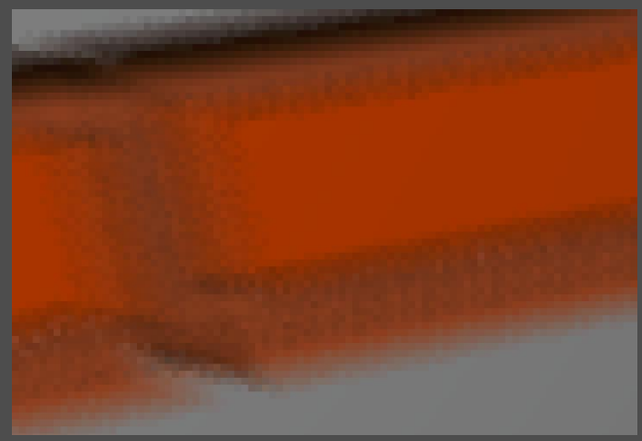
Uncorrelated Sampling

- Artifacts replaced by noise at 16 samples per pixel

Application: Motion Blur



Accumulation Buffer



Interleaved Sampling

Uncorrelated Sampling

- Artifacts replaced by noise at 16 samples per pixel
- Exactly one moment in time for each subimage
  - finite number of time samples and consequently instances of the scene
  - finally correct implementations of REYES/RenderMan and the photon map
  - Sobol' $(0, m, 3)$-net optimally can replace stratified random sampling

Other Applications

- All accumulation buffer techniques
  - weighted sampling
  - extended light source and the N-shadow problem
  - deep shadow maps
  - global illumination by instant radiosity
- CCD chip design
  - high dynamic range capturing

"One-Dimensional" Integration in Computer Graphics

- Linear light sources, spectral effects, volumetric effects
- The (example) problem

$$\int_{I^3} f(x, y, z) dx dy dz$$

  - x, y for ray from the eye through a point in the pixel
  - z for integrating the density f along the ray

"One-Dimensional" Integration in Computer Graphics

- Linear light sources, spectral effects, volumetric effects
- The (example) problem

$$\int_{I^3} f(x, y, z) dx dy dz$$

  - x, y for ray from the eye through a point in the pixel
  - z for integrating the density f along the ray
- The z-component requires presmoothing

"One-Dimensional" Integration in Computer Graphics

- Linear light sources, spectral effects, volumetric effects
- The (example) problem

$$\int_{I^3} f(x,y,z)dxdydz$$

  - x, y for ray from the eye through a point in the pixel
  - z for integrating the density f along the ray
- The z-component requires presmoothing
- **Bad:** Using one-dimensional stratified Monte Carlo for

$$g(x,y) = \int_I f(x,y,z)dz$$

  - uncorrelated ray marching: Fur, photon map with participating media, ...

"One-Dimensional" Integration in Computer Graphics

- Linear light sources, spectral effects, volumetric effects
- The (example) problem

$$\int_{I^3} f(x,y,z)dxdydz$$

  - x, y for ray from the eye through a point in the pixel
  - z for integrating the density f along the ray
- The z-component requires presmoothing
- **Bad:** Using one-dimensional stratified Monte Carlo for

$$g(x,y) = \int_I f(x,y,z)dz$$

  - uncorrelated ray marching: Fur, photon map with participating media, ...
- **Good:** **Use dependent splitting**, e.g. by restricted Cranley-Patterson rotations

$$\int_{I^3} \frac{1}{M} \sum_{k=0}^{M-1} f(x,y,R_k(z))dxdydz$$

  - correlated ray marching: Less random numbers and faster convergence

Remember...

- Discontinuous integrand mainly in x, y

Remember...

- Discontinuous integrand mainly in x, y

- For $s = 1$
  - lattices and $(0, m, 1)$-nets become identical, in fact the rectangle rule
  - the best discrepancy is $D^*(P_N) \geq \frac{1}{N}$
    - obtained by equidistant set of samples (correlated)
    - (stratified) random sampling $D^*(P_N) \in \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$ (uncorrelated)

Remember...

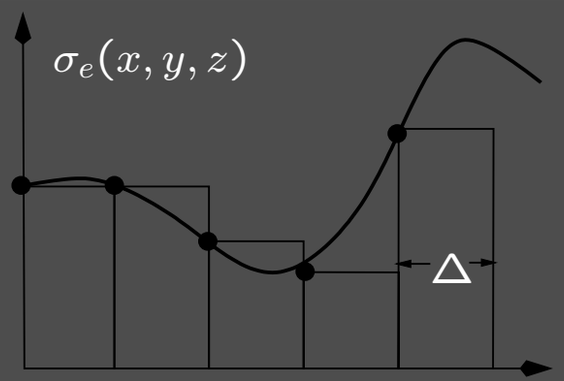
- Discontinuous integrand mainly in x, y

- For $s = 1$
  - lattices and $(0, m, 1)$-nets become identical, in fact the rectangle rule
  - the best discrepancy is $D^*(P_N) \geq \frac{1}{N}$
    - obtained by equidistant set of samples (correlated)
    - (stratified) random sampling $D^*(P_N) \in \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$ (uncorrelated)

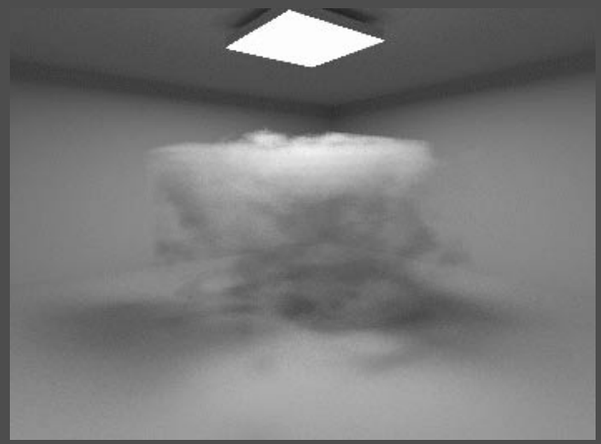
$\Rightarrow$ **Never use one-dimensional stratified Monte Carlo !!!**

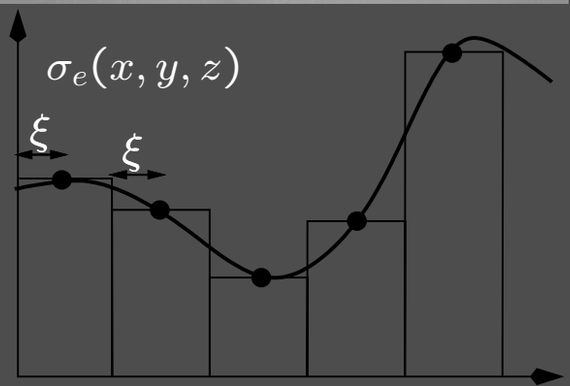
$\Rightarrow$ **Use randomized quasi-Monte Carlo instead**

Volume Rendering

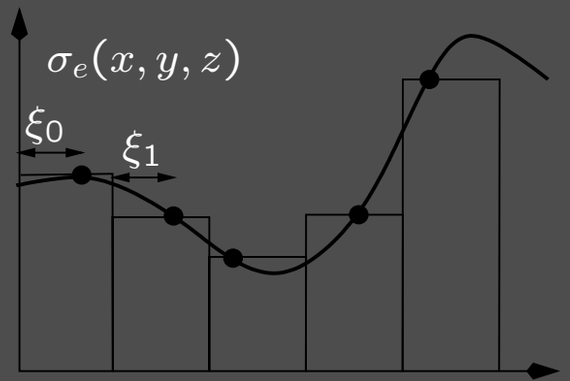


equidistant



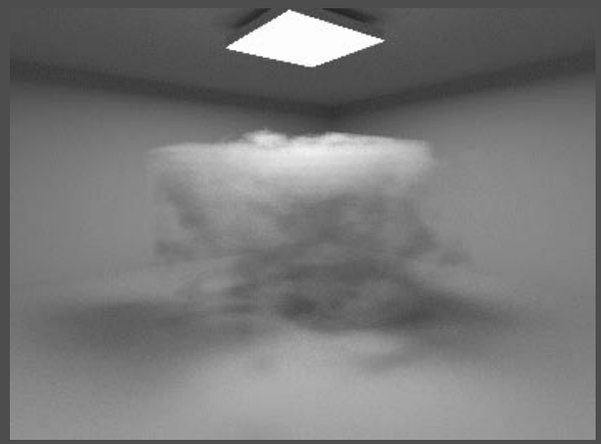


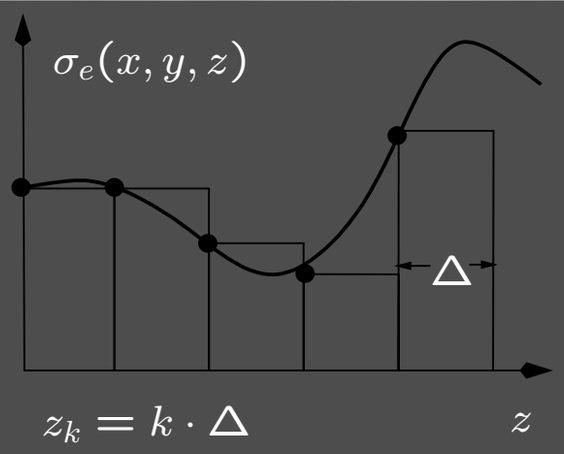
dependent



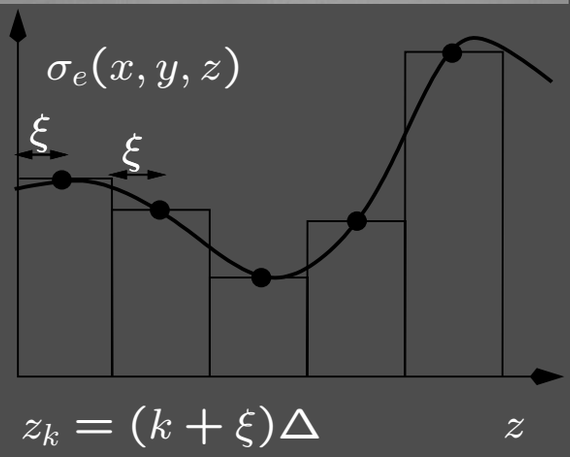
stratified

Volume Rendering

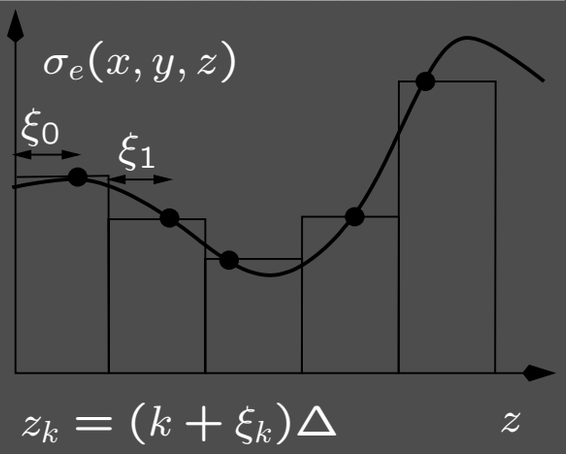




equidistant



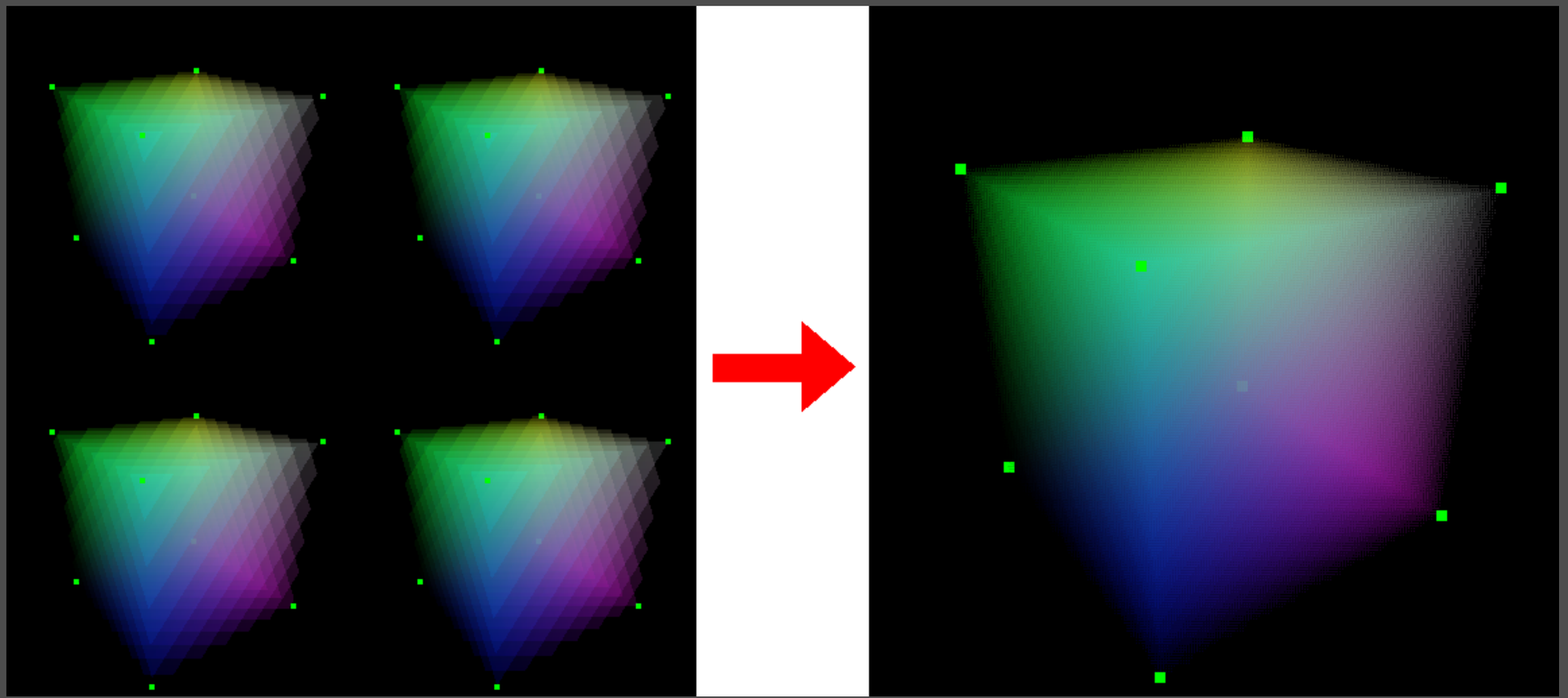
dependent



stratified

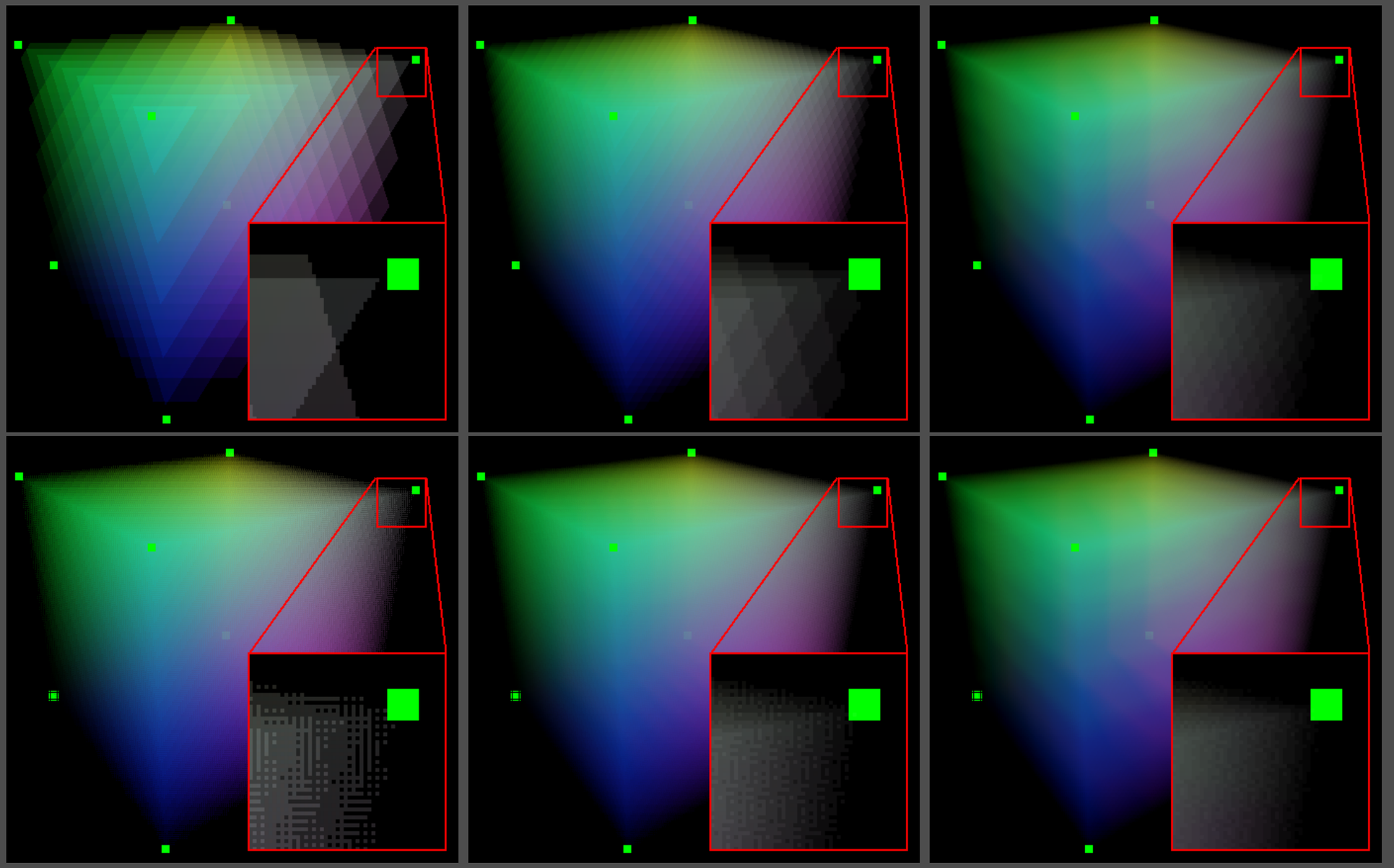
- Dependent sampling saves ∼20% rendering time wasted for random numbers
- Equidistant, i.e. correlated, samples have lower discrepancy that stratified samples
- Combine with interleaved sampling: Coherent ray marching

Application: Volume Rendering

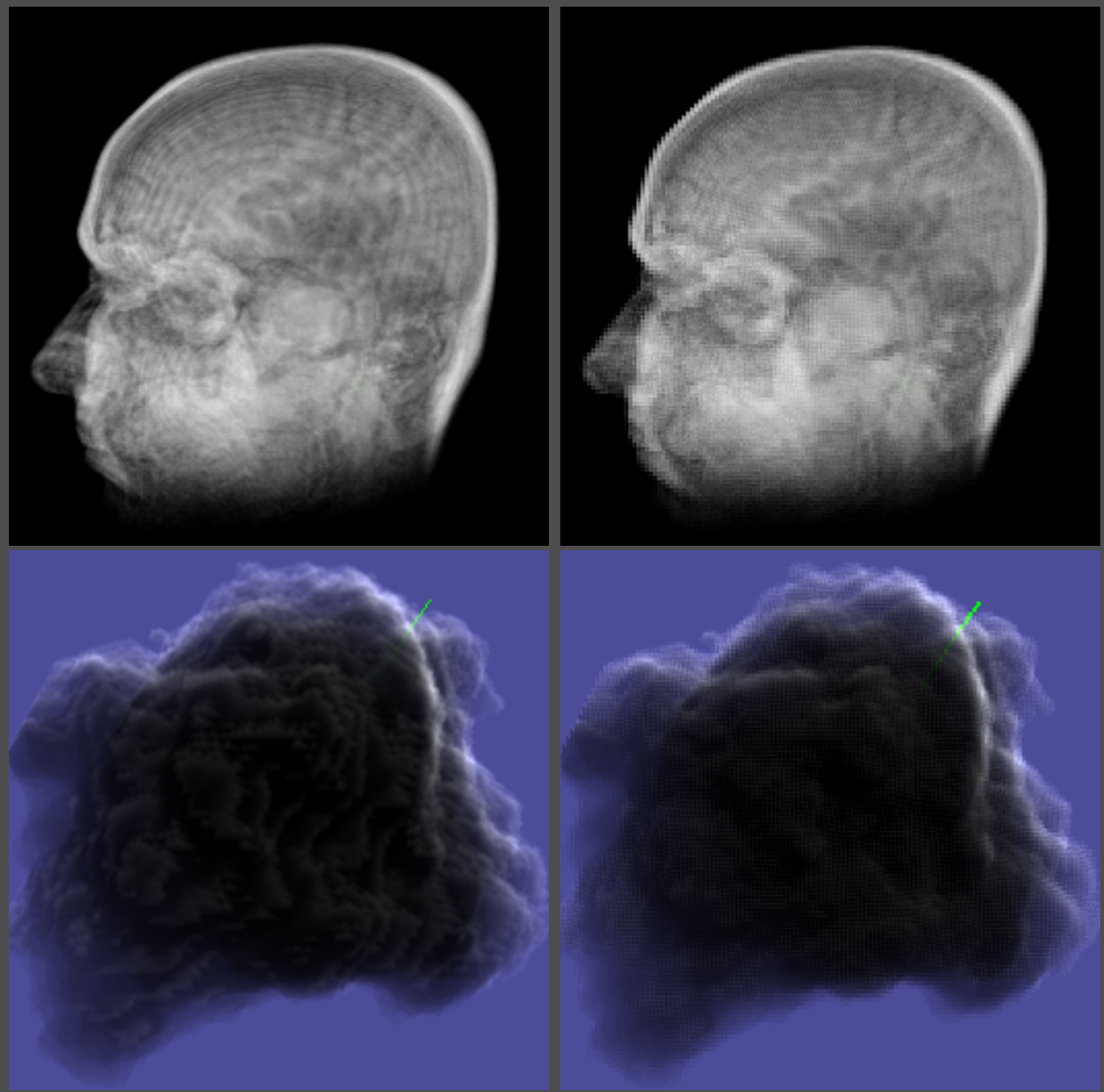


- Much improved depth antialiasing (unbiased)
- Simply interleaving images
  - **coherent** ray marching

Application: Volume Rendering



Application: Volume Rendering



Application: Volume Rendering

The Global Illumination Problem

- Three-point form of the light transport equation

$$L(y \rightarrow z) = L_e(y \rightarrow z) + \int_S L(x \rightarrow y) f_s(x \rightarrow y \rightarrow z) G(x \leftrightarrow y) dA(x)$$

The Global Illumination Problem

- Three-point form of the light transport equation

$$L(y \rightarrow z) = L_e(y \rightarrow z) + \int_S L(x \rightarrow y) f_s(x \rightarrow y \rightarrow z) G(x \leftrightarrow y) dA(x)$$

  Measurement equation

$$I_j = \int_{S \times S} W_e^{(j)}(x \rightarrow y) L(x \rightarrow y) G(x \leftrightarrow y) dA(y) dA(x)$$

The Global Illumination Problem

- Three-point form of the light transport equation

$$L(y \rightarrow z) = L_e(y \rightarrow z) + \int_S L(x \rightarrow y) f_s(x \rightarrow y \rightarrow z) G(x \leftrightarrow y) dA(x)$$

Measurement equation

$$I_j = \int_{S \times S} W_e^{(j)}(x \rightarrow y) L(x \rightarrow y) G(x \leftrightarrow y) dA(y) dA(x)$$

$\Rightarrow$ Path integral formulation

$$I_j = \sum_{k=1}^{\infty} \int_{\mathcal{P}_k} f_j(\bar{x}) d\mu_k(\bar{x}) = \int_{\mathcal{P}} f_j(\bar{x}) d\mu(\bar{x})$$

The Global Illumination Problem

- Three-point form of the light transport equation

$$L(y \rightarrow z) = L_e(y \rightarrow z) + \int_S L(x \rightarrow y) f_s(x \rightarrow y \rightarrow z) G(x \leftrightarrow y) dA(x)$$

  Measurement equation

$$I_j = \int_{S \times S} W_e^{(j)}(x \rightarrow y) L(x \rightarrow y) G(x \leftrightarrow y) dA(y) dA(x)$$

  $\Rightarrow$ Path integral formulation

$$I_j = \sum_{k=1}^{\infty} \int_{\mathcal{P}_k} f_j(\bar{x}) d\mu_k(\bar{x}) = \int_{\mathcal{P}} f_j(\bar{x}) d\mu(\bar{x})$$

- Bidirectional path tracing
  - Multiple importance sampling for quasi-Monte Carlo integration
  - How much is sacrificed by randomized quasi-Monte Carlo integration ?
  - Adapt to two-dimensional structure of integral equation
    - padded replications sampling

Path Integral Formulation

- Path space and path measure

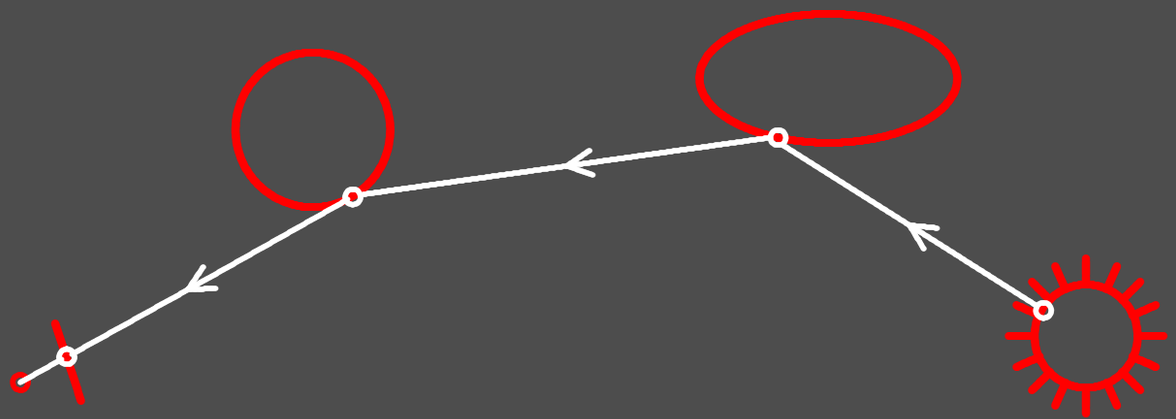
$$\mathcal{P}_k = \{\bar{x} = x_0 x_1 \dots x_k \mid x_i \in S\} \qquad d\mu_k(\bar{x}) = \prod_{i=0}^{k} dA(x_i)$$

Path Integral Formulation

- Path space and path measure

$$\mathcal{P}_k = \{\bar{x} = x_0 x_1 \dots x_k \mid x_i \in S\} \qquad d\mu_k(\bar{x}) = \prod_{i=0}^{k} dA(x_i)$$

- Measurement contribution function

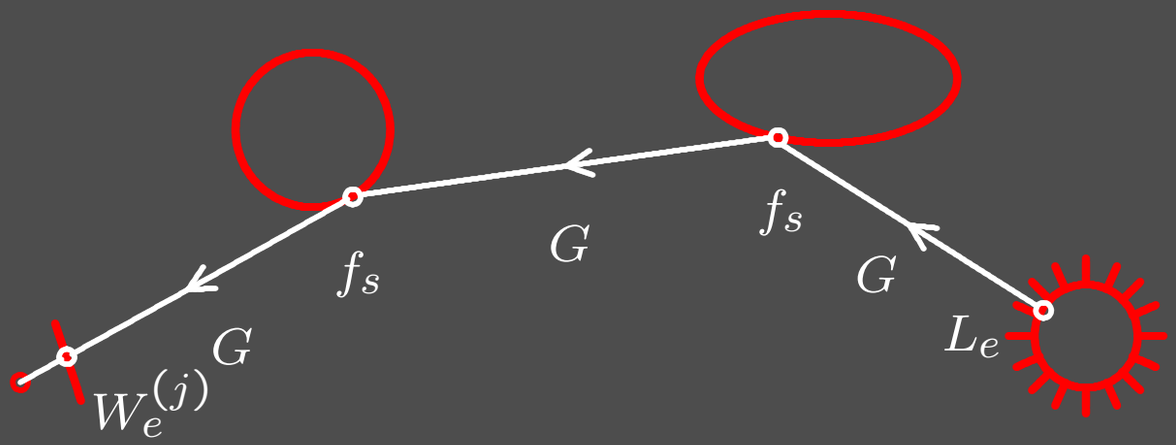


Path Integral Formulation

- Path space and path measure

$$\mathcal{P}_k = \{\bar{x} = x_0 x_1 \dots x_k \mid x_i \in S\} \qquad d\mu_k(\bar{x}) = \prod_{i=0}^{k} dA(x_i)$$

- Measurement contribution function

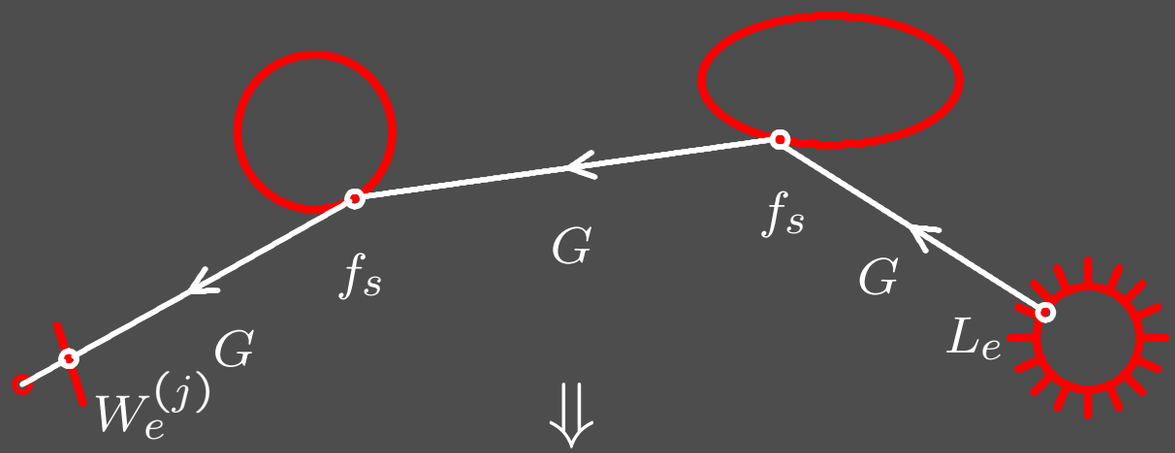


Path Integral Formulation

- Path space and path measure

$$\mathcal{P}_k = \{\bar{x} = x_0 x_1 \dots x_k \mid x_i \in S\} \qquad d\mu_k(\bar{x}) = \prod_{i=0}^{k} dA(x_i)$$

- Measurement contribution function



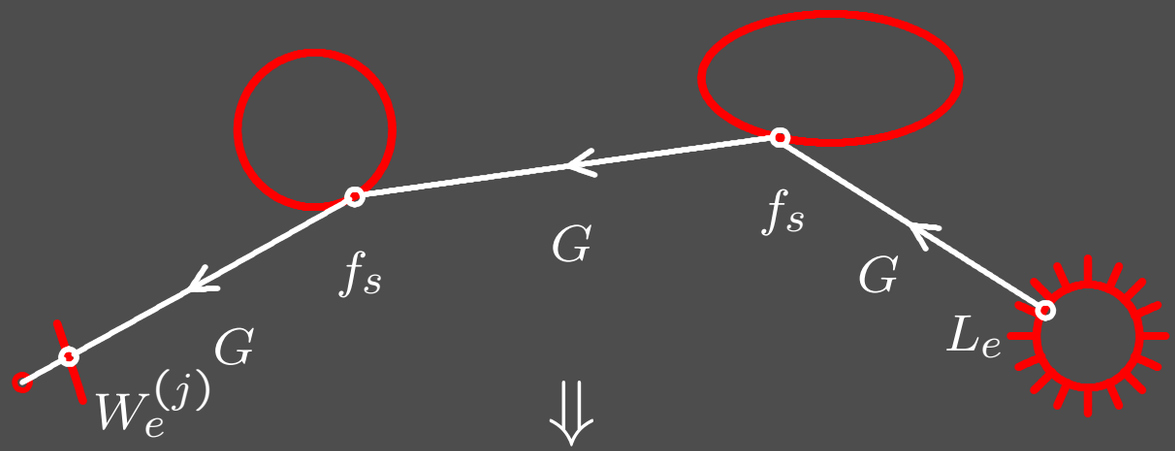
$$f_j = L_e \cdot G \cdot f_s \cdot G \cdot f_s \cdot G \cdot W_e^{(j)}$$

Path Integral Formulation

- Path space and path measure

$$\mathcal{P}_k = \{\bar{x} = x_0 x_1 \dots x_k \mid x_i \in S\} \qquad d\mu_k(\bar{x}) = \prod_{i=0}^{k} dA(x_i)$$

- Measurement contribution function



$$f_j = L_e \cdot G \cdot f_s \cdot G \cdot f_s \cdot G \cdot W_e^{(j)}$$

- Integral for path length k

$$\int_{\mathcal{P}_k} f_j(\bar{x}) d\mu_k(\bar{x})$$

Multiple Importance Sampling

- N techniques to generate samples with associated probability density functions

$$p_1, p_2, \ldots, p_N \; : \; D \rightarrow \mathbb{R}_0^+$$

- Heuristic

$$w_1, w_2, \ldots, w_N \; : \; D \rightarrow \mathbb{R}_0^+$$

  - $\sum_{i=1}^{N} w_i(x) = 1$ for all $x \in D$ with $f(x) \neq 0$
  - $w_i(x) = 0$ for all $x \in D$ with $p_i(x) = 0$

Multiple Importance Sampling

- N techniques to generate samples with associated probability density functions

$$p_1, p_2, \ldots, p_N \ : \ D \rightarrow \mathbb{R}_0^+$$

- Heuristic

$$w_1, w_2, \ldots, w_N \ : \ D \rightarrow \mathbb{R}_0^+$$

  - $\sum_{i=1}^{N} w_i(x) = 1$ for all $x \in D$ with $f(x) \neq 0$
  - $w_i(x) = 0$ for all $x \in D$ with $p_i(x) = 0$

- Estimator

$$\int_D f(x)dx \approx \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{N} w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})} \qquad \text{where} \quad x_{i,j} \sim p_i$$

Multiple Importance Sampling

- N techniques to generate samples with associated probability density functions

$$p_1, p_2, \ldots, p_N \; : \; D \rightarrow \mathbb{R}_0^+$$

- Heuristic

$$w_1, w_2, \ldots, w_N \; : \; D \rightarrow \mathbb{R}_0^+$$

  - $\sum_{i=1}^{N} w_i(x) = 1$ for all $x \in D$ with $f(x) \neq 0$
  - $w_i(x) = 0$ for all $x \in D$ with $p_i(x) = 0$

- Estimator

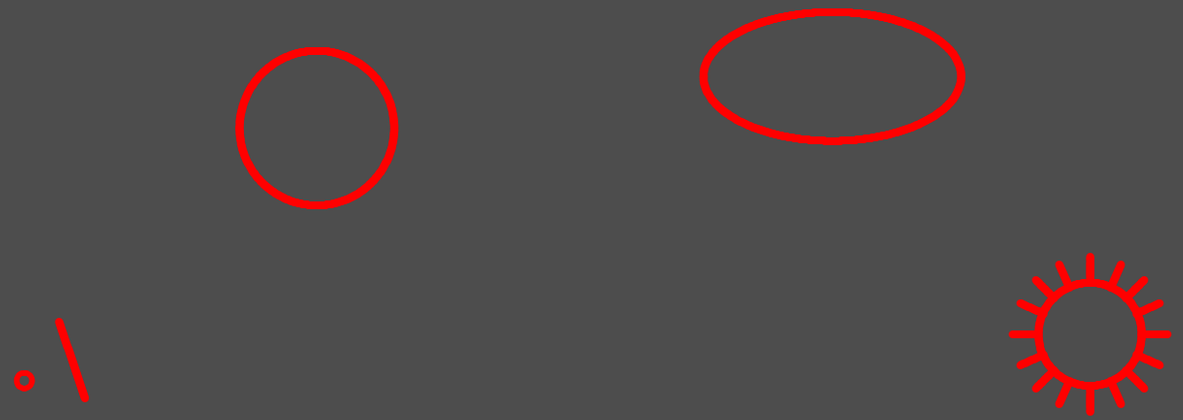
$$\int_D f(x)dx \approx \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{N} w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})} \qquad \text{where} \quad x_{i,j} \sim p_i$$

- Example: Balance heuristic

$$w_i(x) := \frac{p_i(x)}{\sum_{\ell=1}^{N} p_\ell(x)} \qquad \Rightarrow \qquad \int_D f(x)dx \approx \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{N} \frac{f(x_{i,j})}{\sum_{\ell=1}^{N} p_\ell(x_{i,j})}$$

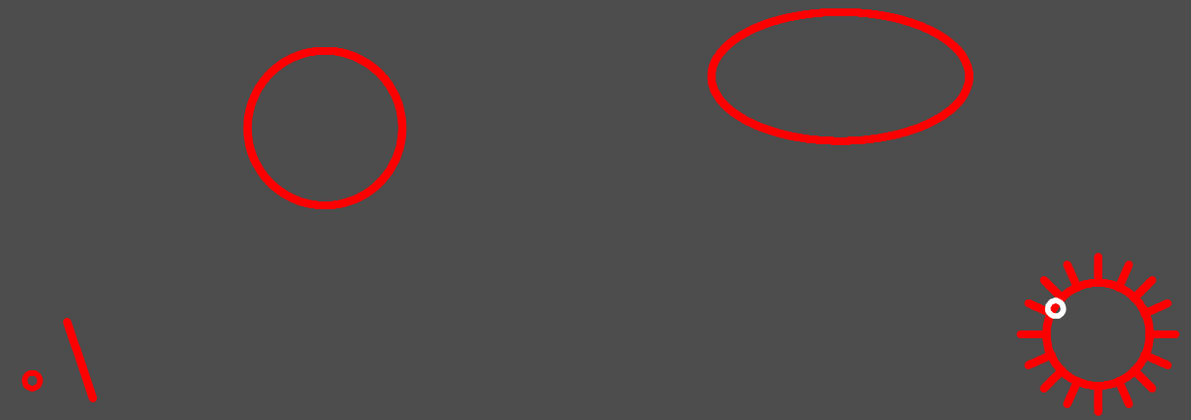
Bidirectional Path Tracing

- Generation of path space samples



Bidirectional Path Tracing

- Generation of path space samples



1. generate light subpath

Bidirectional Path Tracing

- Generation of path space samples

1. generate light subpath

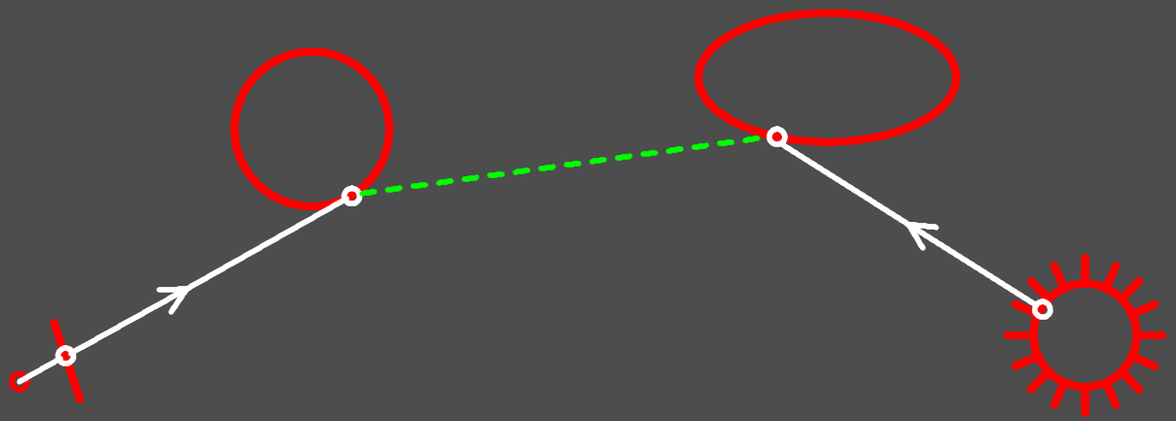
Bidirectional Path Tracing

- Generation of path space samples

1. generate light subpath
2. generate eye subpath

Bidirectional Path Tracing

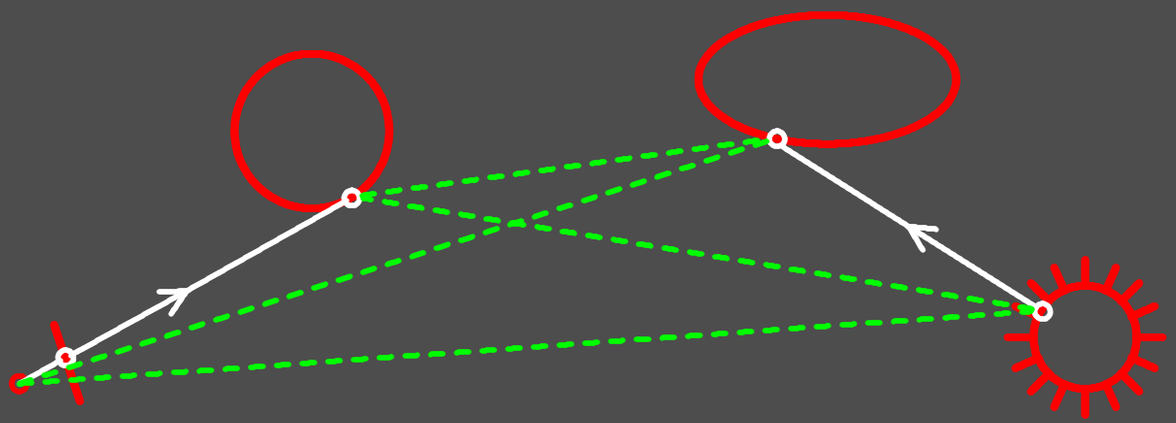
- Generation of path space samples



1. generate light subpath
2. generate eye subpath
3. connect deterministically

Bidirectional Path Tracing

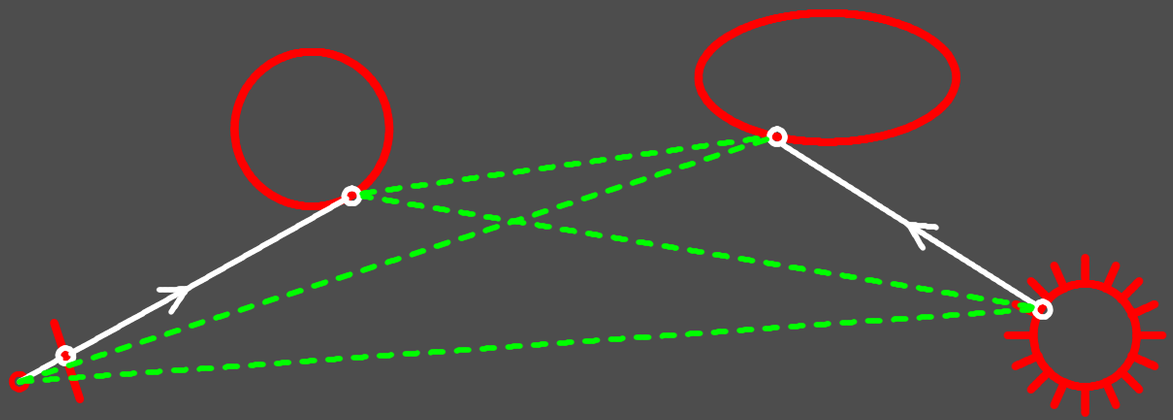
- Generation of path space samples



1. generate light subpath
2. generate eye subpath
3. connect deterministically and use all possibilities

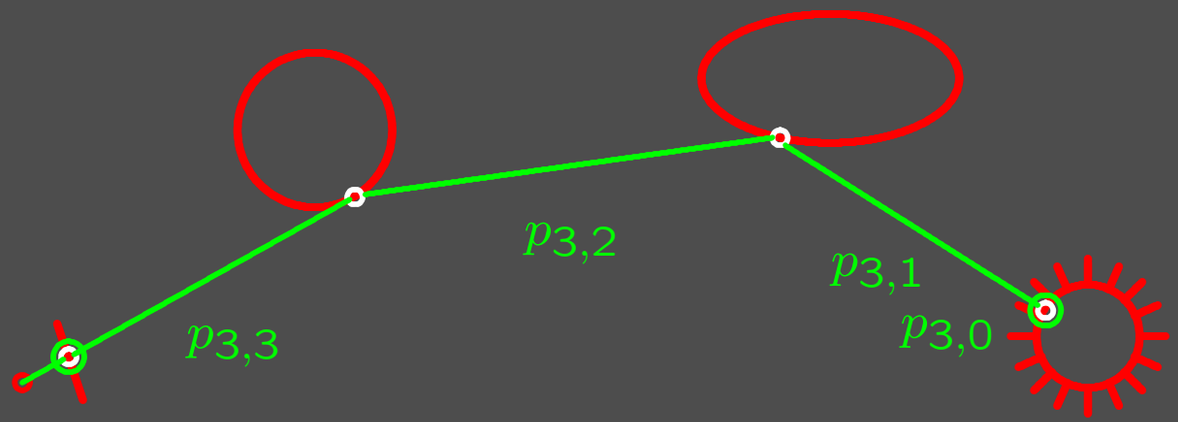
Bidirectional Path Tracing

- Generation of path space samples



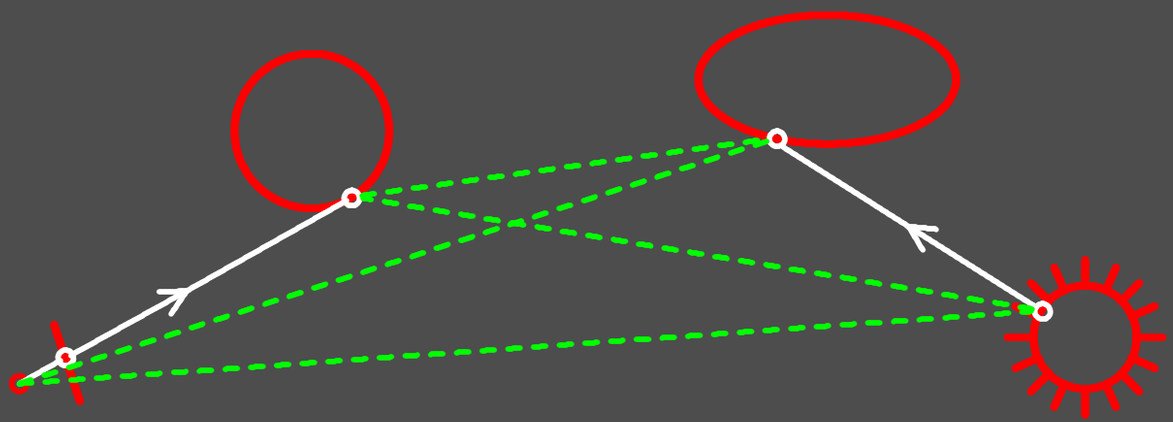
1. generate light subpath
2. generate eye subpath
3. connect deterministically and use all possibilities

