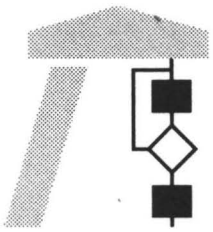
Interner Bericht

Monte Carlo & Beyond
– Course Material –

A. Keller

320/02



FACHBEREICH
INFORMATIK

UNIVERSITÄT
KAISERSLAUTERN

Postfach 3049 · D-67653 Kaiserslautern

Monte Carlo & Beyond
– Course Material –

A. Keller

320/02

Universität Kaiserslautern
AG Numerische Algorithmen
Postfach 30 49
67653 Kaiserslautern
Germany

Februar 2002

Herausgeber: AG Numerische Algorithmen
Leiter: Professor Dr. S. Heinrich

Monte Carlo and Beyond

Alexander Keller
keller@informatik.uni-kl.de

$\int$ Numerical Algorithms Group

Dept. of Computer Science
University of Kaiserslautern

'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
- **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
- **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
- 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

- **Day 1**
  - **computer graphics and visual effects**
  - principles of rendering algorithms
- **Day 2**
  - Monte Carlo integration
  - quasi-Monte Carlo points
- **Day 3**
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo
- **Day 4**
  - application to computer graphics

A Brief History of Interactive Computer Graphics

- Interactive computer graphics from 2d to 3d
- Applications of computer graphics
  - visualization
  - entertainment industry
  - image processing
  - modelling
  - image synthesis: Realism and Art
- Computer graphics is interdisciplinary

Early Interactive Computer Graphics

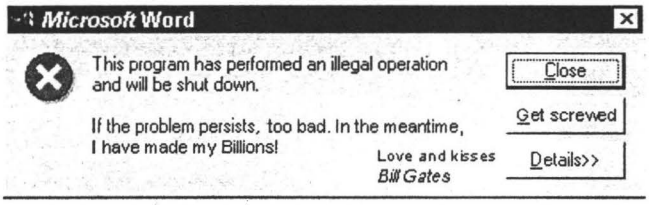
ca. 1960:
Sketchpad by Sutherland

ca. 1970:
Xerox Alto

- **1965:** First CAD systems (GM, Lockheed)
- **1976:** First computers with color raster display (Apple)

Graphical User Interfaces

- **1987:** Apple introduces the Mac II
  ⇒ workbench metaphor

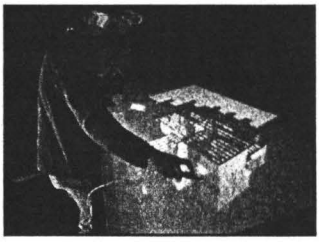


- **1993:** Introduction of the Mosaic web-browser under X-Windows

Interactive Computer Graphics Today

- **1988:** First public demonstration of virtual reality
- **1992:** First cave

Head mounted devices

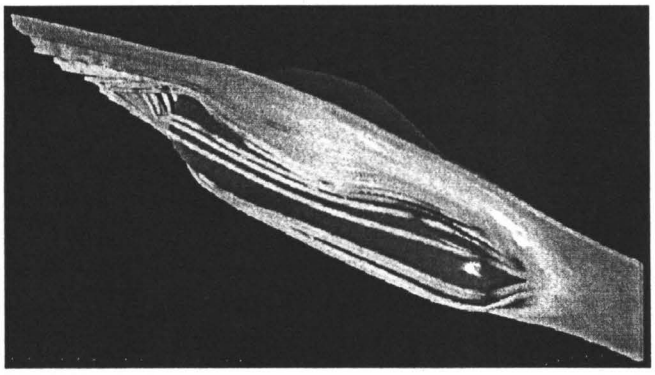


Responsive workbench

- What metaphor in space ?

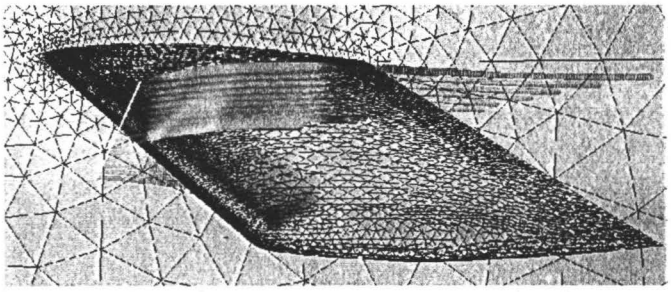
  ⇒ immersive user interfaces

Visualization



- Simulation of streamlines

Visualization



- Computational geometry

Entertainment Industry

- Sony Playstation II
- Movies: The Matrix, Titanic, Fifth Element, Starship Troopers, ...