- Techniques and probability density functions



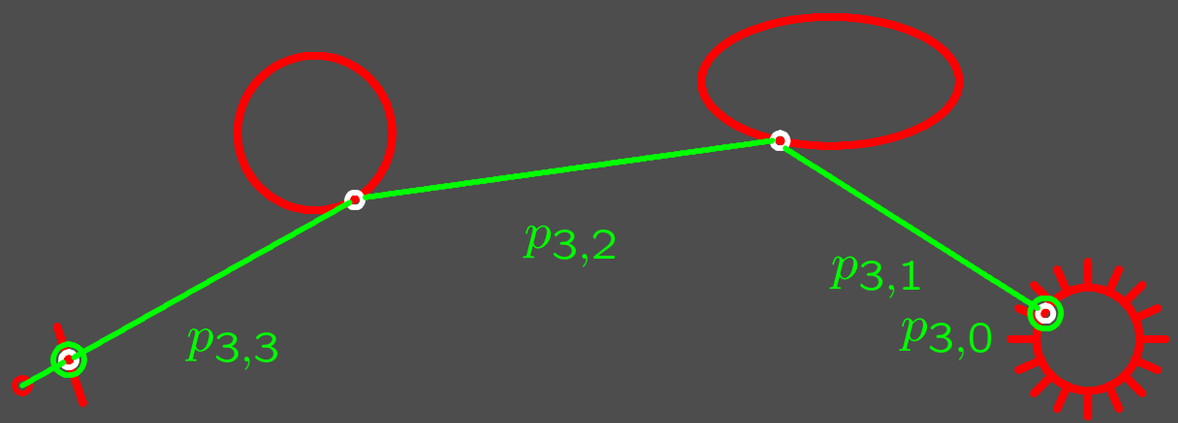
Bidirectional Path Tracing

- Generation of path space samples



1. generate light subpath
2. generate eye subpath
3. connect deterministically and use all possibilities

- Techniques and probability density functions



- Estimator

$$\int_{\mathcal{P}_k} f_j(\bar{x}) d\mu_k(\bar{x}) \approx \frac{1}{n} \sum_{j=1}^{n} \sum_{i=0}^{k} \frac{f_j(\bar{x}_{i,j})}{\sum_{\ell=0}^{k} p_{k,\ell}(\bar{x}_{i,j})} \qquad \text{where} \quad \bar{x}_{i,j} \sim p_{k,i}$$

Randomized Quasi-Monte Carlo Integration

- Low discrepancy point set

  r randomized replications of A

$A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$

$$\begin{matrix} x_{1,1} \ldots & x_{i,1} & \ldots x_{m,1} \\ & \vdots & \\ x_{1,j} \ldots & x_{i,j} & \ldots x_{m,j} \\ & \vdots & \\ x_{1,r} \ldots & x_{i,r} & \ldots x_{m,r} \end{matrix}$$

Randomized Quasi-Monte Carlo Integration

- Low discrepancy point set

  r randomized replications of A

$A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$

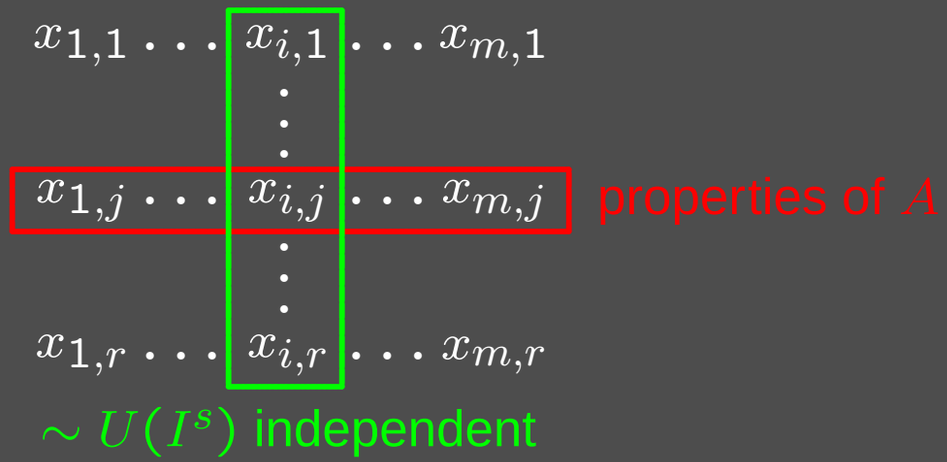
$$
\begin{array}{c}
x_{1,1} \ldots\ x_{i,1} \ldots x_{m,1} \\
\vdots \\
\boxed{x_{1,j} \ldots\ x_{i,j} \ldots x_{m,j}} \quad \text{properties of } A \\
\vdots \\
x_{1,r} \ldots\ x_{i,r} \ldots x_{m,r}
\end{array}
$$

Randomized Quasi-Monte Carlo Integration

- Low discrepancy point set

  r randomized replications of A

$A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$



Randomized Quasi-Monte Carlo Integration

- Low discrepancy point set

  r randomized replications of A

$A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$

$$\begin{array}{ccccc} x_{1,1} & \cdots & x_{i,1} & \cdots & x_{m,1} \\ \vdots & & \vdots & & \vdots \\ x_{1,j} & \cdots & x_{i,j} & \cdots & x_{m,j} \\ \vdots & & \vdots & & \vdots \\ x_{1,r} & \cdots & x_{i,r} & \cdots & x_{m,r} \end{array}$$

properties of A

$\sim U(I^s)$ independent

Estimator

$$\int_{I^s} f(x)dx \approx \frac{1}{r}\sum_{j=1}^{r}\frac{1}{m}\sum_{i=1}^{m} f(x_{i,j})$$

Randomized Quasi-Monte Carlo Integration

- Low discrepancy point set

  $A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$

  r randomized replications of A

$$\begin{array}{ccccc} x_{1,1} & \cdots & x_{i,1} & \cdots & x_{m,1} \\ \vdots & & \vdots & & \vdots \\ x_{1,j} & \cdots & x_{i,j} & \cdots & x_{m,j} \\ \vdots & & \vdots & & \vdots \\ x_{1,r} & \cdots & x_{i,r} & \cdots & x_{m,r} \end{array}$$

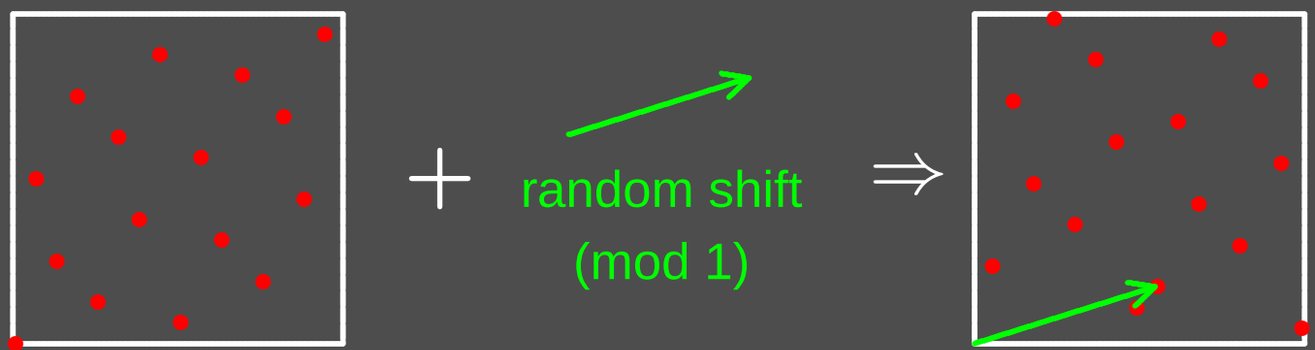
properties of A

$\sim U(I^s)$ independent

Estimator

$$\int_{I^s} f(x)dx \approx \frac{1}{r}\sum_{j=1}^{r}\frac{1}{m}\sum_{i=1}^{m} f(x_{i,j})$$

- Example: Cranley-Patterson rotation (1976)



$x_{i,j} = a_i \oplus \xi_j := (a_i + \xi_j) \bmod 1$ where $\xi_j \sim U(I^s)$ independent

Application to Bidirectional Path Tracing

$$\frac{1}{r}\sum_{j=1}^{r}\frac{1}{m}\sum_{i=1}^{m} f(x_{i,j}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{r}\sum_{j=1}^{r} f(\xi_j \oplus a_i)$$
$$\approx \frac{1}{m}\sum_{i=1}^{m}\int_{I^s} f(x \oplus a_i)dx = \int_{I_s} f(x)dx$$

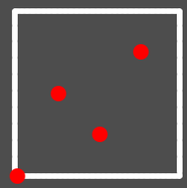
- Use high dimensional low discrepancy point set
- Structure of subpath generation

  area sampling
scattering } $\Rightarrow$ 2d problems
- Padded replications sampling

  For each 2d problem one random shift of the same 2d basis pattern

Subpath Generation by Padded Replications Sampling

Basis pattern

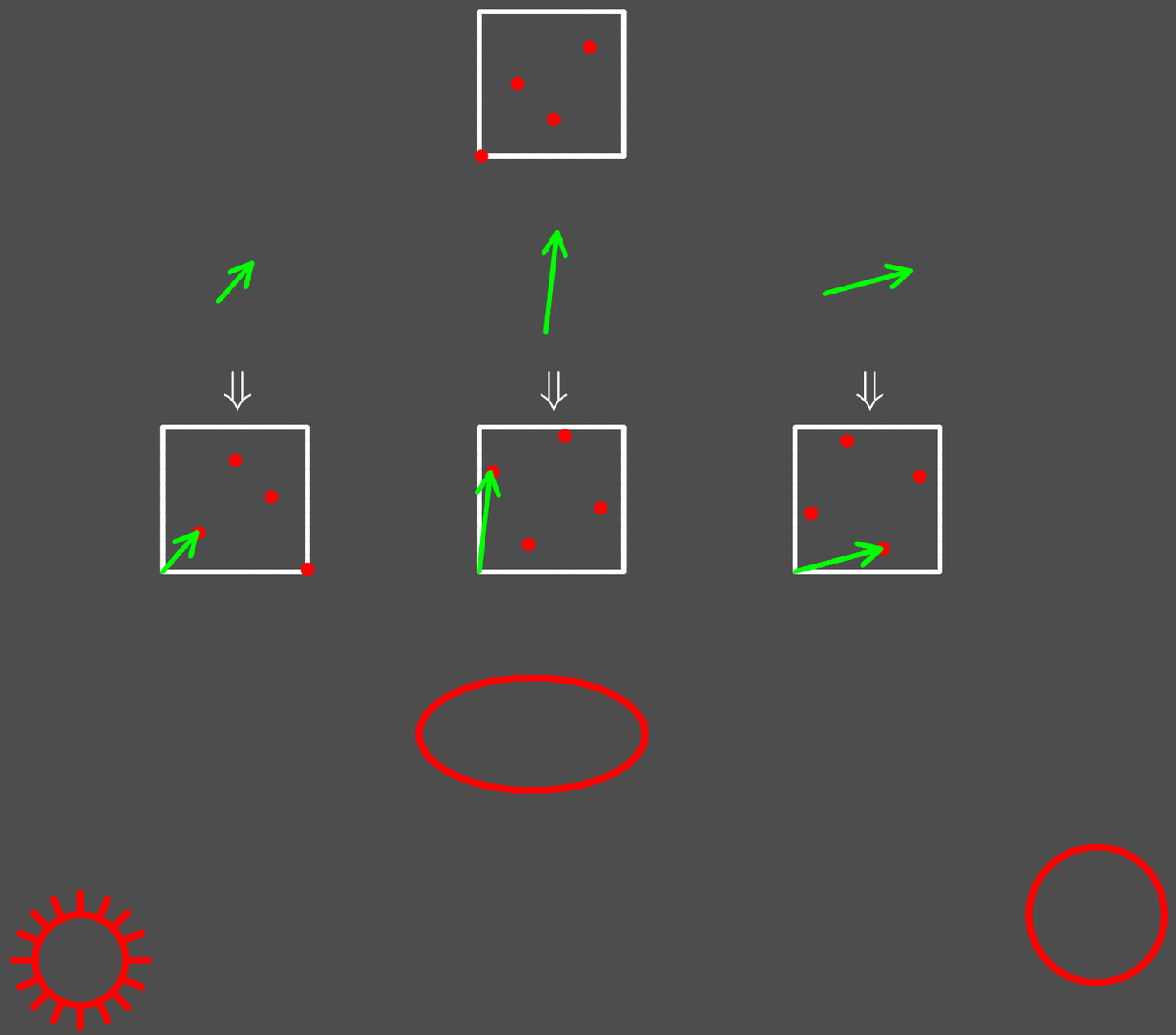


Subpath Generation by Padded Replications Sampling

Basis pattern

Random shifts

Randomized patterns

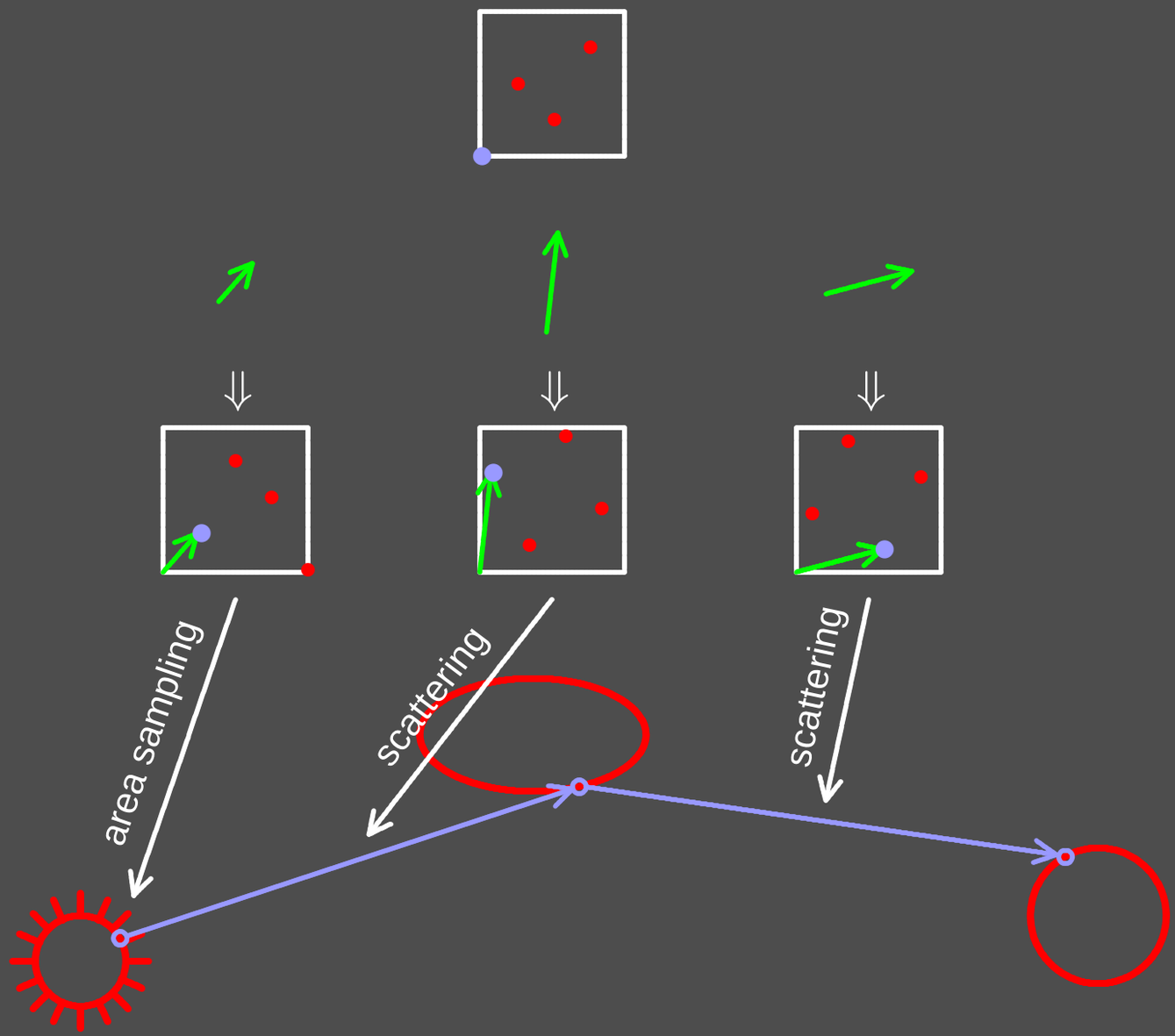


Subpath Generation by Padded Replications Sampling

Basis pattern

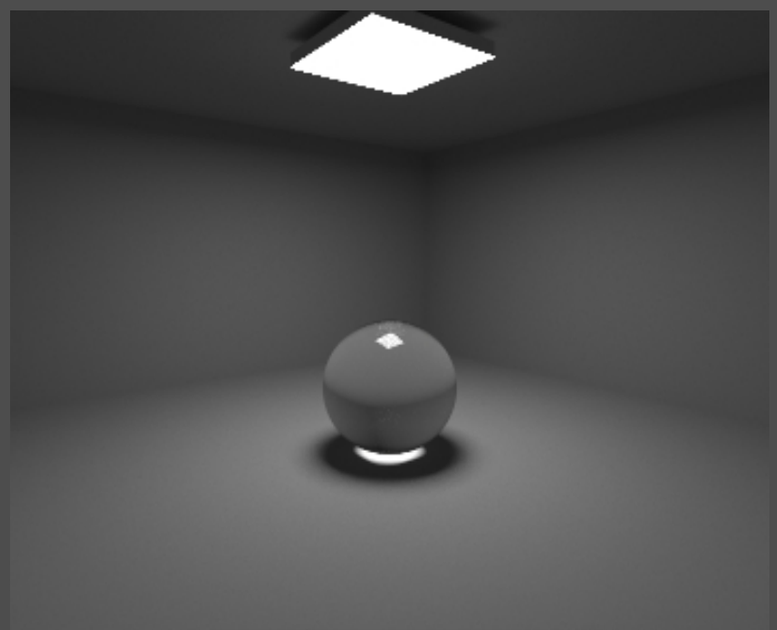
Random shifts

Randomized patterns

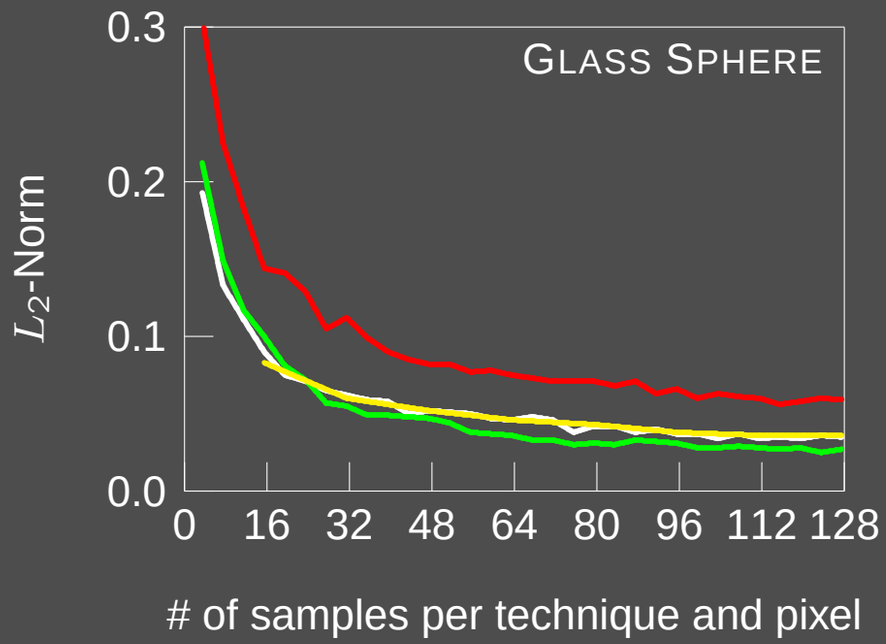


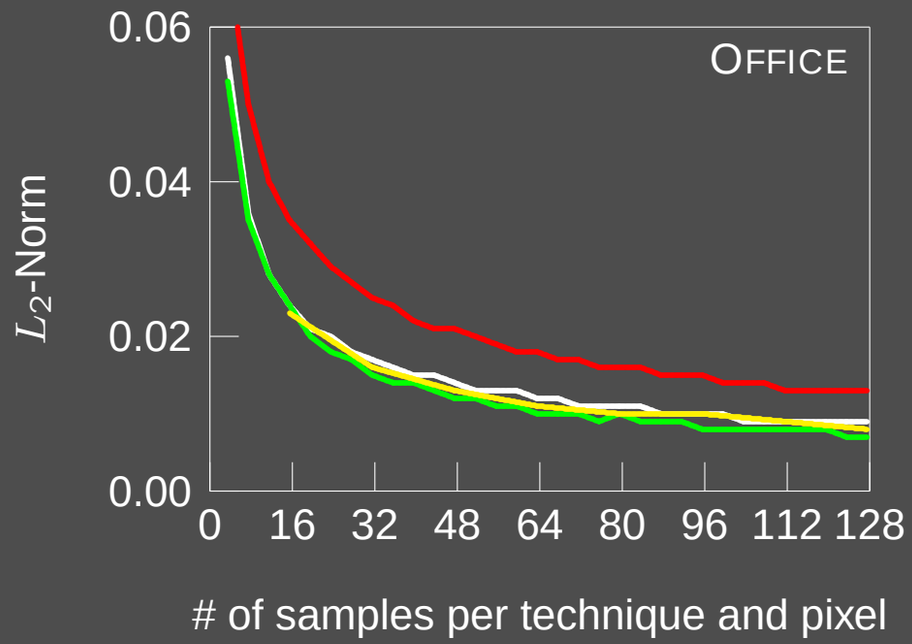
Numerical Experiments

- Comparison of bidirectional path tracing algorithms
  - MC: Random sampling
  - LHS: Latin hypercube sampling
  - RQMC: Randomized scrambled Hammersley, padded Hammersley
  - QMC: Scrambled Halton
- GLASS SPHERE and OFFICE scene



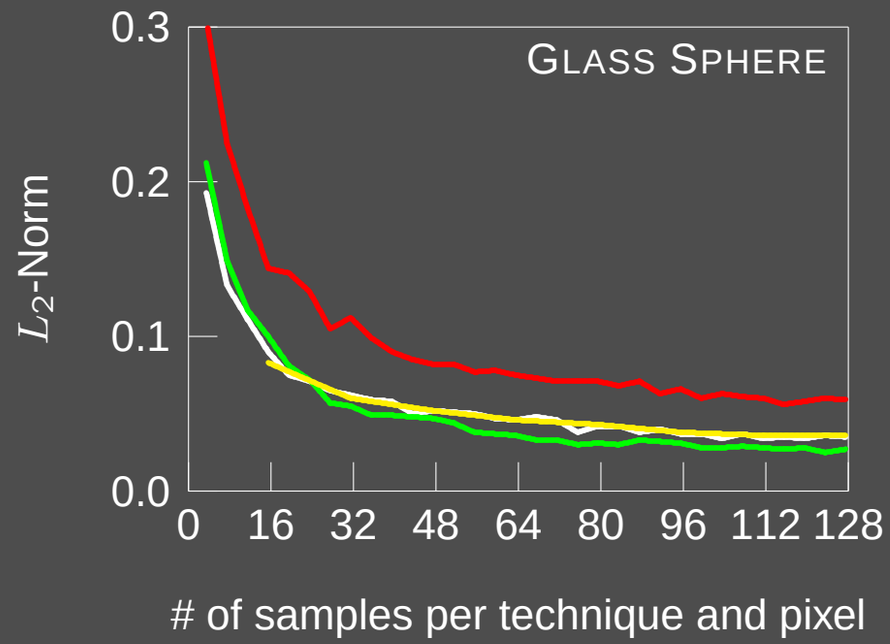
Results

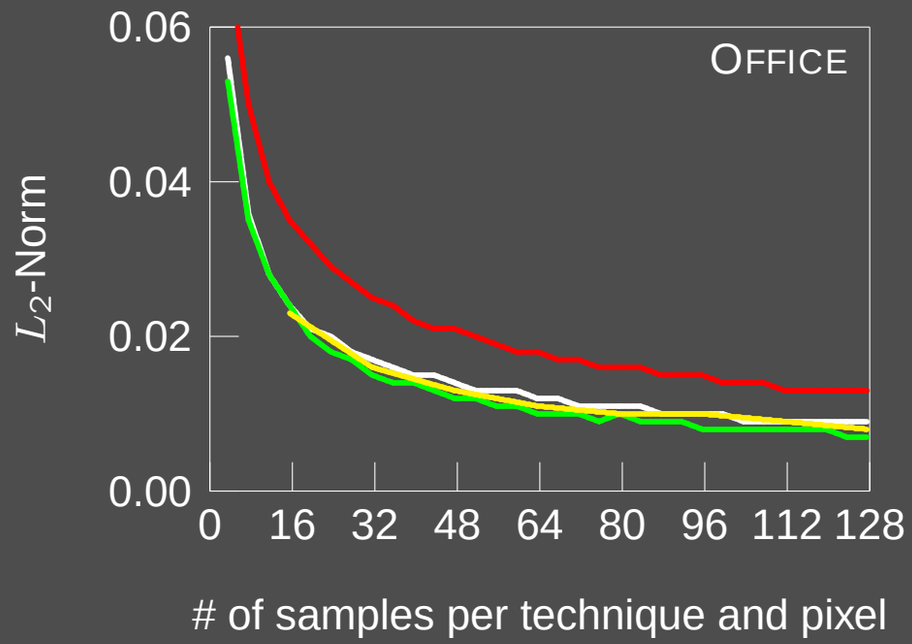


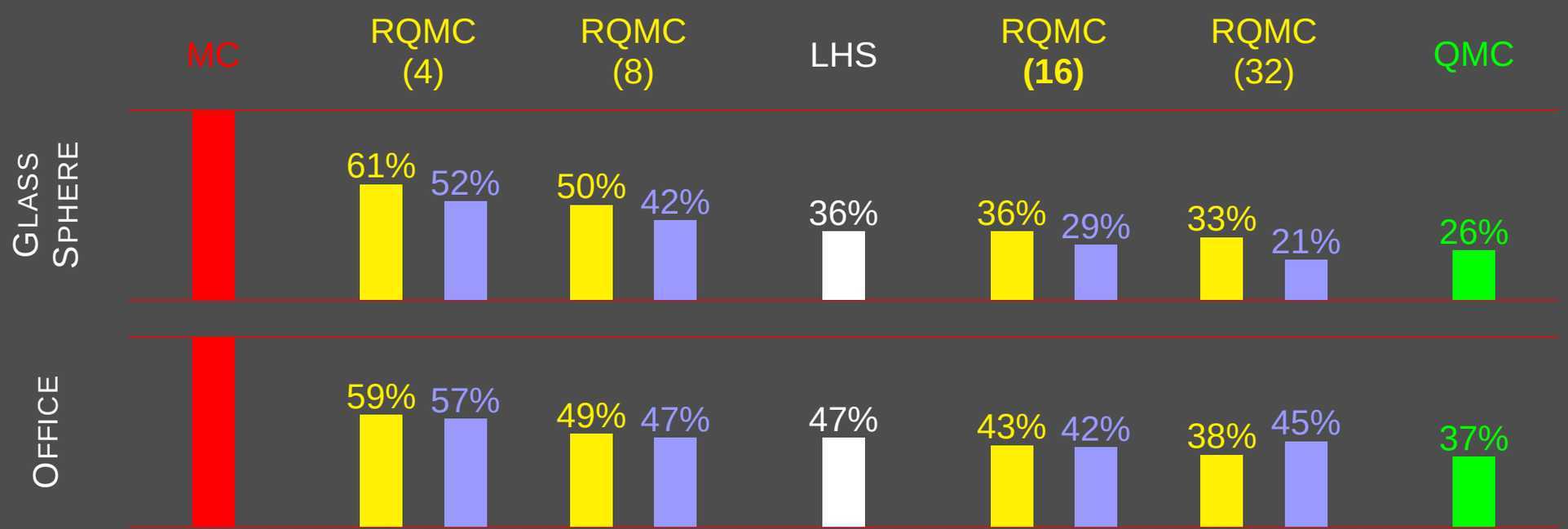


MC LHS RQMC **(16)** QMC

Results


GLASS SPHERE
L_2-Norm
0.3
0.2
0.1
0.0
0 16 32 48 64 80 96 112 128
# of samples per technique and pixel


OFFICE
L_2-Norm
0.06
0.04
0.02
0.00
0 16 32 48 64 80 96 112 128
# of samples per technique and pixel


MC
RQMC (4)
RQMC (8)
LHS
RQMC **(16)**
RQMC (32)
QMC
GLASS SPHERE
61% 52% 50% 42% 36% 36% 29% 33% 21% 26%
OFFICE
59% 57% 49% 47% 47% 43% 42% 38% 45% 37%

Visual Comparison of Images

Monte Carlo
(230s)

padded
Hammersley
(202s)

Strictly Deterministic Sampling in Computer Graphics

- Developed by derandomizing randomized quasi-Monte Carlo integration
  - quasi-Monte Carlo points
  - dependent splitting by restricted Cranley-Patterson rotations

Strictly Deterministic Sampling in Computer Graphics

- Developed by derandomizing randomized quasi-Monte Carlo integration
  - quasi-Monte Carlo points
  - dependent splitting by restricted Cranley-Patterson rotations

- Why it was not used in computer graphics
  - misinterpretation of the Koksma-Hlawka as rate instead of upper bound !!!
  - no scrambling was used
  - misbelief that deterministic sampling must alias

Strictly Deterministic Sampling in Computer Graphics

- Developed by derandomizing randomized quasi-Monte Carlo integration
  - quasi-Monte Carlo points
  - dependent splitting by restricted Cranley-Patterson rotations

- Why it was not used in computer graphics
  - misinterpretation of the Koksma-Hlawka as rate instead of upper bound !!!
  - no scrambling was used
  - misbelief that deterministic sampling must alias

- Why it should be used in computer graphics
  - high dimension
  - unknown discontinuities
  - much better discrete density approximation
  - much easier to parallelize and reproduce

Strictly Deterministic Sampling in Computer Graphics

- Developed by derandomizing randomized quasi-Monte Carlo integration
  - quasi-Monte Carlo points
  - dependent splitting by restricted Cranley-Patterson rotations

- Why it was not used in computer graphics
  - misinterpretation of the Koksma-Hlawka as rate instead of upper bound !!!
  - no scrambling was used
  - misbelief that deterministic sampling must alias

- Why it should be used in computer graphics
  - high dimension
  - unknown discontinuities
  - much better discrete density approximation
  - much easier to parallelize and reproduce

  - **simpler and faster !!!**

You already saw it in...

- Fight Club

You already saw it in...

- Fight Club

You already saw it in...

- Grinch, Walking with Dinosaurs, The Cell, The City of Lost Children, ...
- Product design at Mercedes Benz
- Universal Studio’s Terminator 2/3D
- The game Riven
- etc.

What's behind: Distribution Ray Tracing

- Compute functionals $\langle \Psi, L \rangle$ of the solution of the radiance integral equation

$$L(x,\omega) \;=\; L_e(x,\omega) + (TL)(x,\omega)$$

What's behind: Distribution Ray Tracing

- Compute functionals $\langle \Psi, L \rangle$ of the solution of the radiance integral equation

$$\begin{aligned} L(x,\omega) &= L_e(x,\omega) + (TL)(x,\omega) \\ &= \left(\sum_{i=0}^{\infty} T^i L_e \right)(x,\omega) \end{aligned}$$

  Application of Neumann series yields distribution ray tracing

What's behind: Distribution Ray Tracing

- Compute functionals $\langle \Psi, L \rangle$ of the solution of the radiance integral equation

$$\begin{aligned} L(x,\omega) &= L_e(x,\omega) + (TL)(x,\omega) \\ &= \left(\sum_{i=0}^{\infty} T^i L_e \right)(x,\omega) \end{aligned}$$

  Application of Neumann series yields distribution ray tracing

- Example problem: Direct illumination

$$\langle \Psi, T_{f_r} L_e \rangle = \int_{I^2} \int_{I^2} f(x,y) dx dy$$

  - x point in pixel
  - y point on light source, i.e. $\operatorname{supp} L_e$

What's behind: Distribution Ray Tracing

- Compute functionals $\langle \Psi, L \rangle$ of the solution of the radiance integral equation

$$\begin{aligned} L(x,\omega) &= L_e(x,\omega) + (TL)(x,\omega) \\ &= \left(\sum_{i=0}^{\infty} T^i L_e \right)(x,\omega) \end{aligned}$$

  Application of Neumann series yields distribution ray tracing

- Example problem: Direct illumination

$$\langle \Psi, T_{f_r} L_e \rangle = \int_{I^2} \int_{I^2} f(x,y) dx dy$$

  - x point in pixel
  - y point on light source, i.e. $\mathrm{supp}\, L_e$

- **Tracing rays and shader calls are expensive**

  $\Rightarrow$ efficient, parallel, and deterministic solution

Structure of $(0, 2n, 2)$-Nets

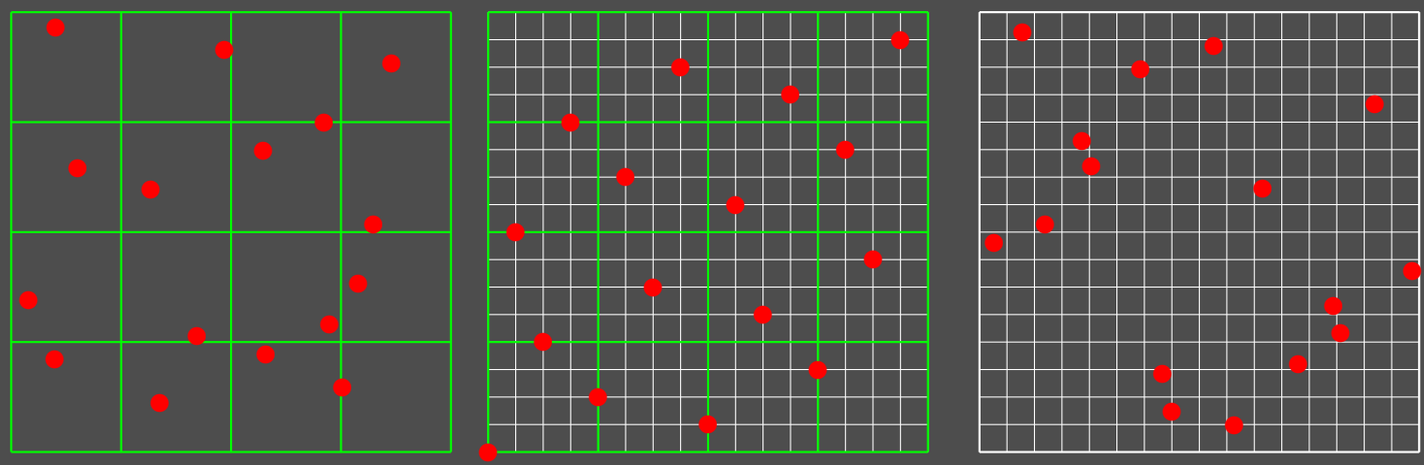
- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points

Structure of $(0, 2n, 2)$-Nets

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points
- $(0, 2n, 2)$-net in base $b = 2$
  - Set P_N of $N = (2^n)^2$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-2n} = \frac{1}{N}$ contains exactly 1 point

Structure of $(0, 2n, 2)$*-Nets*

- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points
- $(0, 2n, 2)$-net in base $b = 2$
  - Set P_N of $N = (2^n)^2$ 2-dimensional points of low discrepancy
  - Every *elementary interval* of volume $2^{-2n} = \frac{1}{N}$ contains exactly 1 point
  - Stratification of the Hammersley points $P_N = \left(\frac{i}{N}, \Phi_2(i)\right)_{i=0}^{N-1}$

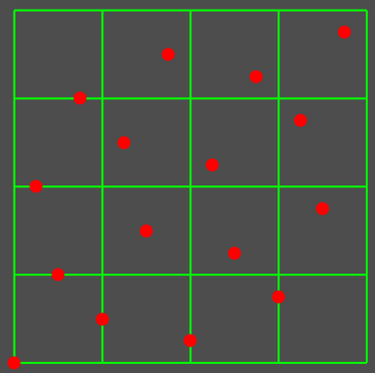


jittered **and** LHS (N -rooks)

$\Rightarrow$ Low dimension, low discrepancy

Efficient $(0, 2n, 2)$*-Net Generation*

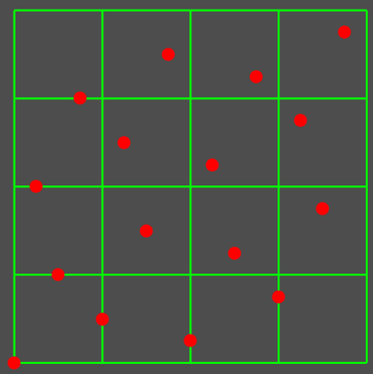
- Access by cell index $(j, k) \in \{0, \ldots, 2^n - 1\}^2$



$$
\begin{aligned}
i(j,k) &= 2^n \left(j + \Phi_2(k)\right) \in \{0, \ldots, N-1\} \\
x(j,k) &= \left(\frac{j + \Phi_2(k)}{2^n}, \frac{k + \Phi_2(j)}{2^n}\right)
\end{aligned}
$$

Efficient $(0, 2n, 2)$*-Net Generation*

- Access by cell index $(j, k) \in \{0, \ldots, 2^n - 1\}^2$



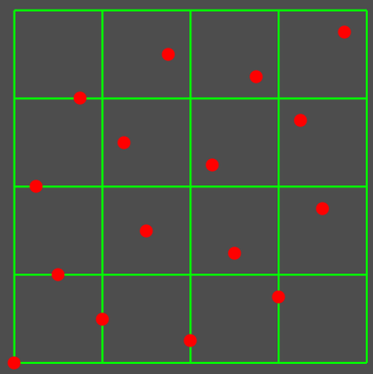
$$i(j,k) = 2^n (j + \Phi_2(k)) \in \{0, \ldots, N-1\}$$

$$x(j,k) = \left(\frac{j + \Phi_2(k)}{2^n}, \frac{k + \Phi_2(j)}{2^n} \right) = \left(\frac{i(j,k)}{N}, \Phi_2(i(j,k)) \right) = x_{i(j,k)}$$

  - table size for the permutation $2^n \Phi_2$ only 2^n instead of $(2^n)^2$
  - Bit parallel computation of $\Phi_2(i)$ in $\mathcal{O}(\log w)$
  - index $i(j,k)$ used to start out Faure-scrambled Hammersley point set for ray tree

Efficient $(0, 2n, 2)$*-Net Generation*

- Access by cell index $(j, k) \in \{0, \ldots, 2^n - 1\}^2$



$$
\begin{aligned}
i(j,k) &= 2^n \left(j + \Phi_2(k)\right) \in \{0, \ldots, N-1\} \\
x(j,k) &= \left(\frac{j + \Phi_2(k)}{2^n}, \frac{k + \Phi_2(j)}{2^n}\right) = \left(\frac{i(j,k)}{N}, \Phi_2(i(j,k))\right) = x_{i(j,k)}
\end{aligned}
$$

  - table size for the permutation $2^n \Phi_2$ only 2^n instead of $(2^n)^2$
  - Bit parallel computation of $\Phi_2(i)$ in $\mathcal{O}(\log w)$
  - index $i(j, k)$ used to start out Faure-scrambled Hammersley point set for ray tree

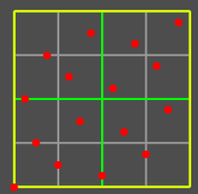
- Cover whole plane by tiling the pattern

$$
\begin{aligned}
\mathbb{Z}^2 &\rightarrow \{0, \ldots, 2^n - 1\}^2 \\
(s_x, s_y) &\mapsto (j, k) := (s_x \bmod 2^n, s_y \bmod 2^n)
\end{aligned}
$$

Antialiased Image Synthesis using $(0, 2n, 2)$*-Nets*

- Color of pixel $P_{m,n}$, $(m, n) \in \{0, \ldots, R_x - 1\} \times \{0, \ldots, R_y - 1\}$

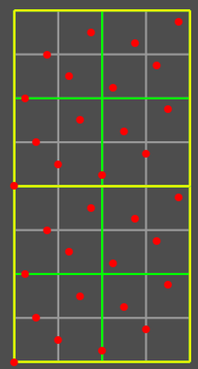
$$\frac{1}{|P_{m,n}|} \int_{I^2} \chi_{P_{m,n}}(x) f(x) dx \approx \frac{1}{|P_{m,n}| N} \sum_{i=0}^{N-1} \chi_{P_{m,n}}(x_i) f(x_i)$$



Antialiased Image Synthesis using $(0, 2n, 2)$*-Nets*

- Color of pixel $P_{m,n}$, $(m, n) \in \{0, \ldots, R_x - 1\} \times \{0, \ldots, R_y - 1\}$

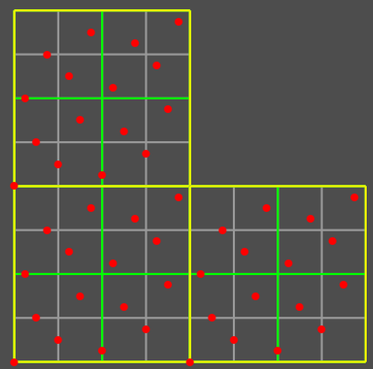
$$\frac{1}{|P_{m,n}|} \int_{I^2} \chi_{P_{m,n}}(x) f(x) dx \approx \frac{1}{|P_{m,n}| N} \sum_{i=0}^{N-1} \chi_{P_{m,n}}(x_i) f(x_i)$$



Antialiased Image Synthesis using $(0, 2n, 2)$*-Nets*

- Color of pixel $P_{m,n}$, $(m, n) \in \{0, \ldots, R_x - 1\} \times \{0, \ldots, R_y - 1\}$

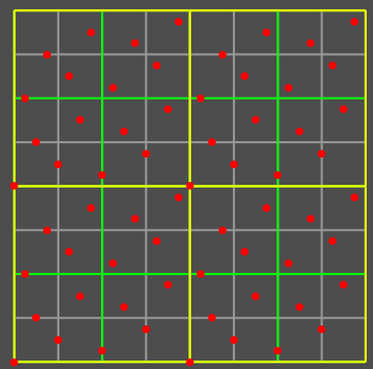
$$\frac{1}{|P_{m,n}|} \int_{I^2} \chi_{P_{m,n}}(x) f(x) dx \approx \frac{1}{|P_{m,n}| N} \sum_{i=0}^{N-1} \chi_{P_{m,n}}(x_i) f(x_i)$$



Antialiased Image Synthesis using $(0, 2n, 2)$*-Nets*

- Color of pixel $P_{m,n}$, $(m, n) \in \{0, \ldots, R_x - 1\} \times \{0, \ldots, R_y - 1\}$

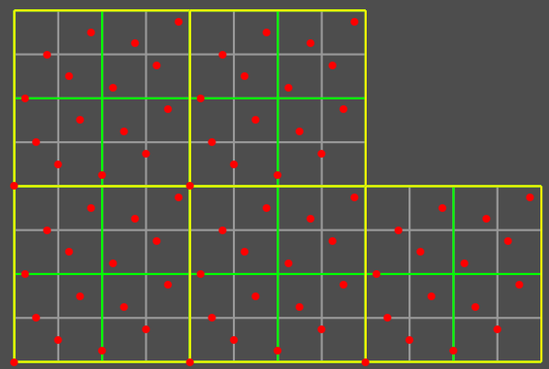
$$\frac{1}{|P_{m,n}|} \int_{I^2} \chi_{P_{m,n}}(x) f(x) dx \approx \frac{1}{|P_{m,n}| N} \sum_{i=0}^{N-1} \chi_{P_{m,n}}(x_i) f(x_i)$$



Antialiased Image Synthesis using $(0, 2n, 2)$*-Nets*

- Color of pixel $P_{m,n}$, $(m, n) \in \{0, \ldots, R_x - 1\} \times \{0, \ldots, R_y - 1\}$

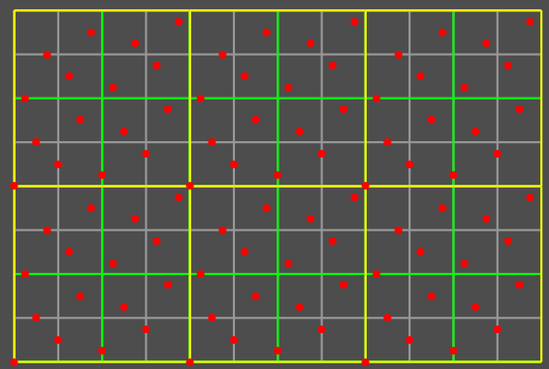
$$\frac{1}{|P_{m,n}|} \int_{I^2} \chi_{P_{m,n}}(x) f(x) dx \approx \frac{1}{|P_{m,n}| N} \sum_{i=0}^{N-1} \chi_{P_{m,n}}(x_i) f(x_i)$$



Antialiased Image Synthesis using $(0, 2n, 2)$*-Nets*

- Color of pixel $P_{m,n}$, $(m, n) \in \{0, \ldots, R_x - 1\} \times \{0, \ldots, R_y - 1\}$

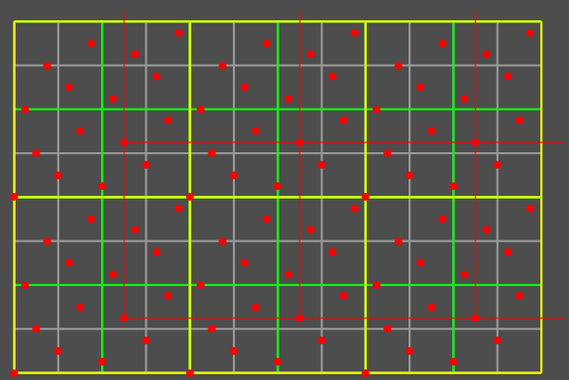
$$\frac{1}{|P_{m,n}|} \int_{I^2} \chi_{P_{m,n}}(x) f(x) dx \approx \frac{1}{|P_{m,n}| N} \sum_{i=0}^{N-1} \chi_{P_{m,n}}(x_i) f(x_i)$$



Antialiased Image Synthesis using $(0, 2n, 2)$*-Nets*

- Color of pixel $P_{m,n}$, $(m, n) \in \{0, \ldots, R_x - 1\} \times \{0, \ldots, R_y - 1\}$

$$\frac{1}{|P_{m,n}|} \int_{I^2} \chi_{P_{m,n}}(x) f(x) dx \approx \frac{1}{|P_{m,n}| N} \sum_{i=0}^{N-1} \chi_{P_{m,n}}(x_i) f(x_i)$$

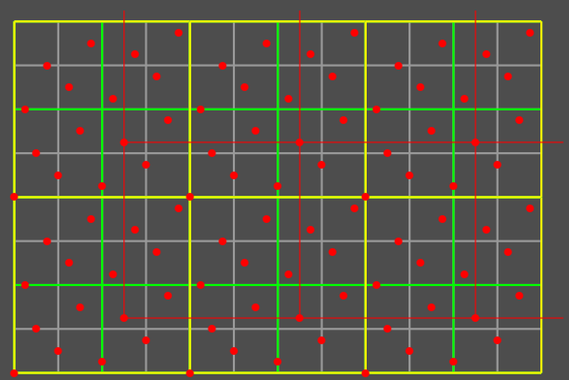


- Interleaved method of dependent tests: Samples of same instance i form regular grid
  - different sampling patterns in adjacent pixels
  - low discrepancy properties preserved

Antialiased Image Synthesis using $(0, 2n, 2)$*-Nets*

- Color of pixel $P_{m,n}$, $(m, n) \in \{0, \ldots, R_x - 1\} \times \{0, \ldots, R_y - 1\}$

$$\frac{1}{|P_{m,n}|} \int_{I^2} \chi_{P_{m,n}}(x) f(x) dx \approx \frac{1}{|P_{m,n}| N} \sum_{i=0}^{N-1} \chi_{P_{m,n}}(x_i) f(x_i)$$

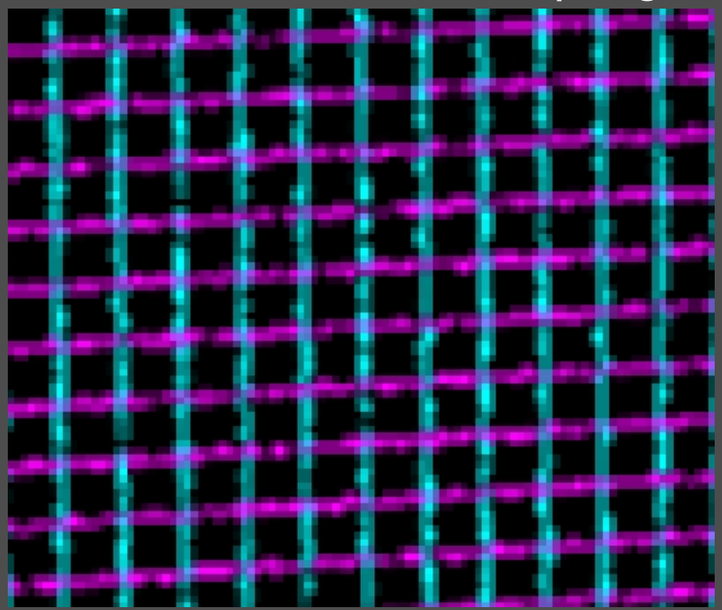


- Interleaved method of dependent tests: Samples of same instance i form regular grid
  - different sampling patterns in adjacent pixels
  - low discrepancy properties preserved
- Parallelization
  - deterministic
  - high coherence due to same instance i
  - almost optimal load balancing
  - Hardware: Simply interleave regular raster images in accumulation buffer

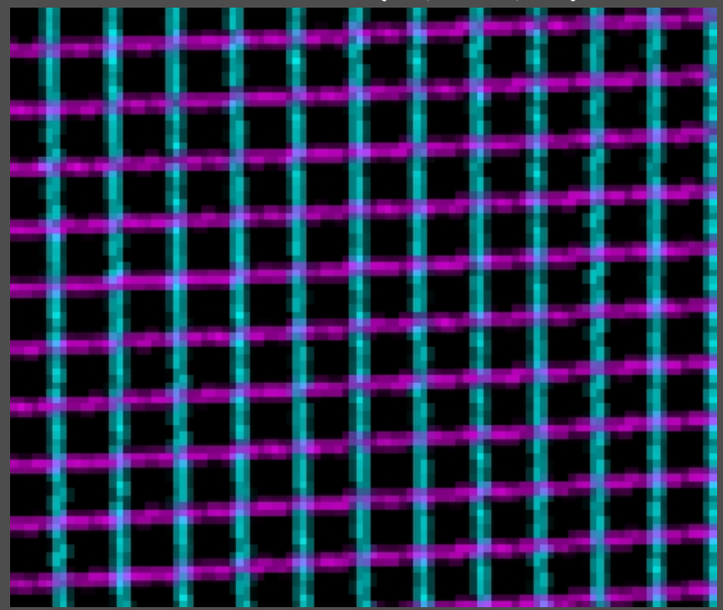
Antialiased Image Synthesis

- Comparison at 16 samples per pixel

stratified random sampling



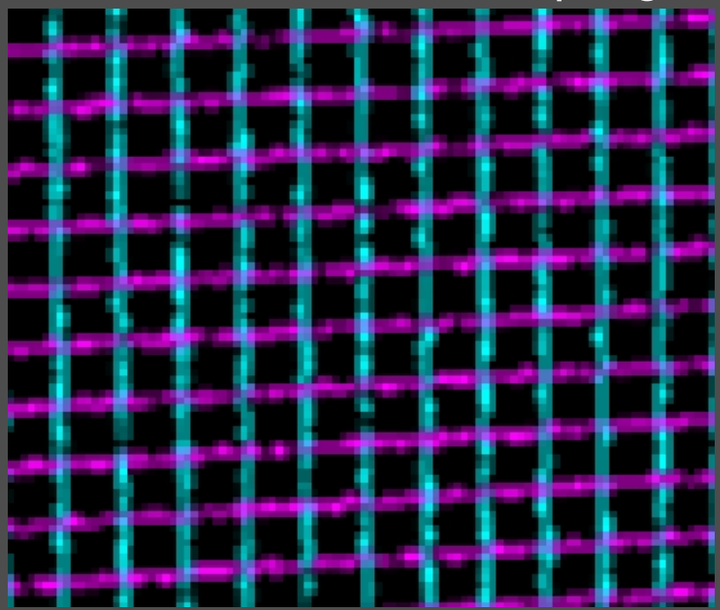
deterministic $(0, 2n, 2)$-net



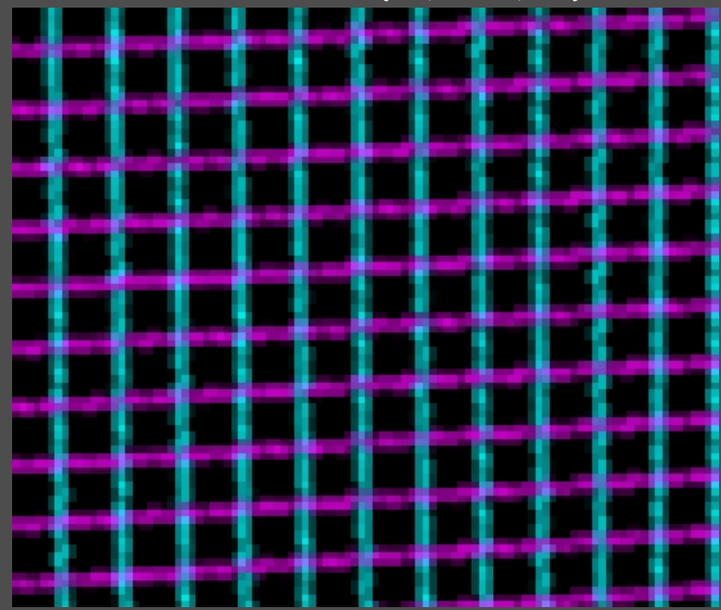
Antialiased Image Synthesis

- Comparison at 16 samples per pixel

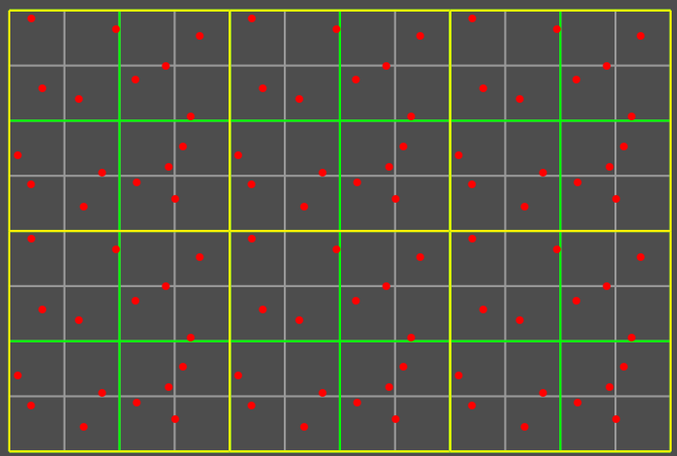
stratified random sampling



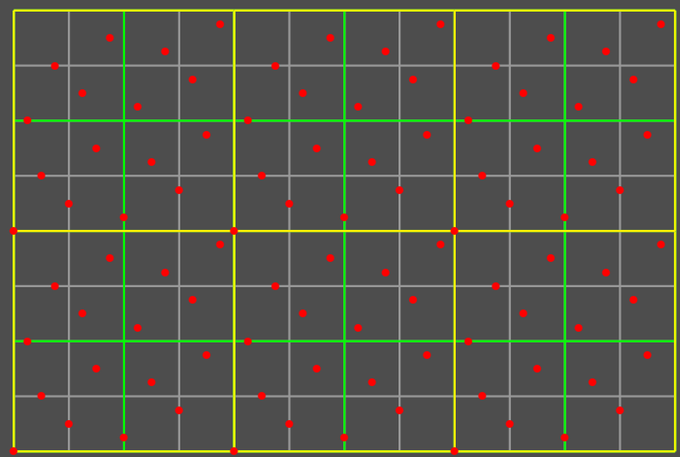
deterministic $(0, 2n, 2)$-net



uncorrelated



correlated



Dependent Splitting and Low Discrepancy Sampling

- Choice of
  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$
  - local quadrature rule $P_{M,s_2} = (z_j)_{j=0}^{M-1}$

$$\int_{I^{s1}} \int_{I^{s2}} f(x,y) dy dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i + z_j) \bmod 1)$$

1. random P_{N,s_1+s_2}, random P_{M,s_2}

Dependent Splitting and Low Discrepancy Sampling

- Choice of
  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$
  - local quadrature rule $P_{M,s_2} = (z_j)_{j=0}^{M-1}$

$$\int_{I^{s_1}} \int_{I^{s_2}} f(x,y) dy dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i + z_j) \bmod 1)$$

1. random P_{N,s_1+s_2}, random P_{M,s_2}
2. random P_{N,s_1+s_2}, low discrepancy P_{M,s_2}
   - variance reduction by restricted Cranley-Patterson rotations
   - benefit from superior discrepancy at special M and low dimensions s_2
   - benefit from intrinsic stratification of $(0, m, 2)$-nets in base 2

Dependent Splitting and Low Discrepancy Sampling

- Choice of
  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$
  - local quadrature rule $P_{M,s_2} = (z_j)_{j=0}^{M-1}$

$$\int_{I^{s_1}} \int_{I^{s_2}} f(x,y) dy dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i + z_j) \bmod 1)$$

1. random P_{N,s_1+s_2}, random P_{M,s_2}
2. random P_{N,s_1+s_2}, low discrepancy P_{M,s_2}
   - variance reduction by restricted Cranley-Patterson rotations
   - benefit from superior discrepancy at special M and low dimensions s_2
   - benefit from intrinsic stratification of $(0, m, 2)$-nets in base 2
3. deterministic low discrepancy P_{N,s_1+s_2}, deterministic low discrepancy P_{M,s_2}

   $\Rightarrow$ **derandomized dependent splitting**

Dependent Splitting and Low Discrepancy Sampling

- Choice of
  - global quadrature rule $P_{N,s_1+s_2} = (x_i, y_i)_{i=0}^{N-1}$
  - local quadrature rule $P_{M,s_2} = (z_j)_{j=0}^{M-1}$

$$\int_{I^{s_1}} \int_{I^{s_2}} f(x,y)dydx \approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{M} \sum_{j=0}^{M-1} f(x_i, (y_i+z_j) \bmod 1)$$

1. random P_{N,s_1+s_2}, random P_{M,s_2}
2. random P_{N,s_1+s_2}, low discrepancy P_{M,s_2}
   - variance reduction by restricted Cranley-Patterson rotations
   - benefit from superior discrepancy at special M and low dimensions s_2
   - benefit from intrinsic stratification of $(0, m, 2)$-nets in base 2
3. deterministic low discrepancy P_{N,s_1+s_2}, deterministic low discrepancy P_{M,s_2}

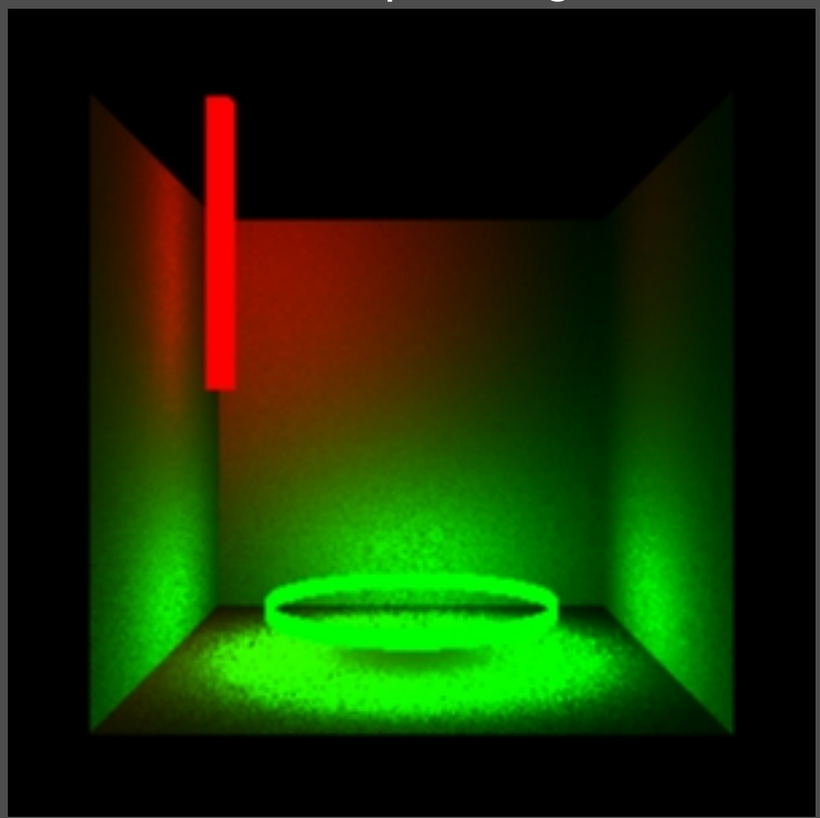
   $\Rightarrow$ **derandomized dependent splitting**

- Adaptive sampling by using low-discrepancy sequences for P_{M,s_2}
  - considering local minima of discrepancy

Strictly Deterministic Distribution Ray Tracing

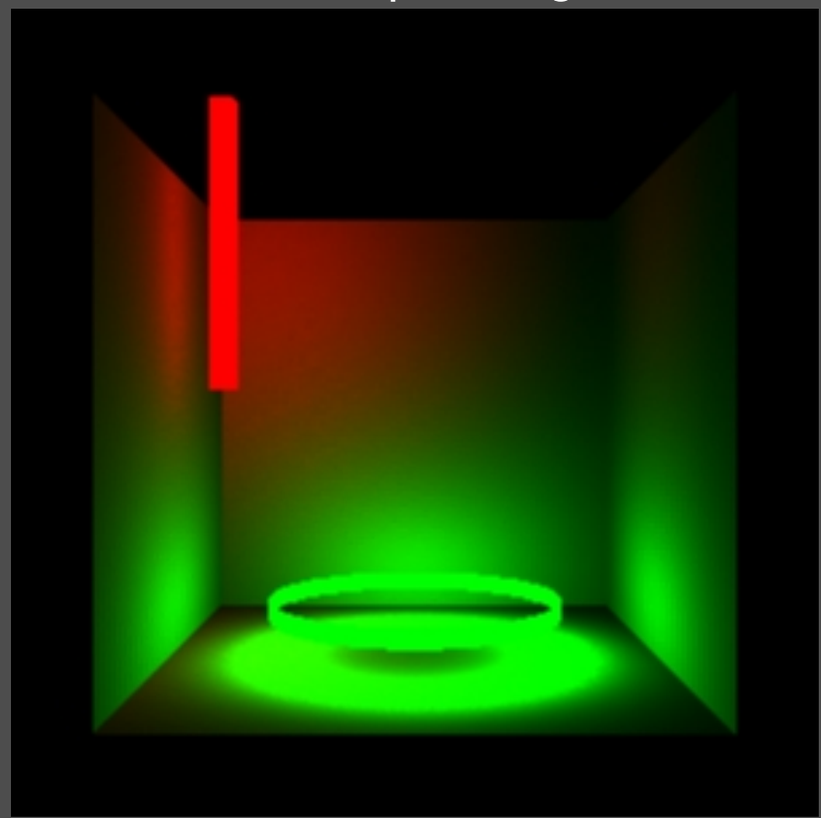
- Comparison for the example of direct illumination

stratified random sampling
25 samples / light



uncorrelated

strictly deterministic ray tree
16 samples / light



correlated

- Faure-scrambled Halton sequences and Hammersley point sets

Architecture

Architecture

Product Design

Conclusion

- Simple, compact, parallel, and strictly deterministic implementation ***mental ray***
  - anti-aliasing, motion blur, depth of field
  - area light sources, glossy scattering, participating media
  - global illumination

Conclusion

- Simple, compact, parallel, and strictly deterministic implementation ***mental ray***
  - anti-aliasing, motion blur, depth of field
  - area light sources, glossy scattering, participating media
  - global illumination
- Perfect reproducibility on parallel computer architectures
- No correlation problems of pseudo-random number generators

Conclusion

- Simple, compact, parallel, and strictly deterministic implementation ***mental ray***
  - anti-aliasing, motion blur, depth of field
  - area light sources, glossy scattering, participating media
  - global illumination
- Perfect reproducibility on parallel computer architectures
- No correlation problems of pseudo-random number generators

Images courtesy mental images and umlaut

Acknowledgements for their Support

- Peter Schröder, Caltech MultiRes Group, Pasadena CA, USA

  `www.multires.caltech.edu`

- Markus Gross, Computer Graphics Lab, Zürich, Swiss

  `graphics.ethz.ch`

- Rolf Herken, mental images, Berlin, Germany

  `www.mentalimages.com`

- Special thanks to

  Thomas Kollig

  Mark Pauly

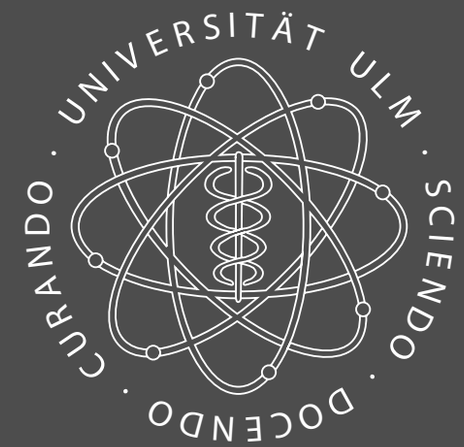
Our Research

- Monte Carlo methods
- Quasi-Monte Carlo methods
- Randomized quasi-Monte Carlo methods
- Realtime rendering
- High end computer graphics (***mental ray***)

  Check out the report: Strictly Deterministic Sampling Methods in Computer Graphics

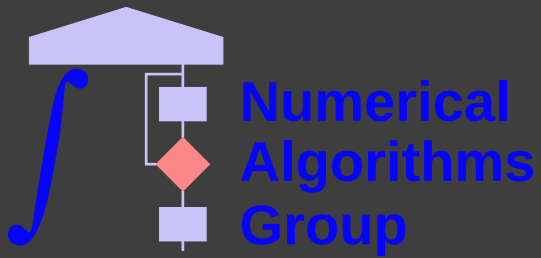
Visit us at

medien.informatik.uni-ulm.de/~keller



Efficient Multidimensional Sampling

Thomas Kollig



University of Kaiserslautern

Alexander Keller

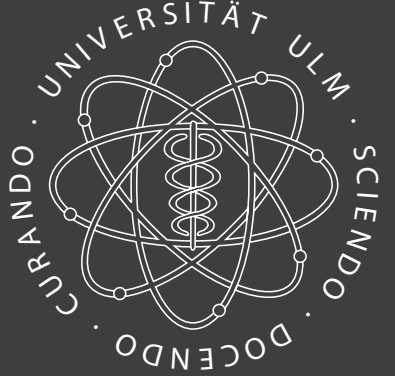


Image Synthesis

- Pixel antialiasing, area light sources, glossy reflections, motion blur, depth of field, . . .
  $\Rightarrow$ integration of multidimensional discontinuous functions

Image Synthesis

- Pixel antialiasing, area light sources, glossy reflections, motion blur, depth of field, . . .
  $\Rightarrow$ integration of multidimensional discontinuous functions

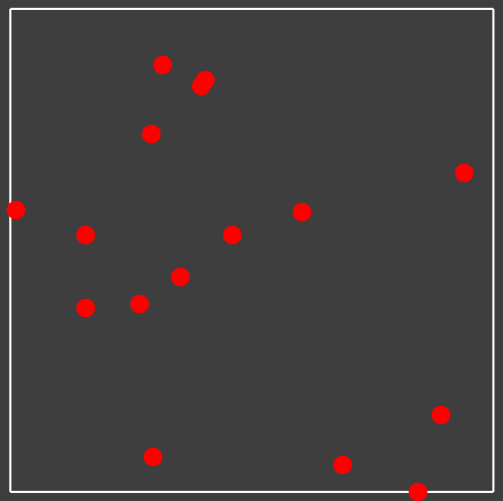
- Monte Carlo integration: $\int_{[0,1)^s} f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(\xi_i)$
  - ξ_i uniformly distributed $\Rightarrow$ unbiased estimator

Image Synthesis

- Pixel antialiasing, area light sources, glossy reflections, motion blur, depth of field, . . .
  $\Rightarrow$ integration of multidimensional discontinuous functions

- Monte Carlo integration: $\int_{[0,1)^s} f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(\xi_i)$
  - ξ_i uniformly distributed $\Rightarrow$ unbiased estimator

- Stratification



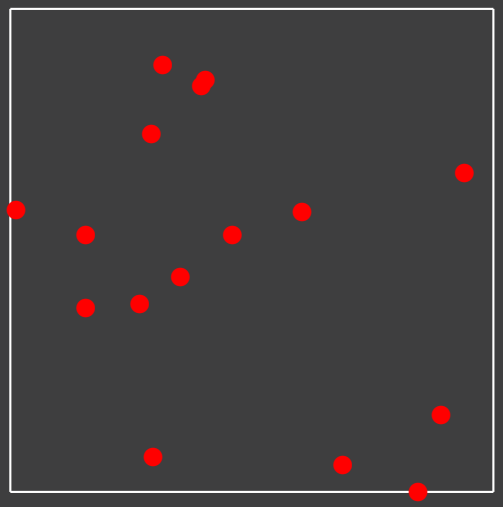
independent

Image Synthesis

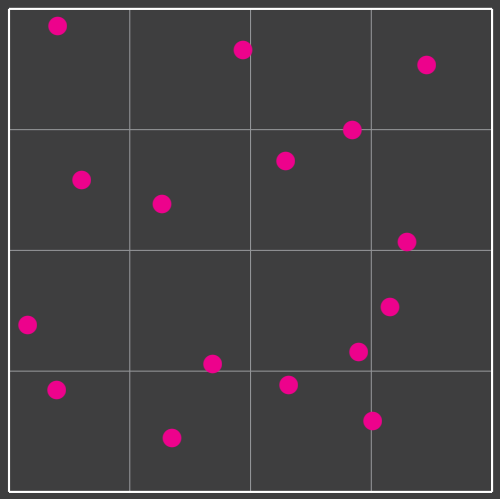
- Pixel antialiasing, area light sources, glossy reflections, motion blur, depth of field, . . .
  $\Rightarrow$ integration of multidimensional discontinuous functions

- Monte Carlo integration: $\int_{[0,1)^s} f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(\xi_i)$
  - ξ_i uniformly distributed $\Rightarrow$ unbiased estimator

- Stratification



independent



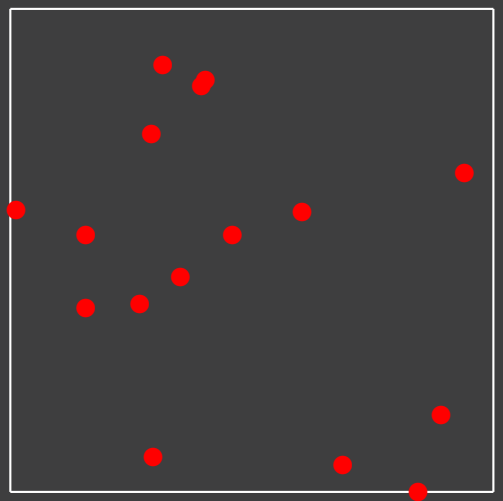
jittered

Image Synthesis

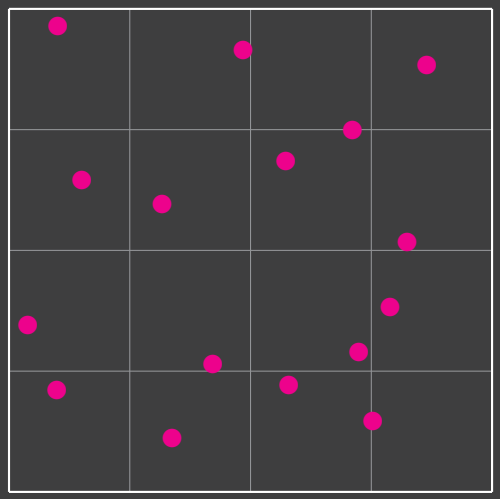
- Pixel antialiasing, area light sources, glossy reflections, motion blur, depth of field, . . .
  $\Rightarrow$ integration of multidimensional discontinuous functions

- Monte Carlo integration: $\int_{[0,1)^s} f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(\xi_i)$
  - ξ_i uniformly distributed $\Rightarrow$ unbiased estimator

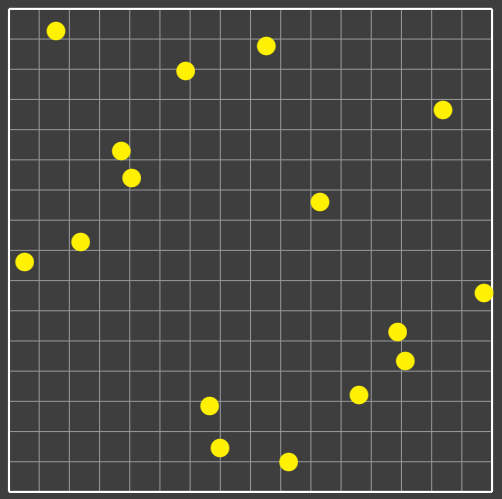
- Stratification



independent



jittered



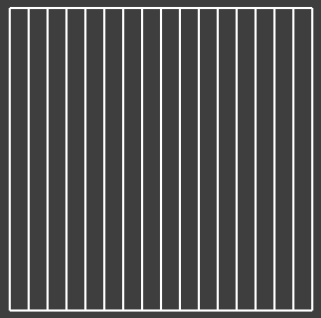
Latin hypercube

Generalization

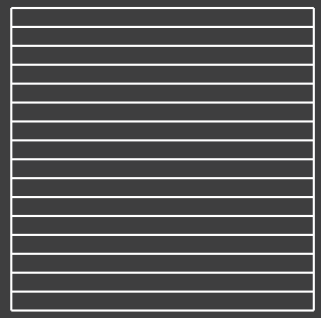
- ”All” stratifications

Generalization

- ”All” stratifications



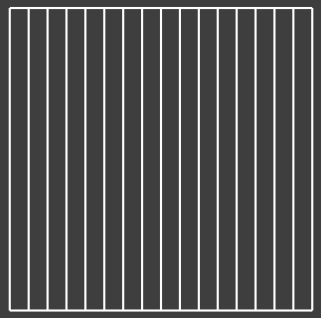
Latin hypercube



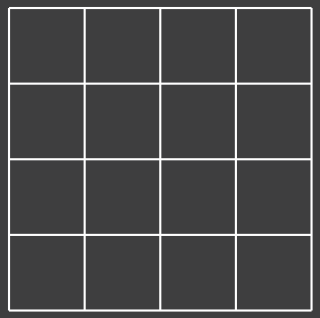
Latin hypercube

Generalization

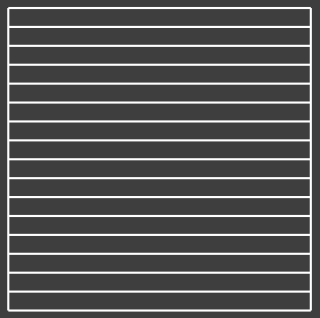
- ”All” stratifications



Latin hypercube



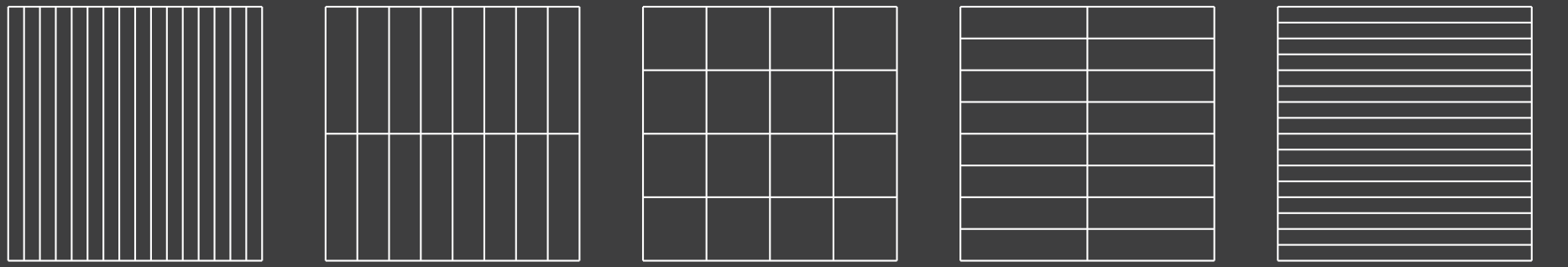
jittered



Latin hypercube

Generalization

- ”All” stratifications



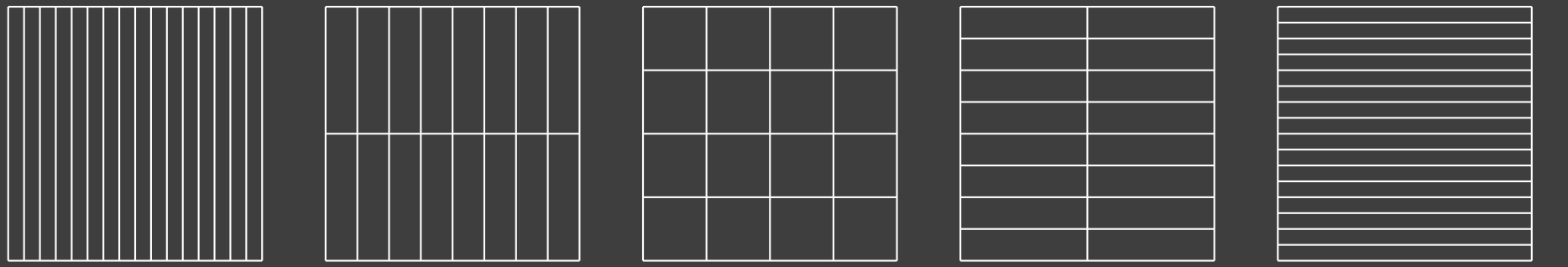
Latin hypercube

jittered

Latin hypercube

Generalization

- ”All” stratifications

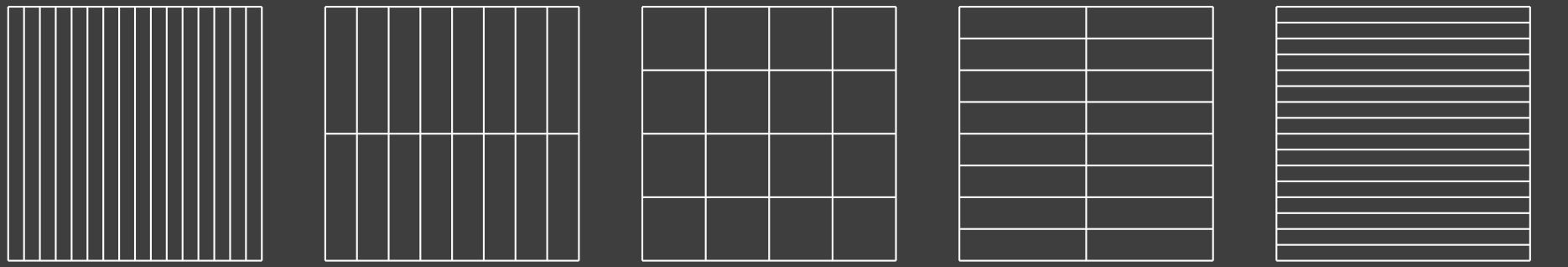


Latin hypercube jittered Latin hypercube

$\Rightarrow$ all elementary intervals in base 2 and dimension 2 with volume $\frac{1}{16}$

Generalization

- ”All” stratifications



$\Rightarrow$ all elementary intervals in base 2 and dimension 2 with volume $\frac{1}{16}$

- Elementary interval

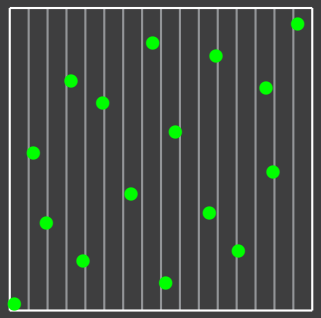
$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j + 1}{b^{l_j}} \right) \subseteq [0,1)^s$$

- Volume

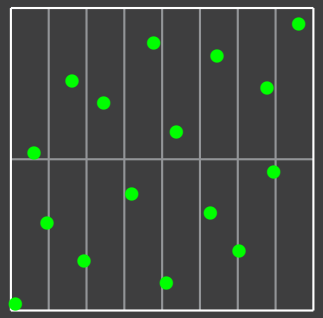
$$\text{Vol}(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = b^{-\sum_{j=1}^{s} l_j}$$

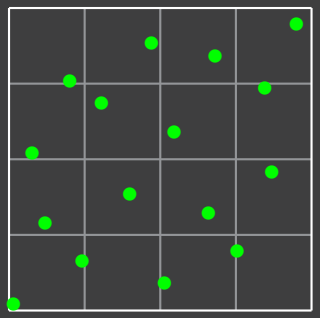
Generalization

- ”All” stratifications

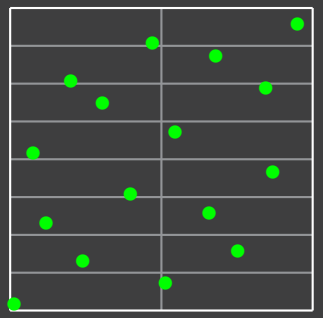


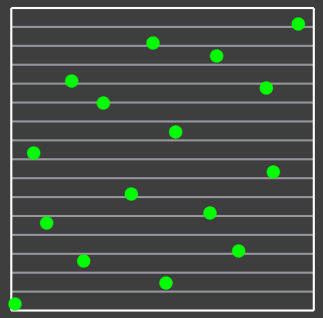
Latin hypercube





jittered

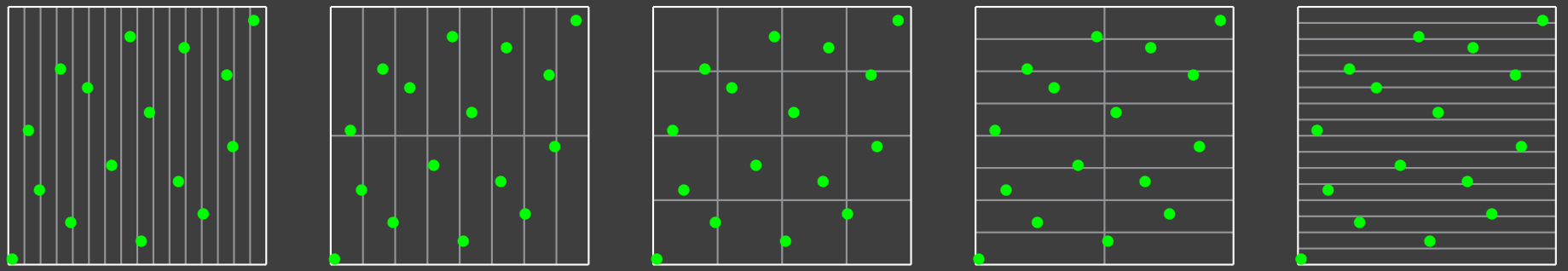




Latin hypercube

Generalization

- ”All” stratifications

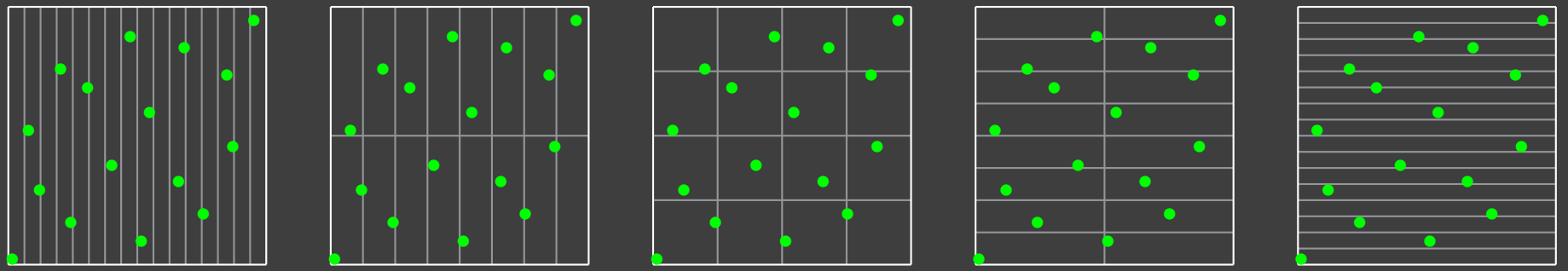


Latin hypercube jittered Latin hypercube

$\Rightarrow$ $(0, 4, 2)$-net in base 2

Generalization

- ”All” stratifications

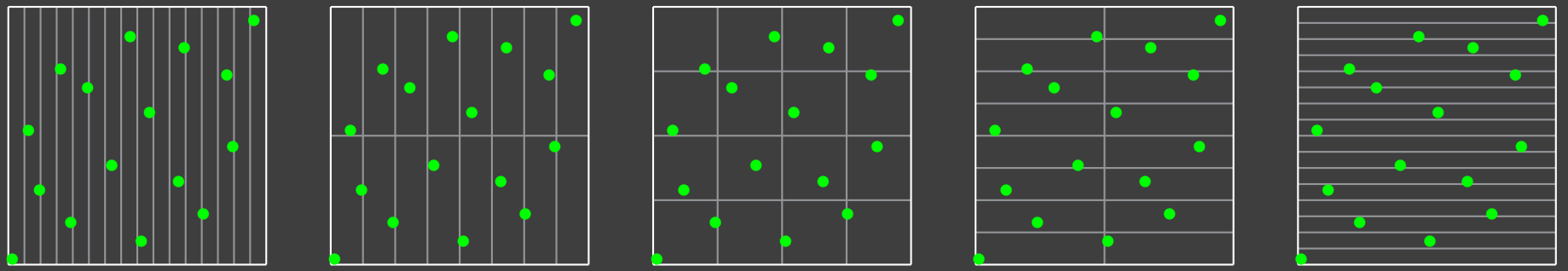


$\Rightarrow$ $(0, 4, 2)$-net in base 2

- **Definition:**
  Given two integers $0 \leq t \leq m$ a set of $N = b^m$ s-dimensional points x_i is called a (t, m, s)-net in base b if every elementary interval with volume $\text{Vol}(E) = b^{t-m}$ contains exactly b^t points. t is called quality parameter.

Generalization

- "All" stratifications



$\Rightarrow$ $(0, 4, 2)$-net in base 2

- **Definition:**
  Given two integers $0 \leq t \leq m$ a set of $N = b^m$ s-dimensional points x_i is called a (t, m, s)-net in base b if every elementary interval with volume Vol$(E) = b^{t-m}$ contains exactly b^t points. t is called quality parameter.

- **Definition:**
  For an integer $t \geq 0$ an infinite point sequence $(y_i)_{i=0}^{\infty}$ is called a (t, s)-sequence in base b, if for all $k \geq 0$ and $m > t$ the point set $\{y_{kb^m}, \ldots, y_{(k+1)b^m-1}\}$ is a (t, m, s)-net.

Randomization

- Monte Carlo integration

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)$$

  - x_i uniformly distributed

    $\Rightarrow$ unbiased estimator

Randomization

- Monte Carlo integration

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)$$

  - x_i uniformly distributed

    $\Rightarrow$ unbiased estimator

  - $X := \{x_0, x_1, \ldots, x_{N-1}\}$ is a (t, m, s)-net (with probability 1)

    $\Rightarrow$ for variance reduction by stratification

Randomization

- Monte Carlo integration

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i)$$

  - x_i uniformly distributed

    $\Rightarrow$ unbiased estimator

  - $X := \{x_0, x_1, \ldots, x_{N-1}\}$ is a (t, m, s)-net (with probability 1)

    $\Rightarrow$ for variance reduction by stratification

- Randomize deterministic (t, m, s)-net $A := \{a_0, a_1, \ldots, a_{N-1}\}$

  $X =$ randomize(A)

One Way: Owen Scrambling

- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes in an independent way.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

- Example for $b = 2$:

One Way: Owen Scrambling

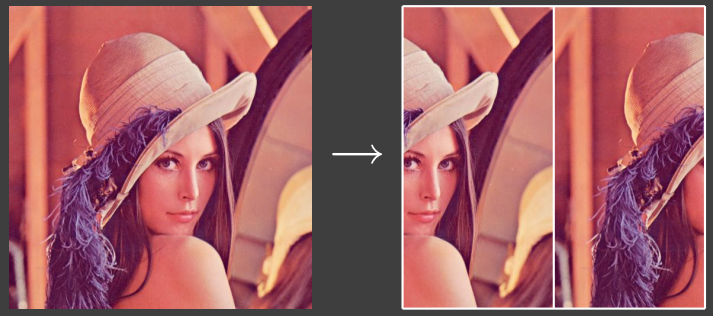
- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes in an independent way.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

- Example for $b = 2$:

One Way: Owen Scrambling

- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes in an independent way.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

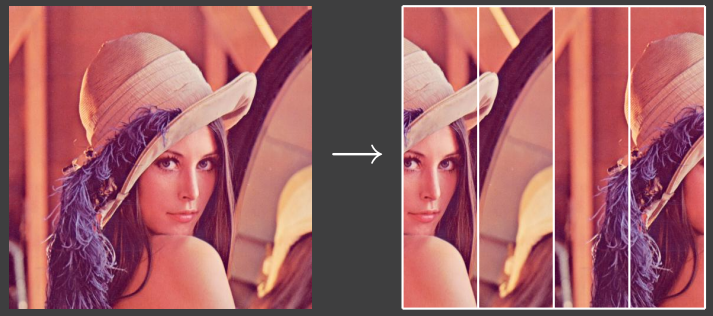
- Example for $b = 2$:



One Way: Owen Scrambling

- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes in an independent way.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

- Example for $b = 2$:



One Way: Owen Scrambling

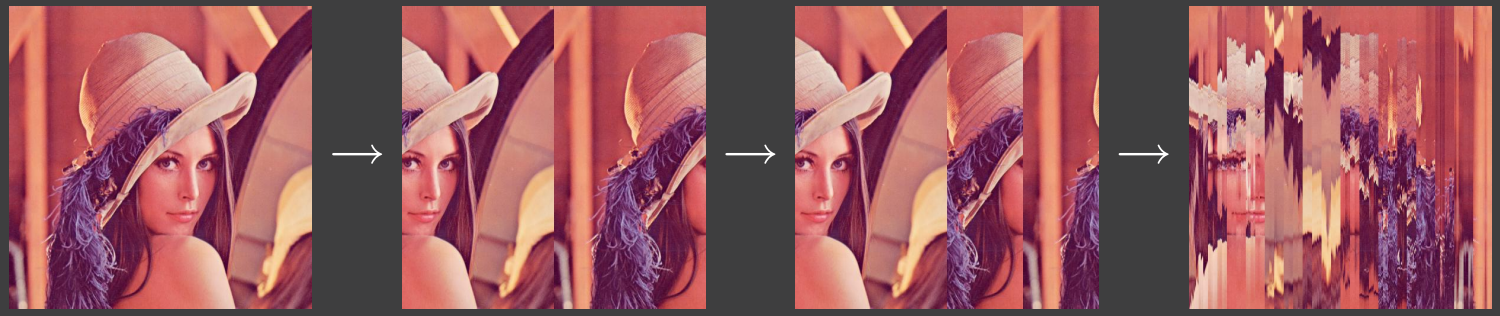
- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes in an independent way.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

- Example for $b = 2$:

One Way: Owen Scrambling

- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes in an independent way.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

- Example for $b = 2$:



One Way: Owen Scrambling

- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes in an independent way.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

- Example for $b = 2$:

- Properties
  - preserves (t, m, s)-net structure (with probability 1)
  - each point is uniformly distributed
  - finite precision terminates algorithm

One Way: Owen Scrambling

- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes in an independent way.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

- Example for $b = 2$:

- Properties
  - preserves (t, m, s)-net structure (with probability 1)
  - each point is uniformly distributed
  - finite precision terminates algorithm
  - **requires full b-ary tree of random permutations**

Cheap Way: Random Digit Scrambling

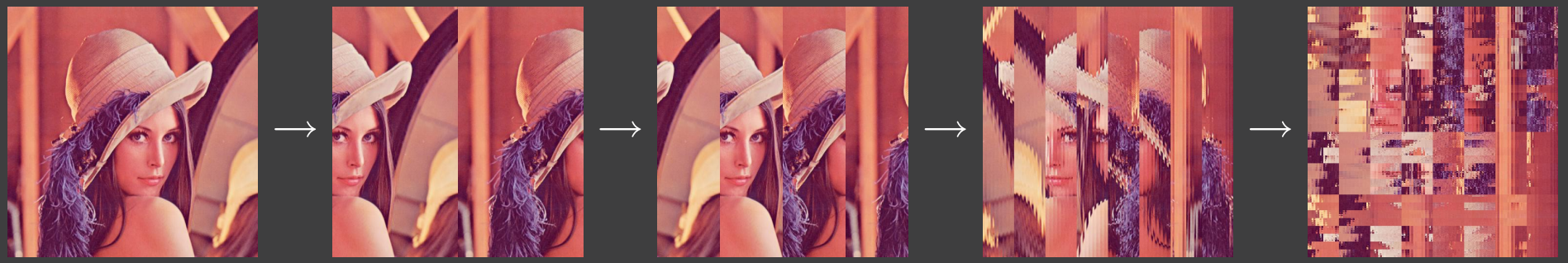
- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

- Example for $b = 2$:

Cheap Way: Random Digit Scrambling

- Algorithm starting with $H = [0, 1)^s$ (for each coordinate):
  1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
  2. Randomly permute these volumes.
  3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

- Example for $b = 2$:



- Properties
  - preserves (t, m, s)-net structure (with probability 1)
  - each point is uniformly distributed
  - finite precision terminates algorithm
  - efficient implementation for base 2: $x_i := \frac{2^n a_i \text{ XOR } \xi}{2^n}$

Implementation

```
double RI_vdC(uint bits, uint r = 0) {
    bits = ( bits               << 16) | ( bits               >> 16);
    bits = ((bits & 0x00ff00ff) <<  8) | ((bits & 0xff00ff00) >>  8);
    bits = ((bits & 0x0f0f0f0f) <<  4) | ((bits & 0xf0f0f0f0) >>  4);
    bits = ((bits & 0x33333333) <<  2) | ((bits & 0xcccccccc) >>  2);
    bits = ((bits & 0x55555555) <<  1) | ((bits & 0xaaaaaaaa) >>  1);

    bits ^= r;

    return (double) bits / (double) 0x100000000LL;
}

double RI_S(uint i, uint r = 0) {
    for(uint v = 1<<31; i; i >>= 1, v ^= v>>1)
        if(i & 1)
            r ^= v;

    return (double) r / (double) 0x100000000LL;
}

double RI_LP(uint i, uint r = 0) {
    for(uint v = 1<<31; i; i >>= 1, v |= v>>1)
        if(i & 1)
            r ^= v;

    return (double) r / (double) 0x100000000LL;
}
```

Implementation

- Random digit scrambled $(0, 1)$-sequences in base 2
  - RI_vdC van der Corput

```
double RI_S(uint i, uint r = 0) {
    for(uint v = 1<<31; i; i >>= 1, v ^= v>>1)
        if(i & 1)
            r ^= v;

    return (double) r / (double) 0x100000000LL;
}

double RI_LP(uint i, uint r = 0) {
    for(uint v = 1<<31; i; i >>= 1, v |= v>>1)
        if(i & 1)
            r ^= v;

    return (double) r / (double) 0x100000000LL;
}
```

Implementation

- Random digit scrambled $(0, 1)$-sequences in base 2
  - `RI_vdC` van der Corput
  - `RI_S` Sobol'

```
double RI_LP(uint i, uint r = 0) {
    for(uint v = 1<<31; i; i >>= 1, v |= v>>1)
        if(i & 1)
            r ^= v;

    return (double) r / (double) 0x100000000LL;
}
```

Implementation

- Random digit scrambled $(0, 1)$-sequences in base 2
  - `RI_vdC` van der Corput
  - `RI_S` Sobol'
  - `RI_LP` Larcher and Pillichshammer

Implementation

- Random digit scrambled $(0, 1)$-sequences in base 2
  - `RI_vdC` van der Corput
  - `RI_S` Sobol'
  - `RI_LP` Larcher and Pillichshammer