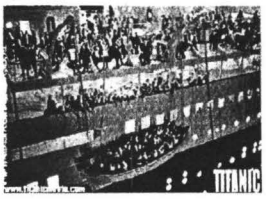
Image Processing

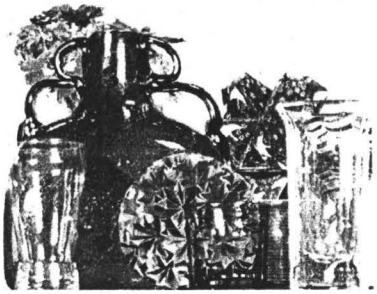
- Image reconstruction, enhancement, and compression

Image Processing

- Morphing

Image Processing

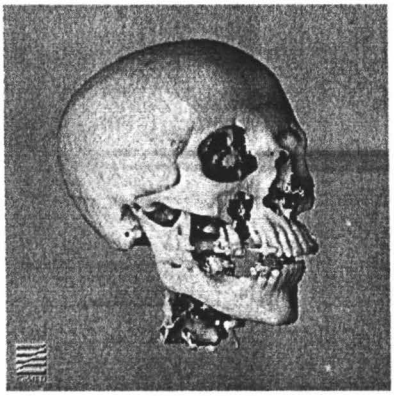




- Compositing

Image Based Rendering

Medical Image Processing

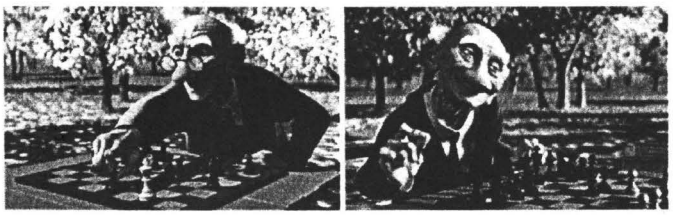


Free Form Surfaces

- **1975:** The 'Trademark' of computer graphics: *The Utah Teapot*

- Modelling and animation

Subdivision Surfaces

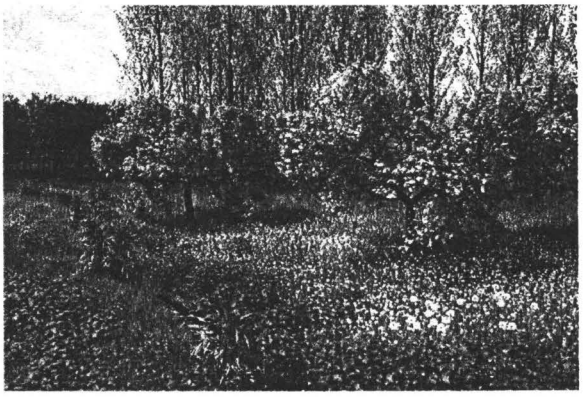


- Modelling and animation
- New, compact data structures, easier to handle

Rendering of Complex Models

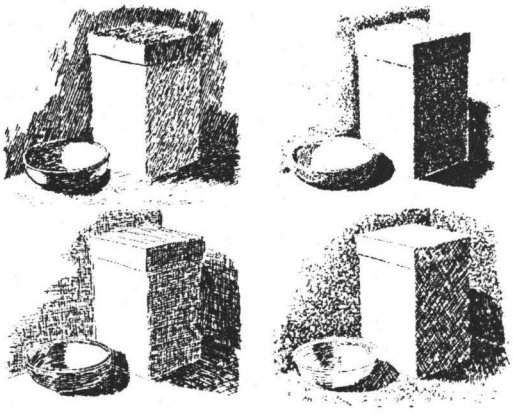
- Modelling of natural systems by artificial life
- Procedural models

Photorealistic Image Synthesis

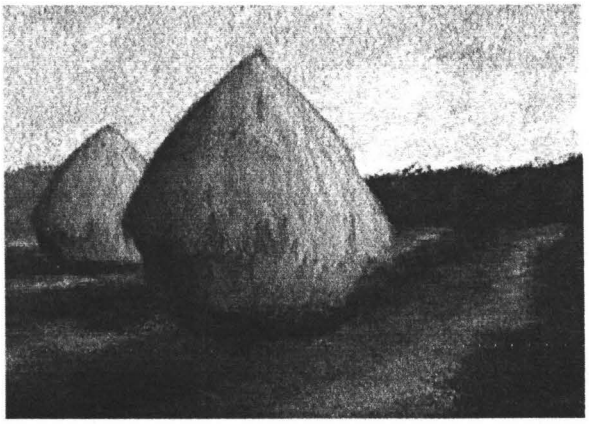


- New, faster numerical algorithms

Illustration



Art: Watercolor



Art: Oilpainting

Computer Graphics is Interdisciplinary

- Computer science
- Mathematics
- Physics
- Engineer sciences
- Biology
- Psychophysics
- Art
- Patent law

Oscars for Visual Effects

- **1977:** **'Star Wars'**.
- **1982:** **'Tron'**: First movie with huge amounts of computer graphics (Disney).
- **1984:** First IMAX-movie completely generated by computer graphics: **'The Magic Egg'**.
- **1989:** SFA-Oscar for the animated short **'Tin Toy'** (Pixar).
- **1993:** VFX-Oscar for **'Jurassic Park'** (Industrial Light and Magic).
- **1995:** SAA-Oscar for **'Toy Story'** (Pixar).
- **1997:** SFA-Oscar for **'Geri's Game'** (Pixar).
- **1998:** SFA-Oscar for **'Bunny'** (Blue Sky Studios).
- **1998:** VFX-Oscar for **'What Dreams May Come'**.
- **1999:** VFX-Oscar for **'The Matrix'** (Manex).
- **2000:** VFX-Oscar for **'Gladiator'** (Mill Film London)

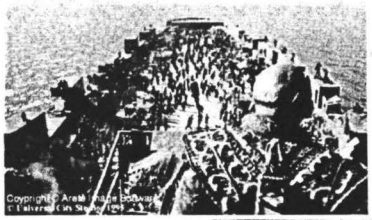
www.VFXPro.com, www.cinefex.com

Visual Effects by the Computer

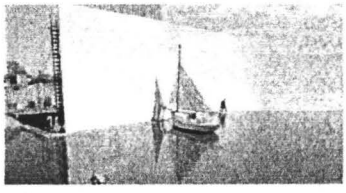
- Why ?
  - It's cheaper.
  - You can do things that are impossible.
  - You can do things that are really impossible.

'Visual effects will become important in world domination.'
John Gaeta, ESC (former MANEX) Group CEO

It's cheaper...



It's cheaper...



You can do Things that are Impossible...

You can do Things that are really Impossible...

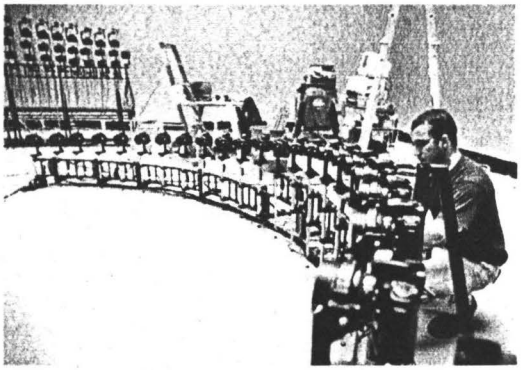
- Things that are physically impossible...

  ... like the extremely fast camera movements in 'The Matrix'

- Procedure
  - green shot
  - lightprobe
  - background shot
  - geometry reconstruction
  - rendering

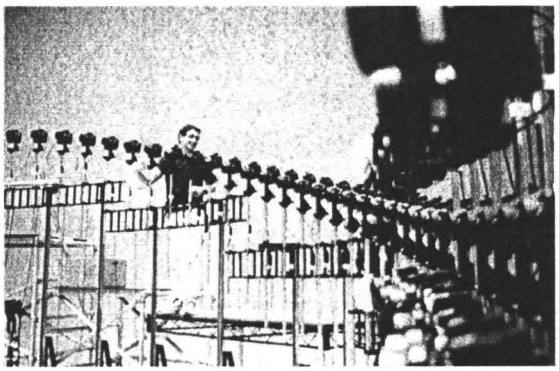
Green Shot

- Setup of the array of digital cameras



Green Shot

- Adjustment of height

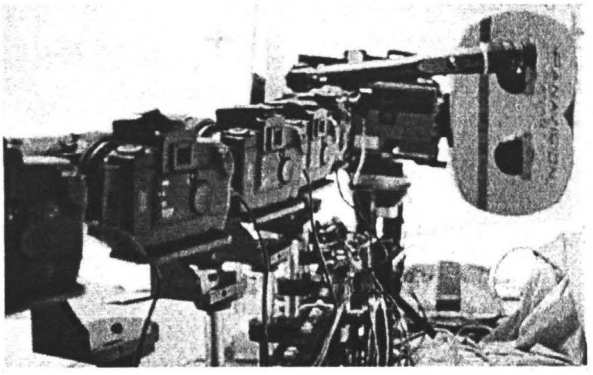


Green Shot

- Adjustment of direction

Green Shot

- Movie cameras at both ends of the array

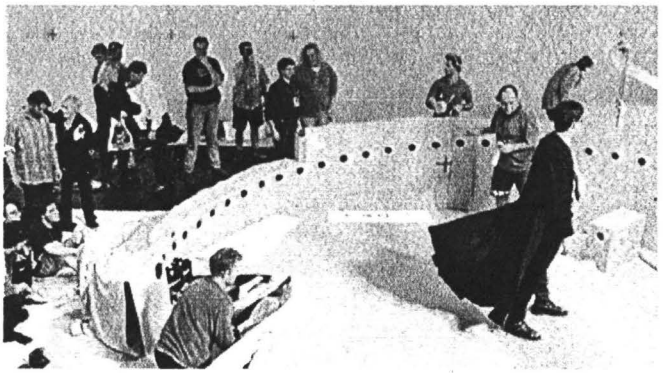


Green Shot

- Completion of the green box

Green Shot

- Keanu Reeves on the set...



Digital Green Shot

Digital Green Shot

Digital Green Shot

Digital Green Shot

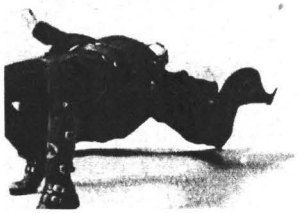
Digital Green Shot

Digital Green Shot

Digital Green Shot

Digital Green Shot

Digital Green Shot



Digital Green Shot: Raw Material

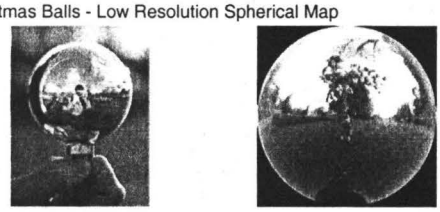
Digital Green Shot Finalized Material

- Flow-Mo (flow motion contrary to freeze frame motion)
  - 12000 frames per second
  - cell animation
- Compensation of systematic and measurement errors
- Inbetweening by morphing

Light probe: Real Light by High Dynamic Range Imaging

- Q: What do have the 'Terminator', 'The Matrix', and Christmas have in common ?