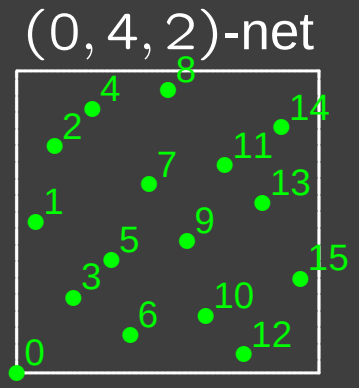
$$\left(\frac{i}{2^m}, \texttt{RI\_vdC}(i)\right)_{i=0}^{2^m-1} \quad \Rightarrow \quad (0, m, 2)\text{-net (Hammersley)}$$

$$\left(\frac{i}{2^m}, \texttt{RI\_LP}(i)\right)_{i=0}^{2^m-1} \quad \Rightarrow \quad (0, m, 2)\text{-net}$$

$$\left(\texttt{RI\_vdC}(i), \texttt{RI\_S}(i)\right)_{i=0}^{\infty} \quad \Rightarrow \quad (0, 2)\text{-sequence (Sobol')}$$

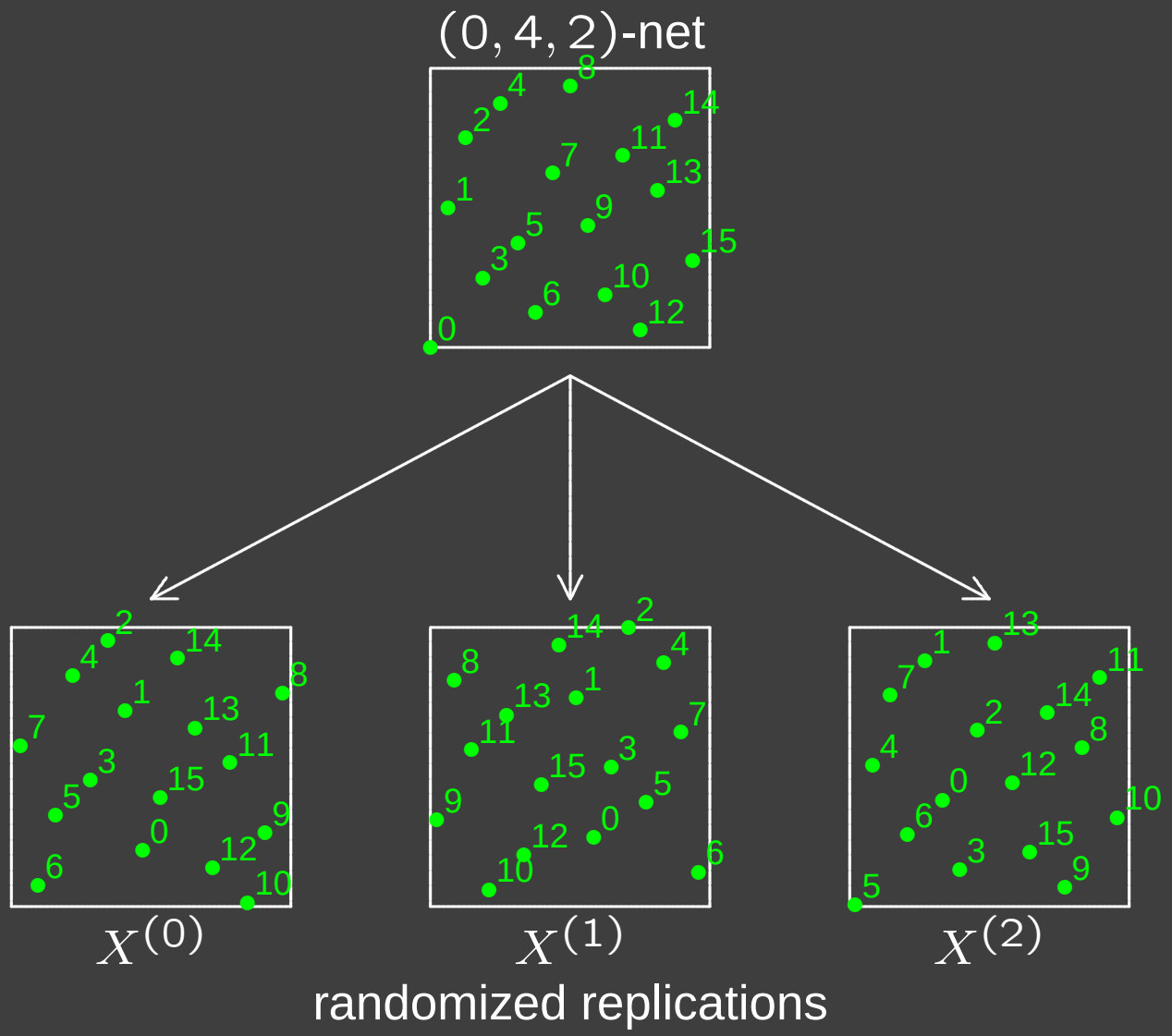
Efficient Multidimensional Sampling

- Exploit low dimensional structure of the integrand
- Multidimensional samples padded from independent randomized replications



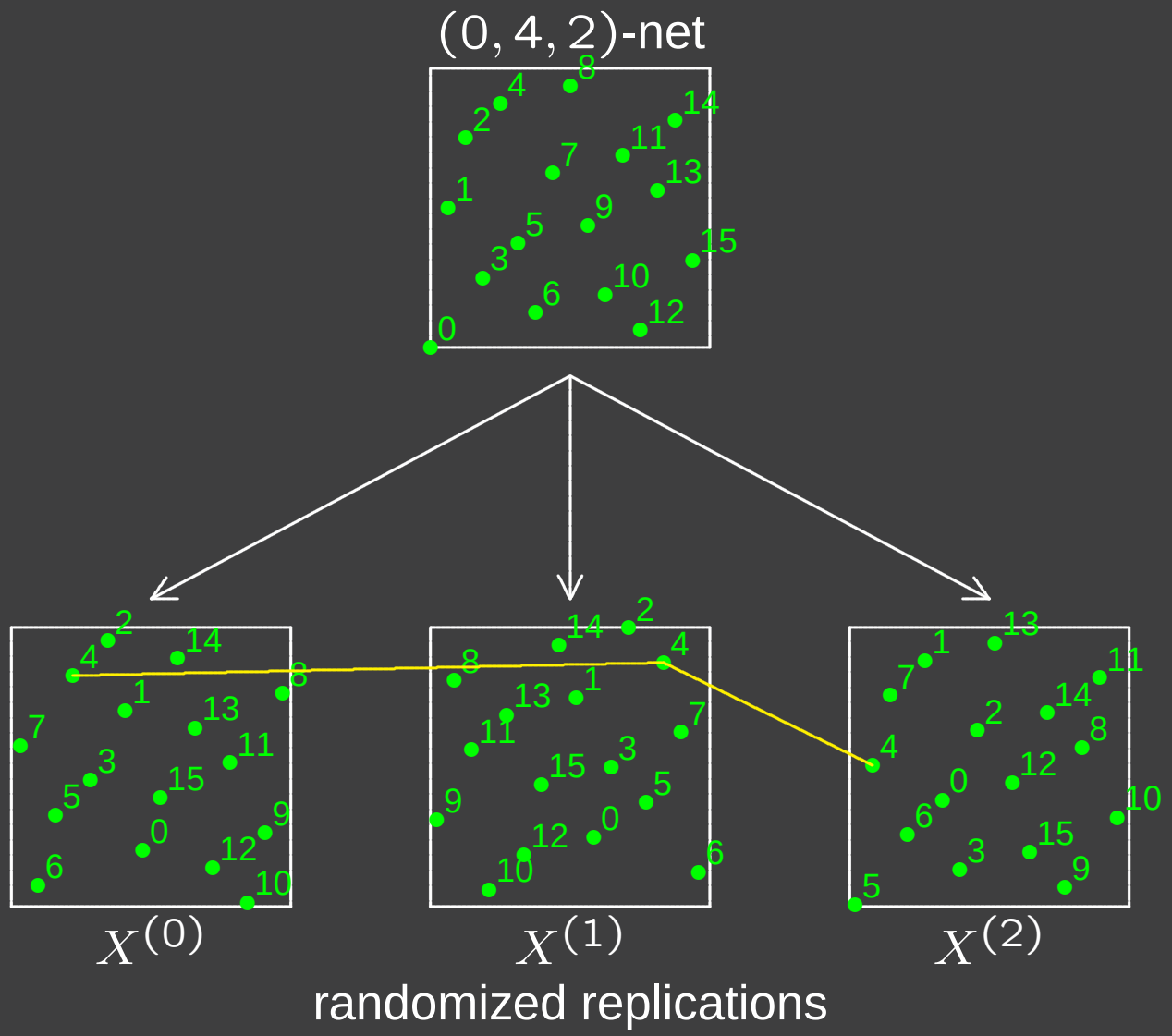
Efficient Multidimensional Sampling

- Exploit low dimensional structure of the integrand
- Multidimensional samples padded from independent randomized replications

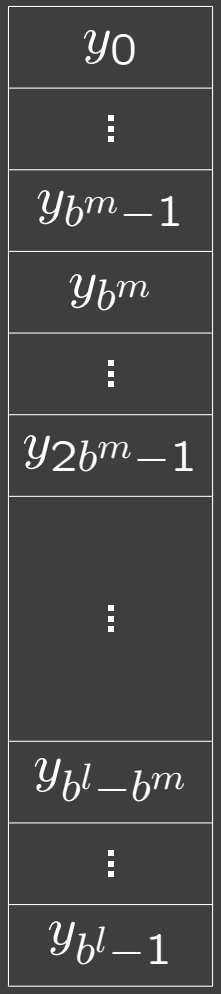


Efficient Multidimensional Sampling

- Exploit low dimensional structure of the integrand
- Multidimensional samples padded from independent randomized replications

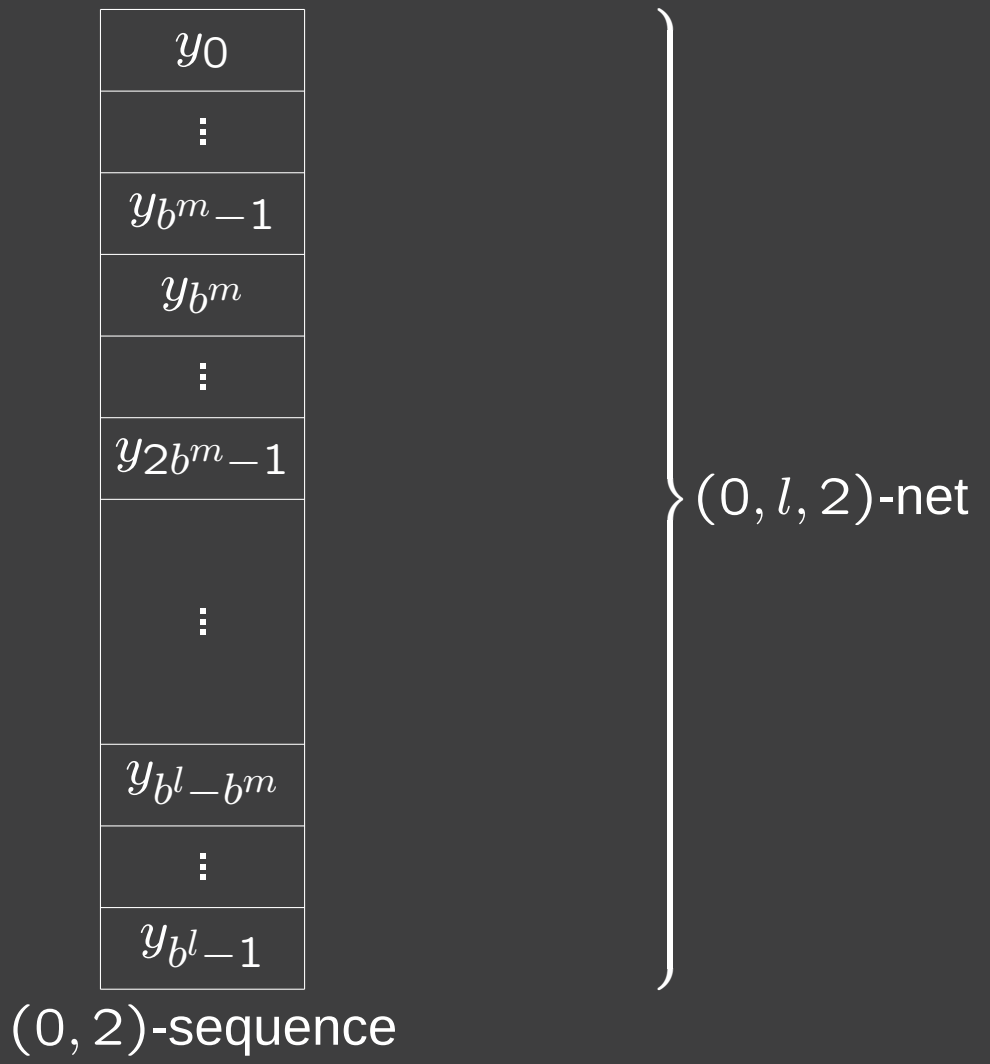


Efficient Trajectory Splitting

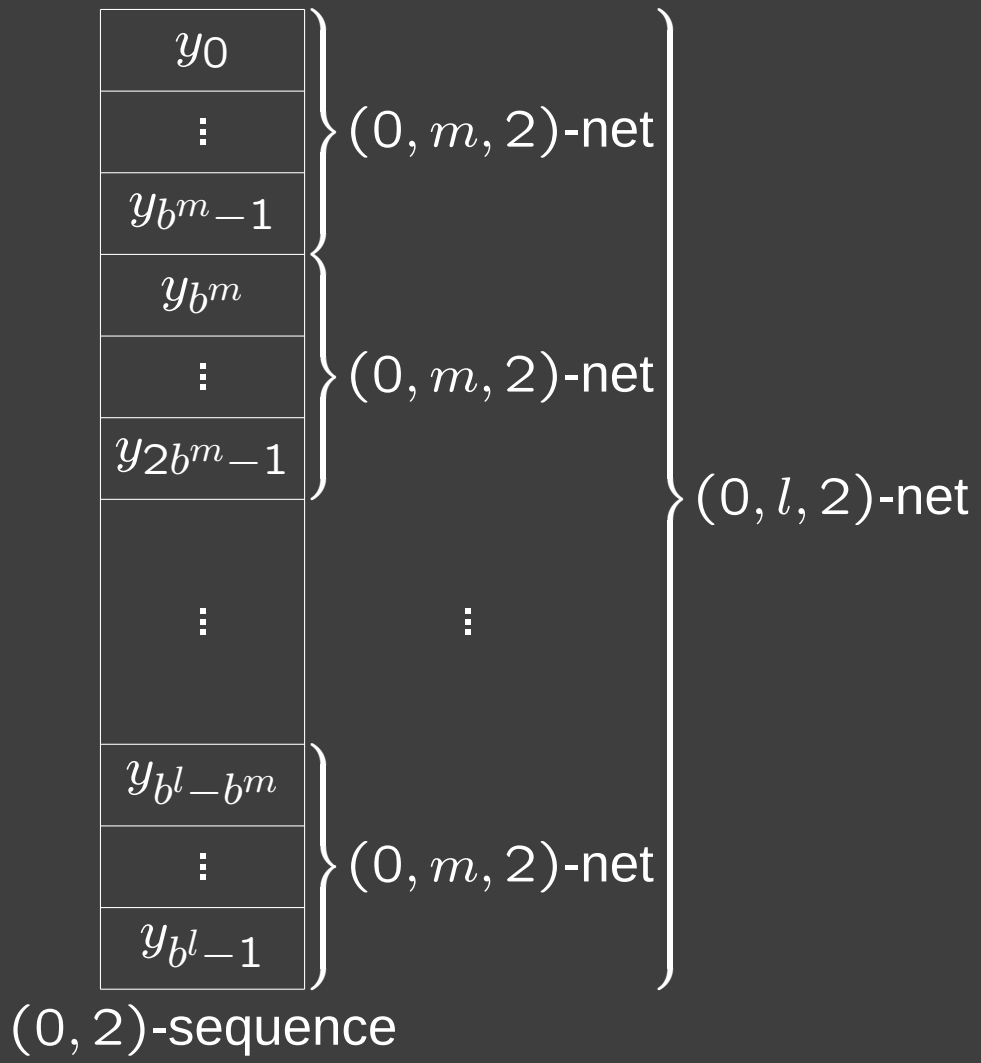


$(0, 2)$-sequence

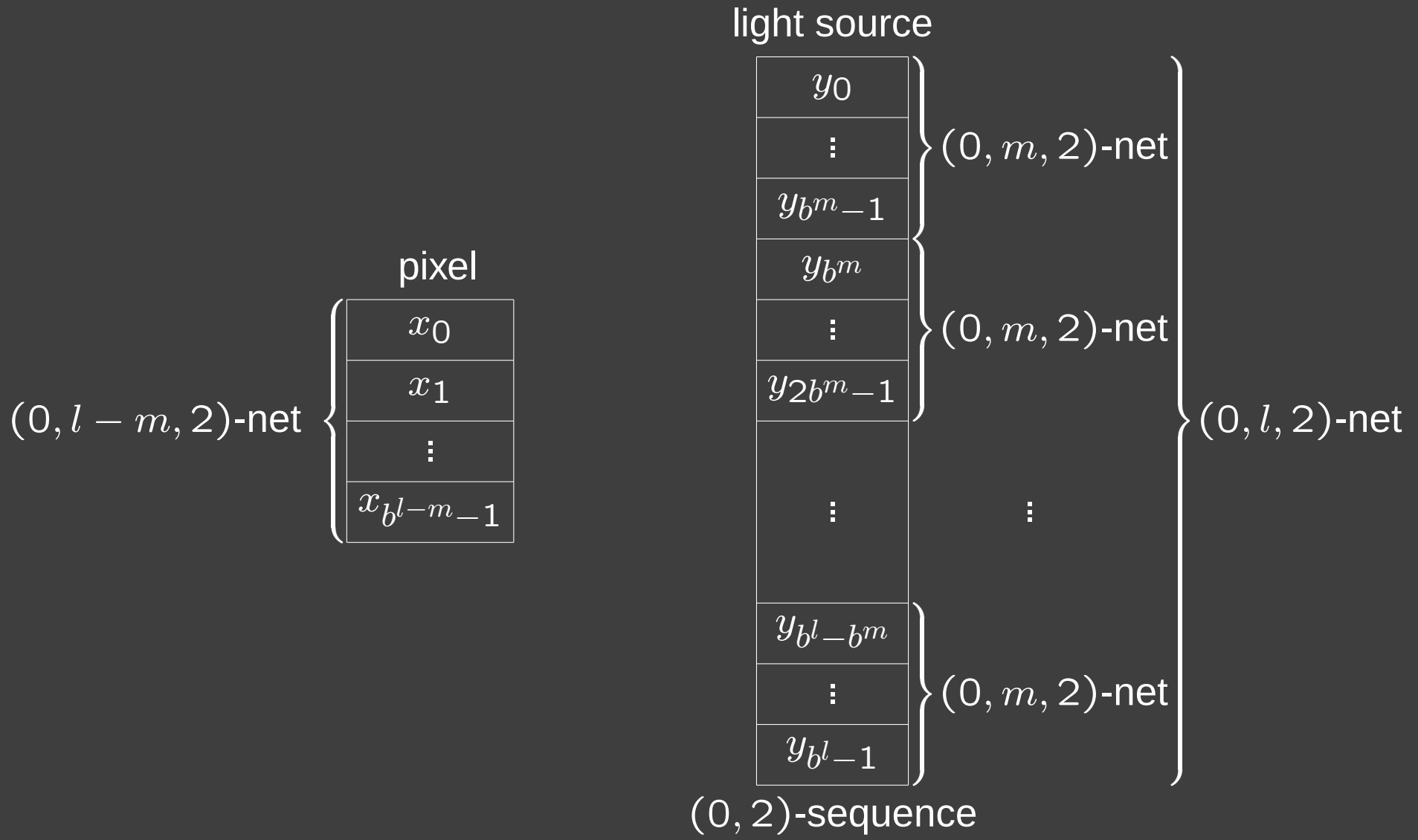
Efficient Trajectory Splitting



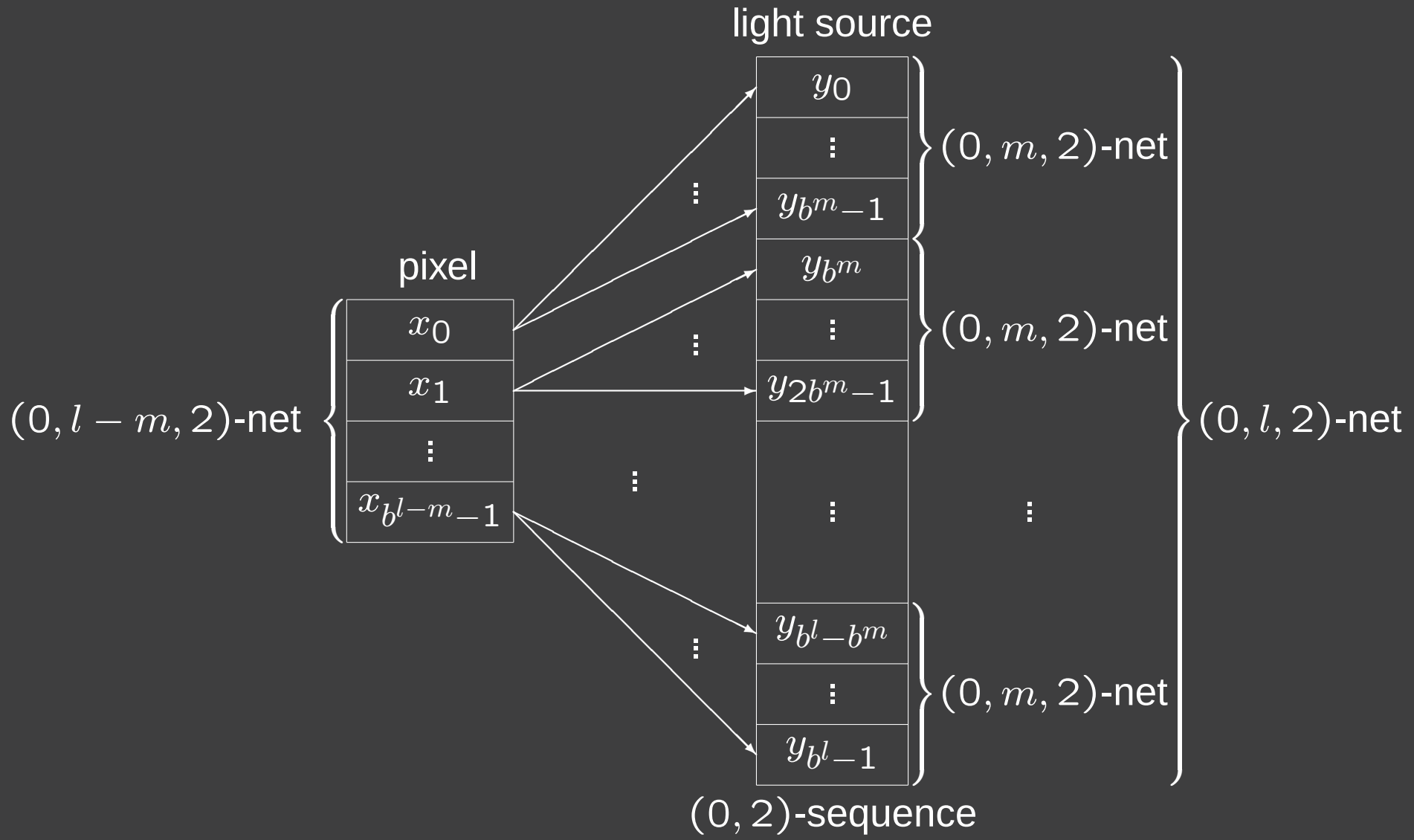
Efficient Trajectory Splitting



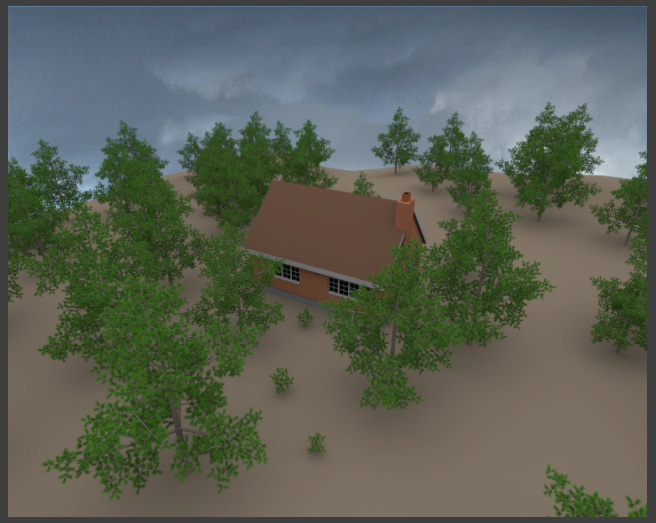
Efficient Trajectory Splitting

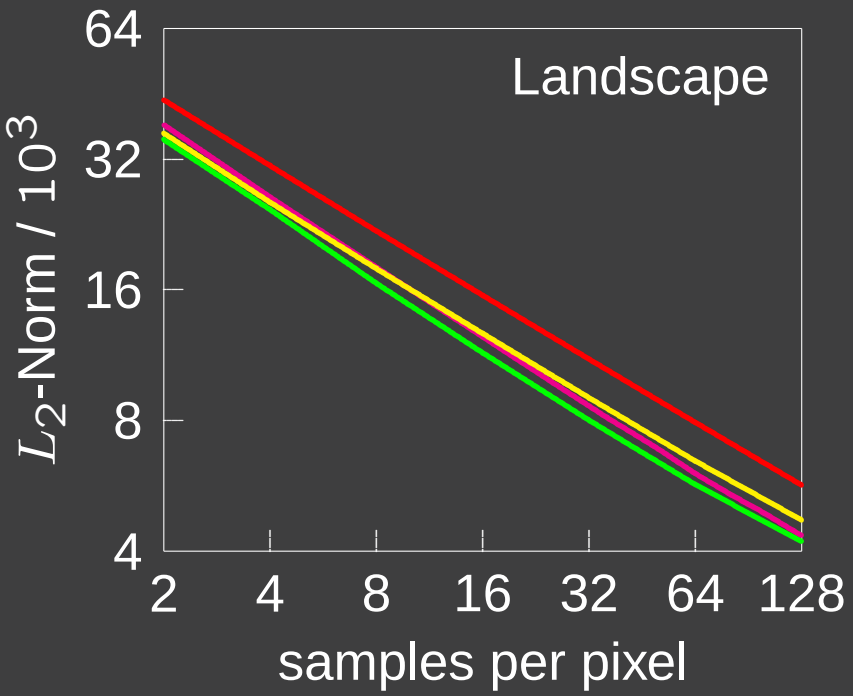


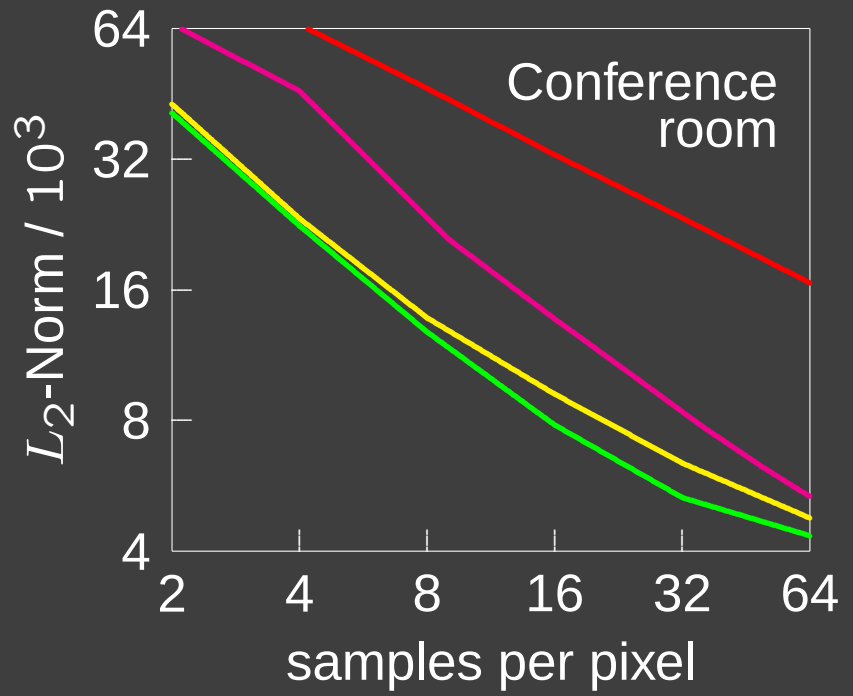
Efficient Trajectory Splitting



Results: Direct Illumination

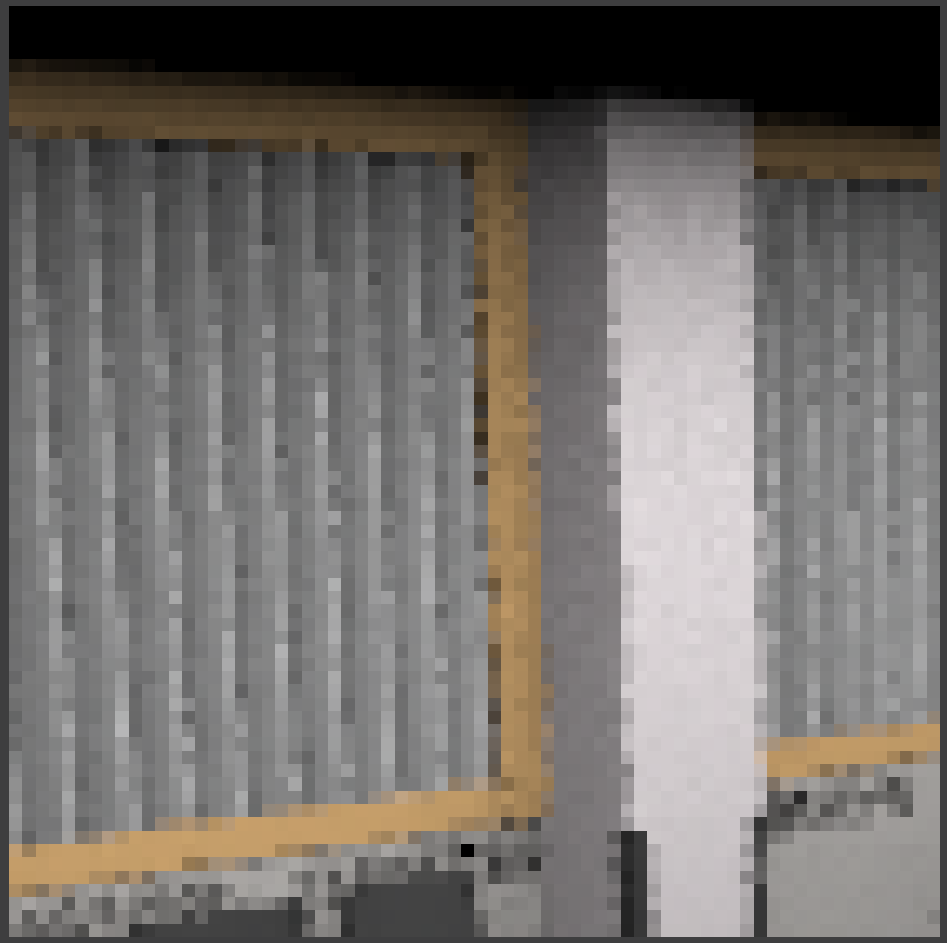




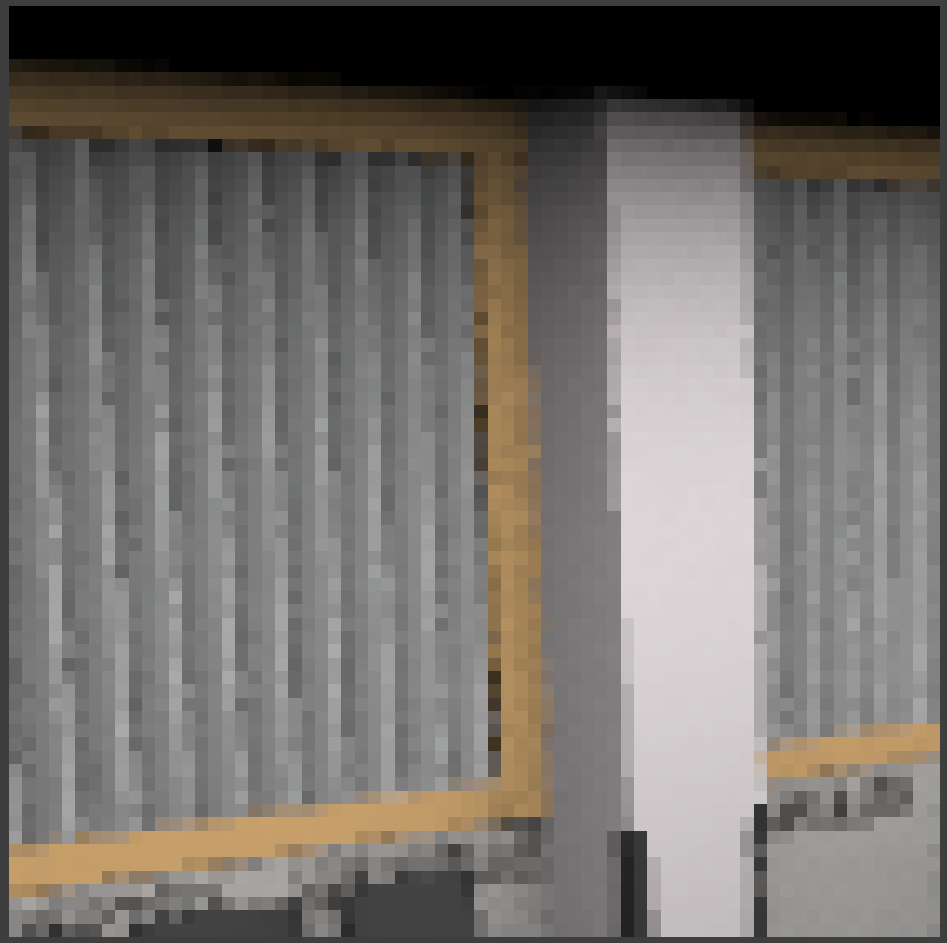


independent
jittered
Latin hypercube
random digit scrambled nets

Results: Antialiasing

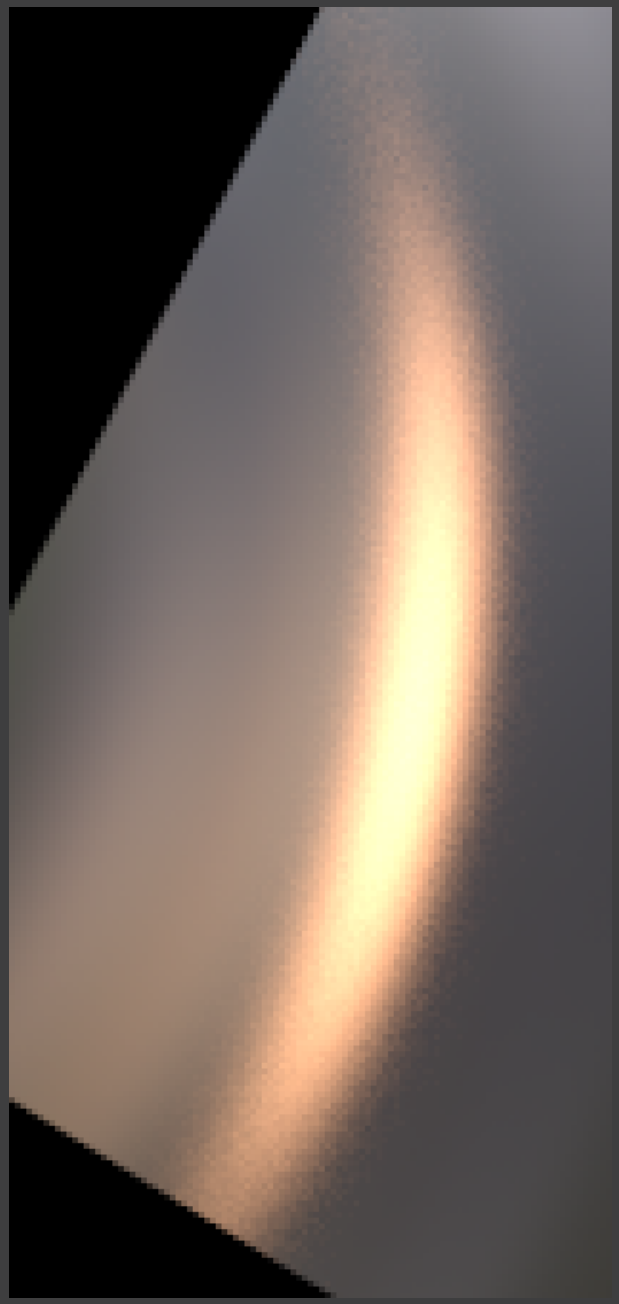


Latin hypercube

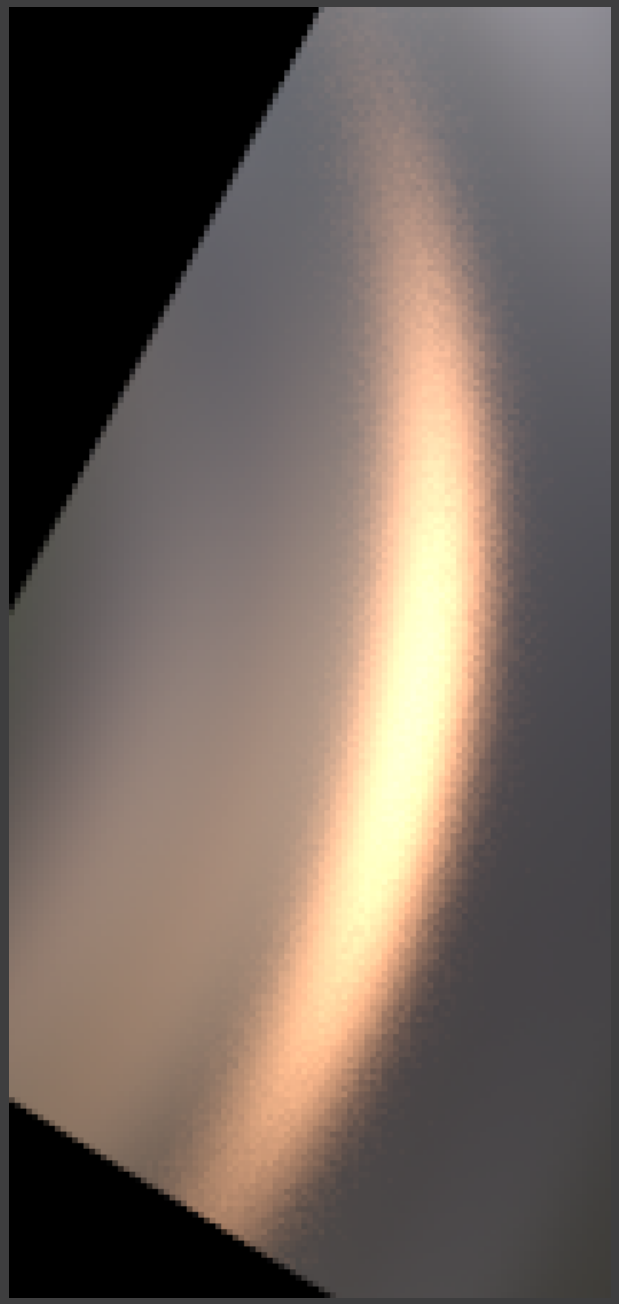


random digit scrambled nets

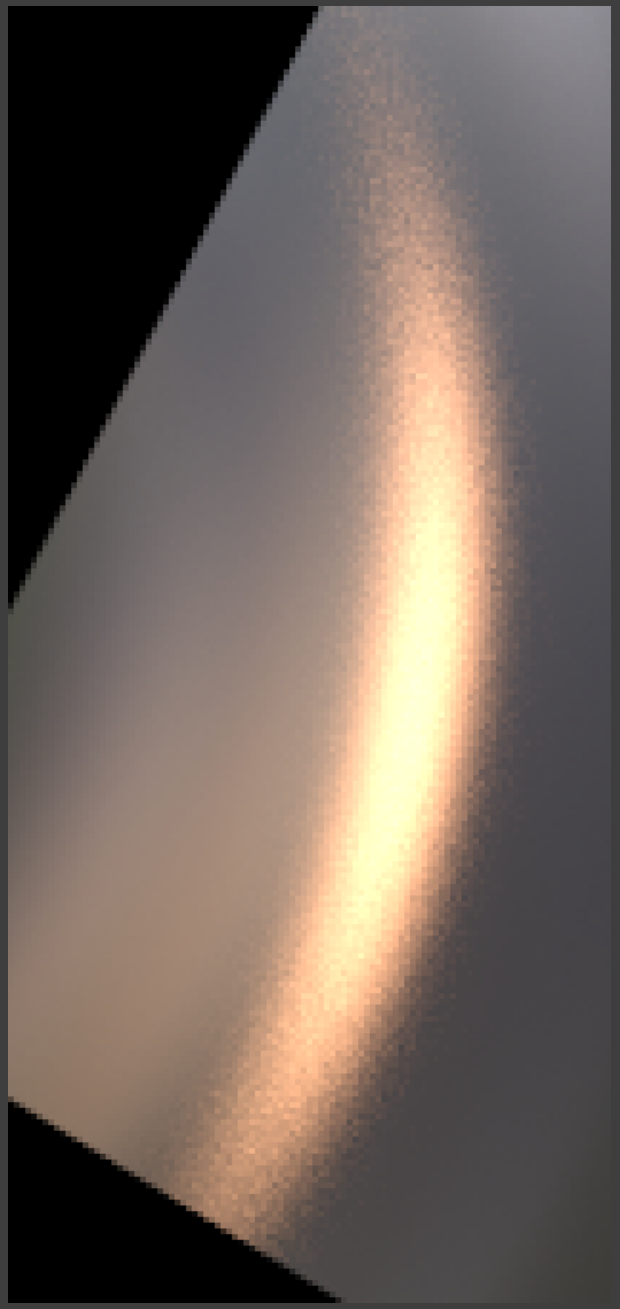
Results: HDRi-lighted Tetrahydron



jittered



random digit scrambled nets



Latin hypercube

Summary

- Generalized concept of stratification
  - extremely simple implementations
  - very efficient sample generation
- Efficient multidimensional sampling
- Efficient trajectory splitting
- Outperforms previous sampling schemes

- Check out our code and more at

www.uni-kl.de/AG-Heinrich/SamplePack.html

Interactive Global Illumination

Using

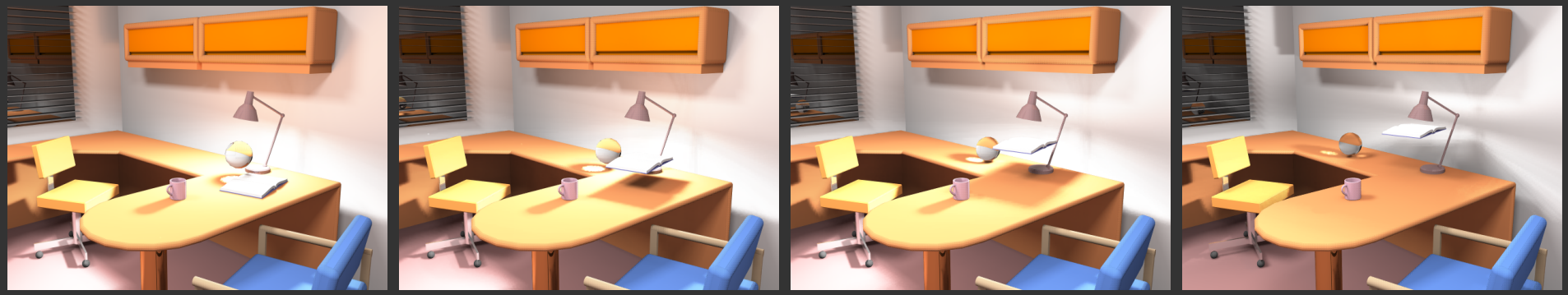
Fast Ray Tracing

Thomas Kollig

joint work with

Ingo Wald, Carsten Benthin, Alexander Keller, and Philipp Slusallek

Goal: Interactive Global Illumination



- Immediate feedback (at least 1 fps) in dynamic environments for
  - direct and indirect lighting by area light sources
  - reflections and refractions
  - caustics

  $\Rightarrow$ preliminary restriction to
    - diffuse, specular, and refractive material properties
    - direct caustics

- High image quality after a short time of no interaction

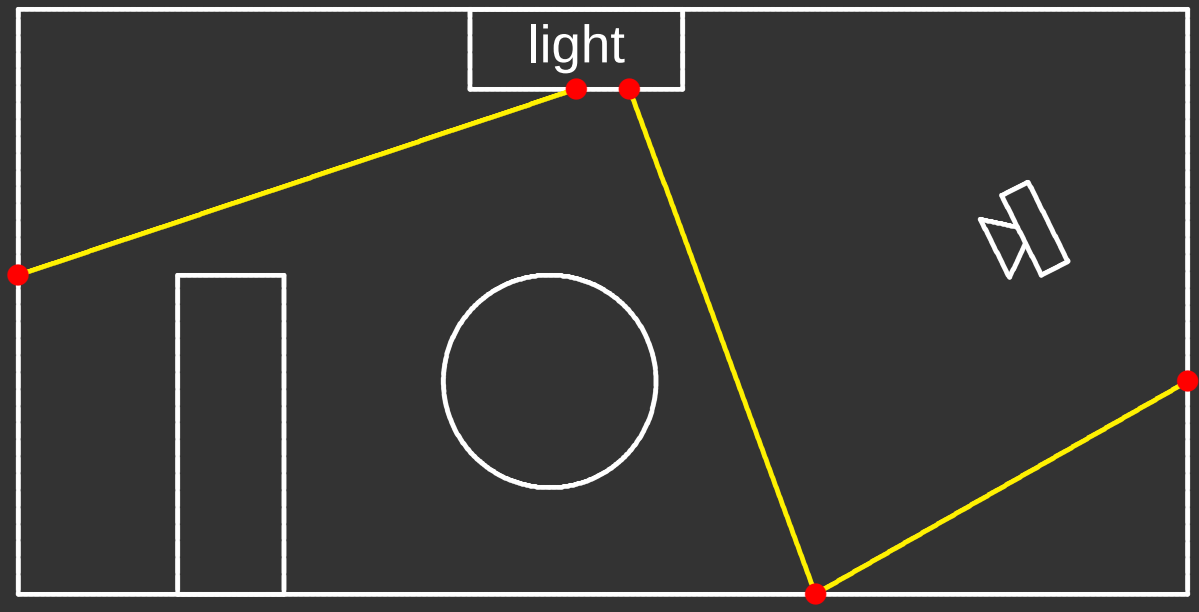
The Framework: Fast Ray Tracing

- Optimized ray tracing engine handles
  - distribution over a cluster of PCs
  - user intervention

- Constraints imposed by the system
  - coherent rays for efficient caching
  - small budget of rays per pixel and frame
  - parallelization on a non-shared memory system
  - precomputations unlikely to be amortized

- Global illumination algorithm is realized as shaders

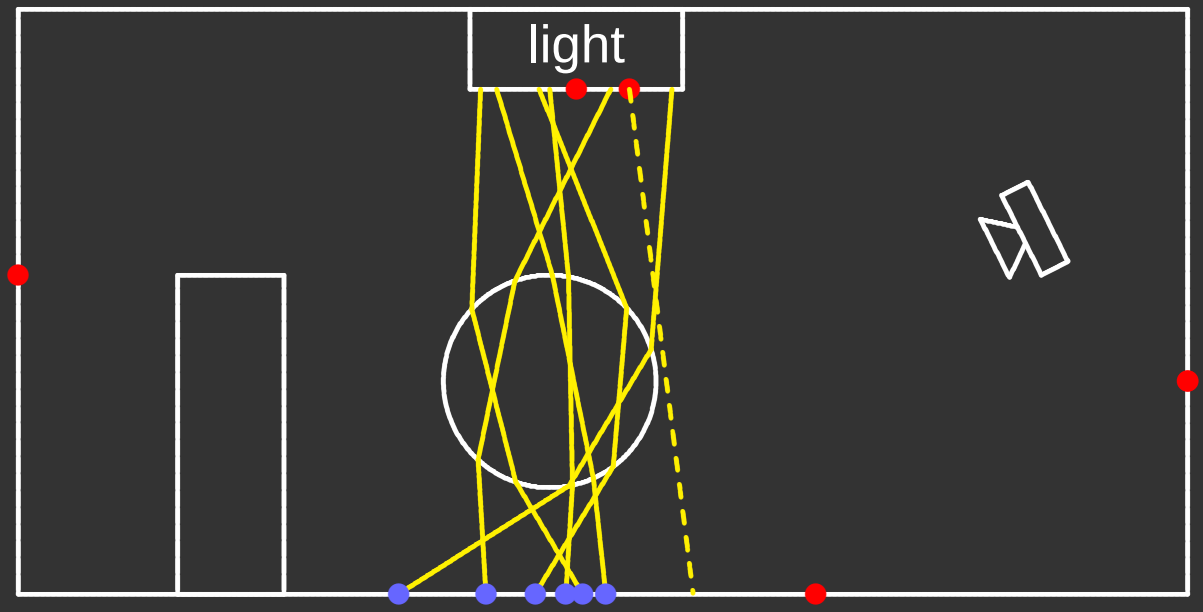
Algorithm Overview: Preprocessing



Point lights

$$P := (y_j, L_j)_{j=1}^{M}$$

Algorithm Overview: Preprocessing



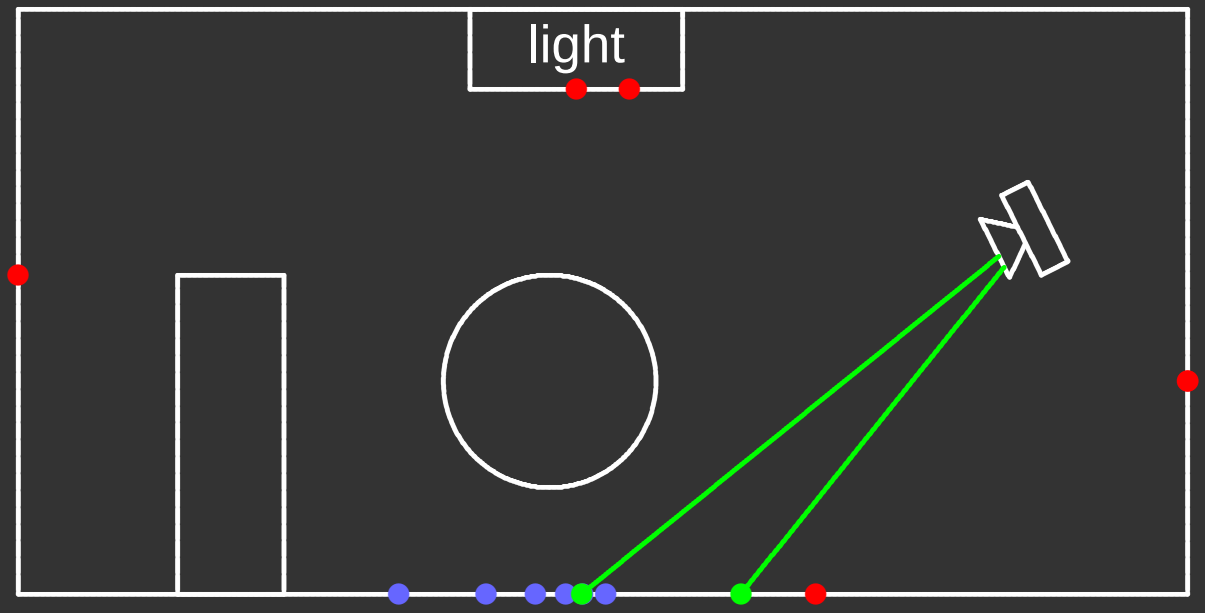
Point lights

$$P := (y_j, L_j)_{j=1}^{M}$$

Caustic photons

$$C := (z_j, \omega_j, \Phi_j)_{j=1}^{N}$$

Algorithm Overview: Illumination



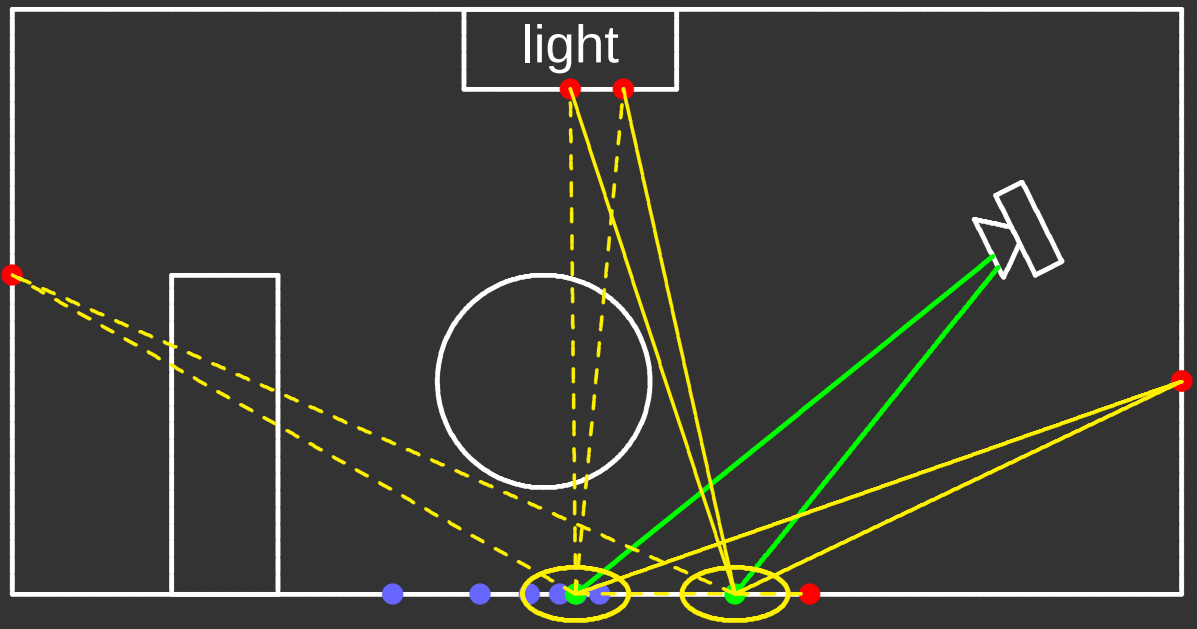
Point lights

$$P := (y_j, L_j)_{j=1}^{M}$$

Caustic photons

$$C := (z_j, \omega_j, \Phi_j)_{j=1}^{N}$$

Algorithm Overview: Illumination



Point lights

$$P := (y_j, L_j)_{j=1}^{M}$$

Caustic photons

$$C := (z_j, \omega_j, \Phi_j)_{j=1}^{N}$$

- Illumination computed by

$$L(x,\omega) \approx L_e(x,\omega) + \sum_{j=1}^{M} V(y_j,x) f_r(\omega_{y_j x}, x, \omega) L_j \frac{\cos\theta_{y_j} \cos\theta_x}{|y_j - x|^2}$$

$$+ \frac{1}{\pi r^2} \sum_{j=1}^{N} B_r(z_j, x) f_r(\omega_j, x, \omega) \Phi_j$$

Algorithm Overview: Rendering

- Eye path generation
  - random decision for

    diffuse $\rightarrow$ end eye path generation
    specular / refractive $\rightarrow$ prolong eye path

  - variance reduction by splitting the eye path at the first hitpoint

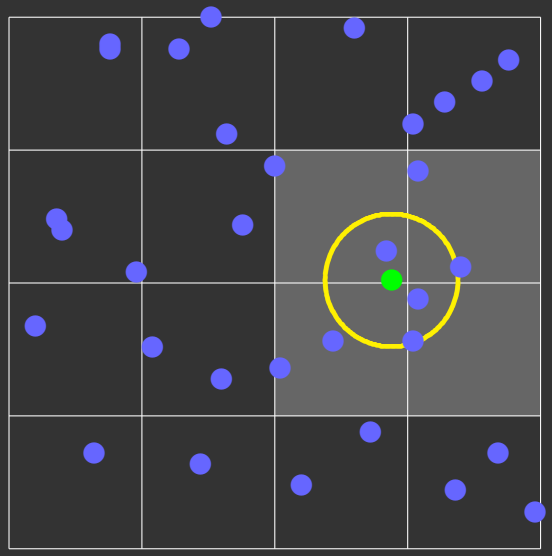
    $\Rightarrow$ at most 3 points per pixel have to be shaded

- Avoid spatial and temporal flickering by using the same random numbers
- Full solution during interaction
- Anti-aliasing by accumulation buffer during times of no interaction

Algorithm Details: Caustics

- Original photon mapping algorithms for storage and query are too slow
- Use fixed filter size r and store photons in a grid of resolution $2r$

  $\Rightarrow$ 8 voxels have to be looked up



- Only a fraction of the voxels contain photons

  $\Rightarrow$ hashing

Algorithm Details: Interleaved Sampling

- Pad 3×3 tiles with 9 different identifications over the whole image plane

6	7	8
3	4	5
0	1	2

Algorithm Details: Interleaved Sampling

- Pad 3×3 tiles with 9 different identifications over the whole image plane

6	7	8	6	7	8	6	7	8
3	4	5	3	4	5	3	4	5
0	1	2	0	1	2	0	1	2
6	7	8	6	7	8	6	7	8
3	4	5	3	4	5	3	4	5
0	1	2	0	1	2	0	1	2

Algorithm Details: Interleaved Sampling

- Pad 3×3 tiles with 9 different identifications over the whole image plane

6	7	8	6	7	8	6	7	8
3	4	5	3	4	5	3	4	5
0	1	2	0	1	2	0	1	2
6	7	8	6	7	8	6	7	8
3	4	5	3	4	5	3	4	5
0	1	2	0	1	2	0	1	2

- Different sets P_k of point lights and C_k of caustic photons for each identification k

- Straightforward and efficient parallelization
  - P_k and C_k computed on demand by each client
  - clients predominantly process pixels with equal k

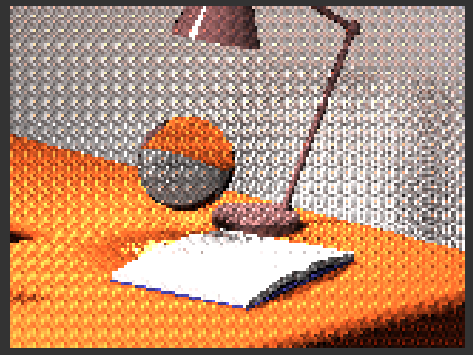
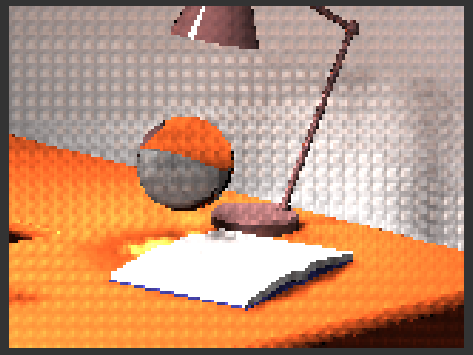
    $\Rightarrow$ no synchronization for global data structures

    $\Rightarrow$ no network communication between clients

Algorithm Details: Discontinuity Buffer

- Averaging the irradiance values of neighboring pixels if continuity is detected
  $\Rightarrow$ variance reduction
- Continuity checked by
  - path distance between endpoint of eye path and eye point
  - normal at the endpoint of the eye path

interleaving

no	5×5	5×5	5×5
			
no	no	3×3	5×5

discontinuity buffer

- Interleaved sampling and the discontinuity buffer perfectly complement each other

Algorithm Details: Randomized quasi-Monte Carlo

- Low discrepancy point set

  r randomized replications of A

$A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$

$$\begin{matrix} x_{1,1} & \ldots & x_{i,1} & \ldots & x_{m,1} \\ & & \vdots & & \\ x_{1,j} & \ldots & x_{i,j} & \ldots & x_{m,j} \\ & & \vdots & & \\ x_{1,r} & \ldots & x_{i,r} & \ldots & x_{m,r} \end{matrix}$$

Algorithm Details: Randomized quasi-Monte Carlo

- Low discrepancy point set

  r randomized replications of A

$A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$

$$\begin{matrix} x_{1,1} & \ldots & x_{i,1} & \ldots & x_{m,1} \\ \vdots & & \vdots & & \vdots \\ x_{1,j} & \ldots & x_{i,j} & \ldots & x_{m,j} \\ \vdots & & \vdots & & \vdots \\ x_{1,r} & \ldots & x_{i,r} & \ldots & x_{m,r} \end{matrix}$$

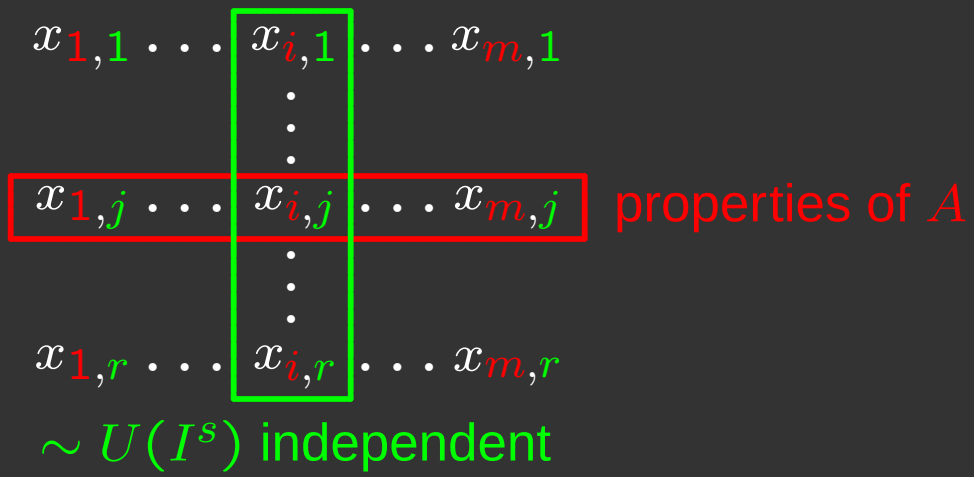
properties of A

Algorithm Details: Randomized quasi-Monte Carlo

- Low discrepancy point set

  r randomized replications of A

$A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$

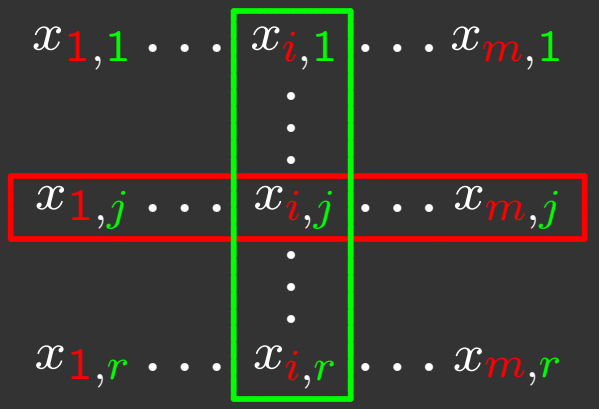


Algorithm Details: Randomized quasi-Monte Carlo

- Low discrepancy point set

  r randomized replications of A

$A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$



Unbiased estimator

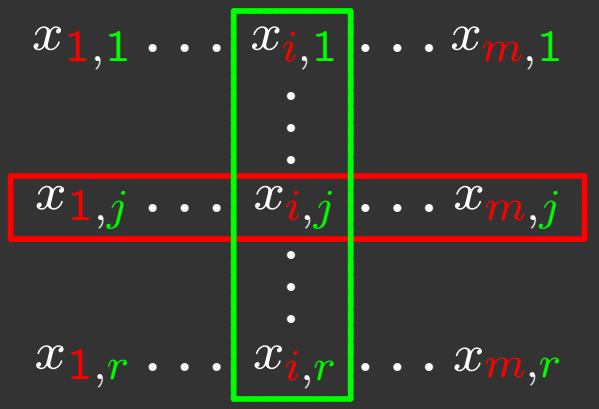
$$\int_{I^s} f(x)dx \approx \frac{1}{r}\sum_{j=1}^{r}\frac{1}{m}\sum_{i=1}^{m} f(x_{i,j})$$

Algorithm Details: Randomized quasi-Monte Carlo

- Low discrepancy point set

  r randomized replications of A

$A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$



Unbiased estimator

$$\int_{I^s} f(x)dx \approx \frac{1}{r}\sum_{j=1}^{r}\frac{1}{m}\sum_{i=1}^{m} f(x_{i,j})$$

- Randomization: $x_{i,j} := ((2^{32} \cdot a_i) \text{ xor } b_j) \cdot 2^{-32}$

Algorithm Details: Randomized quasi-Monte Carlo

- Low discrepancy point set $A = \{a_1, a_2, \ldots, a_m\}$ where $a_i \in I^s$

  r randomized replications of A

$$\begin{array}{ccccc} x_{1,1} & \cdots & x_{i,1} & \cdots & x_{m,1} \\ \vdots & & \vdots & & \vdots \\ x_{1,j} & \cdots & x_{i,j} & \cdots & x_{m,j} \\ \vdots & & \vdots & & \vdots \\ x_{1,r} & \cdots & x_{i,r} & \cdots & x_{m,r} \end{array}$$

properties of A

$\sim U(I^s)$ independent

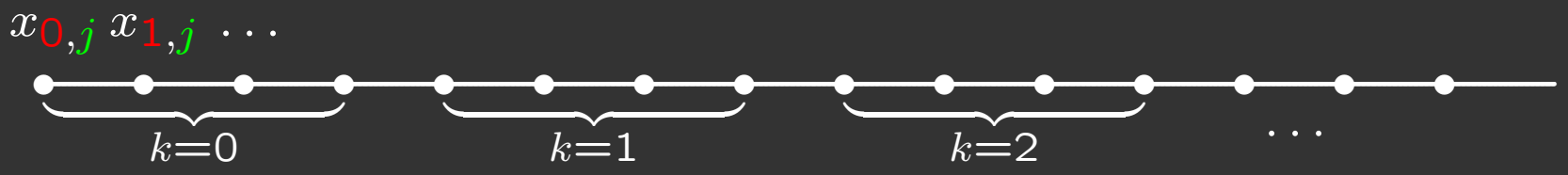
Unbiased estimator

$$\int_{I^s} f(x)dx \approx \frac{1}{r}\sum_{j=1}^{r}\frac{1}{m}\sum_{i=1}^{m} f(x_{i,j})$$

- Randomization: $x_{i,j} := ((2^{32} \cdot a_i) \text{ xor } b_j) \cdot 2^{-32}$

- Subsequent subsequences for generating the sets P_k and C_k

  $\Rightarrow$ discontinuity buffer joins different subsequences in case of continuity



Results

- Cluster of dual AMD AthlonMP 1800+ machines with 512 Mb of RAM
  - fully switched 100 Mb Ethernet
  - server connected by a single Gigabit uplink

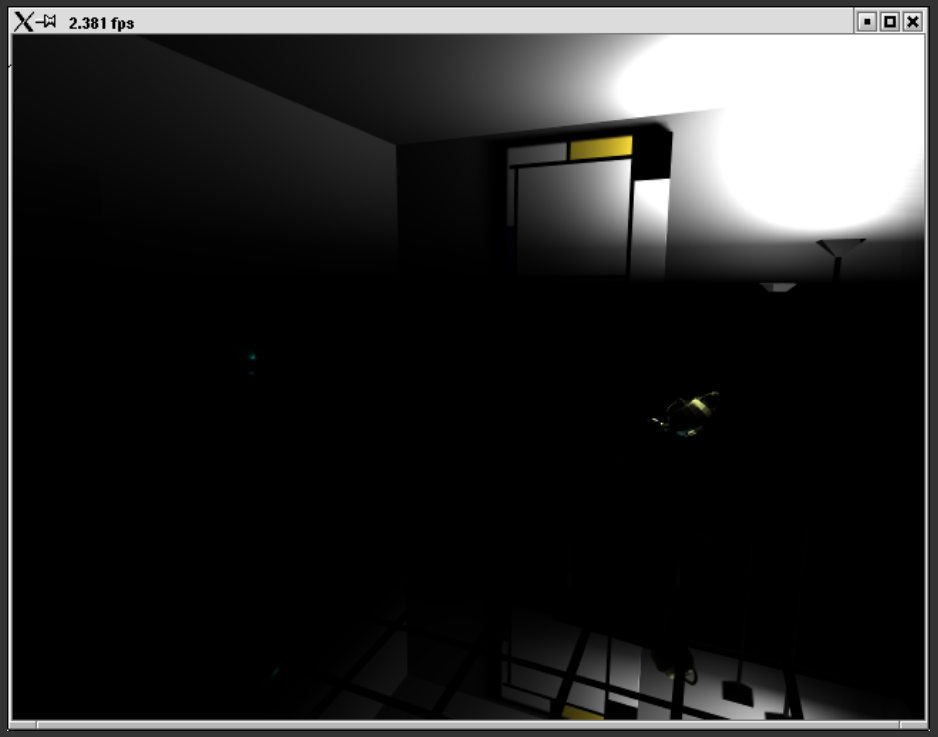
Results

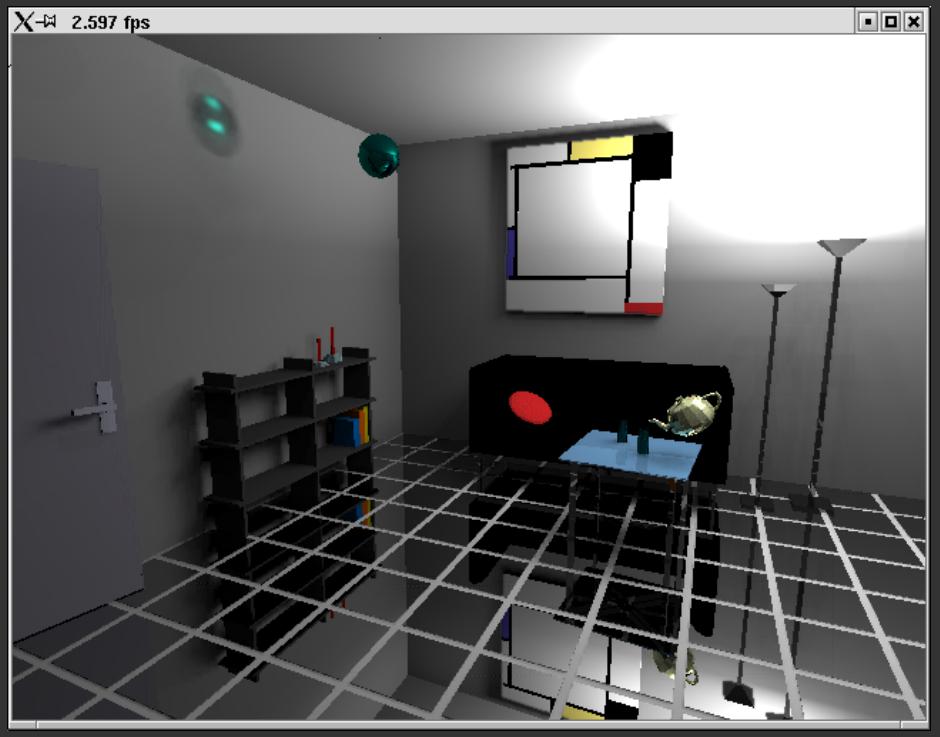
- Cluster of dual AMD AthlonMP 1800+ machines with 512 Mb of RAM
  - fully switched 100 Mb Ethernet
  - server connected by a single Gigabit uplink

- Small set of parameters
  $\Rightarrow$ interactively trade off rendering speed for image quality
- Frame rate scales up to 5 fps at a resolution of 640×480 pixels:
  - discontinuity buffer calculations have to be done on the server
  - network bandwidth
  - server workload
- Image quality scales without bottleneck over the range of available clients

Simulation Quality: “Invisible Date”

- 9,000 triangles, 2 area light sources
- 2.6 fps on 8 clients





- Direct versus indirect lighting

Simulation Quality: Conference Room Scene

- 290,000 triangles, 104 area light sources
- 1.7 fps on 12 clients

- Dynamic versus converged image

Summary and Future Work

- Interactive global illumination system by
  - distributed, fast ray tracing engine
    - constraining global illumination algorithm design
  - hashed photon maps
  - parallelization by interleaved sampling
  - variance reduction by discontinuity buffer
  - variance reduction by randomized quasi-Monte Carlo
- Server bottleneck limits resolution and frame rate
- Image quality scales without bottleneck
- Extensions to arbitrary material properties and high order caustics

Random Walk Radiosity with generalized transition probabilities

Mateu Sbert
Institut d'Informàtica i Aplicacions, Universitat de Girona
Alex Brusi
Universitat de Girona
Robert Tobler
Institut für Computegraphik, Vienna University of Technology
Werner Purgathofer
Institut für Computergraphik, Vienna University of Technology

Abstract

In this paper we study random walk estimators for radiosity with generalized transition and absorption probabilities. That is, a path will travel from patch to patch according to an arbitrary transition probability, and survive or be absorbed in it according to another arbitrary absorption probability. The estimators studied so far, those with arbitrary absorption probabilities but with the Form Factors as transition probabilities, are obviously a particular case of the more general case presented here. Practical applications of random walks with generalized probabilities are given. Closed forms for the variances are found, together with necessary and sufficient conditions for their existence. The variances are shown to fulfill a system of equations, which is a classical result by Halton. Some particular cases are studied, including null variance estimators, which represent the optimal case.

Keywords: Radiosity, Monte Carlo, Random Walk, Variance

1 Introduction

Discrete or continuous random walk estimators have been widely used in radiosity. Gathering random walk proceeds sending paths from the patches of interest to *gather* energy when a source is hit. Path-tracing [7], and even distributed ray-tracing [2, 24] can be considered as the limiting case of gathering random walk for the non-discrete case (without shadow rays). Shooting random walk *shoots* paths carrying energy from the sources, to update the visited patches [11], [1]. The techniques in [20, 4] can be seen as a breadth-first approach to a shooting random walk estimator, which in turn would be the depth-first approach. Bidirectional ray-tracing [23, 9] is a mixture of non-discrete shooting and gathering. The random walk proceeds according to the Form Factor probability transitions [20, 11, 4], or to biased ones [12, 10]. The survival (or not absorption) probability on a patch has usually been considered equal to its reflectivity. An exception to this survival probability is found in [10], where the received importance was considered instead of the reflectivity. In [12] we also find a short discussion under the term *survival biasing*. Also, infinite path length estimators can be considered those where the survival probability is equal to one. A study of generalized absorption probabilities is found in [19]. In it the finite path length estimators [14] and infinite ones [16] are derived as particular cases of this generalized one.

In this paper we will study shooting and gathering estimators resulting of considering any transition and survival (or not absorption) probability. That is, we will relax the assumption that the transition probabilities are the Form Factors.

The organization of this paper is as follows: In section 2 the previous work on random walk radiosity estimators is presented. In section 3 we study the gathering estimator with generalized absorption and transition probabilities. A necessary and sufficient condition for the existence of the variances is given, together with an heuristics to be used in some practical situations. A system of equations that fulfill the variances is given, and some particular estimators are studied. The generalized gathering estimator is also shown to have the same complexity (under certain constraints) as established in [14]. And we show how to generalize the given formulae for the non-diffuse case, that is, the general Rendering equation [7]. Next, in section 4, the shooting estimator is studied. Optimal survive probabilities, for the case when we keep the Form Factors as transition probabilities and are interested in the whole scene, are given. The resulting estimator happens to be the one with survival probability equal to the reflectivity. Some particular cases are given, and the same complexity results as for the gathering case are shown. We present some experimental evidence in section 5, and finally, in section 6 we present some conclusions and ideas for future research.

2 Previous Work

In [14] the three estimators defined in [21] were studied, together with their gathering dual ones. In [16] the i nfinite path length estimators were characterized , and in [18] it was proved that the best finite path length estimator better than the *biased* infinite path length one. Finally, in [17] the variances for the previous shooting estimators for any general source selection probability were obtained, and it was also proved that the results obtained so far were extensible to the pure particle tracing case, that is, when we keep the impinging point on a patch as next exiting point. The obtained results are summarized in table 1.

Table 1: Different Random Walk estimators. The meaning of the different quantities is in table 3.

Shooting	**Patch scored**	**Variance**
$\frac{\Phi_T}{1-R_i}$	last	$\sum_s \Phi_s \frac{R_i b_{is}}{A_i p_s (1-R_i)} - b_i^2$
$\frac{\Phi_T}{R_i}$	all but last	$\sum_s \Phi_s \frac{R_i b_{is}}{A_i p_s} (\frac{1}{R_i} + 2\xi_i) - b_i^2$
Φ_T	all	$\sum_s \Phi_s \frac{R_i (1+2R_i\xi_i) b_{is}}{A_i p_s} - b_i^2$
$infinite$	all	$\sum_s \Phi_s \frac{(1+2R_i\xi_i)\beta_{is}}{A_i p_s} - b_i^2$
Gathering	**Patch scored**	**Variance**
$\frac{E_s}{1-R_s}$	last	$\frac{R_i}{p_i} \sum_s \frac{E_s}{(1-R_s)} b_{is} - {b_i}^2$
$\frac{E_s}{R_s}$	all but last	$\frac{R_i}{p_i} \sum_s \frac{E_s+2b_s}{R_s} b_{is} - {b_i}^2$
E_s	all	$\frac{R_i}{p_i} \sum_s (E_s + 2b_s) b_{is} - {b_i}^2$
$infinite$	all	$\frac{1}{p_i} \sum_s (E_s + 2b_s) \beta_{is} - {b_i}^2$

A feature common to the studied estimators is that the transition

Table 2: Variances for Random Walk estimators with generalized absorption probabilities. The meaning of the different quantities is in table 3.

$shooting, \Phi_T$	$\theta_i \sum_s \Phi_s \frac{(1+2R_i\xi_i)\epsilon_{is}}{A_i p_s} - b_i^2$
$gathering, E_s$	$\frac{\theta_i}{p_i} \sum_s (E_s + 2b_s)\epsilon_{is} - {b_i}^2$

probabilities used are the Form Factors, and as survival (or not absorption) probability is used the reflectivity of the patch (except of course for the infinite path length, where the survival probability is always 1). In [19] this second assumption was relaxed, that is, generalized absorption probabilities were considered. The resulting variances for the so generalized Φ_T and E_s estimators are in table 2. The already studied cases, when the survival probability is equal to reflectivity and the infinite case, can then be seen as particular cases. The Form Factors were still the transition probabilities. The usefulness of the estimators obtained can be seen when considering survival probabilities proportional to importance (or better to received importance). This has been used in [10].
In this paper we will relax the assumption of the Form Factors as being the transition probabilities. Thus, we will study a generalized random walk with any arbitrary survival and transition probabilities. Next we will study the gathering case.

Table 3: Meaning of the different quantities appearing in table 1. The suffix i means for patch i, suffix s indexes the sources.

E_i	Emissivity
b_i	Reflected radiosity = $B_i - E_i$
β_i	idem with each reflectivity substituted by its square
Φ_s	Emitted power
A_i	Area
R_i	Reflectivity
θ_i	Generalized survival probability
ξ_i	Received power (or radiosity) due to self-emitted unit power (or emittance)
b_{is}	Reflected radiosity on i due to source s. $b_i = \sum_s b_{is}$
β_{is}	idem with each reflectivity substituted by its square. $\beta_i = \sum_s \beta_{is}$
ϵ_{is}	Reflected "radiosity"with each reflectivity substituted by its square divided by the survival probability. Doesn't always have a physical meaning. $\epsilon_i = \sum_s \epsilon_{is}$
p_i	Probability for a path to begin at i

3 A gathering estimator with generalized transition and absorption probabilities

We will consider the discrete random walk here, that is, the one which proceeds according to patch-to-patch Form Factors. However, the formulae and results obtained are also valid for point-to-point Form Factors, as shown in [17].

Let us first consider what the expected value of any unbiased Monte Carlo estimator should be for the radiosity of a patch. Let us suppose that the emittance of source s is E_s, b_i is the reflected radiosity of patch i due to the received power (that is, $b_i = B_i - E_i$, and so for a non-emitter patch, it equals the total radiosity), F_{kl} denotes the Form Factor from patch k to patch l, and R_k denotes the reflectance of patch k. Then we have, by developing the radiosity system in a Neumann series (dropping the zero order term):

$$\begin{aligned} b_i &= R_i \sum_s E_s F_{is} + R_i \sum_h \sum_s E_s F_{ih} R_h F_{hs} \\ &\quad + R_i \sum_h \sum_j \sum_s E_s F_{ih} R_h F_{hj} R_j F_{js} + \cdots \end{aligned}$$

This can be expressed as:

$$b_i = b_i^{(1)} + b_i^{(2)} + b_i^{(3)} + \cdots$$

where $b_i^{(1)} = R_i \sum_s E_s F_{is}$, $b_i^{(2)} = R_i \sum_s \sum_h E_s F_{ih} R_h F_{hs}$, $b_i^{(3)} = R_i \sum_s \sum_h \sum_j E_s F_{ih} R_h F_{hj} R_j F_{js}$ and so on. That is, $b_i^{(1)}$ represents the radiosity due to direct illumination, $b_i^{(2)}$ represents the radiosity after one bounce, and so on. It is also useful to define the following quantities:

$$b_{is} = b_{is}^{(1)} + b_{is}^{(2)} + \cdots$$

$b_{is}^{(1)}$ represents the radiosity due to direct illumination from source s, $b_{is}^{(2)}$ represents the radiosity after one bounce from source s, and so on. It is clear that:

$$b_i = \sum_s b_{is}$$

Now let us consider the following simulation. A path γ starts from patch i with probability p_i (this probability can be considered as the initial or emitted importance of the patch), and from here on it evolves according to the transition probabilities p_{ij}. It will then be absorbed in patch j with probability $1 - \theta_j$, and survive with probability θ_j. Next we define the random variables $\widehat{b}_i^{(1)}, \widehat{b}_i^{(2)}, \widehat{b}_i^{(3)}, \ldots$ in the following way:
All of these random variables are initially null. If the path γ happens to arrive at source s with length l, and if $i, h_1, h_2, \ldots, h_{l-1}, s$ is the trajectory the path has followed, then the value of $\widehat{b}_i^{(l)}$ is set to $R_i \frac{F_{ih_1}}{p_{ih_1}} \frac{R_{h_1}}{\theta_{h_1}} \frac{F_{h_1h_2}}{p_{h_1h_2}} \cdots \frac{R_{h_{l-1}}}{\theta_{h_{l-1}}} \frac{F_{h_{l-1}s}}{p_{h_{l-1}s}} \frac{E_s}{p_i}$. Let us also define a new random variable $\widehat{b}_i$ as:

$$\widehat{b}_i = \sum_l \widehat{b}_i^{(l)}$$

Now let us find the expected value of these random variables $\widehat{b}_i^{(l)}$. Applying the definition of expected value, and remembering that the probability of selecting patch i is p_i, and that the conditional probability of landing on source s just when leaving patch i is F_{is}, we have

$$E(\widehat{b}_i^{(1)}) = \sum_s R_i \frac{F_{is}}{p_{is}} \frac{E_s}{p_i} \times p_i \times p_{is} = b_i^{(1)}$$

To go from patch i to a source s in a path of length two we can pass through any patch h (after surviving on it with probability θ_h), so we have

$$\begin{aligned} E(\widehat{b}_i^{(2)}) &= \sum_h \sum_s R_i \frac{F_{ih}}{p_{ih}} \frac{R_h}{\theta_h} \frac{F_{hs}}{p_{hs}} \frac{E_s}{p_i} \times p_i p_{ih} \theta_h p_{hs} \\ &= b_i^{(2)} \end{aligned}$$

and so on. Then, we have

$$E(\widehat{b}_i) = E(\widehat{b}_i^{(1)} + \widehat{b}_i^{(2)} + \cdots) = E(\widehat{b}_i^{(1)}) + E(\widehat{b}_i^{(2)}) + \cdots$$
$$= b_i^{(1)} + b_i^{(2)} + \cdots = b_i$$

So it is clear that the random variable $\widehat{b}_i^{(l)}$ is a centered estimator for the radiosity due to the power arrived on patch i after $l-1$ bounces, and the sum of this whole family of estimators gives a new centered estimator $\widehat{b}_i$ which corresponds to the total radiosity of patch i due to the power arrived after any number of bounces. Our next aim is to obtain the variance for this estimator. We will use a similar approach to the one in [14], [16] and [19]. We can decompose $Var(\widehat{b}_i)$ in the following way

$$\begin{aligned} Var(\widehat{b}_i) &= Var(\widehat{b}_i^{(1)} + \widehat{b}_i^{(2)} + \cdots) \\ &= E(\left(\widehat{b}_i^{(1)} + \widehat{b}_i^{(2)} + \cdots\right)^2) - \left(E(\widehat{b}_i)\right)^2 \\ &= E(\widehat{b}_i^{(1)2}) + E(\widehat{b}_i^{(2)2}) + \cdots \\ &+ 2\sum_{1\le n<m} E(\widehat{b}_i^{(n)}\widehat{b}_i^{(m)}) - {b_i}^2 \end{aligned} \quad (1)$$

The terms of the form $E(\widehat{b}_i^{(n)}\widehat{b}_i^{(m)})$ are not zero, because if a path arrives with length n on source s it can also arrive later at source s' with length m. Next we find them:

$$\begin{aligned} & E(\widehat{b}_i^{(n)}\widehat{b}_i^{(m)}) \\ &= \sum_s \sum_{s'} \sum_{h_1} \cdots \sum_{h_{n-1}} \sum_{h_{n+1}} \cdots \sum_{h_{m-1}} R_i \frac{F_{ih_1}}{p_{ih_1}} \frac{R_{h_1}}{\theta_{h_1}} \frac{F_{h_1h_2}}{p_{h_1h_2}} \\ &\quad \cdots \frac{R_{h_{n-1}}}{\theta_{h_{n-1}}} \frac{F_{h_{n-1}s}}{p_{h_{n-1}s}} \frac{E_s}{p_i} \cdot R_i \frac{F_{ih_1}}{p_{ih_1}} \frac{R_{h_1}}{\theta_{h_1}} \frac{F_{h_1h_2}}{p_{h_1h_2}} \\ &\quad \cdots \frac{R_{h_{n-1}}}{\theta_{h_{n-1}}} \frac{F_{h_{n-1}s}}{p_{h_{n-1}s}} \frac{R_s}{\theta_s} \frac{F_{sh_{n+1}}}{p_{sh_{n+1}}} \frac{R_{h_{n+1}}}{\theta_{h_{n+1}}} \frac{F_{h_{n+1}h_{n+2}}}{p_{h_{n+1}h_{n+2}}} \\ &\quad \cdots \frac{R_{h_{m-1}}}{\theta_{h_{m-1}}} \frac{F_{h_{m-1}s'}}{p_{h_{m-1}s'}} \frac{E_{s'}}{p_i} \cdot \\ &\quad p_i p_{ih_1} \theta_{h_1} \cdots p_{h_{n-1}s} \theta_s p_{sh_{n+1}} \theta_{h_{n+1}} \cdots p_{h_{m-1}s'} \\ &= \theta_i \sum_s \sum_{h_1} \cdots \sum_{h_{n-1}} \frac{R_i^2}{\theta_i} \frac{F_{ih_1}^2}{p_{ih_1}} \frac{R_{h_1}^2}{\theta_{h_1}} \frac{F_{h_1h_2}^2}{p_{h_1h_2}} \\ &\quad \cdots \frac{R_{h_{n-1}}^2}{\theta_{h_{n-1}}} \frac{F_{h_{n-1}s}^2}{p_{h_{n-1}s}} \frac{E_s}{p_i} \cdot p_i p_{sh_1} \cdots p_{h_{n-1}s} \cdot \\ &\quad \sum_{s'} \sum_{h_{n+1}} \cdots \sum_{h_{m-1}} R_s F_{sh_{n+1}} R_{h_{n+1}} \\ &\quad \cdots R_{h_{m-1}} F_{h_{m-1}s'} \frac{E_{s'}}{p_i} \\ &= \frac{\theta_i}{p_i} \sum_s \kappa_{is}^{(n)} \sum_{s'} b_{ss'}^{(m-n)} = \frac{\theta_i}{p_i} \sum_s \kappa_{is}^{(n)} b_s^{(m-n)} \end{aligned}$$

where $\kappa_{is}^{(n)}$ is the i component of the nth term corresponding to the development in Neumann series of the linear system

$$\kappa_{is} = \sum_j \frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}} (\kappa_{js} + \delta_{js} E_s) \quad (2)$$

Then

$$\sum_{1\le n} \left(\sum_{n<m} E(\widehat{b}_i^{(n)}\widehat{b}_i^{(m)}) \right) = \frac{\theta_i}{p_i} \sum_s \sum_{1\le n} \kappa_{is}^{(n)} \sum_{n<m} b_s^{(m-n)}$$
$$= \frac{\theta_i}{p_i} \sum_s \kappa_{is} b_s$$

and also

$$\begin{aligned} E(\widehat{b}_i^{(1)2}) &= \sum_s \left(R_i \frac{F_{is}}{p_{is}} \frac{E_s}{p_i} \right)^2 \times p_i p_{is} \\ &= \sum_s \frac{\theta_i E_s}{p_i} \frac{R_i^2 F_{is}^2}{\theta_i p_{is}} E_s \\ &= \frac{\theta_i}{p_i} \sum_s E_s \kappa_{is}^{(1)} \end{aligned}$$

$$E(\widehat{b}_i^{(2)2}) = \sum_h \sum_s \left(R_i \frac{F_{ih}}{p_{ih}} \frac{R_h F_{hs}}{\theta_h p_{hs}} \frac{E_s}{p_i} \right)^2 \times p_i p_{ih} \theta_h p_{hs}$$
$$= \frac{\theta_i}{p_i} \sum_s E_s \kappa_{is}^{(2)}$$

and so on. Then we obtain

$$\begin{aligned} Var(\widehat{b}_i) &= \frac{\theta_i}{p_i} \sum_s E_s (\kappa_{is}^{(1)} + \kappa_{is}^{(2)} + \cdots) \\ &+ 2\frac{\theta_i}{p_i} \sum_s \kappa_{is} b_s - {b_i}^2 \\ &= \frac{\theta_i}{p_i} \sum_s (E_s + 2b_s)\kappa_{is} - {b_i}^2 \end{aligned}$$

where κ_{is} is the solution of the system (2).

For the radiosity our estimator is simply $\widehat{b}_i + E_i$, and as E_i is a constant we have

$$Var(\widehat{B}_i) = Var(\widehat{b}_i + E_i) = \frac{\theta_i}{p_i} \sum_s (E_s + 2b_s)\kappa_{is} - {b_i}^2 \quad (3)$$

If we consider each patch in turn, $p_i = 1$, and for N_i paths:

$$Var(\widehat{B}_i) = \frac{1}{N_i} \left(\theta_i \sum_s (E_s + 2b_s)\kappa_{is} - {b_i}^2 \right) \quad (4)$$

3.1 Existence of the variance

We have that the κ_i are the solutions of system (2). But a necessary and sufficient condition for the system to have an iterative solution [22] (such as the one given by the Neumann decomposition) is that

$$\rho(\frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}}) < 1 \quad (5)$$

where ρ is the spectral radius. It is obvious that if system (2) is solvable with the Neumann decomposition, the solutions must all be finite and positive, and thus the variances exist. On the other hand, it is very easy to show that if a solution of the system exists such that all values are positive, this solution can be obtained in an iterative way with the Neumann decomposition. Thus, a *necessary and sufficient condition for the existence of the variances is condition (5).*

An example where this condition is not satisfied is given when taking $\frac{R_i^2}{\theta_i} \ge 1$ and $p_{ij} = F_{ij}$ for all i, j. The spectral radius in this case is greater or equal than 1. Alternatively, it can be shown that

the infinite sums in formula (1) are not convergent. This is simply done by showing that $\lim_{n\to\infty} E(\widehat{b}_i^{(n)2}) \neq 0$, and thus the sum can not converge.

$$
\begin{aligned}
E(\widehat{b}_i^{(n)2}) &= \sum_{h_1} \cdots \sum_{h_{n-1}} \sum_{s} \left(R_i \frac{R_{h_1}}{\theta_{h_1}} \cdots \frac{R_{h_{n-1}}}{\theta_{h_{n-1}}} \frac{E_s}{p_i} \right)^2 \cdot \\
&\quad p_i F_{i_{h_1}} \theta_{h_1} \cdots F_{h_{n-2} h_{n-1}} \theta_{h_{n-1}} F_{h_{n-1} s} \\
&= \sum_{h_1} \cdots \sum_{h_{n-1}} \sum_{s} R_i^2 \frac{R_{h_1}^2}{\theta_{h_1}} \cdots \frac{R_{h_{n-1}}^2}{\theta_{h_{n-1}}} \frac{E_s^2}{p_i} \cdot \\
&\quad F_{i_{h_1}} \cdots F_{h_{n-1} s} \\
&\geq \frac{R_i^2}{p_i} \sum_{h_1} \cdots \sum_{h_{n-1}} \sum_{s} F_{i_{h_1}} \cdots F_{h_{n-1} s} E_s^2
\end{aligned}
\tag{6}
$$

because we have assumed $\frac{R_i^2}{\theta_i} \geq 1$ for all i. But we have

$$
\sum_{h_1} \cdots \sum_{h_{n-1}} F_{i_{h_1}} \cdots F_{h_{n-1} s} = (F^n)_{is}
$$

and in [15] it is proven that, whenever the Form Factor matrix is irreducible and aperiodic

$$
\lim_{n\to\infty} (F^n)_{is} = \frac{A_s}{A_T} \tag{7}
$$

where A_T is the total area in the scene. We do not consider it geometrically meaningful for a Form Factor matrix to be periodic. If there are closed rooms, it is reducible with one submatrix for each room. In this case, we consider in turn each room and we have irreducible submatrices.
But now from (6) and (7) we obtain:

$$
\begin{aligned}
\lim_{n\to\infty} E(\widehat{b}_i^{(n)2}) &\geq \sum_{s} \left(\lim_{n\to\infty} (F^n)_{is} \right) E_s^2 \\
&= \sum_{s} \frac{A_s}{A_T} E_s^2 > 0
\end{aligned}
\tag{8}
$$

3.2 Transition probabilities equal to the Form Factors

When $p_{ij} = F_{ij}$ we have the case studied in [19]. Formula (2) becomes:

$$
\kappa_{is} = \sum_{j} \frac{R_i^2 F_{ij}}{\theta_i} (\kappa_{js} + \delta_{js} E_s) \tag{9}
$$

The solution of this system was called ϵ_{is} in [19].
We have seen in the previous section that when for all i, $\theta_i \leq R_i^2$, then the variances do not exist. But as the condition $\theta_i > R_i^2$ for all i is too restrictive for the existence of the variance, the question arises whether a weaker condition can be given. We have run simulations and found that the spectral radius (5) is approximated by, and in most cases, is less than, the average value

$$
(\frac{R^2}{\theta})_{ave} = \frac{1}{A_T} \sum A_i \frac{R_i^2}{\theta_i} \tag{10}
$$

This finding is similar to the results in [5]. It is based on the following:

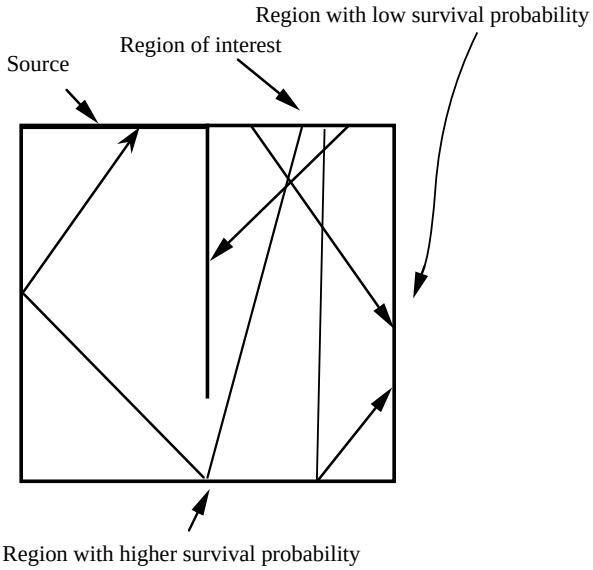


Figure 1: Paths traced from the region of interest have a higher survival probability where the reflected radiosity is higher.