- A: Christmas Balls - Low Resolution Spherical Map

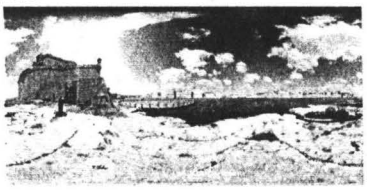


Rendering with Natural Light

Background Shot

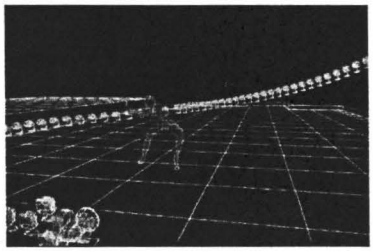
- High resolution panorama camera

- Digital virtual set



Rendering

- Simulation of light transport
  - light sources: Light probe
  - geometry:



    - reconstructed by correspondences in single frames
    - insertion of artificial objects
  - reflectivity: Green and panorama shot as texture on geometry
  - camera path through the positions given by the camera array

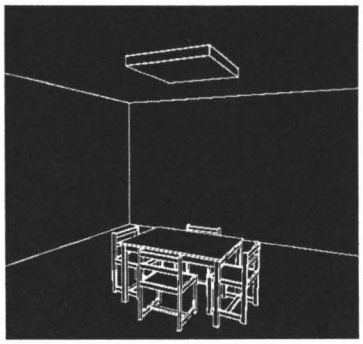
The Final Shot

Monte Carlo and Beyond

- **Day 1**
  - computer graphics and visual effects
  - **principles of rendering algorithms**
- **Day 2**
  - Monte Carlo integration
  - quasi-Monte Carlo points
- **Day 3**
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo
- **Day 4**
  - application to computer graphics

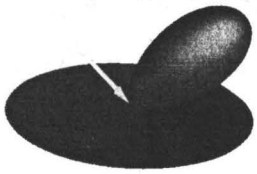
Scene Geometry

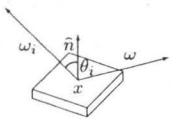
- Scene surface $\partial V := \cup_{i=1}^{K} S_i$
  - S_i surface primitive, e.g. triangle
- Scene $V :=$ BoundingBox(∂V)
- Set Ω of all unit directions ω
  - surface of a unit sphere
- Surface normal $\hat{n} : \partial V \to \Omega$
- Ray $(x, \omega) \in V \times \Omega$
- Hitpoint $h : V \times \Omega \to \partial V \cup \{\infty\}$
  - 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 $f_s(\omega_i, x, \omega) = f_s(\omega, x, \omega_i)$





- Scattered radiance

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

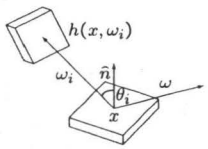
- Integral operator shorthand

$$L_s = T_{f_s} L_i$$

Vacuum Radiance Transport

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

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

- 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

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

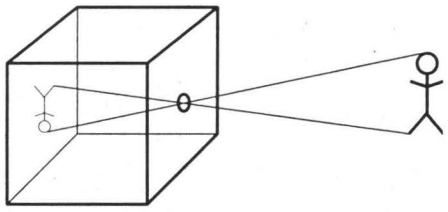
$\Rightarrow$ the global illumination problem is reduced to an integration problem

Principles of Rendering Algorithms

- **Pipeline:** Transformation, opt. cull, shade, clip, rasterize with Z-buffer
  - GeForceIII rasterization hardware
    - nVidea, California
- **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

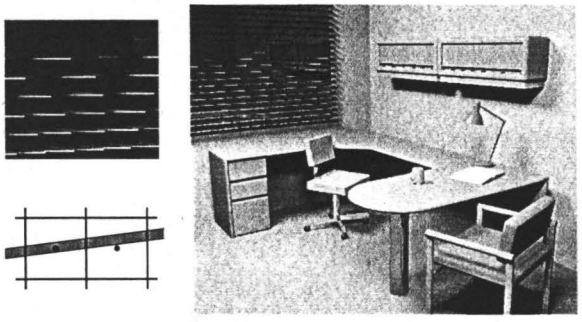
- Image: Matrix of pixels
  - Pix-el = **Pic**ture **El**ement
- **1980:** Turner Whitted: An Improved Illumination Model for Shaded Display.
- Trace ray from center of pixel through focal point into the scene

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.

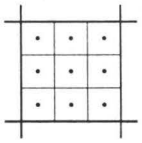


Anti-Aliasing by Supersampling

- In fact the pixel is an area, not a point !
  $\Rightarrow$ pixel color is average not a single sample

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

- Multiple samples instead of only pixel center

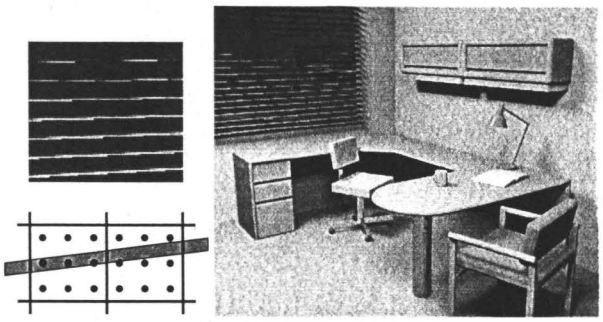


axis-aligned, regular grid

Supersampling

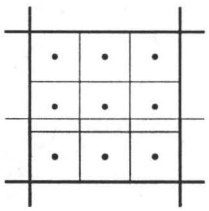
```
...
for (int x = 0; x < SizeX; x++)
   for (int y = 0; y < SizeY; y++)
   {
      SumOfSamples = Black;

      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



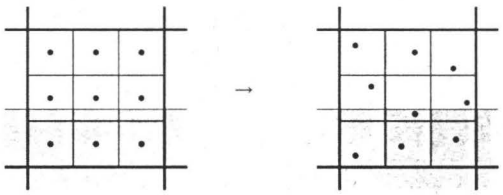
Observation 2: Still Aliasing

- since the 9 points can behave like only 3



Introducing Randomness

- Jittering



⇒ use random numbers

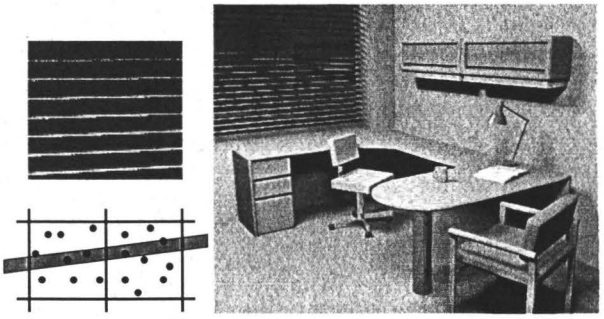
- Estimation by throwing the dice is superior !
  ⇒ Monte Carlo algorithms

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



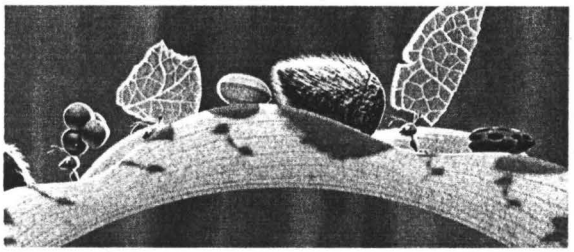
The Effort of A Bug's Life

- Movies: 24 frames/sec. $\Rightarrow$ 90 minutes are 129.600 frames
- Each frame: 2048 x 872 pixels at 4 bytes $\approx$ 7.1MB/frame
- 129.600 frames x 7.1MB/frame $\approx$ 925GB
- In addition: Many tests, the video, ...
  - A Bug's Life: 2TB of disk storage

- Render-farm: 1000 processors at 250MHz
  - ca. 15 hours per frame (mono-processor)

Scene Complexity

- Realism by details: Grass, stone, fur, hair, ...



$\Rightarrow$ natural look by complexity

The REYES-Architecture: Assumptions and Goals

- Scene complexity and variety
  - procedural models
  - online renderer
- Complex shading
  - procedural shaders
  - vast amount of textures
- Minimal ray casting
- Efficiency: Performance and image quality
  - anti-aliasing
- Flexibility

Design Principles

- Canonical frames
- Vectorization (streaming)
- Common graphic primitive
  - micro-polygons
- Locality
  - geometry
  - textures
- Linearity
- No limit on model size
- Backdoor for extensions

Principle: Spatial Locality

- Secondary rays
  - page thrashing
- Approximation of non-local effects by textures
  - reflections, by e.g. environment-maps
  - refractions
  - shadows

Principle: Point Sampling

- Simple, mighty, general
- Anti-aliasing
  - Monte Carlo integration by jittered sampling
- Visibility by Z-buffer
  - online algorithm
  - fits jittered sampling
  - backdoor for exact compositing

Principle: Micro-Polygon Nets

- Micro-polygon: Shaded quadrangles of 1/2 pixel size and one color
  - Nyquist limit
- Micro-polygon nets by splitting along (u, v) coordinates
  - in camera frame
  - resolution by estimating the object size on the screen
- Shading of all micro-polygons of a net
  - before visibility
  - backface culling

Are Micro-Polygons Waste ?

- No !
  - can be vectorized
  - texture locality
  - simple texture filtering
  - coherent subdivision (arrays)
  - no clipping
  - simple application of displacement mapping
  - no perspective projection computations

Principle: Texture Locality

- Important modelling mechanism: Textures
  - surface properties
  - approximation of reflections, refraction, and shadows
  - less ray casting
- Coherent access textures (CAT)
  - $(s, t) = (au + b, cv + d)$
- Random access textures (RAT)
  - all other mappings $(s, t) = f(u, v)$

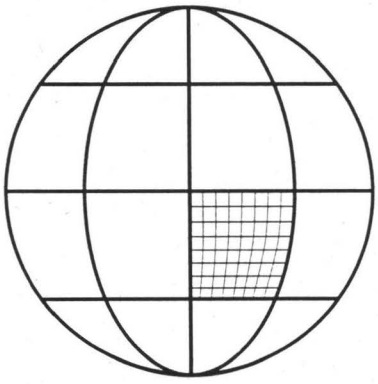
The Algorithm

- Read objects from database
  - potential splitting into other primitives