If we consider the series $\rho(\frac{R_i^2}{\theta_i}(F^n)_{ij})$, its limit is $\rho(\frac{R_i^2}{\theta_i}\frac{A_i}{A_T})$, but it is very easy to check that

$$
\rho(\frac{R_i^2}{\theta_i}\frac{A_i}{A_T}) = \frac{1}{A_T} \sum_i A_i \frac{R_i^2}{\theta_i} = (\frac{R^2}{\theta})_{ave}
$$

Thus what we do is to approximate the first term in the series, $\rho(\frac{R_i^2}{\theta_i} F_{ij})$, by the limit of the series.
Thus, a heuristic such as keeping this average value less than 0.8 or maybe 0.9 should be safe.
An example of the use of such an estimator is the following: Suppose we have obtained a coarse solution for the radiosities. This solution could be used to drive the random walk taking $\theta_i \propto B_i - E_i$, assuring that enough paths will survive in patches with high received radiosity. This is useful in a scene like the one in Figure 1. This case can be considered the dual of the one given in [10] (see Figure 2).

Suppose now that $\theta_i = 1$. This is the infinite path length estimator, $\frac{R_i^2}{\theta_i} = R_i^2$ and κ_{is} in (9) has now a physical meaning: the reflected radiosity of patch i due to source s in a dual scene where we simply have substituted each reflectivity by its square. This quantity was called β_{is} in [16], and we again obtain the formula in table 1. On the other hand, when $\theta_i = R_i$ we have the E_s estimator, $\frac{R_i^2}{\theta_i} = R_i$ and κ_{is} becomes b_{is} (which can be seen by direct substitution in formula (9):

$$
b_{is} = \sum_{j} R_i F_{ij} (b_{js} + \delta_{js} E_s) \tag{11}
$$

and again obtaining the formula for the E_s estimator given in table 1.

3.3 Transition probabilities as biased Form Factors

Now let us consider the biased Form Factors as transition probabilities:

$$p_{ij} = \frac{R_i F_{ij} B_j}{B_i - E_i}$$

It is easy to check that they are probabilities, that is, they are all positive and sum to 1. We will consider two different survival probabilities. The first will be $\theta_i = \frac{B_i - E_i}{B_i}$, and the second $\theta_i = 1$, that is, the infinite estimator. In the first case equations (2) convert into:

$$\kappa_{is} = \sum_j \frac{R_i F_{ij} B_i}{B_j} (\kappa_{js} + \delta_{js} E_s) \tag{12}$$

It is not difficult to obtain the solution of this system. We have

$$\kappa_{is} = \frac{B_i b_{is}}{B_s} \tag{13}$$

The variance is obtained substituting the θ_i values and the found κ_{is} values in formula (3):

$$Var(\widehat{B}_i) = \frac{B_i - E_i}{p_i} \sum_s (E_s + 2b_s) \frac{b_{is}}{B_s} - {b_i}^2 \tag{14}$$

As $E_s + 2b_s = B_s + b_s$, and $b_s < B_s$, this variance can be bounded:

$$Var(\widehat{B}_i) < \frac{B_i - E_i}{p_i} 2 \sum_s b_{is} - {b_i}^2 = \frac{2b_i^2}{p_i} - b_i^2 \tag{15}$$

And for $p_i = 1$

$$Var(\widehat{B}_i) < b_i^2 \tag{16}$$

In the second case, that is, $\theta_i = 1$, equations (2) convert into

$$\kappa_{is} = \sum_j \frac{R_i F_{ij} (B_i - E_i)}{B_j} (\kappa_{js} + \delta_{js} E_s) \tag{17}$$

Multiplying the left and right term in the system by $E_s + 2b_s$ and summing over s:

$$\begin{aligned} \sum_s (E_s + 2b_s)\kappa_{is} &= \sum_j \frac{R_i F_{ij} (B_i - E_i)}{B_j} \cdot \\ &\quad (\sum_s (E_s + 2b_s)\kappa_{js} \\ &+ \quad \sum_s (E_s + 2b_s)\delta_{js} E_s) \\ &= \sum_j \frac{R_i F_{ij} (B_i - E_i)}{B_j} \cdot \\ &\quad (\sum_s (E_s + 2b_s)\kappa_{js} + (E_j + 2b_j)E_j) \end{aligned} \tag{18}$$

This new system can be seen by direct substitution to have the solution

$$\sum_s (E_s + 2b_s)\kappa_{is} = b_i^2 \tag{19}$$

And substituting (19) and $\theta_i = 1$ in (3):

$$Var(\widehat{B}_i) = \frac{B_i - E_i}{p_i} {b_i} - {b_i}^2 = \frac{b_i^2}{p_i} - b_i^2 \tag{20}$$

That means for $p_i = 1$, that is we consider each patch in turn, a null variance estimator. Thus, the probabilities considered are optimal in the sense that they lead to null variance estimators.

3.4 A system of equations for the variances

Let us suppose that for each path from i, we do the survival test *before* it starts. This means that on average only a θ_i fraction of paths will actually start off, and a $1 - \theta_i$ fraction will never start, that is, will have zero length. It is easy to show that in this case the variance is:

$$Var(\widehat{B}_i) = Var(\widehat{b_i + E_i}) = \frac{1}{p_i} \sum_s (E_s + 2b_s)\kappa_{is} - {b_i}^2 \tag{21}$$

Multiplying the left and right terms of system (2) by $E_s + 2b_s$ and summing over s

$$\begin{aligned} \sum_s (E_s + 2b_s)\kappa_{is} &= \sum_j \frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}} \cdot \\ &\quad (\sum_s (E_s + 2b_s)\kappa_{js} + \\ &\quad \sum_s (E_s + 2b_s)\delta_{js} E_s) \\ &= \sum_j \frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}} \cdot \\ &\quad (\sum_s (E_s + 2b_s)\kappa_{js} + (E_j + 2b_j)E_j) \end{aligned} \tag{22}$$

Taking $p_i = 1$ in (3)

$$\sum_s (E_s + 2b_s)\kappa_{is} = Var(\widehat{B}_i) + b_i^2$$

the system (22) can be written as:

$$\begin{aligned} Var(\widehat{B}_i) &= \sum_j \frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}} \cdot \\ &\quad (Var(\widehat{B}_j) + b_j^2 + (E_j + 2b_j)E_j) + b_i^2 \end{aligned} \tag{23}$$

and simplifying

$$Var(\widehat{B}_i) = \sum_j \frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}} (Var(\widehat{B}_j) + B_j^2) + b_i^2 \tag{24}$$

This system was obtained from conditional probability considerations in [6]. However, Halton considered only symmetric matrices (the ones corresponding to our $R_i F_{ij}$), and considered also that system (24) always had a solution. His work is referenced in [3], but Ermakow adds as condition for the existence (and finitness)

of a solution that the $Norm\left(\frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}}\right) < 1$ (translated to the radiosity setting).
Equating to zero the independent term in (24) we will obtain a homogeneous system with null solution for the variances. Thus:

$$\sum_j \frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}} B_j^2 = b_i^2 \tag{25}$$

By direct substitution it can be checked that $p_{ij} = F_{ij}$ and $\theta_i = 1$ are the solutions of (25), and we again obtain the null variance estimator of the previous section.

3.5 Complexity

The same results on complexity as in [14] can be obtained here. We just have to find a bound for the variances which is independent of the number of patches. Let us see it:

$$\begin{aligned} Var(\widehat{B}_i) &= \frac{1}{N_i}\left(\theta_i \sum_s (E_s + 2b_s)\kappa_{is} - {b_i}^2\right) \\ &\leq \frac{1}{N_i}\left((E_{max} + 2b_{max})\sum_s \kappa_{is}\right) \end{aligned} \tag{26}$$

because $\theta_i \leq 1$.
But $\kappa_i = \sum_s \kappa_{is}$ can be bounded summing over s in the system (2):

$$\begin{aligned} \kappa_i &= \sum_j \frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}}(\kappa_j + E_j) \\ &\leq \frac{R_i^2}{\theta_i} K \sum_j F_{ij}(\kappa_{max} + E_{max}) \\ &= \frac{R_i^2}{\theta_i} K(\kappa_{max} + E_{max}) \end{aligned} \tag{27}$$

where $K = max(\frac{F_{ij}}{p_{ij}})$. But then

$$\kappa_{max} \leq max(\frac{R_i^2}{\theta_i}) K(\kappa_{max} + E_{max}) \tag{28}$$

and thus

$$\kappa_{max} \leq \frac{max(\frac{R_i^2}{\theta_i}) K E_{max}}{1 - max(\frac{R_i^2}{\theta_i}) K} \tag{29}$$

This means

$$Var(\widehat{B}_i) \leq \frac{1}{N_i}\left((E_{max} + 2b_{max})\frac{max(\frac{R_i^2}{\theta_i}) K E_{max}}{1 - max(\frac{R_i^2}{\theta_i}) K}\right) \tag{30}$$

and this bound stays valid as long as we keep bounded $K = max(\frac{F_{ij}}{p_{ij}})$, $max(\frac{R_i^2}{\theta_i})$ and E_{max}, as from [14] we know that $b_{max} \leq R_{max} max(\frac{E_i}{1-R_i})$.

3.6 Other unbiased estimators

Just as we have generalized the E_s estimator in section3, the same can be done with the $\frac{E_s}{1-R_s}$ and $\frac{E_s}{R_s}$ estimators. The generalized estimator $\frac{E_s}{1-R_s}$ can be defined in a similar way as the E_s one, just it will only score on a source where the path dies, and the value of $\widehat{b}_i^{(l)}$ is, if $i, h_1, h_2, \ldots, h_{l-1}, s$ is the trajectory of the path:

$$\widehat{b}_i^{(l)} = R_i \frac{F_{ih_1}}{p_{ih_1}} \frac{R_{h_1}}{\theta_{h_1}} \frac{F_{h_1h_2}}{p_{h_1h_2}} \cdots \frac{R_{h_{l-1}}}{\theta_{h_{l-1}}} \frac{F_{h_{l-1}s}}{p_{h_{l-1}s}} \frac{E_s}{p_i(1-\theta_s)}$$

With a similar approach as in section 3 we can find the variance:

$$Var(\widehat{B}_i) = \frac{\theta_i}{p_i} \sum_s \frac{E_s}{1-\theta_s} \kappa_{is} - {b_i}^2 \tag{31}$$

This estimator is interesting because it leads to a (finite path length) null variance estimator when

$$p_{ij} = \frac{R_i F_{ij} B_j}{B_i - E_i}$$

and $\theta_i = \frac{B_i - E_i}{B_i}$. Using these values, the value in (13) and setting $p_i = 1$, (31) converts into:

$$\begin{aligned} Var(\widehat{B}_i) &= \frac{B_i - E_i}{B_i} \sum_s \frac{E_s B_s}{E_s} \frac{B_i b_{is}}{B_s} - {b_i}^2 \\ &= (B_i - E_i) \sum_s b_{is} - {b_i}^2 = 0 \end{aligned} \tag{32}$$

The generalized estimator $\frac{E_s}{R_s}$ is also defined in a similar way as the E_s one, just it will score on each source it hits, except when it dies, and the value of $\widehat{b}_i^{(l)}$ is, if $i, h_1, h_2, \ldots, h_{l-1}, s$ is the trajectory of the path:

$$\widehat{b}_i^{(l)} = R_i \frac{F_{ih_1}}{p_{ih_1}} \frac{R_{h_1}}{\theta_{h_1}} \frac{F_{h_1h_2}}{p_{h_1h_2}} \cdots \frac{R_{h_{l-1}}}{\theta_{h_{l-1}}} \frac{F_{h_{l-1}s}}{p_{h_{l-1}s}} \frac{E_s}{p_i \theta_s}$$

And the variance can be found to be:

$$Var(\widehat{B}_i) = \frac{\theta_i}{p_i} \sum_s \frac{E_s + 2b_s}{\theta_s} \kappa_{is} - {b_i}^2 \tag{33}$$

3.7 The continuous case

So far we have considered patch-to-patch transition probabilities. This means that, in the simulation, the path, after having hit a patch on a given point, exits from a new random point to continue its trajectory. This is different from the point-to-point transition probabilities or pure Particle Tracing simulation as explained in [11]. Thus it leads to a different solution than the one given by the rendering equation (although both solutions coincide in the limiting case, when the size of patches decreases to zero). This is because when using the patch-to-patch transition probabilities what we are really obtaining are solutions of a system of equations, which is an approximation to the rendering equation. In the continuous case, or pure Particle Tracing, what is obtained is the average of the *exact* solution of the rendering equation over a patch. However, using the same argumentation as in [17] it can be seen that the *same* formulae obtained here are valid for the pure Particle Tracing case, just the quantities appearing are obviously not the same. That is, for instance, the b_i quantity will mean now the average on the patch of the *exact* solution of the rendering equation, when for the previous

case it meant the exact solution of the radiosity system of equations. And of course in the continuous case the system (2) becomes an integral equation:

$$\kappa_s(x) = \int_{\mathcal{S}} \frac{R^2(x)F^2(x,y)}{\theta(x)p(x,y)}(\kappa_s(y) + E_s(y))dy \qquad (34)$$

where $\mathcal{S}$ is the set of surfaces of the scene, and $E_s(y)$ is equal to E_s if y happens to be in source s and zero otherwise. The κ_{is} value is then the average over the patch i of $\kappa_s(x)$.

4 The shooting estimator

Now let us consider the shooting estimator with generalized transition and absorption probabilities. Consider the following simulation. A path γ starts from source s with probability p_s, and from here on it evolves according to the transition probabilities p_{ij}. On each hit patch i, a survival-absorption test is done according to the probabilities $\{\theta_i, 1-\theta_i\}$. If the path γ happens to arrive at patch i with length l, then the radiosity of this patch is updated with the quantity $\frac{R_i}{A_i}\frac{F_{sh_1}}{p_{sh_1}}\frac{R_{h_1}}{\theta_{h_1}}\frac{F_{h_1h_2}}{p_{h_1h_2}}\frac{R_{h_2}}{\theta_{h_2}}\cdots\frac{R_{h_{l-1}}}{\theta_{h_{l-1}}}\frac{F_{h_{l-1}i}}{p_{h_{l-1}i}}\frac{\Phi_s}{p_s}$.
Now, the variance can be found either from duality considerations as in [17] or using the same approach as in section 3:

$$Var(\widehat{B}_i) = \theta_i \sum_s \Phi_s \frac{(1+2R_i\xi_i)\kappa^\star_{is}}{A_i^2 p_s} - b_i^2 \qquad (35)$$

where $\kappa^\star_{is}$ is the solution of the system:

$$\kappa^\star_{is} = \sum_j \frac{R_i^2 F_{ji}^2}{\theta_i p_{ji}}(\kappa^\star_{js} + \delta_{js}\Phi_s) \qquad (36)$$

This system can be considered in a certain way the dual of (2), as far as the power system can be considered dual to the radiosity system.

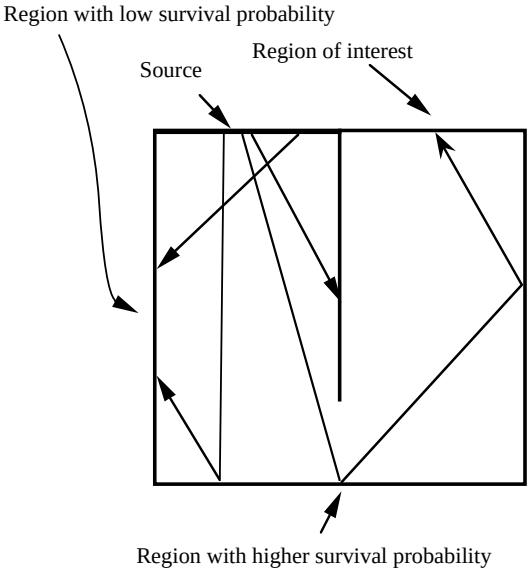


Figure 2: Paths traced from the source have a higher survival probability where the received importance towards the region of interest is higher.

Here the same argumentation as in previous section about the existence of the variance is also valid, and it is easy to check that

$$\rho(\frac{R_i^2}{\theta_i}\frac{F_{ij}^2}{p_{ij}}) = \rho(\frac{R_i^2}{\theta_i}\frac{F_{ji}^2}{p_{ji}})$$

Formula (35), when taking source selection probabilities proportional to the power of the sources $p_s = \frac{\Phi_s}{\Phi_T}$, where Φ_T is the total power, converts into

$$Var(\widehat{B}_i) = \theta_i\Phi_T\frac{(1+2R_i\xi_i)\kappa^\star_i}{A_i^2} - b_i^2 \qquad (37)$$

where $\kappa^\star_i = \sum_s \kappa^\star_{is}$.
Also, taking each source s in turn, with $N_s = Np_s$ paths from it, we obtain

$$Var(\widehat{B}_i) = \frac{1}{N_s}\left(\theta_i\Phi_s\frac{(1+2R_i\xi_i)\kappa^\star_{is}}{A_i^2} - b_{is}^2\right) \qquad (38)$$

And as we have independent simulations:

$$\begin{aligned} Var(\widehat{B}_i) &= Var(\widehat{b}_i) = Var(\sum_s \widehat{b}_{is}) = \sum_s Var(\widehat{b}_{is}) \\ &= \sum_s \frac{1}{N_s}\left(\theta_i\Phi_s\frac{(1+2R_i\xi_i)\kappa^\star_{is}}{A_i^2} - b_{is}^2\right) \end{aligned} \qquad (39)$$

This is the variance we obtain in real simulations, as we usually precompute the number of rays to cast from each source according to its probability, rather than doing it on the fly.

4.1 Transition probabilities equal to the Form Factors

In [19] the case $p_{ij} = F_{ij}$ was studied, but the absorption probabilities were general ones. Formula (36) becomes:

$$\kappa^\star_{is} = \sum_j \frac{R_i^2 F_{ji}}{\theta_i}(\kappa^\star_{js} + \delta_{js}\Phi_s) \qquad (40)$$

which can be shown to have the solution $A_i\epsilon_{is}$, where ϵ_{is} was defined in [19].

An example of the usefulness of such an estimator is the following:
If we make the survival probability proportional to received importance, that is, $\theta_i \propto I_i - V_i$, where the V_i is the initial and I_i the total importance [10] (we use here the notation by Neumann et al.), we assure that the paths will often survive in patches which are important to the selected ones. This could be used to drive a random walk in a scene like the one in Figure 2, the dual of Figure 1. This case was the one considered in [10], although a breadth-first strategy was used there, instead of the considered depth-first strategy here.
Now suppose that $\theta_i = 1$. This is the infinite path length estimator, $\frac{R_i^2}{\theta_i} = R_i^2$ and $\kappa^\star_{is}$ in (40) now has the solution $A_i\beta_i$: the reflected power of patch i due to source s in a dual scene where we simply have substituted each reflectivity by its square. Substituting in (35), we obtain again the formula for the infinite estimator in table 1. On the other hand, when $\theta_i = R_i$ we have the Φ_T estimator, $\frac{R_i^2}{\theta_i} = R_i$ and $\kappa^\star_{is}$ becomes $A_i b_{is}$, which can be seen by direct substitution in formula (40):

$$A_i b_{is} = \sum_j R_i F_{ji}(A_j b_{js} + \delta_{js}\Phi_s) \qquad (41)$$

and again obtain the formula for the Φ_T estimator given in table 1.

Another interesting case is the estimator presented in [8], although Keller introduces it in the context of Quasi-Monte Carlo estimators. Keller takes the average reflectivity of the scene as survival probability on all patches.

4.1.1 Optimal survival probabilities

Suppose we are interested in all patches, not just in a single region. We want to find out the optimal survival probabilities θ_i in the sense to maximize the efficiency. This can be defined as the inverse of the product of the variance times the cost [13], for a single patch, or the average weighted variances (expected value of the Mean Square Error) times the cost, for the whole scene. This means, taking as average cost $\frac{1}{1-\theta_{ave}}$, to minimize the quantity

$$E(MSE) \cdot \frac{1}{1-\theta_{ave}} \tag{42}$$

Now, using the definition of the Mean Square, formula (37) for the variance (we consider a reasonable hypothesis $p_s = \frac{\Phi_s}{\Phi_T}$ when interested on the whole scene) and the approximation

$$\epsilon_i \approx \frac{\frac{R_i^2}{\theta_i}\Phi_T}{A_T(1-(\frac{R^2}{\theta})_{ave})}$$

we obtain, following the same approach as in [18]:

$$E(MSE) \approx \frac{\Phi_T^2 R_{ave}^2}{A_T A_{ave}(1-(\frac{R^2}{\theta})_{ave})}$$

which substituted in (42), after approximating $(\frac{R^2}{\theta})_{ave}$ by $\frac{R_{ave}^2}{\theta_{ave}}$, gives as quantity to minimize:

$$\frac{\Phi_T^2 R_{ave}^2}{A_T A_{ave}(1-(\frac{R_{ave}^2}{\theta_{ave}}))}\frac{1}{1-\theta_{ave}} \tag{43}$$

The behaviour of this quantity, taking $\frac{\Phi_T^2 R_{ave}^2}{A_T A_{ave}} = 1$, is shown in Figure 3, for values of R_{ave} 0.3, 0.5 and 0.8, respectively.

The analytical solution is $\theta_{ave} = R_{ave}$. This will obviously happen when for all i $\theta_i = R_i$. Thus we can state the result:
Of all unbiased shooting random walk estimators with generalised absorption probabilities and transition probabilities equal to the Form Factors, the most efficient for calculating all radiosities is the one with survival probability equal to the reflectivity.
Remember that the infinite path length estimator is not considered here (the cost would be infinite), but from [18] we know that biasing it we obtain a much worse estimator than Φ_T. That means that Φ_T is the best of all shooting random walk estimators with transition probabilities equal to the Form Factors, biased or not. And from [14] we know that shooting estimators are much better when dealing with the whole scene than gathering estimators, so we can extend this result to all random walk estimators studied till now. It must be remembered here that the Φ_T estimator in its breadth-first approach was the one used by Shirley [20] and Feda and Purgathofer [4].

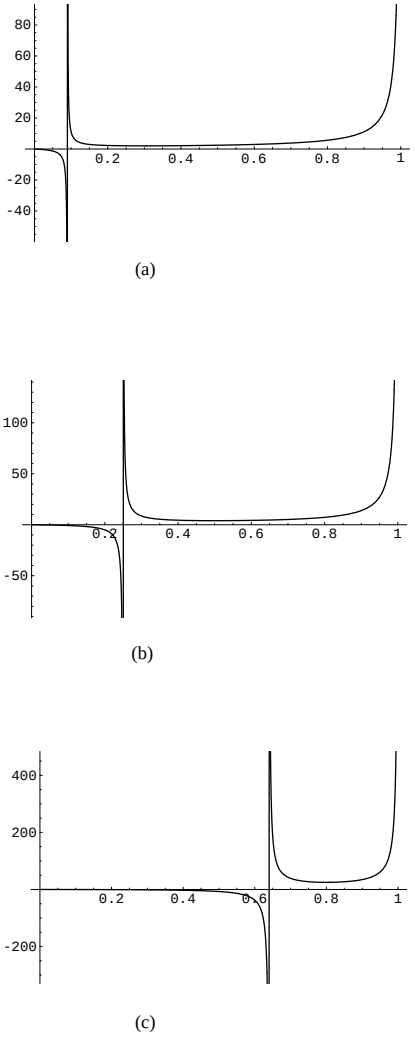


Figure 3: Behaviour of the inverse of efficiency against θ_{ave} for $R_{ave} = 0.3, 0.5$ and 0.8, respectively. The vertical asymptote corresponds to R_{ave}^2

4.2 Transition probabilities as biased Form Factors

Now let us consider as transition probabilities the biased Form Factors:

$$p_{ij} = \frac{R_j F_{ij} I_j^q}{I_i^q - q_i}$$

where the I_i^q values are the solutions of the system (dual of the power one):

$$I_i^q = \sum_j R_j F_{ij} I_j^q + q_i \tag{44}$$

and q_i is the vector of initial importance [17]. It is easy to check that the p_{ij} quantities are probabilities, that is, they are all positive and sum to 1 for a fixed i. We will consider two different survival probabilities. The first will be $\theta_i = \frac{I_i^q - q_i}{I_i^q}$, and the second $\theta_i = 1$, that is, the infinite estimator. In the first case equations (36) convert into:

$$\kappa_{is}^\star = \sum_j \frac{R_i F_{ji}(I_j^q - q_j)}{I_i^q - q_i}(\kappa_{js}^\star + \delta_{js}\Phi_s) \tag{45}$$

It is not difficult to obtain the solution of this system. It is:

$$\kappa_{is}^\star = \frac{A_i(I_s^q - q_s)b_{is}}{I_i^q - q_i} \tag{46}$$

The variance is then obtained substituting the θ_i values and the found $\kappa_{is}^\star$ values in (35):

$$Var(\widehat{B}_i) = \frac{1}{I_i^q}\sum_s \frac{\Phi_s(1+2R_i\xi_i)(I_s^q - q_s)b_{is}}{A_i p_s} - {b_i}^2 \tag{47}$$

Suppose now we are interested only in a single patch k (thus $q_i = \delta_{ki}$), and that we take $p_s = \frac{\Phi_s(I_s^q - q_s)}{\sum_s \Phi_s(I_s^q - q_s)}$. As now

$$I_s^q - q_s = I_{ks} - \delta_{ks}$$

where I_{ks} is the importance of source s to illuminate patch k (the relation $\sum_s I_{ks}\Phi_s = A_k B_k$ holds), we obtain for the variance of the patch k, taking into account that $\sum_s \Phi_s(I_{ks} - \delta_{ks}) = A_k(B_k - E_k) = A_k b_k$ and $1 + 2R_k\xi_k = 2I_{kk} - 1$ [17]:

$$\begin{aligned} Var(\widehat{B}_k) &= \frac{1+2R_k\xi_k}{A_k I_{kk}}\sum_s \Phi_s(I_{ks} - \delta_{ks})(\sum_s b_{ks}) - {b_k}^2 \\ &= \frac{2I_{kk}-1}{I_{kk}} b_k(\sum_s b_{ks}) - {b_k}^2 \\ &= {b_k}^2(1 - \frac{1}{I_{kk}}) \end{aligned} \tag{48}$$

This quantity is always strictly positive, because $I_{kk} > 1$.
Another interesting case happens when considering $q_i = \frac{1}{R_i} - 1$ (we do not use here a normalized q vector). We can check by direct substitution in the system (44) that $I_i^q = \frac{1}{R_i}$. With those values plugged in equation (47) we obtain:

$$Var(\widehat{B}_i) = R_i \sum_s \frac{\Phi_s(1+2R_i\xi_i)b_{is}}{A_i p_s} - {b_i}^2 \tag{49}$$

which is exactly the same variance as the corresponding to the Φ_T estimator. In fact, both estimators are the same, because plugging the values for I_i^q and q_i in the survival and transition probabilities we obtain $\theta_i = \frac{I_i^q - q_i}{I_i^q} = R_i$, and $p_{ij} = \frac{R_j F_{ij} I_j^q}{I_i^q - q_i} = F_{ij}$. This tells us that the Φ_T estimator can be considered a particular case of importance biasing in which the initial importance of each patch is the inverse of its reflectivity minus one.

In the second case, that is, $\theta_i = 1$, equations (36) convert into

$$\kappa_{is}^\star = \sum_j \frac{R_i F_{ji}(I_j^q - q_j)}{I_i^q}(\kappa_{js}^\star + \delta_{js}\Phi_s) \tag{50}$$

Instead of solving this system in general, we will only obtain the solution for the particular case $q_i = \delta_{ik}$. We will do it from the duality between this case and the one given by the equation (20). We obtain, following the same approach as in [17]:

$$Var(\widehat{I}_{ks}) = \frac{(I_{ks} - \delta_{ks})^2}{p_s} - (I_{ks} - \delta_{ks})^2 \tag{51}$$

Considering each source in turn ($p_s = 1$), and taking into account that $b_{ks} = \frac{1}{A_k}(I_{ks} - \delta_{ks})\Phi_s$,

$$\begin{aligned} Var(\widehat{b}_{ks}) &= Var(\frac{\Phi_s}{A_s}\widehat{I}_{ks}) \\ &= \frac{\Phi_s^2}{A_s^2}\left((I_{ks} - \delta_{ks})^2 - (I_{ks} - \delta_{ks})^2\right) = 0 \end{aligned} \tag{52}$$

And as the b_{is} are independent estimators for all s, $Var(\widehat{B}_i) = Var(\widehat{b}_i) = \sum_s Var(\widehat{b}_{is}) = 0$.
Thus the transition probabilities considered are optimal in the sense that they lead to a null variance estimator. Compare them with the ones used in [12]:

$$p_{ij} = \frac{F_{ij} I_j^q}{\sum_h F_{ih} I_h^q}$$

Our I_j^q quantities are Pattanaik's hemispherical potential.

4.2.1 Biasing with importances as dual of radiosities

Consider the dual of the radiosity system:

$$J_i^q = \sum_j R_j F_{ji} J_j^q + q_i \tag{53}$$

and suppose we use the probabilities:

$$p_{ij} = \frac{R_j F_{ji} J_j^q}{J_i^q - q_i}$$

and

$$\theta_i = \frac{J_i^q - q_i}{J_i^q}$$

It can be easily proven that the same correspondence between systems (53) and (44) exists as between the radiosity and power systems. Thus, for $q_i' = A_i q_i$, then $I_i^{q'} = A_i J_i^q$. Thus, the transition and survival probabilities can be expressed simply as

$$p_{ij} = \frac{R_j F_{ij} I_j^{q'}}{I_i^{q'} - q_i'}$$

and

$$\theta_i = \frac{I_i^{q'} - q_i'}{I_i^{q'}}$$

where $q_i' = A_i q_i$. In this way we obtain the same results as in the previous section.

4.3 Complexity

We can use the same argumentation as in section 3.5, and also the observation that $K = max(\frac{F_{ij}}{p_{ij}}) = max(\frac{F_{ji}}{p_{ji}})$ to bound $\sum_s \kappa^\star_{is}$ using K. From this bounding we can obtain the same results as in [14].

4.4 Other unbiased estimators

Just as we have generalized the Φ_T estimator in section 4, the same can be done with the $\frac{\Phi_T}{1-R_i}$ and $\frac{\Phi_T}{R_i}$ estimators. The generalized estimator $\frac{\Phi_T}{1-R_i}$ can be defined in a similar way as the Φ_T one, just it will only score on the patch where the path dies, and the value of $\widehat{b}_i^{(l)}$ is, if $s, h_1, h_2, \ldots, h_{l-1}, i$ is the trajectory of the path:

$$\widehat{b}_i^{(l)} = \frac{R_i}{A_i}\frac{F_{sh_1}}{p_{sh_1}}\frac{R_{h_1}}{\theta_{h_1}}\frac{F_{h_1h_2}}{p_{h_1h_2}}\frac{R_{h_2}}{\theta_{h_2}}\cdots\frac{R_{h_{l-1}}}{\theta_{h_{l-1}}}\frac{F_{h_{l-1}i}}{p_{h_{l-1}i}}\frac{\Phi_s}{p_s(1-\theta_i)}$$

With the same approach as in section 3 we can find the variance:

$$Var(\widehat{B}_i) = \frac{\theta_i}{1-\theta_i}\sum_s \Phi_s \frac{\kappa^\star_{is}}{A_i^2 p_s} - b_i^2 \qquad (54)$$

This estimator is interesting because it leads to a (finite path length) null variance estimator when

$$p_{ij} = \frac{R_j F_{ij} I_j^q}{I_i^q - q_i}$$

$q_i = \delta_{ki}$, $\theta_i = \frac{I_i^q - q_i}{I_i^q}$ and $p_s = \frac{\Phi_s(I_s^q - q_s)}{\sum_s \Phi_s(I_s^q - q_s)}$. Using these values and the value in (46), (54) converts (for the patch i such that $q_i = 1$) into:

$$\begin{aligned} Var(\widehat{B}_i) &= \left(\frac{1}{I_i^q}\sum_s \frac{b_{is}}{A_i}\sum_s \Phi_s(I_s^q - q_s)\right) - b_i^2 \\ &= \left(\frac{b_i}{A_i} b_i A_i\right) - b_i^2 = 0 \end{aligned} \qquad (55)$$

because $\sum_s \Phi_s(I_s^q - q_s) = b_i A_i$. This null variance case is obviously optimal and is in apparent contradiction with the result in [17]. There the optimal probability for source selection was given as $p_s \propto \Phi_s\sqrt{I_s^q - q_s}$. But note that in [17] the transition probabilities considered were pure, not biased, Form Factors.

The generalized estimator $\frac{\Phi_T}{R_i}$ is also defined in a similar way as the Φ_T one, just it will score on each patch it hits, except when it dies, and the value of $\widehat{b}_i^{(l)}$ is updated with, if $i, h_1, h_2, \ldots, h_{l-1}, s$ is the trajectory of the path:

$$\widehat{b}_i^{(l)} = \frac{R_i}{A_i}\frac{F_{sh_1}}{p_{sh_1}}\frac{R_{h_1}}{\theta_{h_1}}\frac{F_{h_1h_2}}{p_{h_1h_2}}\frac{R_{h_2}}{\theta_{h_2}}\cdots\frac{R_{h_{l-1}}}{\theta_{h_{l-1}}}\frac{F_{h_{l-1}i}}{p_{h_{l-1}i}}\frac{\Phi_s}{p_s\theta_i}$$

And the variance can be found to be:

$$Var(\widehat{B}_i) = \sum_s \Phi_s \frac{(1+2R_i\xi_i)\kappa^\star_{is}}{A_i^2 p_s} - b_i^2 \qquad (56)$$

4.5 The continuous case

The same remarks as in section 3.7 are valid here. That is, all the results obtained are also valid for the point-to-point Form Factors, or pure Particle Tracing. Again, system (36) becomes an integral equation.

5 Results

Here we present in figure 5 some experiments performed on a very simple scene, a cubical enclosure with each face divided into nine equal size patches (Fig.4), the reflectivities of the faces being 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 respectively, and a source with emissivity 1 in the middle of the first face, in patch 5. Thus patches 1 to 9 receive no direct lighting and have reflectivity 0.3, patches 10 to 18 reflectivity 0.4, and so on. For this scene we computed a reference solution with the Φ_T estimator (as defined in [14]) and 10^7 paths. This provided us with the b_i values. The κ_{is} values were obtained with a simulation with the *infinite* shooting estimator (as defined in [14], with threshold 0.001) for the same scene with the reflectivities substituted by its square divided by the average reflectivity, that is, we obtained a solution of equation 9. We could not use the Φ_T estimator because the reflectivity for the sixth face is greater than one, and thus it makes no sense to use it as survival probability.

Then 100 runs of 10^4 paths each for both shooting and gathering estimators taking as survival probability for each patch the average reflectivity (which is 0.55) and for gathering $p_i = \frac{A_i}{A_T}$ (the fraction of total area). The transition probabilities were the Form Factors. We used the 100 sets of results to obtain the square errors, and thus an estimated value of the variances for a single path. The formulae for the expected variances are the formulae 3 and 35, with the approximation $\xi_i = 0$ (and for each patch $\theta_i = 0.55$, the average reflectivity). Figure 5 shows that the obtained results are in concordance with the theoretically expected ones. Although the scene used in the test has no occlusions, it should be noted that the variance of a patch radiosity does not depend on whether it is due to direct or indirect illumination.

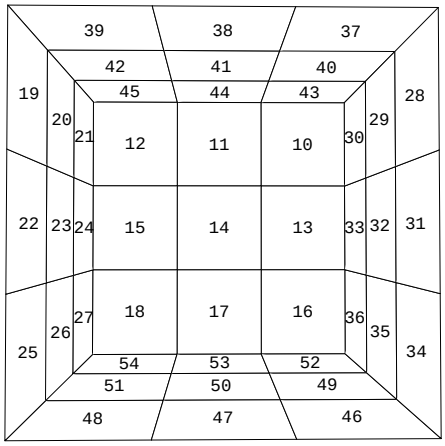
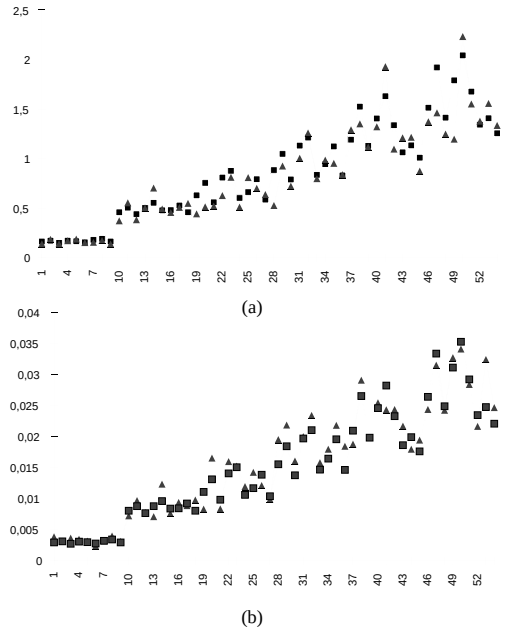


Figure 4: Numbering the patches in the test scene. Patches 1 to 9, with reflectance 0.3, are not shown. Patch 5 is the emitter. Patches 10 to 18 have reflectance 0.4, 19 to 27 0.5 and so on.

6 Conclusions and future research

We have generalized the results of [14], [16] and [19] to the case of generalized transition probabilities, obtaining closed formulae for the variances of the estimators studied. Those are presented in table 4. Some particular cases are studied, including different null variance estimators. A necessary and sufficient condition for the existence of the variance is also given. A heuristic for the existence of the variance in the general case will also be searched for, similar to the one given for the generalized absorption case. The study



(a) (b)

Figure 5: Comparison of the expected variances (plotted as square dots) and the experimentally obtained square errors for the gathering (a) and shooting (b) estimator with survival probability equal to the average reflectivity, for the 54 equal area patches of a cube (on x axis), with face reflectivities 0.3, 0.4, 0.5, 0.6, 0.7, 0.8. A source with emittance 1 is in the middle of the first face.

of the general (non-diffuse) Rendering equation will be undertaken in two ways. First, the results obtained here will be extended to the continuous, non-diffuse case in a similar way as in [17]. Second, the results in [3] on different estimators for the second kind Fredholm integral equation will be analysed and compared to the previously obtained extensions.

Table 4: Variances for the Random Walk estimators with generalized transition, p_{ij}, and survival probabilities, θ_i. κ_{is} is the solution of the system $\kappa_{is} = \sum_j \frac{R_i^2 F_{ij}^2}{\theta_i p_{ij}}(\kappa_{js} + \delta_{js} E_s)$ and $\kappa_{is}^\star$ is the solution of the system $\kappa_{is}^\star = \sum_j \frac{R_i^2 F_{ji}^2}{\theta_i p_{ji}}(\kappa_{js}^\star + \delta_{js}\Phi_s)$

Shooting	**Patch scored**	**Variance**
$\frac{\Phi_T}{1-R_i}$	last	$\frac{\theta_i}{1-\theta_i}\sum_s \Phi_s \frac{\kappa_{is}^\star}{A_i^2 p_s} - b_i^2$
$\frac{\Phi_T}{R_i}$	all but last	$\sum_s \Phi_s \frac{(1+2R_i\xi_i)\kappa_{is}^\star}{A_i^2 p_s} - b_i^2$
Φ_T	all	$\theta_i \sum_s \Phi_s \frac{(1+2R_i\xi_i)\kappa_{is}^\star}{A_i^2 p_s} - b_i^2$
Gathering	**Patch scored**	**Variance**
$\frac{E_s}{1-R_s}$	last	$\frac{\theta_i}{p_i}\sum_s \frac{E_s}{1-\theta_s}\kappa_{is} - {b_i}^2$
$\frac{E_s}{R_s}$	all but last	$\frac{\theta_i}{p_i}\sum_s \frac{E_s+2b_s}{R_s}\kappa_{is} - {b_i}^2$
E_s	all	$\frac{\theta_i}{p_i}\sum_s (E_s + 2b_s)\kappa_{is} - {b_i}^2$

References

[1] Philip Dutre and Yves D. Willems, "Importance-driven Monte Carlo light tracing", *Fifth Eurographics Workshop on Rendering*, pp. 185–194, Darmstadt, Germany, June 1994. Appeared in *Photorealistic Rendering Techniques* (eds. G. Sakas, P. Shirley, and S. Müller), Springer, Heidelberg.

[2] R.L. Cook, T. Porter, L. Carpenter, "Distributed Ray Tracing", *Computer Graphics (SIGGRAPH '84 Proc.)*, 18(3), pp. 137-145, July 1984.

[3] S.M. Ermakow, *Die Monte-Carlo-Methode und verwandte Fragen*. R. Oldenbourg Verlag, München-Wien,1975.

[4] Martin Feda and Werner Purgathofer, "Progressive ray refinement for Monte Carlo radiosity", *Fourth Eurographics Workshop on Rendering*, pp. 15–26, Paris, France, June 1993. Eurographics Technical Report Series EG 93 RW.

[5] Gladimir Baranoski, Randall Bramley and Peter Shirley, "Fast radiosity solutions for environments with high average reflectance", *Sixth Eurographics Workshop on Rendering* Appeared in *Rendering Techniques '95* , pp. 345–356, Springer, Wien, 1995.

[6] J.H. Halton, "Sequential Monte Carlo", *Proc. Cambridge Philos. Soc.*,Vol. N. pp. 57–78, 1961.

[7] J.T. Kajiya, "The Rendering Equation", *Computer Graphics (SIGGRAPH '86 Proc.)*, vol. 20, pp. 143-150, August 1986.

[8] A. Keller, "Quasi Monte Carlo Radiosity", *Proceedings of the Seventh Eurographics Workshop on Rendering.* Appeared in *Rendering Techniques '96* , pp. 102–111, Springer, Wien, 1996.

[9] Eric P. Lafortune and Yves D. Willems, "Bi-directional path tracing", *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pp. 145–153, Alvor, Portugal, December 1993.

[10] Attila Neumann, Laszlo Neumann, Philippe Bekaert, Yves Willems and Werner Purgathofer, "Importance-Driven Stochastic Ray Radiosity", *Proceedings of the Seventh Eurographics Workshop on Rendering.* Appeared in *Rendering Techniques '96* , pp. 111–122, Springer, Wien, 1996.

[11] S. N. Pattanaik and S. P. Mudur, "Computation of global illumination by Monte Carlo simulation of the particle model of light", *Proceedings of the Third Eurographics Workshop on Rendering*, pp. 71–83, Bristol, UK, May 1992.

[12] S. Pattanaik and S. Mudur, "Efficient potential equation solutions for global illumination computation", *Computers & Graphics*, **17**(4):387–396, 1993.

[13] Reuven Y. Rubinstein. *Simulation and the Monte Carlo method.* John Wiley & Sons, New York, 1981.

[14] Mateu Sbert, "Error and complexity of random walk Monte Carlo radiosity", *IEEE Transactions on Visualization and Computer Graphics*, **3**(1), March 1997.

[15] Mateu Sbert. *The use of global random directions to compute radiosity. Global Monte Carlo methods.* Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, March 1997. Available in *http://ima.udg.es/˜ mateu*

[16] Mateu Sbert, "Variance of two infinite path length random walk radiosity estimators", *Proceedings of WSCC'97*, pp. 475-483, Plzen, February 1997. An extended version to appear in *Computers&Graphics*.

[17] Mateu Sbert, "Optimal source selection in shooting random walk radiosity", *Computer Graphics Forum (Proceedings of Eurographics'97)*, **16**(3), pp. C301–C308, Budapest, September 1997.

[18] Mateu Sbert, Alex Brusi "Comparing finite and biased infinite path length shooting random walk estimators for radiosity", *Proceedings of SCCG'97*, Budmerice, Slovakia, June 1997.

[19] Mateu Sbert "Random Walk radiosity with generalized absorption probabilities", Submitted to *CGI'98*.

[20] Peter Shirley, "A ray tracing method for illumination calculation in diffuse-specular scenes", *Proceedings of Graphics Interface '90*, pp. 205–212, Toronto, Ontario, May 1990.

[21] P. Shirley, "Time Complexity of Monte Carlo Radiosity", *Proceedings of Eurographics'91*, Vienna, pp. 459-465, Elsevier Science Publishers, Amsterdam, North-Holland, September 1991.

[22] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, 2nd edition, Springer, New York, 1993.

[23] Eric Veach and Leonidas J. Guibas, "Optimally combining sampling techniques for Monte Carlo rendering", *ACM Computer Graphics Proceedings, (ACM SIGGRAPH '95 Proceedings)*, pp. 419–428, 1995.

[24] G.J. Ward, F.M. Rubinstein, R.D. Clear, "A Ray Tracing Solution for Diffuse Interreflection", *Computer Graphics (SIGGRAPH '88 Proc.)*, vol. 22, pp. 85-92, August 1988.

EUROGRAPHICS 2002 / G. Drettakis and H.-P. Seidel
(Guest Editors)

Volume 21 (2002), Number 3

Efficient Multidimensional Sampling

Thomas Kollig and Alexander Keller

Department of Computer Science, Kaiserslautern University, Germany

Abstract

Image synthesis often requires the Monte Carlo estimation of integrals. Based on a generalized concept of stratification we present an efficient sampling scheme that consistently outperforms previous techniques. This is achieved by assembling sampling patterns that are stratified in the sense of jittered sampling and N-rooks sampling at the same time. The faster convergence and improved anti-aliasing are demonstrated by numerical experiments.

Categories and Subject Descriptors (according to ACM CCS): G.3 [Probability and Statistics]: Probabilistic Algorithms (including Monte Carlo); I.3.2 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

1. Introduction

Many rendering tasks are given in integral form and usually the integrands are discontinuous and of high dimension, too. Since the Monte Carlo method[22] is independent of dimension and applicable to all square-integrable functions, it has proven to be a practical tool for numerical integration. It relies on the point sampling paradigm and such on sample placement. Increasing the uniformity of the samples is crucial for the efficiency of the stochastic method and the level of noise contained in the rendered images.

The most popular uniform sampling schemes in graphics are jittered and Latin hypercube sampling. Jittered sampling[2] profoundly has been analyzed by Mitchell[13] and in fact can only improve efficiency. Chiu et al.[1] joined the concepts of jittered and Latin hypercube sampling obtaining an increased uniformity of the samples, but no minimum distance property can be guaranteed that has been proved to be useful in graphics[2]. In consequence care of the choice of the strata has to be taken manually, since warping[19] these point sets in order to e.g. sample long thin light sources can dramatically reduce the benefits of stratification.

We present an unbiased Monte Carlo integration scheme that consistently outperforms the previous approaches, is trivial to implement, and robust to use even with warping. This is obtained by an even more general concept of stratification than just joining jittered and Latin hypercube sampling. Since our samples are highly correlated and satisfy a minimum distance property, noise artifacts are attenuated much more efficiently and anti-aliasing is improved.

2. Monte Carlo Integration

The Monte Carlo method of integration estimates the integral of a square-integrable function f over the s-dimensional unit cube by

$$\int_{[0,1)^s} f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(\xi_i) \ , \qquad (1)$$

where the $\xi_i \in [0,1)^s$ are independent uniform random samples. The efficiency of the stochastic method is inversely proportional to the variance σ^2_{MC} of the estimator (1). Among many variance reduction techniques[22, 23, 11], increasing the uniformity of the samples by stratification has been proven to be beneficial in graphics[13, 2]. We briefly review the facts relevant to this paper; for a more complete survey we refer to e.g. Glassner's book[6] or the notes[9] of the course 'Beyond Monte Carlo'.