- Dicing into micro-polygon nets
  - only one net in main memory at a time $\Rightarrow$ memory control
  - Dicing along (u, v) (natural)
- Shade each micro-polygon in the net
- Sample and potential entry into the Z-buffer
  - no return to object space required

Methods of the Primitives

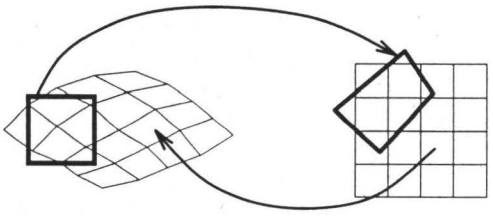
- **Bound**
  - bound in camera space then in image space
  - consider displacements
- **Split**
  - split into other primitives
  - must stay in bounding box
  - spline patches, subdivison surfaces, ...
- **DiceableTest**
  - possible ?
  - too many micro-polygons ?
  - or split...
- **Dice**
  - dice into micro-polygons

Split Primitives and Dice into Micro-Polygons



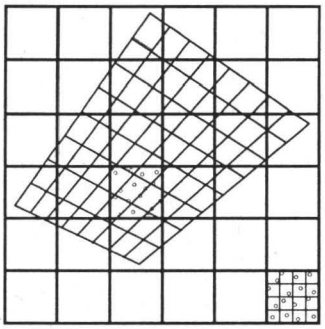
- **Split, Dice, DiceableTest**

Texturing



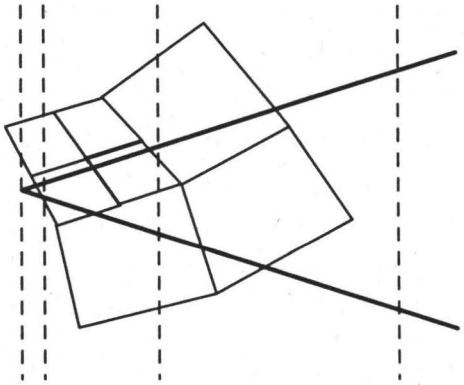
- No back-transformation
- Mip-maps

Sampling



- Jittered sampling
- Z-buffer

Culling



- no clipping

Extensions

- Motion blur
  - each sample is attached a moment in time
  - move micro-polygons to that moment in time
- Depth of Field
  - sample on the lens
  - move micro-polygons according to these samples
- Constructive solid geometry and transparency
  - multiple Z-entries per sample
- Shadows
  - shadow maps, deep shadow maps

Implementation

- Tile image into buckets
  1. primitives are sorted into the buckets by their left upper corner
  2. process buckets from left to right, top down
     (a) **Split** or **Dice**, shade, sort again
     (b) sample the elements in a bucket
- Memory control by **DiceableTest**
- Rendering time proportional to the number of primitives

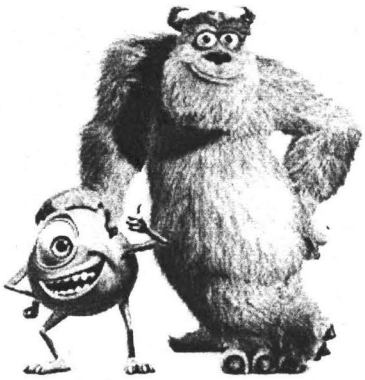
Disadvantages

- Shading before sampling
  - shading incorrect for motion blur
  - coherence of shading after sampling ?
- Dicing to micro-polygons may be difficult
  - e.g. implicit surfaces like blobs
- Bounding box difficult
  - e.g. particle systems
- canonical frames difficult
  - e.g. polygons
  - simple for bicubic patches

Advantages

- No inversions
- No clipping
- Vectorization of computations
- No texture memory thrashing
- No prefiltering
- Micro-polygons as common primitiv
- Rendering time proportional to scene complexity
- No limit on scene size
  - online algorithm

Visual Effects: Fur and Hair in Monsters Inc.



Monte Carlo and Beyond

- **Day 1**
  - computer graphics and visual effects
  - principles of rendering algorithms
- **Day 2**
  - **Monte Carlo integration**
  - quasi-Monte Carlo points
- **Day 3**
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo
- **Day 4**
  - application to computer graphics

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

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 i ***discrete***.

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

  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
  - $\lim_{x \to -\infty} P_X(x) = 0$
  - $\lim_{x \to \infty} P_X(x) = 1$
- **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

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

- 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

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

- 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 $\textbf{Prob}(\omega_i) = w_i$
  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
- 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 $(X, Y) \sim \mathcal{N}(0,1) \times \mathcal{N}(0,1)$

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

- 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

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

- 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

- $\mathcal{O}(n^s \log n)$ 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
- **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**

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

- 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

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

- Increase efficiency, not only variance reduction !!!

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

Error Control

- Unbiased estimator Y

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

- 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

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

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

- Variance of Monte Carlo part

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

- Lower bound $\mathcal{O}\left(N^{-\frac{r}{s}-\frac{1}{2}}\right)$ for $f \in C^r_M([0,1)^s)$ 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

- Computational complexity

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

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

  $\Rightarrow$ exploit induced grid structure

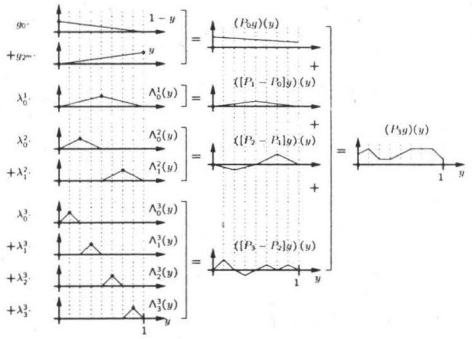
- Examples
  - accumulation buffer
  - multilevel method of dependent tests

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 $y_k = \frac{k}{2^m}$
- Method of dependent tests

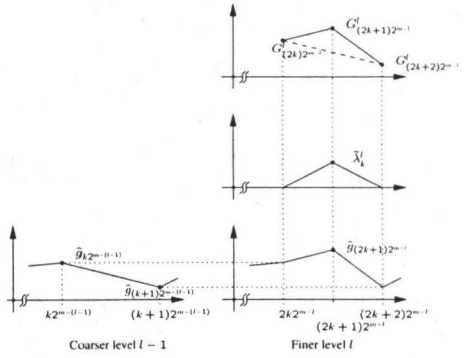
$$G^l_k := \frac{1}{N_l}\sum_{i=0}^{N_l-1} f(x_i,y_k) \text{ with } N_l := N \cdot \frac{2^m+1}{2^l+1}\cdot 2^{\alpha l}\cdot\frac{2^\alpha-1}{2^{\alpha(m+1)}-1}$$

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

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

Implementation

- In-place reconstruction



Efficiency Issues

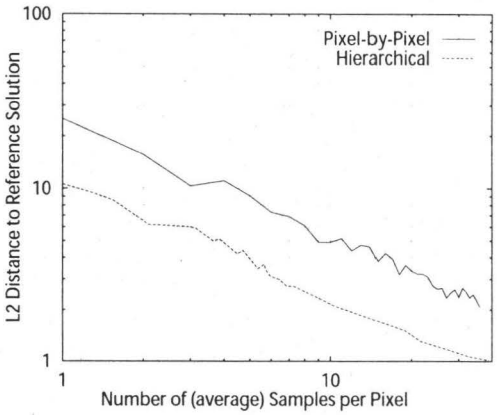
- Individual functionals
  - same high variance
  - same sampling rate, even if correlated
  - converged samples
- One function
  - small detail contribution if correlated

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

  - 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

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

- 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

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

- 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 ($\beta = 0$)	$w_j(x) := \frac{N_j}{p_j(x)\sum_{k=0}^{M-1} N_k}$	$w_j(x) = \frac{1}{M p_j(x)}$

- 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

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

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

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

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

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

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

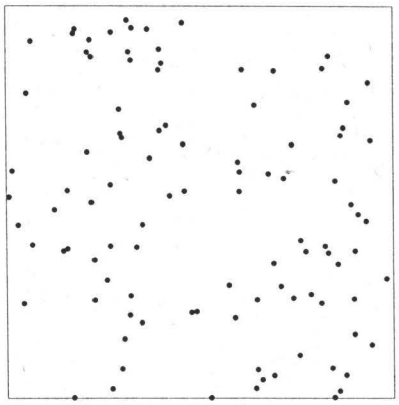
  - Lloyd-relaxation
  - jittered sampling

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