2.1. Jittered Sampling

For jittered sampling[2] the unit cube is subdivided into N cubes of equal measure $\frac{1}{N}$, where in each cube one

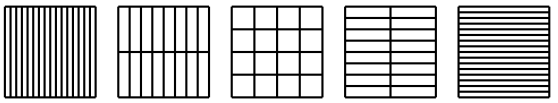


Figure 1: *All elementary intervals in base $b = 2$ and dimension $s = 2$ with volume $\lambda_2(E) = \frac{1}{16}$.*

random sample is taken (see figure 2 (a)). It is simple to show[7] that the variance of the resulting estimator never can be higher than σ^2_{MC}.

2.2. Latin Hypercube Sampling

The idea of Latin hypercube sampling (N-rooks sampling) is to subdivide the unit cube into N intervals along each coordinate. Then the samples are chosen randomly such that each interval contains exactly one point (see figure 2 (c)). Since there are more restrictions in the placement of Latin hypercube samples in comparison to jittered sampling, the variance

$$\sigma^2_{\mathrm{LHS}} \leq \left(\frac{N}{N-1}\right)^{\min\{s-1,1\}} \cdot \sigma^2_{\mathrm{MC}}$$

can slightly increase. Nevertheless it never can be much higher and often is reduced in practical application.

3. Uniform Samples from (t, m, s)-Nets

Chiu et al.[1] combined jittered and Latin hypercube sampling in order to achieve more uniformity. An even more general concept of stratification has been developed by Sobol'[21] that finally yielded the so-called (t, m, s)-nets and (t, s)-sequences[14].

In order to explain the concept, the notion of the elementary interval

$$E := \prod_{j=1}^{s} \left[\frac{a_j}{b^{l_j}}, \frac{a_j+1}{b^{l_j}}\right) \subseteq [0,1)^s$$

is required, where $0 \leq a_j < b^{l_j}$ and $0 \leq l_j$ are integers. Consequently the volume of E is

$$\lambda_s(E) = \prod_{j=1}^{s} \frac{1}{b^{l_j}} = b^{-\sum_{j=1}^{s} l_j} \quad .$$

As an example figure 1 shows the structure of all elementary intervals with the volume $\lambda_2(E) = \frac{1}{16}$ in base $b = 2$ for dimension $s = 2$.

Given two integers $0 \leq t \leq m$ a set of $N = b^m$ s-dimensional points x_i is called a (t, m, s)-net in base b if every elementary interval with volume $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.

t can be considered as a quality parameter that is best if chosen small. For $t = 0$ each elementary interval contains exactly $b^0 = 1$ point. Consequently the b^{ks} points of a $(0, ks, s)$-net in base b with $k \in \mathbb{N}$ are stratified like both jittered and Latin hypercube sampling points at the same time as can be seen in figure 2 (b). In addition the structure of the elementary intervals imposes even more stratification resulting in an increased uniformity of the samples.

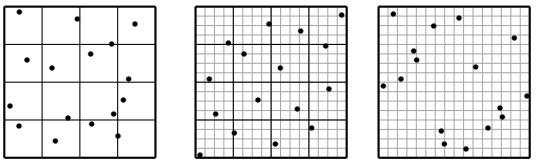


Figure 2: *Realization of (a) jittered and (c) Latin hypercube sampling. The realization of a $(0, 4, 2)$-net in base 2 in (b) not only combines both sampling techniques, but imposes even more stratification as can be seen from the corresponding dyadic elementary intervals in figure 1.*

In the sequel we explain how to efficiently construct such point sets suited for unbiased Monte Carlo integration.

3.1. Deterministic Generation

(t, m, s)-nets are much more uniformly distributed than random samples can be. This is exploited by quasi-Monte Carlo integration[15], where deterministic (t, m, s)-nets are used for the estimator (1): For certain, very restricted function classes a quadratically faster convergence can be guaranteed as compared to random sampling.

Most deterministic constructions of (t, m, s)-nets are based on (t, s)-sequences: For an integer $t \geq 0$ an infinite point sequence $(y_i)_{i=0}^{\infty}$ is called a (t, s)-sequence in base b, if for all $k \geq 0$ and $m > t$ the point set $\{y_{kb^m}, \ldots, y_{(k+1)b^m-1}\}$ is a (t, m, s)-net.

Consequently the first b^m points of a (t, s)-sequence form a (t, m, s)-net. A second approach is to add the component $\frac{i}{b^m}$ to the first b^m points of a (t, s)-sequence always yielding a $(t, m, s + 1)$-net.

Since explaining explicit constructions is beyond the scope of this paper, we refer to Niederreiter's book[15] and provide the compact implementation (section 7) of three $(0, 1)$-sequences that can be used to generate a $(0, 2)$-sequence and $(0, m, 2)$-nets.

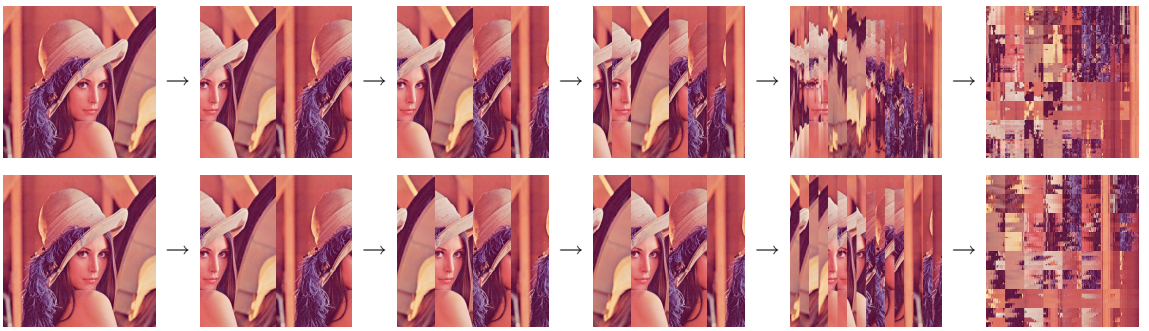


Figure 5: *Owen scrambling (top row) and random digit scrambling (bottom row) in base 2. A difference is hardly perceivable. First intervals are swapped horizontally; the final image then includes the permutations along the vertical direction, too.*

Figure 3: *The effect of a Cranley-Patterson rotation by the random vector ξ.*

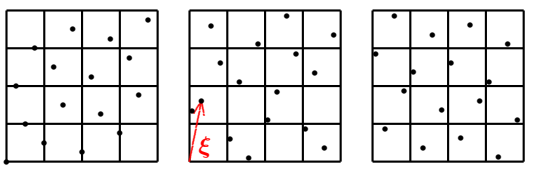


Figure 4: *Randomizing the $(0, 4, 2)$-net in base 2 in a) by a Cranley-Patterson rotation can degrade the uniformity as shown in b), whereas c) random digit scrambling preserves the properties of the net.*

3.2. Randomized Generation

The quasi-Monte Carlo method yields consistent but biased estimators. However, it is possible to randomize a (t, m, s)-net $P := \{a_0, a_1, \ldots, a_{N-1}\}$ in such a way that

a) the randomized point set $X := \{x_0, x_1, \ldots, x_{N-1}\}$ remains a (t, m, s)-net (with probability 1) and
b) x_i is uniformly distributed in $[0, 1)^s$ for $i = 0, 1, \ldots, N-1$.

Condition b) is sufficient to make (1) an unbiased estimator for all square-integrable functions[16, 8]. Preserving the uniformity properties of the samples by condition a) allows one to benefit from the improved convergence of the quasi-Monte Carlo method. The resulting variance reduction technique belongs to the domain of randomized quasi-Monte Carlo integration[18, 10].

3.2.1. Cranley-Patterson Rotations

Cranley and Patterson[3] randomized a point set P by just adding the same random shift ξ to each point $a_i \in P$ modulo 1 as illustrated in figure 3. Originally developed for point sets that tile periodically, applying a so-called Cranley-Patterson rotation to a (t, m, s)-net can destroy its stratification structure (see figure 4) thus violating condition a).

3.2.2. Owen Scrambling

Owen's randomization scheme preserves the structure of (t, m, s)-nets in base b (with probability 1). For the (involved) formulas we refer to the original work[16]. The actual algorithm, however, is simple to explain. Starting with $H = [0, 1)^s$ the following steps are applied to each coordinate (see figure 5):

1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
2. Randomly permute these volumes in an independent way.
3. For each volume H_h recursively repeat the procedure starting out with $H = H_h$.

Owen[17] proved that using an Owen-scrambled $(0, m, s)$-net in (1) yields the upper bound

$$\sigma^2_{\mathrm{OS}} \leq \left(\frac{b}{b-1}\right)^{\min\{s-1,m\}} \cdot \sigma^2_{\mathrm{MC}}$$

for the variance σ^2_{OS} of the resulting estimator. For $b = N$ this $(0, m, s)$-net sampling degenerates to Latin hypercube sampling. Decreasing the base b implies more restrictions to the sample placement resulting in an increased variance bound. Although this variance bound is strict[17], for most functions to be integrated the variance is reduced.

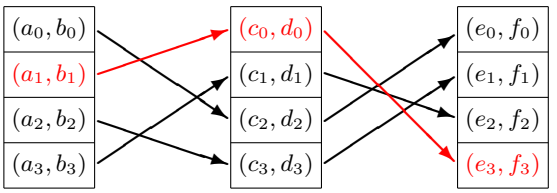


Figure 6: *Multidimensional sampling. The highlighted sample $(a_1, b_1, c_0, d_0, e_3, f_3)$ is padded from the stratified patterns (a_i, b_i),(c_i, d_i), and (e_i, f_i) using random permutations.*

Due to the finite precision of computer arithmetic the infinite scheme in fact becomes a finite algorithm. Nevertheless the number of required random permutations behaves exponentially in the precision so that an efficient implementation remains quite challenging[5].

3.2.3. Random Digit Scrambling

Instead of using independent random permutations in each level of the recursion of Owen scrambling, only one random permutation can be used (see the bottom row of figure 5). This subset of the original method obviously still fulfills the conditions of section 3.2, but requires only a number of permutations linear in the precision. Opposite to Owen's scrambling method, using random digit scrambling preserves minimum distance properties contained in the net to be scrambled.

A highly efficient implementation becomes available for (t, m, s)-nets in base $b = 2$, where a permutation simply can be realized by the `XOR` operation[4, 5]: Each coordinate of the point set is randomized by just performing a bitwise `XOR` of one random bit vector (i.e. a random integer) and the components of the point set (for the trivial realization see section 7).

4. Multidimensional Sampling

Typically the integrands in image synthesis expose high correlation with respect to certain low-dimensional projections, e.g. the pixel area, lens area, or area light sources. Therefore high-dimensional samples are padded using low-dimensional stratified patterns[20]. Correlation artifacts are avoided by randomly permuting the sample order of the low-dimensional patterns (see figure 6). Additionally the number of samples becomes independent of dimension making this approach more practical than jittered sampling.

Although constructions of (t, m, s)-nets exist for any dimension, choosing the optimal quality parameter $t = 0$ requires $b \geq s - 1$ for $m \geq 2$. For $s > 3$ this prohibits to use the extraordinarily efficient vectorized implementations in base $b = 2$. However, using the simple algorithms from section 7, it is possible to pad high-dimensional samples in an even simpler way: Instead of using random permutations we just pad independent realizations[10] of randomly digit scrambled nets (or Owen-scrambled nets). Since condition (2) (section 3.2) holds for the low-dimensional realizations, each resulting high-dimensional sample x_i is uniformly distributed in $[0, 1)^s$ for $i = 0, 1, \ldots, N - 1$, too, guaranteeing an unbiased estimate (1).

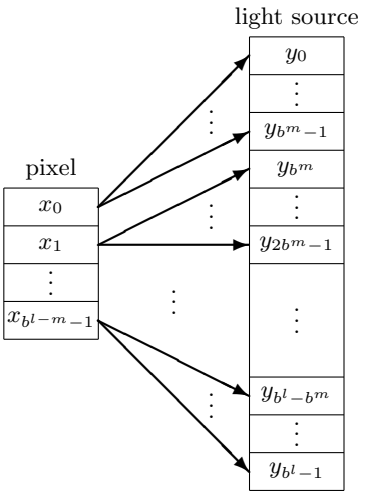


Figure 7: *Trajectory splitting, see the explanation in section 4.1.*

4.1. Trajectory Splitting

Considering the example of distribution ray tracing[2] splitting trajectories[8], e.g. tracing multiple shadow rays for one eye ray, can increase efficiency depending on the correlation coefficient with respect to the split dimensions[22].

From the definition in section 3.1 it follows that the first b^l points of a (t, s)-sequence $(y_j)_{j=0}^{\infty}$ are a (t, l, s)-net. In addition each point set $\{y_{ib^m}, \ldots, y_{(i+1)b^m-1}\}$ is a (t, m, s)-net for $0 \leq i < b^{l-m}$. This observation can be used to realize trajectory splitting by extending the scheme from the previous section:

For the example of pixel anti-aliasing and illumination by an area light source two independent realizations are required: An instance of a randomized $(0, l - m, 2)$-net of b^{l-m} samples x_i in the pixel and the first $b^l = b^{l-m} \cdot b^m$ samples y_j of an instance of a randomized $(0, 2)$-sequence on the area light source. For the i-th sample in the pixel then b^m shadow rays have to be traced towards the samples

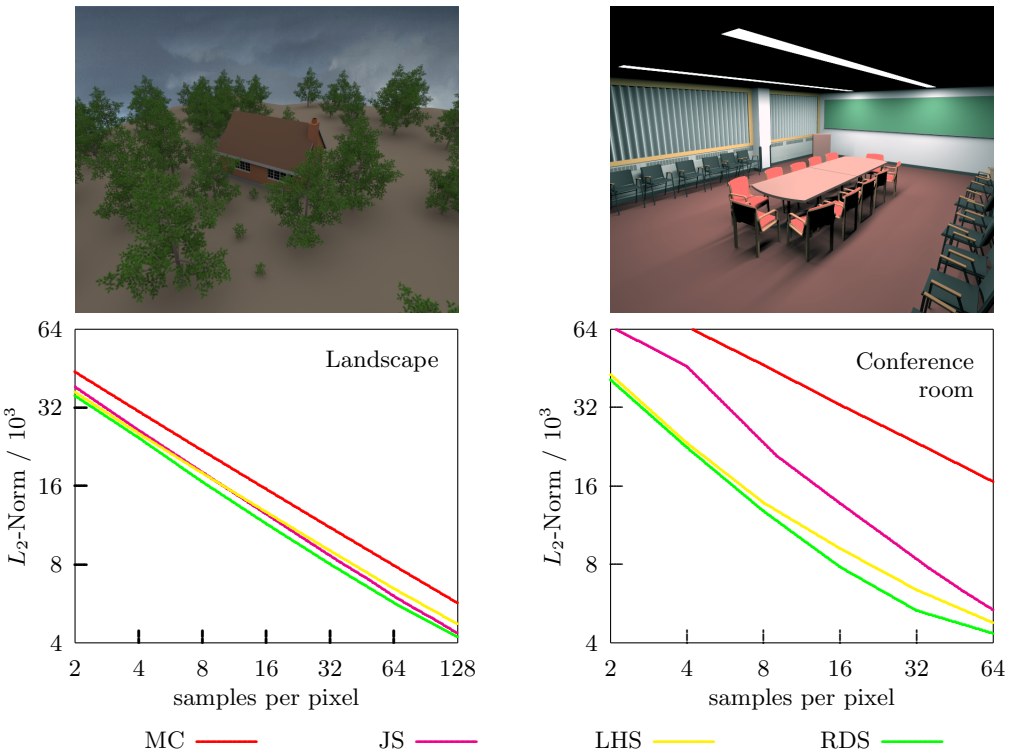


Figure 8: *Comparison of pure random (MC), jittered (JS), and Latin hypercube (LHS) sampling with our approach using random digit scrambling (RDS).*

$\{y_{ib^m}, \ldots, y_{(i+1)b^m-1}\}$ on the light source (see figure 7) yielding the estimator

$$\int_{[0,1)^2}\int_{[0,1)^2} f(x,y)dydx \approx \frac{1}{b^{l-m}}\sum_{i=0}^{b^{l-m}-1}\frac{1}{b^m}\sum_{j=ib^m}^{(i+1)b^m-1} f(x_i,y_j)\ . \quad (2)$$

By using the subsequent $(0,m,2)$-nets of a $(0,2)$-sequence to realize trajectory splitting, the samples on the light source itself form a $(0,l,2)$-net obtaining superior stratification properties in a fully automatic way. This would be rather costly to achieve by jittered or Latin hypercube sampling.

5. Numerical Results

For the application examples two representative settings were selected: An overcast sky model daylight simulation and an indoor scene with very long and thin light sources. The resulting four-dimensional integrals compute pixel anti-aliasing with direct illumination.

The new scheme (2) with x_i and y_i from the algorithms in section 7 is compared to pure random, jittered, and Latin hypercube sampling. In the experiments a splitting rate of 4 was used, i.e. for each eye ray 4 shadow rays were traced. For each pixel an independent realization of the sampling scheme was used.

Trajectory splitting for jittered and Latin hypercube sampling was realized by generalizing the multidimensional sampling scheme[20] in a straightforward way: N samples and $4N$ samples were generated on the pixel and the light source, respectively. Then the set of $4N$ points randomly is split into N sets of 4 points and each set is assigned a pixel sample in canonical order.

The error graphs in figure 8 are determined by computing the L_2-norm of a measurement to a converged master image. For the case of the hemispherical overcast sky integral our scheme slightly outperforms jittered and Latin hypercube sampling, is much simpler to implement, and saves about 10–15% of the total number of rays to be traced in order to obtain the same quality. Due to the complex shadowing the overall gain by stratification is small.

Warping the samples onto the long thin light sources in the conferences room scene exposes the projection regularity of the samples. Therefore Latin hypercube

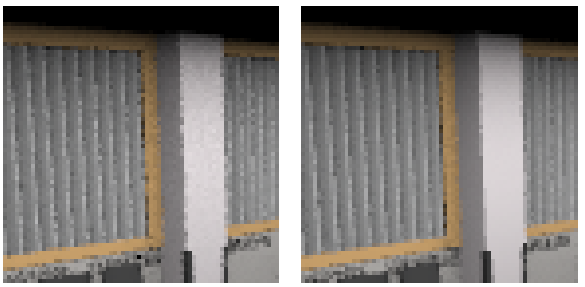


Figure 9: *Improved anti-aliasing and noise reduction for the conference room scene with long light sources. Latin hypercube sampling in the left image and our new sampling scheme on the right.*

sampling significantly outperforms jittered sampling. The samples from the new scheme, however, are stratified in a more general way and satisfy a minimum distance property reducing the error by approximately 15% as compared to Latin hypercube sampling.

Comparing the zoomed images in figure 9 shows that the high correlation of the samples from the new scheme results in superior anti-aliasing and noise reduction as compared to Latin hypercube sampling. This becomes even more apparent in animations, where uncorrelated noise causes distracting flicker.

6. Conclusion

We presented new algorithms for efficiently generating high-dimensional uniform samples yielding unbiased Monte Carlo estimators. The implementation of the highly correlated sampling scheme is extremely simple and due to the generalized concept of stratification previous patterns are outperformed consistently.

7. Appendix: Algorithms

Using the following code fragments it is possible to verify the results of the paper with any ray tracer in a very short amount of time. The routines `RI_vdC`, `RI_S`, and `RI_LP` implement the radical inverse functions by van der Corput[15], Sobol'[21], and Larcher and Pillichshammer[12], respectively, which are $(0,1)$-sequences in base $b = 2$ (see section 3.1). Randomized digit scrambling (section 3.2.3) is realized by just calling the routines with a random integer instead of the default parameter `uint r = 0`. Completing `RI_vdC` with the component $\frac{i}{2^m}$ yields the famous Hammersley point set, which in fact is a $(0, m, 2)$-net. Using $x_i = \left(\frac{i}{2^m}, \texttt{RI\_LP}(i)\right)$ instead, however, results in a $(0, m, 2)$-net of much higher quality. Combining $y_i = (\texttt{RI\_vdC}(i), \texttt{RI\_S}(i))$ results in the first two components of the Sobol' sequence, which form a $(0, 2)$-sequence as used in section 4.1.

```
typedef unsigned int uint;

double RI_vdC(uint bits, uint r = 0)
{
  bits = ( bits                << 16)
         | ( bits                >> 16);
  bits = ((bits & 0x00ff00ff) <<  8)
         | ((bits & 0xff00ff00) >>  8);
  bits = ((bits & 0x0f0f0f0f) <<  4)
         | ((bits & 0xf0f0f0f0) >>  4);
  bits = ((bits & 0x33333333) <<  2)
         | ((bits & 0xcccccccc) >>  2);
  bits = ((bits & 0x55555555) <<  1)
         | ((bits & 0xaaaaaaaa) >>  1);

  bits ^= r;

  return (double) bits / (double) 0x100000000LL;
}

double RI_S(uint i, uint r = 0)
{
  for(uint v = 1<<31; i; i >>= 1, v ^= v>>1)
    if(i & 1)
      r ^= v;

  return (double) r / (double) 0x100000000LL;
}

double RI_LP(uint i, uint r = 0)
{
  for(uint v = 1<<31; i; i >>= 1, v |= v>>1)
    if(i & 1)
      r ^= v;

  return (double) r / (double) 0x100000000LL;
}
```

Acknowledgements

The first author has been funded by the Stiftung Rheinland-Pfalz für Innovation.

References

1. K. Chiu, C. Wang, and P. Shirley. Multi-Jittered Sampling. In S. Heckbert, editor, *Graphics Gems IV*, pages 370–374. Academic Press, 1994. 1, 2
2. R. Cook, T. Porter, and L. Carpenter. Distributed Ray Tracing. In *Computer Graphics (SIGGRAPH 84 Conference Proceedings)*, pages 137–145, 1984. 1, 4
3. R. Cranley and T. Patterson. Randomization of number theoretic methods for multiple integration. *SIAM Journal on Numerical Analysis*, 13:904–914, 1976. 3

4. I. Friedel. Abtastmethoden für die Monte-Carlo und Quasi-Monte-Carlo-Integration. Universität Kaiserslautern, 1998. 4

5. I. Friedel and A. Keller. Fast generation of randomized low-discrepancy point sets. In H. Niederreiter, K. Fang, and F. Hickernell, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 257–273. Springer, 2002. 4

6. A. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, 1995. 1

7. M. Kalos and P. Whitlock. *Monte Carlo Methods, Volume I: Basics*. J. Wiley & Sons, 1986. 2

8. A. Keller. Trajectory Splitting by Restricted Replication. Interner Bericht 316/01, Universität Kaiserslautern, 2001. 3, 4

9. A. Keller. Beyond Monte Carlo – Course Material. Interner Bericht 320/02, University of Kaiserslautern, 2002. Lecture at the Caltech, July 30th – August 3rd, 2001. 1

10. T. Kollig and A. Keller. Efficient bidirectional path tracing by randomized quasi-Monte Carlo integration. In H. Niederreiter, K. Fang, and F. Hickernell, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 290–305. Springer, 2002. 3, 4

11. E. Lafortune. *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*. PhD thesis, Katholieke Universitiet Leuven, Belgium, 1996. 1

12. G. Larcher and F. Pillichshammer. Walsh Series Analysis of the L_2-Discrepancy of Symmetrisized Point Sets. *Monatsh. Math.*, (132):1–18, 2001. 6

13. D. Mitchell. Consequences of Stratified Sampling in Graphics. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 277–280, 1996. 1

14. H. Niederreiter. Point sets and sequences with small discrepancy. *Monatsh. Math.*, 104:273–337, 1987. 2

15. H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, Pennsylvania, 1992. 2, 6

16. A. Owen. Randomly Permuted (t, m, s)-Nets and (t, s)-Sequences. In H. Niederreiter and P. Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, volume 106 of *Lecture Notes in Statistics*, pages 299–315. Springer, 1995. 3

17. A. Owen. Monte Carlo Variance of Scrambled Net Quadrature. *SIAM J. on Numerical Analysis*, 34(5):1884–1910, 1997. 3

18. A. Owen. Monte Carlo Extension of Quasi-Monte Carlo. In *Winter Simulation Conference*, pages 571–577. IEEE Press, 1998. 3

19. P. Shirley. Nonuniform Random Point Sets via Warping. In D. Kirk, editor, *Graphics Gems III*, pages 80–83. Academic Press Professional, 1992. 1

20. P. Shirley. *Realistic Ray Tracing*. AK Peters, Ltd., 2000. 4, 5

21. I. Sobol'. On the Distribution of Points in a Cube and the approximate Evaluation of Integrals. *Zh. vychisl. Mat. mat. Fiz.*, 7(4):784–802, 1967. 2, 6

22. I. Sobol'. *A Primer for the Monte Carlo Method*. CRC Press, 1994. 1, 4

23. E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997. 1

Efficient Bidirectional Path Tracing by Randomized Quasi-Monte Carlo Integration

Thomas Kollig and Alexander Keller

kollig@informatik.uni-kl.de and keller@informatik.uni-kl.de
Computer Science Dept., University of Kaiserslautern
Postfach 3049, D-67653 Kaiserslautern, Germany

Abstract. As opposed to Monte Carlo integration the quasi-Monte Carlo method does not allow for an error estimate from the samples used for the integral approximation and the deterministic error bound is not accessible in the setting of computer graphics, since usually the integrands are of unbounded variation. We investigate the application of randomized quasi-Monte Carlo integration to bidirectional path tracing yielding much more efficient algorithms that exploit low-discrepancy sampling and at the same time allow for variance estimation.

1 Introduction

The global illumination problem consists in rendering photorealistic images of a virtual scene and camera descriptions (for a detailed introduction see [Gla95]). A very basic algorithm for the solution of this light transport problem is the bidirectional path tracing algorithm [LW93,VG94], which in the context of the quasi-Monte Carlo method has been investigated in [Kel98b].

We first generalize this work by introducing multiple importance sampling using the balance heuristic for the quasi-Monte Carlo method, which is superior to the previous approaches of bidirectional path tracing. By sacrificing only little performance randomized quasi-Monte Carlo algorithms allow for integration error estimation. By numerical experiments we compare the efficiency of different randomized quasi-Monte Carlo approaches and illustrate how they smoothly blend between the pure Monte Carlo and the quasi-Monte Carlo case.

As a result the new scheme of *padded replications sampling* yields a bidirectional path tracing algorithm that is highly efficient, allows for an error estimate and is very simple to implement.

2 Bidirectional Path Tracing

We briefly recall the path integral formulation of the global illumination problem [Vea97], which in combination with multiple importance sampling yields the bidirectional path tracing algorithm. Furthermore we define the problem of insufficient techniques that is inherent with multiple importance sampling.

2.1 The Global Illumination Problem in Path Integral Form

A light path $\bar{x} = x_0 x_1 \dots x_k$ of length k is characterized by its interaction points x_i with the scene surface S. The union of all path spaces

$$\mathcal{P}_k := \{\bar{x} = x_0 x_1 \dots x_k \mid x_i \in S \text{ for } 0 \leq i \leq k\}$$

of a specific light path length k forms the path space $\mathcal{P} := \bigcup_{k=1}^{\infty} \mathcal{P}_k$. For Lebesgue measurable subsets $D_0, D_1, \dots, D_k \subseteq S$ we define the measure

$$\mu_k(D_0 \times D_1 \times \dots \times D_k) := A(D_0) \cdot A(D_1) \cdot \dots \cdot A(D_k) \ ,$$

where A is the area measure, and $\mu(D) := \sum_{k=1}^{\infty} \mu_k(D \cap \mathcal{P}_k)$ for $D \subseteq \mathcal{P}$. For a path $\bar{x} \in \mathcal{P}_k$ the measurement contribution function is

$$f_j(\bar{x}) := L_e(x_0 \to x_1)\, G(x_0 \leftrightarrow x_1) \cdot \left(\prod_{i=1}^{k-1} f_s(x_{i-1} \to x_i \to x_{i+1})\, G(x_i \leftrightarrow x_{i+1}) \right) \cdot W_e^{(j)}(x_{k-1} \to x_k) \ ,$$

where the light sources are determined by the emittance L_e and $W_e^{(j)}$ are the detector functionals, which formalize the camera description. The bidirectional scattering distribution function f_s describes the surface properties.

$$G(x \leftrightarrow y) := V(x \leftrightarrow y) \frac{|\cos\theta_x||\cos\theta_y|}{|x-y|^2}$$

is the geometry term, where θ_x is the angle between the surface normal in x and the vector between x and y; θ_y is defined analogously. The visibility function $V(x \leftrightarrow y)$ is 1 if x and y are mutually visible and 0 otherwise. Then the global illumination problem consists in computing detector values I_j by the path integral

$$I_j = \sum_{k=1}^{\infty} \int_{\mathcal{P}_k} f_j(\bar{x}) d\mu_k(\bar{x}) = \int_{\mathcal{P}} f_j(\bar{x}) d\mu(\bar{x}) \ . \tag{1}$$

2.2 Multiple Importance Sampling

The problem of importance sampling [Sob94] is to find an efficient probability density function p. However, often it is possible to specify a whole set $p_1, p_2, \dots, p_N$ of probability density functions instead of just one single p. While each probability density function of the set may reduce the variance of the importance sampling estimator only in a possibly unknown subdomain of D, multiple importance sampling, a variance reduction technique introduced by [VG94] and analyzed in [OZ99], allows for the combination of samples which are distributed according to different probability density functions.

A probability density function p can be used as a *technique*, if we are able to generate p-distributed samples and to evaluate[1] $p(x)$ for a given $x \in D$. Assuming that we have N techniques with their associated probability density functions

$$p_1, p_2, \ldots, p_N \;:\; D \to \mathbb{R}_0^+ \;,$$

a so-called *heuristic* consists of N corresponding weight functions

$$w_1, w_2, \ldots, w_N \;:\; D \to \mathbb{R}_0^+ \;,$$

such that

1. $\sum_{i=1}^{N} w_i(x) = 1$ for all $x \in D$ with $f(x) \neq 0$ and
2. $w_i(x) = 0$ for all $x \in D$ with $p_i(x) = 0$ holds.

Note that these conditions imply that each $x \in \operatorname{supp} f$ can be generated by at least one[2] technique p_i. Then the multiple importance sampling estimator

$$\int_D f(x)dx \approx \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{N} w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})} \tag{2}$$

is unbiased, where the $x_{i,j}$ are p_i-distributed for $1 \leq i \leq N$ and $1 \leq j \leq n$. We use the so called *balance heuristic* which has the weight functions

$$w_i(x) := \frac{p_i(x)}{\sum_{\ell=1}^{N} p_\ell(x)} \;. \tag{3}$$

The behaviour of the estimator (2) using (3) is comparable to importance sampling with $p \equiv \frac{1}{N} \sum_{\ell=1}^{N} p_\ell$.

2.3 The Bidirectional Path Tracing Algorithm

The path space samples $\bar{x} \in \mathcal{P}$ are generated in three steps (see Fig. 1 (a)):

1. generate a light subpath by a random walk starting on a light source,
2. generate an eye subpath by a random walk starting on a detector, and
3. connect both subpaths deterministically.

Since the ray casting function is very expensive, we use all possible connections to form additional path space samples as illustrated in Fig. 1 (a). The resulting associated probability density functions are denoted by $p_{k,i}$, where k is the path length and i the number of points of the light subpath. Figure 1 (b) shows all possible techniques with their associated probability density functions for path length $k = 3$.

[1] At least we must be able to decide whether $x \in \operatorname{supp} p$ holds for a given $x \in D$.
[2] This can happen due to disjoint $\operatorname{supp} p_i$ or *the problem of insufficient techniques* as addressed in Sec. 2.4.

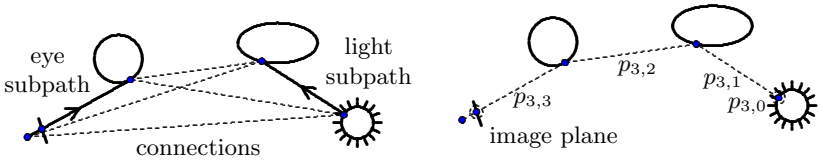


Fig. 1. (a) Generation of path space samples and (b) techniques with their associated probability density functions for path length $k = 3$, where each $p_{3,i}$ is positioned at its deterministic connection. A pinhole camera model is assumed where the eye subpaths originate from the pinhole through the image plane.

Applying multiple importance sampling (2) with the balance heuristic (3) to the path integral formulation of the global illumination problem (1) yields the bidirectional path tracing estimator

$$I_j = \sum_{k=1}^{\infty} \int_{\mathcal{P}_k} f_j(\bar{x}) d\mu_k(\bar{x}) \approx \sum_{k=1}^{\infty} \frac{1}{n} \sum_{j=1}^{n} \sum_{i=0}^{k} \frac{f_j(\bar{x}_{k,i,j})}{\sum_{\ell=0}^{k} p_{k,\ell}(\bar{x}_{k,i,j})} \quad , \tag{4}$$

where n is the number of samples per technique and for $1 \leq j \leq n$ the $\bar{x}_{k,i,j}$ are generated according to $p_{k,i}$. In order to handle the infinite sum we can use absorbing Markov chains for subpath generation. A biased alternative is to compute the approximation up to a maximum path length $k_{\max}$. For example $k_{\max} = 2$ implies that instead of the full solution only direct illumination is calculated.

The possibility to achieve a valid path using the *eye connection techniques* $p_{k,k}$, where the end of a light subpath is connected with the eye point, is very small and most samples are of zero contribution, if the image is computed pixel by pixel. Therefore we allow samples of these techniques to contribute directly to any pixel of the image.

2.4 The Problem of Insufficient Techniques

Multiple importance sampling tries to hide the weaknesses of single probability density functions, but nevertheless can fail. In order to illustrate the limits of the estimator (2) suppose we have a subdomain $G \subseteq D$ for which only one technique is accessible. Then multiple importance sampling degenerates to standard importance sampling on G due to an *insufficient set of techniques*.

For bidirectional path tracing the problem of insufficient techniques arises for singular surface properties, e.g. mirrors that usually are modeled by a Dirac delta distribution in the bidirectional scattering distribution function f_s. In Fig. 2 (a) such a difficult path is sketched: It can only be generated by the technique $p_{k,0}$ that uses no light subpath. The resulting high variance is clearly visible in Fig. 2 (b) and is perceived as white dots on the mirror.

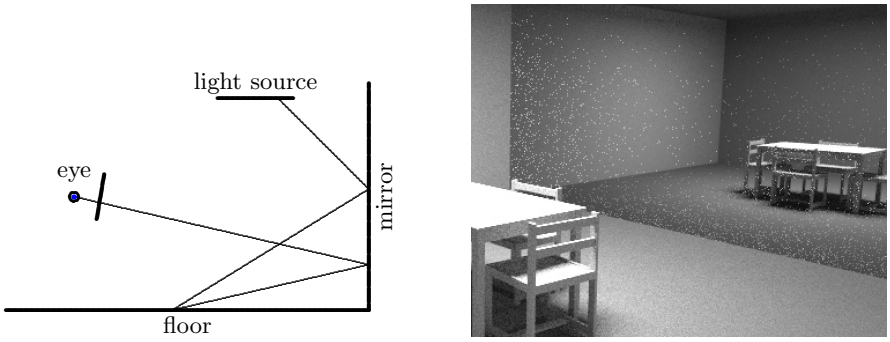


Fig. 2. The problem of insufficient techniques: (a) sketch of difficult path and (b) resulting high variance on the mirror.

3 Quasi-Monte Carlo Bidirectional Path Tracing

The Koksma-Hlawka inequality predicts that quasi-Monte Carlo integration performs superior to Monte Carlo integration for integrands of bounded variation in the sense of Hardy and Krause [Nie92]. For integrands with unknown discontinuities like the measurement contribution function in (1) only pessimistic error bounds are available [Hla71] due to unbounded variation. Nevertheless numerical experiments [Kel98a] reveal that even for these functions low-discrepancy sampling performs better than random sampling.

3.1 Multiple Importance Sampling for Quasi-Monte Carlo

In order to apply quasi-Monte Carlo integration to bidirectional path tracing the subpath generation has to be done using high-dimensional low-discrepancy points, where the dimension depends on the length of the subpaths. Due to the transport operator points at the beginning of a subpath affect the integration error more than points at the end of a subpath. In accordance the lower dimensions of low-discrepancy points often are better equidistributed than their higher dimensions. Therefore the first four dimensions are used to determine the first point of each subpath, the next four for the first scattering events and so on. This interleaving scheme is similar to [Kel98b], however, now we use the multiple importance sampling estimator (2) with the balance heuristic (3) and deterministic low-discrepancy sampling. In order to avoid aliasing different light subpaths have to be used for the estimation of each pixel functional. This is particularly important for the eye connection techniques $p_{k,k}$ (see also Fig. (5)) and is achieved by using consecutive subsequences of a low-discrepancy point sequence instead of a repeated finite point set.

4 Randomized Quasi-Monte Carlo Bidirectional Path Tracing

In [Owe98b] Owen surveys randomization techniques for quasi-Monte Carlo integration. Randomized quasi-Monte Carlo integration exploits the benefits of low-discrepancy sampling and at the same time allows for an efficient error estimate, which is not accessible for quasi-Monte Carlo integration.

From an initial low-discrepancy point set $P := \{a_1, a_2, \ldots, a_m\} \subset I^s$ we generate r randomized replications $X_j := \{x_{1,j}, x_{2,j}, \ldots, x_{m,j}\} \subset I^s$ with $1 \leq j \leq r$ such that

1. each replication X_j preserves the low-discrepancy properties of the initial point set P and
2. the replications $x_{i,1}, x_{i,2}, \ldots, x_{i,r}$ of each point $a_i \in P$ are independent and uniformly distributed on I^s.

Then the randomized quasi-Monte Carlo estimator with a total of $n = rm$ samples

$$\int_{I^s} f(x)dx \approx \frac{1}{r}\sum_{j=1}^{r}\frac{1}{m}\sum_{i=1}^{m} f(x_{i,j}) \tag{5}$$

is unbiased. The expected error is bounded by the square root of the variance σ^2 of the above estimator and can be estimated in an unbiased way using the samples of (5):

$$\sigma^2 \approx \frac{1}{r(r-1)}\sum_{k=1}^{r}\left(\frac{1}{m}\sum_{i=1}^{m} f(x_{i,k}) - \frac{1}{r}\sum_{j=1}^{r}\frac{1}{m}\sum_{i=1}^{m} f(x_{i,j})\right)^2 . \tag{6}$$

Choosing the number r of replications just large enough to obtain a good variance estimate very little performance of the low-discrepancy quadrature is sacrificed and adaptive sampling controlled by error estimation yields much more efficient rendering algorithms.

As long as the replications $x_{i,1}, x_{i,2}, \ldots, x_{i,r}$ are independent and uniformly distributed on I^s the estimator (5) is unbiased and the variance estimator (6) remains valid. Thus the initial point set P and its replications X_j do not need to be of low-discrepancy, however, their choice affects variance and therefore error.

For bidirectional path tracing the initial point set P with m points and the replication scheme have to be selected. Then each pixel functional is estimated by r independent random replications of P. The dimensions are assigned identically to the quasi-Monte Carlo setting in Sect. 3.1.

4.1 Cranley-Patterson Rotations

Cranley and Patterson [CP76] suggested the following form of randomization: For a replication X_j they added a random shift ξ_j to each point a_i of the

initial point set P. Thus we have $x_{i,j} = (a_i + \xi_j) \bmod 1$ with independent realizations $\xi_j \sim U(I^s)$ for $1 \leq i \leq m$ and $1 \leq j \leq r$.

Most low-discrepancy constructions are designed to minimize the star-discrepancy in the sense of (t, m, s)-nets or (t, s)-sequences. By randomly shifting a point set P this discrepancy of a replication X_j can be different and especially worse than the original discrepancy [Tuf96], since the (t, m, s)-net property is not shift invariant. Points designed to also minimize the torus discrepancy [BC87] are better suited for Cranley-Patterson rotations. The equidistribution properties of good lattice points remain almost unaffected when being shifted [CP76,SJ94].

4.2 Owen Scrambling

In [Owe95] Owen presented a randomization scheme for (t, m, s)-nets and (t, s)-sequences in base b. Starting with $H = I^s$ the following algorithm is applied to each coordinate:

1. Slice H into b equal volumes $H_1, H_2, \ldots, H_b$ along the coordinate.
2. Randomly permute these volumes in an independent way.
3. For each volume H_h repeat the procedure with $H = H_h$.

Due to the finite precision of computer arithmetic the infinite scheme in fact becomes a finite algorithm [FK00].

4.3 Padded Replications Sampling

For the light transport problem the benefits of quasi-Monte Carlo integration diminish in high dimensions [Kel98a]. So instead of using a computationally expensive high-dimensional low-discrepancy point set as initial point set P (see Fig. 4 (a)), the structure underneath the transport problem can be exploited: The light and eye subpaths are generated by area sampling and scattering, which both are two-dimensional problems. Thus the idea of padded replications sampling is to use a random replicate of one two-dimensional basis pattern for each two-dimensional subproblem as illustrated in Fig. 3. This in fact only changes the initial point set P (see Fig. 4 (b)) of the replication scheme. Of course P no longer is of low-discrepancy, but is much cheaper to generate and performs at least as good as shown in the experimental section.

5 Latin Supercube Sampling

Similar to padded replications sampling Latin supercube sampling [Owe98a] is a method to expand low-dimensional samples to high dimensions: The low-dimensional point sets are randomly permuted before padding. Suppose that

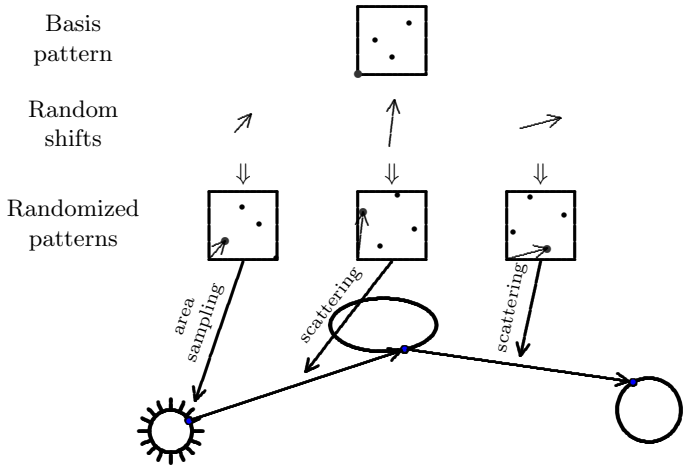


Fig. 3. Subpath generation by padded replications sampling using Cranley-Patterson rotations for e.g. a light subpath. The basis pattern size is $m = 4$ and the dimension of the padded points is $s = 6$.

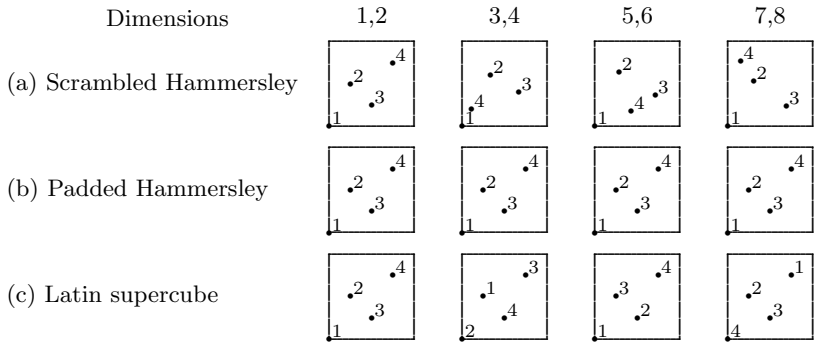


Fig. 4. Illustration of the initial point sets P used by the different approaches for randomized quasi-Monte Carlo integration: (a) high-dimensional low-discrepancy point set, (b) padded replications sampling, and (c) decorrelated padded replications sampling by index permutations.

$Q_i := \{a_{i,1}, a_{i,2}, \ldots, a_{i,m}\} \subset I^{s_i}$ for $1 \leq i \leq q$ with $\sum_{i=1}^{q} s_i = s$ are (randomized) quasi-Monte Carlo point sets. Then the Latin supercube samples are

$$x_j := (a_{1,\pi_1(j)}, a_{2,\pi_2(j)}, \ldots, a_{q,\pi_q(j)}) \subset I^s$$

for $1 \leq j \leq m$, where the π_i are independent uniform random permutations over $\{1, 2, \ldots, m\}$.

In computer graphics Latin supercube sampling has been applied by Cook [Coo86] (later formalized by Shirley [Shi90]) for distribution ray tracing, which is not a consistent algorithm in the sense of (1) since it uses only a subset of the techniques $p_{k,0}$ and $p_{k,1}$, where the end of an eye subpath has to hit a light source or is connected with a point on a light source. However, Cook and Shirley did not use (randomized) quasi-Monte Carlo point sets but stratified random point sets for Latin supercube sampling.

5.1 Latin Supercube Samples from Deterministic Low-Discrepancy Points

Using large two-dimensional quasi-Monte Carlo point sets for Latin supercube sampling is prohibitive due to the considerable amount of permutation memory of order $\mathcal{O}(qm) = \mathcal{O}(k_{\max}m)$, if a finite maximum path length $k_{\max}$ is used. On the other hand the number of different light subpaths that can be generated by Latin supercube sampling is limited by $m^{k_{\max}}$. Therefore for small values of m Latin supercube sampling is only practicable if the eye connection techniques $p_{k,k}$ are not used, otherwise severely disturbing aliasing artifacts will be visible (see Fig. (5)) that only can be avoided by using huge values of m that are of the order of pixels in the image.

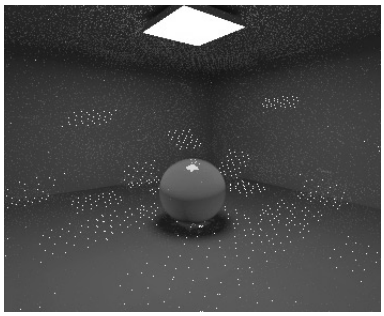


Fig. 5. Aliasing caused by Latin supercube samples from deterministic points with eye connection techniques.

5.2 Latin Supercube Samples from Randomized Low-Discrepancy Points

For padded replications sampling the same basis pattern is padded together to form the initial point set P. The resulting correlation between the dimensions (see Fig. 4 (b)) can cause an increased variance for a larger number m of points in the basis pattern. Latin supercube resolves this correlation when being applied to the initial point set P before replication (see Fig. 4 (c)).

6 Numerical Experiments

For the numerical experiments we chose the Glass Sphere and Office test scenes. Figure 6 shows the master images, which have been rendered with the original bidirectional path tracing algorithm using more than a thousand samples per pixel and technique. In our experiments the error of an image is approximated by its L_2-distance to these master images. Instead of using

Fig. 6. The GLASS SPHERE and OFFICE scene are used as test scenes.

absorbing Markov chains for subpath generation we restricted the maximum path length to $k_{\max} = 6$ for the GLASS SPHERE and to $k_{\max} = 3$ for the OFFICE scene.

The difficulties of the GLASS SPHERE scene are the caustic on the floor and the light, which is reflected by the glass sphere onto the ceiling. The OFFICE scene has only diffuse surface properties. Besides the two big luminaries at the ceiling the small spherical light source of the table lamp makes the rendering complicated. Since here we have no singular surface properties we omit the eye connection techniques $p_{k,k}$, which are only useful in order to render directly seen caustics.

6.1 Quasi-Monte Carlo Bidirectional Path Tracing

In Fig. 7 (a) the performance of several deterministic low-discrepancy sequences is compared. The Niederreiter sequence [Nie92] in base 2 is slightly worse than the scrambled Halton sequence [Fau92] for a small number of samples per technique and pixel and almost as good for more than 64 samples. Optimizing the t parameter by increasing the construction base of the Niederreiter sequence leads to worse results. The Niederreiter-Xing sequence [NX96,Pir00] in base 2 is even far less efficient except for multiples of 64 samples. The reason why the simple scrambled Halton sequence performs best is that it better fits the structure of the global illumination problem: The initial segments of the subpaths contribute most and consequently the lower dimensions of the points are most important. In addition only a very small number of samples is used to estimate a pixel functional. Looking at the low-dimensional projections of short consecutive subsequences yields that the projections of the scrambled Halton sequence expose better discrepancy than the projections of the (t, s)-sequences, causing the superior performance of the scrambled Halton sequence.

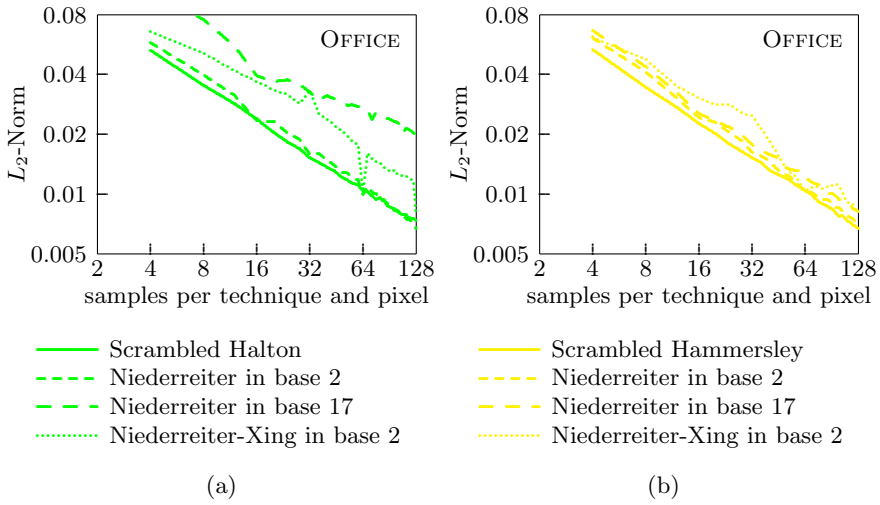


Fig. 7. Convergence graphs for (a) quasi-Monte Carlo bidirectional path tracing using different low-discrepancy sequences and (b) randomized quasi-Monte Carlo bidirectional path tracing using different high-dimensional point sets as initial point set P and different randomization schemes.

6.2 Randomized Quasi-Monte Carlo Bidirectional Path Tracing

Now different approaches to randomized quasi-Monte Carlo bidirectional path tracing are compared, where high-dimensional low-discrepancy point sets are selected as initial point set P. While the scrambled Hammersley point set is chosen for Cranley-Patterson rotations (see 4.1), different (t, s)-nets are used with Owen scrambling (see 4.2). Here each pixel functional is estimated using only one independent replication. The resulting convergence graphs in Fig. 7 (b) show that the scrambled Hammersley version performs best. Using a Niederreiter sequence is slightly worse, where the increased construction base affects the error less than in the quasi-Monte Carlo setting. The Niederreiter-Xing sequence again performs even worse except for multiples of 64 samples.

Usually for randomized quasi-Monte Carlo integration the size m of the initial point set P is fixed and the desired sampling rate is obtained by increasing the number r of replications, yielding a convergence rate of $\mathcal{O}(r^{-\frac{1}{2}})$. Seen that way the points of the graphs in Fig. 7 (b) can be considered as starting points of the convergence graphs for fixed m and increasing r.

6.3 Blending between Monte Carlo and Quasi-Monte Carlo

So far we have analyzed quasi-Monte Carlo and randomized quasi-Monte Carlo separately. Now the best of the above sampling schemes are compared:

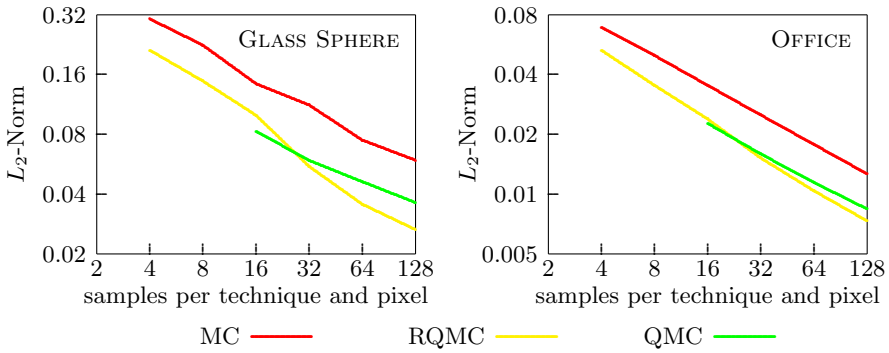


Fig. 8. Convergence graphs for MC, RQMC ($m = 16$), and QMC.

- **Monte Carlo (MC).** The original bidirectional path tracing algorithm uses pure random sampling. For comparison we also implemented a version using Latin hypercube sampling (LHS).
- **Randomized quasi-Monte Carlo (RQMC).** Besides using Cranley-Patterson rotations with the scrambled Hammersley point set also its padded replications sampling (see Sect. 4.3) counterpart using the two-dimensional Hammersley point set has been applied. In both approaches we can choose the fixed size m of the initial point set P.
- **Quasi-Monte Carlo (QMC)** using the scrambled Halton sequence.

In Fig. 8 convergence graphs are shown. As expected the convergence rates for the Monte Carlo and randomized quasi-Monte Carlo versions are $\mathcal{O}(n^{-1/2})$, where n denotes the total number of samples per technique and pixel. A slightly improved rate of $\mathcal{O}(n^{-1/2-\alpha})$ can be observed for the quasi-Monte Carlo approach, where $\alpha \in \left[0, \frac{1}{2}\right]$ decreases with the maximum path length $k_{\max}$ used in the simulation due to the discontinuities in the measurement contribution function.

For a more detailed comparison we measured the number of samples required to achieve a given error in relation to the number needed by the original bidirectional path tracing algorithm, i.e. the pure Monte Carlo algorithm. The results are shown in Fig. 9. Far more than half of the expensive samples can be saved by the quasi-Monte Carlo version. The randomized quasi-Monte Carlo approaches form a smooth transition between the pure random and the deterministic algorithm. With an increasing size m of the initial point set P the error decreases due to the better equidistribution of the samples. It is an interesting result that for bidirectional path tracing padded replications sampling performs at least as good as a high-dimensional low-discrepancy point set.

Along the lines of Cook [Coo86] and Shirley [Shi90] also Latin supercube sampling by deterministic low-discrepancy points (see Sect. 5.1) has been

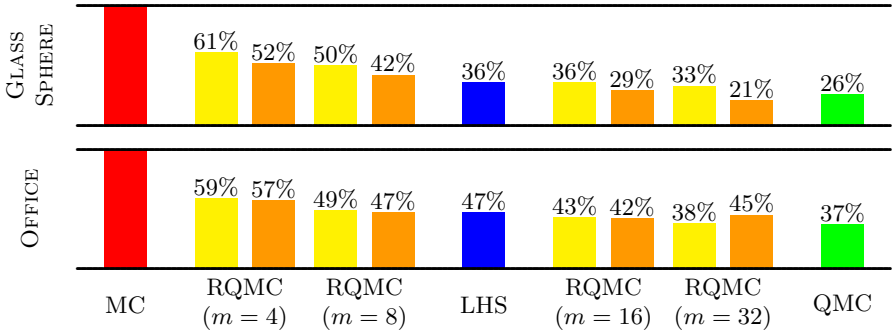


Fig. 9. Number of samples needed to achieve a given error in relation to MC. Padded replications sampling (right bars of RQMC) performs at least as good as the straight forward approach (left bars of RQMC).

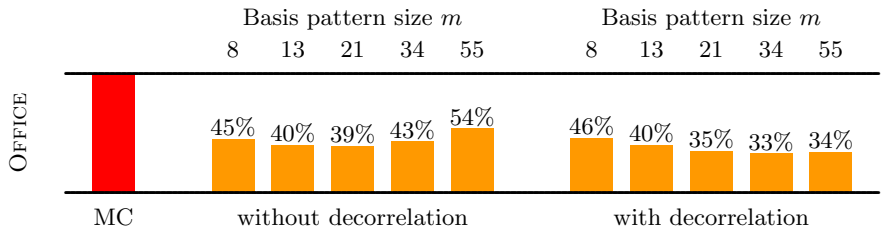


Fig. 10. Using Latin supercube samples as input for Cranley-Patterson rotation reduces the effect of correlation caused by padded replications sampling. For this experiment Fibonacci lattice points have been used as basis pattern.

applied to render the Office scene, since here the eye connection techniques are not required. For padding the two-dimensional Hammersley point set was chosen. In comparison to the pure Monte Carlo algorithm only 35% of the samples are needed to achieve the same error. Thus it performs similar to the purely deterministic quasi-Monte Carlo approach (37% of the samples, see Fig. 9).

6.4 Decorrelation of Padded Replications Sampling

Latin supercube sampling can reduce the correlation between the dimensions of the points used by padded replications sampling. An increased variance due to correlation becomes visible in Fig. 9 for the Office scene when increasing the basis pattern size m from 16 (42%) to 32 (45%).

In Fig. 10 padded replications sampling with and without decorrelation using Latin supercube samples is compared, where we padded replications of the Fibonacci lattice points [SJ94]. The reduced correlation results in a better performance when using bigger basis pattern sizes m. In computer

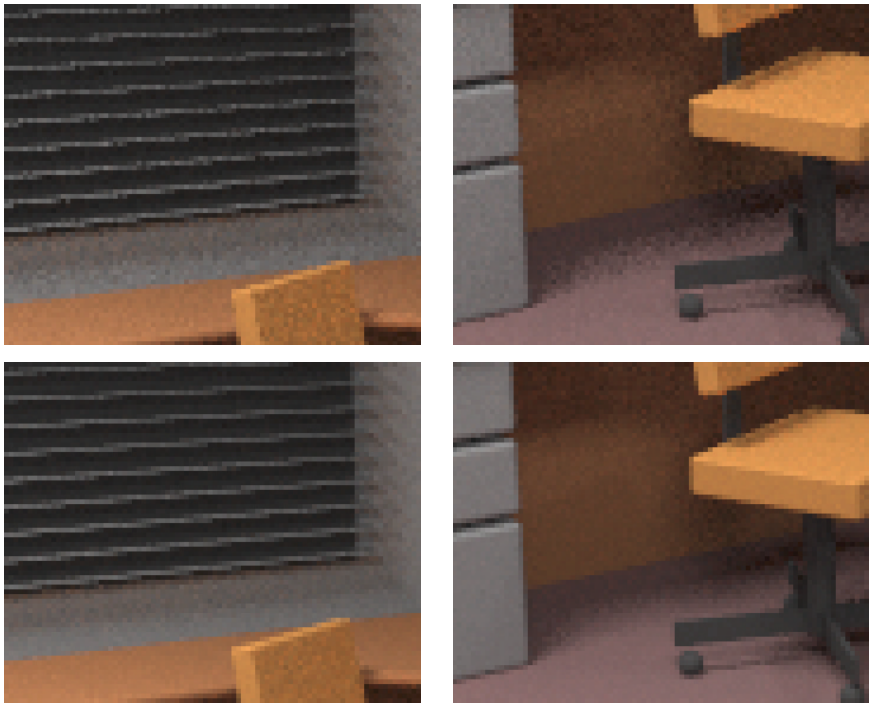


Fig. 11. Image comparison. The close-ups in the upper row were rendered with pure Monte Carlo bidirectional path tracing and the close-ups in the lower row were rendered by the padded replications sampling approach using the Hammersley point set without decorrelation.

graphics, however, $rm = n \leq 128$ so that the effect of decorrelation by Latin supercube sampling is hardly perceivable.

6.5 Visual Comparison

For a visual comparison of images we rendered the OFFICE scene with the original bidirectional path tracing algorithm and with the padded replications sampling using 16 samples per technique and pixel. Since the padded replications sampling version needs fewer pseudo random numbers its rendering time was about 12% shorter. Figure 11 shows two close-ups of the images. The reduced error results in a less noisy image. Even in only indirectly illuminated regions (right column) there is less noise.

7 Conclusion

We investigated several new sampling approaches to bidirectional path tracing speeding up the original algorithm by a factor of 2 to 5. By numerical

evidence we showed that *padded replications sampling* is almost as efficient as the best quasi-Monte Carlo integration approach. However, *padded replications sampling* allows for variance estimation, is much simpler to implement as compared to the high-dimensional low-discrepancy constructions, requires much less random numbers than pure random sampling, and perfectly fits the intrinsic two-dimensional structure of the global illumination problem.

Acknowledgement

The authors would like to thank Fred Hickernell for his comments.

References

BC87. J. Beck and W. Chen, *Irregularities of Distribution*, Cambridge University Press, 1987. 4.1

Coo86. R. Cook, *Stochastic Sampling in Computer Graphics*, ACM Transactions on Graphics **5** (1986), no. 1, 51–72. 5, 6.3

CP76. R. Cranley and T. Patterson, *Randomization of Number Theoretic Methods for Multiple Integration*, SIAM Journal on Numerical Analysis **13** (1976), 904–914. 4.1

Fau92. H. Faure, *Good Permutations for Extreme Discrepancy*, J. Number Theory **42** (1992), 47–56. 6.1

FK00. I. Friedel and A. Keller, *Fast Generation of Randomized Low Discrepancy Point Sets*, 2000, this volume. 4.2

Gla95. A. Glassner, *Principles of Digital Image Synthesis*, Morgan Kaufmann, 1995. 1

Hla71. E. Hlawka, *Discrepancy and Riemann Integration*, Studies in Pure Mathematics (L. Mirsky, ed.), Academic Press, New York, 1971, pp. 121–129. 3

Kel98a. A. Keller, *Quasi-Monte Carlo Methods for Photorealistic Image Synthesis*, Ph.D. thesis, Shaker, Aachen, 1998. 3, 4.3

Kel98b. ———, *The Quasi-Random Walk*, Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing 1996 (H. Niederreiter, P. Hellekalek, G. Larcher, and P. Zinterhof, eds.), Lecture Notes in Statistics, vol. 127, Springer, 1998, pp. 277–291. 1, 3.1

LW93. E. Lafortune and Y. Willems, *Bidirectional Path Tracing*, Proc. 3rd International Conference on Computational Graphics and Visualization Techniques (Compugraphics), 1993, pp. 145–153. 1

Nie92. H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Pennsylvania, 1992. 3, 6.1

NX96. H. Niederreiter and C. Xing, *Quasirandom Points and Global Function Fields*, Finite Fields and Applications (S. Cohen and H. Niederreiter, eds.), Cambridge University Press, 1996, 269–296. 6.1

Owe95. A. Owen, *Randomly Permuted* (t, m, s)*-Nets and* (t, s)*-Sequences*, Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing (H. Niederreiter and P. Shiue, eds.), Lecture Notes in Statistics, vol. 106, Springer, 1995, pp. 299–317. 4.2

Owe98a. ———, *Latin Supercube Sampling for Very High Dimensional Simulations*, ACM Transactions on Modeling and Computer Simulation **8** (1998), 71–102. 5

Owe98b. ———, *Monte Carlo Extension of Quasi-Monte Carlo*, Winter Simulation Conference, IEEE Press, 1998, pp. 571–577. 4

OZ99. A. Owen and Y. Zhou, *Safe and Effective Importance Sampling*, Tech. report, Stanford University, Goldman-Sachs, 1999. 2.2

Pir00. G. Pirsic, *A Software Implementation of Niederreiter-Xing Sequences*, 2000, this volume. 6.1

Shi90. P. Shirley, *Physically Based Lighting Calculations for Computer Graphics*, Ph.D. thesis, University of Illinois, Urbana-Champaign, 1990. 5, 6.3

SJ94. I. Sloan and S. Joe, *Lattice Methods for Multiple Integration*, Clarendon Press, Oxford, 1994. 4.1, 6.4

Sob94. I. Sobol, *A Primer for the Monte Carlo Method*, CRC Press, 1994. 2.2

Tuf96. B. Tuffin, *On the Use of Low Discrepancy Sequences in Monte Carlo Methods*, Monte Carlo Methods and Applications **2** (1996), no. 4, 295–320. 4.1

Vea97. E. Veach, *Robust Monte Carlo Methods for Light Transport Simulation*, Ph.D. thesis, Stanford University, December 1997. 2

VG94. E. Veach and L. Guibas, *Bidirectional Estimators for Light Transport*, Proc. 5th Eurographics Workshop on Rendering (Darmstadt, Germany), June 1994, pp. 147–161. 1, 2.2

Combining Global and Local Global-Illumination Algorithms

György Antal, Budapest University of Technology*
Roel Martinez, University of Girona †
Ferenc Csonka, Budapest University of Technology‡
Mateu Sbert, University of Girona §
László Szirmay-Kalos, Budapest University of Technology¶

Abstract

Global illumination algorithms can be classified as local and global transfer methods. Local methods find a single point (or patch) in a given step and transfer its radiance towards other point(s). Global methods, on the other hand, select the source and the target of the transfer simultaneously. Local methods are better if the radiance distribution is heterogeneous and the scene is sparse, while global methods can win for dense scenes of homogeneous radiance. This paper proposes the combination of global and local global illumination algorithms in the sense of multiple importance sampling. In this way, the combined method can eliminate the higher noise at the corners produced by local methods and the need for first-shot for global techniques.

Keywords: Global illumination, stochastic iteration, finite-element techniques, Monte-Carlo methods

1 Introduction

Global illumination algorithms simulate the light transport. If the radiance estimate is represented by function $L(\vec{y},\omega')$, then the light transport produces a single reflection of the radiance function, which is obtained by applying the light transport operator $\mathcal{T}_{f_r}$:

$$L^r(\vec{x},\omega) = \mathcal{T}_{f_r} L(\vec{y},\omega') =$$

$$\int_\Omega L(\vec{y},\omega') \cdot f_r(\omega',\vec{x},\omega) \cdot \cos\theta'_{\vec{x}}\, d\omega',$$

where $\vec{y}$ is seen from $\vec{x}$ at direction ω' and $\theta'_{\vec{x}}$ is the angle between this direction and the surface normal.

Since the light traverses the space along straight lines, the simulation requires the generation of lines to identify the points between the light is transported. There are many different possibilities of this line generation. Lines can be obtained deterministically or randomly, which is used in Monte-Carlo algorithms. Monte-Carlo methods randomize the light transport operator, that is, they use a random operator $\mathcal{T}^*$ that gives back the effect of $\mathcal{T}$ in the average case:

$$E[\mathcal{T}^*_{f_r} L] = \mathcal{T}_{f_r} L.$$

In order to find the expected value, Monte-Carlo algorithms have to obtain many samples and approximate the expected value as the average of these samples. The method may produce individual lines or a bundle of lines of certain similarity. Working with bundle of lines can exploit the coherence of the scene and can thus significantly increase the computation speed. The formation of the bundles depends on what kind of similarity can be taken advantage of the algorithm. For example, hemicube based radiosity algorithms consider lines of the same origin and passing through a regular grid, since the first intersection of these lines can be computed by the z-buffer hardware. Parallel ray bundles can transfer the radiance of all points of the scene parallel to a random direction. The visibility needed by this parallel transfer can also be computed efficiently by incremental algorithms. Even if conventional ray-shooting is used, it is worth computing those lines simultaneously that visit the same nodes of the space partitioning data structure [4]. Realizing that current processors can execute four floating point instructions concurrently, it also seems advantageous to follow always four nearby lines [12]. Finally, line generation can also be classified according to the strategy of finding the starting point and its direction vector. *Local line methods* find the starting point of the half-line first, then they obtain the direction of the line, which will identify the intersection point or the other point of the transfer. An alternative is the *global line approach* which samples the two points simultaneously.

There have been many discussions about the comparative advantages of these algorithms, but no method can be claimed to be the best. This is not surprising since each method has advantages and disadvantages in certain situations. Thus instead of insisting to a given technique, it is worth combining several of them, in a way that the ad-

*gyantal@mailbox.hu
†roel@ima.udg.es
‡fcsonka@graphisoft.hu
§mateu@ima.udg.es
¶szirmay@iit.bme.hu

vantages are preserved. Such quasi-optimal combination of Monte-Carlo sampling techniques is offered by *multiple importance sampling*. In this paper, the combination of global and local line methods is considered.

2 Multiple importance sampling

In this section we recall the fundamental theory of multiple importance sampling [11, 10]. Assume that integral $L = \int_{\mathcal{P}} l(z)dz$ needs to be evaluated. Monte-Carlo quadratures generate samples with certain probability density. Suppose that we have N different sampling techniques. Sampling method i uses probability density $p_i(z)$, thus the primary estimator of this method is $l(z)/p_i(z)$. Assume also that with method i we obtain N_i samples $z_{i1}, \ldots, z_{iN_i}$. The combined estimator is computed from the samples of all sampling techniques, applying appropriate weighting functions $w_i(z)$, and summing the results:

$$\langle L \rangle_c = \sum_{i=1}^{N} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(z_{ij}) \cdot \frac{l(z_{ij})}{p_i(z_{ij})} = \sum_{i=1}^{N} \sum_{j=1}^{N_i} \frac{l(z_{ij})}{d(z_{ij})} \tag{1}$$

where the divider is

$$d_i(z) = \frac{p_i(z) N_i}{w_i(z)}.$$

The combined estimator is unbiased, i.e. the expected value of this estimator gives back the original integral, if for all z values $\sum_{i=1}^{N} w_i(z) = 1$. In order to find an optimal weighting, the variance of the combined estimator $\langle L \rangle_c$ should be minimized by setting the weights appropriately and also taking into account the constraint of unbiasedness. Unfortunately, this optimization problem cannot be solved analytically, but different quasi-optimal solutions can be obtained. One such approximate solution is called the *balance heuristic* [11]:

$$w_i(z) = \frac{N_i p_i(z)}{\sum_{k=1}^{N} N_k p_k(z)}. \tag{2}$$

Substituting these weights into equation 1, we can conclude that balance heuristic divides with the total density

$$d(z) = \sum_{k=1}^{N} N_k p_k(z)$$

instead of the original densities $N_i p_i(z)$ of the individual methods.

This formula can be efficiently used in random walk algorithms that obtain samples independently. However, the application of multiple importance sampling in iteration like algorithms requires further considerations. We could, for example, use all sampling techniques to obtain a tentative value in the next iteration step then find the real value as the weighted average of the results of the individual methods, but this method would slow down the progress of the iteration and thus the introduction of higher order terms. Thus we propose to randomly select just a single technique in each iteration step, compute just a single sample, and apply the other techniques to the already iterated value.

To consider the random selection formally, let us assume that the sample is computed with method i with probability P_i.

The modified estimator uses the indicator functions ξ_i, which are 1 if the respective method generates a sample:

$$\langle L \rangle_c = \sum_{i=1}^{N} w_i(z_i) \cdot \frac{l(z_i)}{p_i(z_i)} \cdot \xi_i. \tag{3}$$

The requirement of the unbiasedness becomes:

$$\sum_{i=1}^{N} P_i \cdot w_i(z) = 1.$$

The modified formulae of balanced heuristics is the following:

$$w_i(z) = \frac{p_i(z)}{\sum_{k=1}^{N} P_k \cdot p_k(z)}.$$

Thus when a sample is computed, its contribution is always divided by

$$d(z) = \sum_{k=1}^{N} P_k \cdot p_k(z)$$

no matter which sample strategy is used.

We are going to apply this approach for two sets of algorithms. The first set contains local and global ray-shooting to solve the diffuse radiosity problem, while the second set includes parallel and perspective ray-bundle based transfers and working in the general non-diffuse setting.

3 Combination of local and global ray transfers

One of the simplest tool to transfer the radiance in the scene is the generation of random lines and the identification of those points that are intersected by these lines.

3.1 Transfer with local lines

Local line methods sample first the source of the oriented lines, called rays, then they decide on the direction of these rays. Suppose that patch j is selected with probability density p_j as a source patch and the starting point of the ray with uniform distribution. Thus the probability density of sampling point $\vec{y}$ as the starting point of the ray is:

$$p(\vec{y}) = \frac{p_j}{A_j}.$$

Figure 1: An office that is tessellated to 14 thousand patches and lit by a blue light source, and is rendered using 5 million global and 5 million local lines and 180 seconds computation time

According to the concepts of importance sampling, it is worth setting p_j to be proportional to the power of the patch:

$$p_j = \frac{\Phi_j}{\sum_{k=1}^{n} \Phi_k}.$$

If the surfaces are diffuse, then $\Phi_j = L_j A_j \pi$, where L_j is the radiance of the patch. It also means that the density of the selection is proportional to the radiance:

$$p(\vec{y}) = \frac{\Phi_j}{A_j \sum_{k=1}^{n} \Phi_k} = \frac{L_j}{\sum_{k=1}^{n} A_k L_k}.$$

Having found the starting point, the direction of the ray is sampled, which can take into account the local BRDF. If the surface is diffuse, cosine distribution can be applied, i.e. the density of the direction is $\cos\theta_{\vec{y}}/\pi$. The starting point and the direction establish the ray, which is traced and its hit point $\vec{x}$ is identified.

Since the solid angle in which a differential area $d\vec{x}$ around $\vec{x}$ is seen from $\vec{y}$ is $d\vec{x} \cdot \cos\theta_{\vec{x}}/|\vec{x}-\vec{y}|^2$, the probability that this strategy transfers the light from differential area $d\vec{y}$ to $d\vec{x}$ is the following:

$$\Pr(d\vec{y} \to d\vec{x}) = \frac{p_j}{A_j} \cdot \frac{d\vec{y} \cdot \cos\theta_{\vec{y}} \cdot d\vec{x} \cdot \cos\theta_{\vec{x}}}{|\vec{x}-\vec{y}|^2}.$$

Thus the density is:

$$p_l(\vec{y} \to \vec{x}) = \frac{p_j}{A_j} \cdot \frac{\cos\theta_{\vec{y}} \cdot \cos\theta_{\vec{x}}}{\pi|\vec{x}-\vec{y}|^2} = \frac{L_j}{\sum_{k=1}^{n} A_k L_k} \cdot \frac{\cos\theta_{\vec{y}} \cdot \cos\theta_{\vec{x}}}{\pi|\vec{x}-\vec{y}|^2}. \quad (4)$$

3.2 Transfer with global lines

Global line algorithms use uniformly distributed lines and transfer the light between those points that are intersected by the lines. Note that a line may intersect many patches, when the radiance is transfered between all subsequent pairs of patches.

In order to compute the probability density of such transfers, the theory of integral geometry [7] can be used here. The measure of the set of those uniformly distributed lines, which intersect differential areas $d\vec{x}$ and $d\vec{y}$ is:

$$\mu(d\vec{y}, d\vec{x}) = \frac{d\vec{y} \cdot \cos\theta_{\vec{y}} \cdot d\vec{x} \cdot \cos\theta_{\vec{x}}}{|\vec{x}-\vec{y}|^2}.$$

Note that this is only an unnormalized measure and is not a probability. To obtain a probability, we should compute the ratio of this measure and the measure of the lines crossing the sphere enclosing the whole scene. From integral geometry we know that the measure of the set of lines intersecting a convex body is $\pi S/2$, where S is the surface area of the body. Denoting the area of the enclosing sphere by S, the probability of selecting differential areas $d\vec{y}$ and $d\vec{x}$ as candidates for the transfer is:

$$\Pr(d\vec{y} \to d\vec{x}) = \frac{2\mu(d\vec{y}, d\vec{x})}{\pi S} = \frac{2}{S} \cdot \frac{d\vec{y} \cdot \cos\theta_{\vec{y}} \cdot d\vec{x} \cdot \cos\theta_{\vec{x}}}{|\vec{x}-\vec{y}|^2}.$$

The density of global line transfer is then:

$$p_g(\vec{y} \to \vec{x}) = \frac{2}{S} \cdot \frac{\cos\theta_{\vec{y}} \cdot \cos\theta_{\vec{x}}}{\pi|\vec{x}-\vec{y}|^2}. \quad (5)$$

3.3 Combination of local and global line methods

Looking at equations 4 and 5 we can realize that the local line methods will probably generate oriented lines starting at the high radiance points. Global line methods, on the other hand, provide more samples at average radiance points. If the radiance distribution is homogeneous, then the global line method will carry out roughly $2\sum A_k/S$

more transfers. Since the ratio of the total area of the object surfaces and the surface of the enclosing sphere is about 5 in everyday scenes, global lines can result in the multiplication of the effective samples by about 10.

In order to preserve this benefit, but to get also the good properties of the local lines for heterogeneous radiance scenes, the two methods are combined. Suppose that in each step we decide randomly whether a local or a global line is generated. The probability of the local line method is P_l, while the global line method is applied with probability $1 - P_l$.

Having computed the transfer, the transferred radiance is multiplied by the weights of multiple importance sampling. The weight of local line method is:

$$w_l = \frac{p_j S}{2A_j(1 - P_l) + p_j S P_l}$$

The weight of the global line method is:

$$w_g = \frac{2A_j}{2A_j(1 - P_l) + p_j S P_l}$$

The samples of both techniques are thus divided by the following density:

$$d(\vec{y}) = \frac{2}{S}(1 - P_l) + \frac{p_j}{A_j} P_l.$$

In figure 1 we show the result of the proposed algorithm when rendering a diffuse scene. Note that we should not use first shot, and thus algorithm is unbiased.

4 Combination of methods using ray-bundles

In this section we combine methods that transfer locally and globally sampled bundles of rays. First, we quickly review the individual methods that discuss their combination.

4.1 Method 1: Parallel ray-bundle tracing

Parallel ray-bundle tracing transfers the radiance of all patches parallel to a randomly selected global line in each iteration cycle [8]. The random transport operator is:

$$\mathcal{T}_1^* L = 4\pi \cdot L(\vec{y}, \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta'_{\vec{x}}.$$

where $\vec{y} = h(\vec{x}, -\omega')$ is the point visible from $\vec{x}$ at direction $-\omega'$.

Indeed, if the orientation is sampled uniformly, then its probability density is $p(\omega') = 1/4\pi$, thus the expectation of the random transport operator gives back the effect of the light transport operator $\mathcal{T}_{f_r} L$:

$$E[\mathcal{T}_1^* L] = \int_{\Omega'} 4\pi \cdot L(h(\vec{x}, -\omega'), \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta' \cdot \frac{d\omega'_{\vec{x}}}{4\pi}.$$

It is straightforward to extend the method to be bi-directional, which transfers the radiance not only into direction ω', but also to $-\omega'$. Note that this does not even require additional visibility computation. For bi-directional transfers, the density of sampling is $p(\omega') = 1/2\pi$.

The radiance transfer needs the identification of those points that are mutually visible in the global direction. In order to solve this global visibility problem, a window is placed perpendicular to the global direction. The window is decomposed into a number of pixels. A pixel is capable to store a list of patch indices and z-values. The lists are sorted according to the z-values. The collection of these pixels is called the *transillumination buffer*[6]. The patches are rendered one after the other into the buffer using a modified z-buffer algorithm which keeps all visible points not just the nearest one. Traversing the generated lists the pairs of mutually visible points can be obtained. For each pair of points, the radiance transfer is computed and the transferred radiance is multiplied by the BRDF, resulting in the reflected radiance L^r.

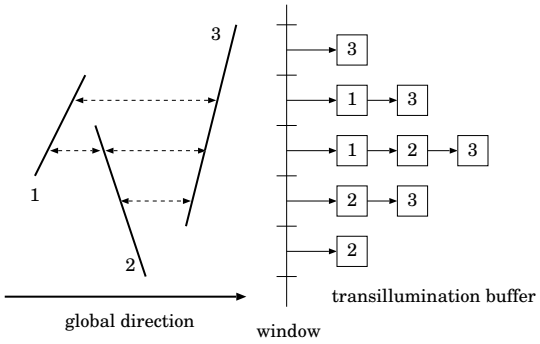


Figure 2: Organization of the transillumination buffer

From the reflected radiance the patch radiance can be obtained by a simple averaging operation. Note that if the integral is evaluated on the window, then the cosine factor is compensated:

$$L(m)|_i = \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}_{f_r}^* L(m-1)\, d\vec{x} \approx$$

$$\frac{4\pi \cdot \delta P}{A_i} \cdot \sum_P L^{in}(P) \cdot f_r(\omega', P, \omega),$$

where P runs on the pixels covering the projection of patch i, $L^{in}(P)$ is the radiance of the surface point visible in pixel P, $f_r(\omega', P, \omega)$ is the BRDF of that point which receives this radiance coming through pixel P and δP is the area of the pixels.

4.2 Parallel ray-bundle tracing with a single transillumination plane

The drawback of the previous algorithm is that it cannot exploit the hardware z-buffer since it can store only a sin-

gle value per pixel, but the algorithm requires all patches that are projected onto this pixel. Fortunately, this requirement can be eliminated, thus the algorithm can be executed on the hardware, if the visibility algorithm is further randomized in the following way [5].

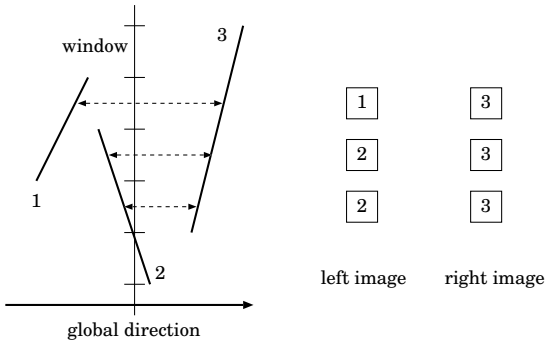


Figure 3: Transferring the radiance through a single plane

Let us find randomly a point on the line of the transillumination direction and place the transillumination window at this point. The scene is rendered from the two sides of the window supposing that the color of patch i is i. Having read the two images from the frame buffer, the patches that see each other from the opposite sides of the window can be identified, and the radiance can be transferred between them. Of course, this method finds two points that see each other in the transillumination direction only with some probability. This probability is proportional to the distance between the two points. If the distance of the front and back clipping planes is R, then the probability is $|\vec{x} - \vec{y}|/R$. When the scene is rendered, the z coordinates are transformed in a way that they fit in the [0,1] range for the whole scene, that is, the distances are normalized with R. It means that this probability equals to the sum of the z values of the two visible points, as read out from the z-buffer. In order to compensate those cases when the two points are not on the opposite sides of the transillumination window, when the radiance is transferred, it is divided by the selection probability, i.e. by the normalized distance of the two points. Formally, the random transport operator is:

$$\mathcal{T}_2^* L = 2\pi \cdot L(\vec{y}, \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta'_{\vec{x}} \cdot \xi(\vec{x}, \vec{y}) \cdot \frac{R}{|\vec{x} - \vec{y}|},$$

where $\xi(\vec{x}, \vec{y})$ is the indicator function, which is 1 if and only if the transillumination plane is between $\vec{x}$ and $\vec{y}$.

4.3 Perspective ray-bundle shooting

Perspective ray-bundle shooting selects a single patch randomly and sends its radiance from one of its randomly selected point towards all directions [1]. According to importance sampling, it is worth setting the selection probability p_i proportional to the powers of the patches.

If patch j is selected with probability p_j and point $\vec{y}$ on this patch with uniform $1/A_j$ probability, then the random transport operator is

$$(\mathcal{T}_3^* L)(\vec{x}, \omega) =$$

$$\frac{A_j}{p_j} \cdot v(\vec{x}, \vec{y}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{x}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x} - \vec{y}|^2},$$

where $v(\vec{x}, \vec{y})$ is the mutual visibility indicator, which is 1 if the two points are visible from each other.

The expected value of this random variable is:

$$E[\mathcal{T}_3^* L] = \sum_j p_j \cdot \int_{A_j} (\mathcal{T}_{pers}^* L)(\vec{x}, \omega) \, \frac{d\vec{y}}{A_j} =$$

$$\sum_j \int_{A_j} v(\vec{x}, \vec{y}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{x}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x} - \vec{y}|^2} \, d\vec{y}.$$

Using the formula of solid angles

$$d\vec{y} \cdot \cos\theta_{\vec{y}} / |\vec{x} - \vec{y}|^2 = d\omega_{\vec{x}}$$

and assuming that illumination can only come from surfaces — i.e. there is no external sky light illumination — the integration over all surfaces can be replaced by an integration over all incoming solid angles:

$$E[\mathcal{T}_{pers}^* L] = \int_{\Omega'} L(h(\vec{x}, -\omega'), \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta'_{\vec{x}} \, d\omega'_{\vec{x}}.$$

To obtain the patch radiance, the radiances of the points are averaged:

$$L(m)|_i = \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}_{f_r}^* L(m-1) \, d\vec{x} =$$

$$\frac{A_j}{p_j A_i} \int_{A_i} v(\vec{x}, \vec{y}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{x}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x} - \vec{y}|^2} \, d\vec{x}. \tag{6}$$

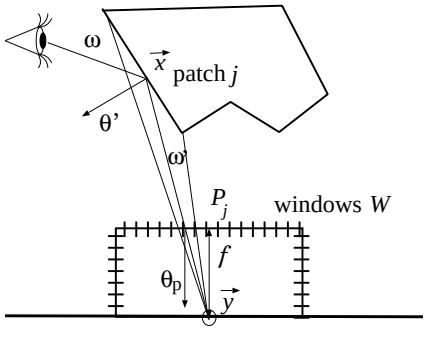


Figure 4: Perspective ray-bundle tracing with hemicubes

The integral in equation (6) can also be evaluated on the five window surfaces (W) that form a hemicube around the

source $\vec{y}$ (figure 4). Note that this is similar to the famous hemicube approach of the diffuse radiosity problem [3]. In fact, radiance shooting requires the vertex-patch form factors that can be computed by the hemicube. In this section, we re-derive the basic formulae to show that they can also be used in cases when the reflection is non-diffuse.

To find formal expressions, let us express the solid angle $d\Omega_p$, in which a differential surface area $d\vec{x}$ is seen through pixel area $d\vec{p}$, both from the surface area and from the pixel area:

$$d\Omega_p = \frac{d\vec{x} \cdot \cos\theta'_{\vec{x}}}{|\vec{y} - \vec{x}|^2} = \frac{d\vec{p} \cdot \cos\theta_p}{|\vec{y} - \vec{p}|^2}, \qquad (7)$$

where θ_p is the angle between direction pointing to $\vec{x}$ from $\vec{y}$ and the normal of the window (figure 4). The distance $|\vec{y} - \vec{p}|$ between pixel point $\vec{p}$ and the lightsource $\vec{y}$ equals to $f/\cos\theta_p$ where f is the distance from $\vec{y}$ to the window plane, that is also called the *focal distance*. Using this and equation (7), differential area $d\vec{x}$ can be expressed and substituted into equation (6), thus we can obtain:

$$L(m)|_i = \frac{A_j}{p_j A_i f^2} \cdot$$

$$\int_W v(\vec{y}, \vec{x}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{p}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \cos\theta_{\vec{y}} \cdot \cos^3\theta_p \, d\vec{p}.$$

Let P_i be the set of those pixels in which patch i is visible from the lightsource. P_i is computed by running a z-buffer/constant shading rendering step for each sides of the window surface, assuming that the color of patch i is i, then reading back the "images". The reflected radiance on patch i is approximated by a discrete sum as follows:

$$L(m)|_i \approx \frac{A_j \delta P}{p_j A_i f^2} \cdot$$

$$\sum_P L(\vec{y}, \omega'_{\vec{y}\to\vec{p}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}(\vec{p}), \omega) \cdot \cos\theta_{\vec{y}} \cdot \cos^3\theta_p,$$

where δP is the area of a single pixel in the image. If R is the resolution of the image — i.e. the top of the hemicube contains $R \times R$ pixels, while the side faces contain $R \times R/2$ pixels – then $\delta P = 4f^2/R^2$.

4.4 Representation of the temporary radiance

The discussed ray-bundle methods sample the radiance function in each step and obtain a new function. The radiance is a four variate function and usually has high variation, thus its accurate finite-element representation would require many basis functions. Instead, in an iteration step we compute only the irradiance on each patch, which is independent of the transfer direction of the next step. With the irradiance information we also store the incoming direction. In the next iteration step, when the output radiance of a patch in a given direction is needed, it is obtained on the fly, multiplying the irradiance by the BRDF of the patch taking into account the previous and current directions.

In order to establish importance sampling for perspective ray-bundle shooting, the powers of the patches should also be known. The computation of the powers from the irradiance values is also straightforward, the irradiances should be multiplied by the albedos $a_i(\omega)$ of the patches.

4.5 The combination of the ray-bundle based strategies

So far, we introduced three different random radiance transfer methods that use different sampling probabilities. Parallel ray-bundle tracing samples the direction from point $\vec{x}$ with a uniform density, i.e. the probability of generating a direction in $d\omega$ is

$$\frac{d\omega}{2\pi}.$$

Note that we use 2π due to the bi-directionality of the algorithm.

When just a single plane is used, contribution to point $\vec{x}$ is possible only if the plane is between point $\vec{x}$ and that point $\vec{y}$ which is visible from here. If the maximum size of the scene is R, then the probability that a contributing direction is in $d\omega$ is

$$\frac{|\vec{x} - \vec{y}| \cdot d\omega}{2\pi R}.$$

For perspective ray-bundle shooting, the probability that shooting point is in differential area $d\vec{y}$ of patch j is

$$\frac{p_j \cdot d\vec{y}}{A_j} = \frac{\Phi_j \cdot d\vec{y}}{A_j \sum_i \Phi_i}.$$

Before applying the concept of multiple importance sampling, we have to solve the problem that different methods formulate the light transport problem with different integrals. Parallel ray-bundles use directional integrals while perspective ray-bundle shooting applies surface integrals. Converting directional integrals to surface integrals, the probability densities used by the discussed ray-bundle based methods are the following:

$$p_1(\vec{y}) = \frac{\cos\theta_{\vec{y}}}{2\pi \cdot |\vec{x} - \vec{y}|^2},$$

$$p_2(\vec{y}) = \frac{\cos\theta_{\vec{y}}}{2\pi R \cdot |\vec{x} - \vec{y}|},$$

$$p_3(\vec{y}) = \frac{p_j}{A_j}.$$

Each of them is good for particular illumination conditions. Parallel ray-bundles are effective if the scene consists of patches of similar radiance, while perspective ray bundles are effective if one or several patches are much brighter than the others (note that these bright points are

Parallel Perspective Combined

Figure 5: Comparison of stochastic iteration using parallel (left), perspective (middle) ray-bundles and the combination of the two methods (right) using the same computation time (7 seconds on a P4/1.2GHz computer)

selected with much higher probability by perspective ray-bundle shooting). Thus perspective ray-bundle shooting is the best method if the scene contains small light sources. It is thus highly intuitive why parallel ray-bundle algorithms always apply a first shot to distribute the illumination of the light sources, letting the algorithm compute only the indirect illumination.

On the other hand, the transfer of nearby points is better coped by parallel transfers than by perspective transfers. Close points are obtained by parallel ray-bundle tracing with the highest probability, this probability is smaller if just a single plane is used and the smallest for perspective ray-bundle shooting. Thus dense scenes and corners can be rendered in a better way by parallel ray-bundle transfers.

In order to obtain a method that does not require first shot and can nicely render corners and close objects, the presented techniques are combined according to multiple importance sampling.

Suppose that each of the three methods is used with probability P_1, P_2 and P_3, respectively. Since one method is applied in each step $P_1 + P_2 + P_3 = 1$. These probabilities can be specified by the user, taking into account the features of the scene and the time of the application of the methods.

When combining the three ray-bundle algorithms, the divider of balanced heuristic becomes:

$$d(\vec{y}) = P_1 \frac{\cos\theta_{\vec{y}}}{2\pi|\vec{x}-\vec{y}|^2} + P_2 \frac{\cos\theta_{\vec{y}}}{2\pi R|\vec{x}-\vec{y}|} + P_3 \frac{p_j}{A_j}.$$

When parallel ray-bundles are used, this weight should be multiplied by $d\vec{y}/d\omega = |\vec{x}-\vec{y}|^2/\cos\theta_{\vec{y}}$ in order to replace differential surfaces areas $d\vec{y}$ by differential solid angles $d\omega$:

$$d(\omega) = P_1 \frac{1}{2\pi} + P_2 \frac{|\vec{x}-\vec{y}|}{2\pi R} + P_3 \frac{p_j}{A_j} \frac{|\vec{x}-\vec{y}|^2}{\cos\theta_{\vec{y}}}.$$

In order to test the proposed method we have selected the standard Cornell Box scene (figure 5). The images have been rendered with 500×500 resolution. The transillumination buffer contained 1000×1000 pixels. Figure 5 shows the Cornell box rendered by parallel, perspective ray bundles and by the combined method. The computation time was 7 seconds in all cases. Note that parallel bundles distribute the energy with higher noise generally, but are good in rendering corners. The result of perspective bundles is better except for the annoying spikes at the corners. The combined method can combine the advantages of both techniques and results in much more pleasing image than its parents.

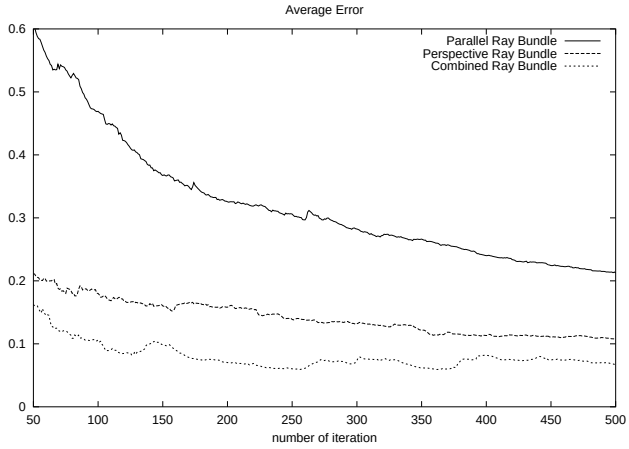


Figure 6: Error of the parallel, perspective and combined ray-bundle shooting algorithms for the Cornell box.

5 Conclusions

In this paper we proposed the combination of local and global radiance transfer methods according to the concept of multiple importance sampling. First local and global lines based techniques were combined, where we could preserve the efficiency of global transfers, but could get rid of the necessity of the first shot. Secondly we proposed the combination of three ray-bundle transfer methods. The combined method is able to render complex glossy scenes in a few tens of seconds and is particularly effective if the surfaces are not highly specular. This estimation requires just one or a few radiance values per patch, thus the storage requirements is modest.

6 Acknowledgements

This work has been supported by the National Scientific Research Fund (OTKA ref. No.: T029135 and T042735), the Bolyai Scholarship, Sloven-Hungarian Action Fund and Intel Corp. The scenes have been modelled by Maya that was generously donated by AliasWavefront.

References

[1] Gy. Antal, L. Szirmay-Kalos, and F. Csonka. Hemicube shooting for non-diffuse global illumination scenes. In *Proc. Spring Conference on Computer Graphics (SCCG '2002)*, pages 89–96. Comenius University Press, 2002.

[2] Ph. Bekaert. *Hierarchical and stochastic algorithms for radiosity*. PhD thesis, University of Leuven, 1999. http://www.cs.leuven.ac.be/ cwis/research/graphics/ CGRG.PUBLICATIONS/PHBPPHD.

[3] M. Cohen and D. Greenberg. The hemi-cube, a radiosity solution for complex environments. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, pages 31–40, 1985.

[4] V. Havran. *Heuristic Ray Shooting Algorithms*. Czech Technical University, Ph.D. dissertation, 2001.

[5] R. Martinez, L. Szirmay-Kalos, and M. Sbert. A hardware based implementation of the multipath method. In *Computer Graphics International*, Bradford, UK, 2002.

[6] L. Neumann. Monte Carlo radiosity. *Computing*, 55:23–42, 1995.

[7] M. Sbert. *The Use of Global Directions to Compute Radiosity*. PhD thesis, Catalan Technical University, Barcelona, 1996.

[8] L. Szirmay-Kalos. Stochastic iteration for non-diffuse global illumination. *Computer Graphics Forum (Eurographics'99)*, 18(3):233–244, 1999.

[9] L. Szirmay-Kalos, F. Csonka, and Gy. Antal. Global illumination as a combination of continuous random walk and finite-element based iteration. *Computer Graphics Forum (Eurographics'2001)*, 20(3):288–298, 2001.

[10] E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, http://graphics.stanford.edu/papers/veach_thesis, 1997.

[11] E. Veach and L. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Rendering Techniques '94*, pages 147–162, 1994.

[12] I. Wald, C. Benthin, P. Slussalek, and M. Wagner. Interactive rendering with coherent ray tracing. In *Eurographics '01*, 2001.