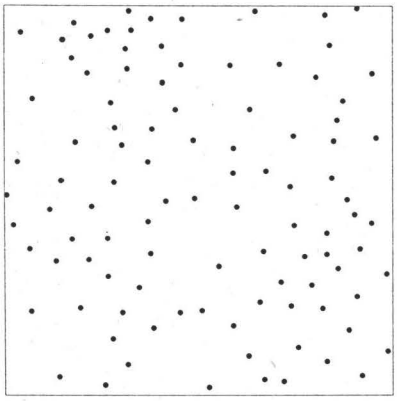
  $\Rightarrow$ *Small* number of relaxation steps yields blue-noise-samples
- \- Expensive iteration step
- \- No incremental sampling

Stratification by Lloyd-Relaxation



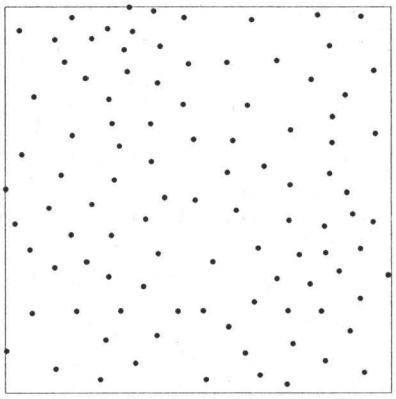
- Iteration 0

Stratification by Lloyd-Relaxation



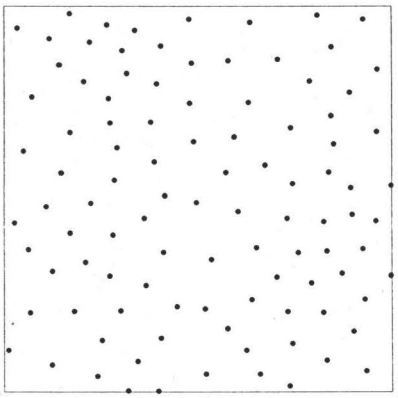
- Iteration 1

Stratification by Lloyd-Relaxation



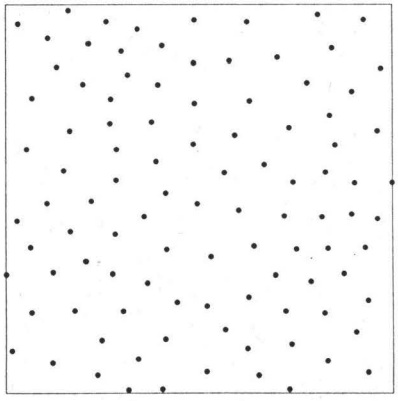
- Iteration 2

Stratification by Lloyd-Relaxation



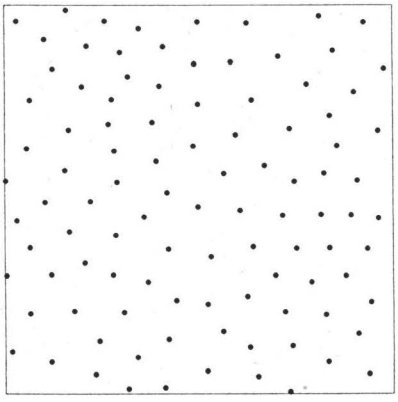
- Iteration 3

Stratification by Lloyd-Relaxation



- Iteration 4

Stratification by Lloyd-Relaxation



- Iteration 5

Replication Heuristics: Regularization

- Antithetic variables

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

  - 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

$$\int_{I^{s_1}}\int_{I^{s_2}} f(x,y)dydx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y_i)$$

computational complexity can be improved by

$$\int_{I^{s_1}}\int_{I^{s_2}} f(x,y)dydx \approx \frac{1}{NM}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} f(x_i,y_{i,j})$$

- 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

$$\begin{aligned}\int_{I^{s_1}}\int_{I^{s_2}} f(x,y)dydx &= \int_{I^{s_1}}\int_{I^{s_2}}\sum_{j=0}^{M-1} w_j(x,y)f(x,R_j(x,y))dydx \\ &\approx \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} w_j(x_i,y_i)f(x_i,R_j(x_i,y_i))dydx\end{aligned}$$

- 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

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

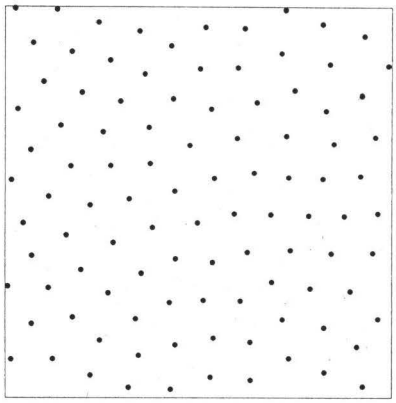
Monte Carlo and Beyond

- **Day 1**
  - computer graphics and visual effects
  - principles of rendering algorithms
- **Day 2**
  - Monte Carlo integration
  - **quasi-Monte Carlo points**
- **Day 3**
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo
- **Day 4**
  - application to computer graphics

Quasi-Monte Carlo Points

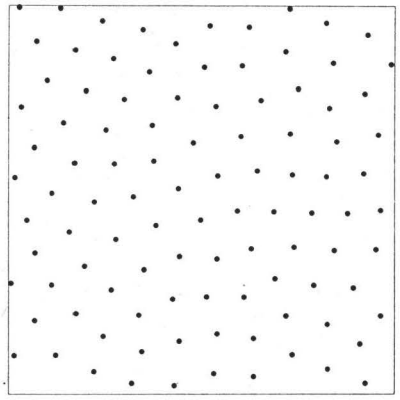
- Discrepancy
- Deterministic low discrepancy
  - Halton and Hammersley points
  - Scrambling
  - (t,m,s)-nets and (t,s)-sequences
  - Digital constructions
  - Good lattice points

Stratification by Lloyd-Relaxation



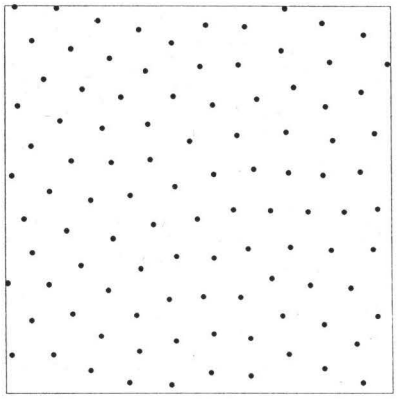
- Iteration 18

Stratification by Lloyd-Relaxation



- Iteration 19

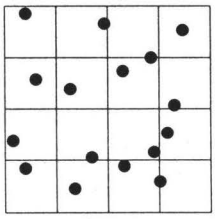
Stratification by Lloyd-Relaxation



- Iteration 20

Stratification: Jittered Sampling

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



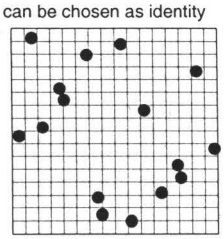
- Increased efficiency by increased uniformity of distribution
- Problem: N must be factorized

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



- Cannot be much worse than uniform random sampling

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

Replication Heuristics: Stratification

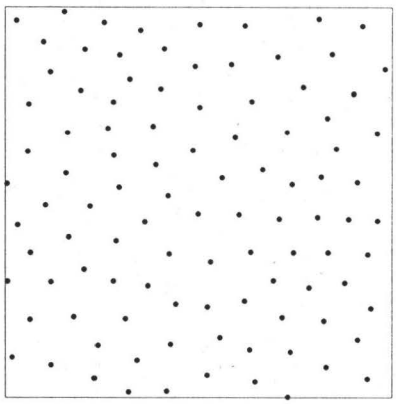
- Heuristic with
  - weights $w_j = \lambda_s(A_j)$, and
  - mappings $R_j : I^s \to A_j$
- Independent sampling for $N_j = \lambda_s(A_j)N$

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

- Dependent sampling

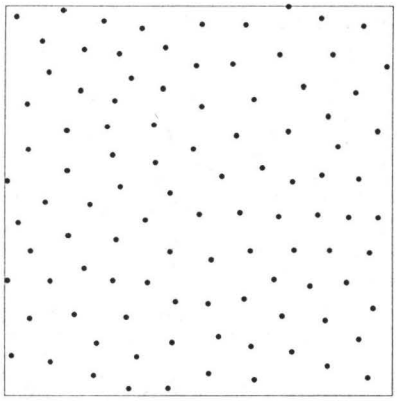
$$\int_{I^s} f(x)dx \approx \frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \lambda_s(A_j) f(R_j(x_i))$$

Stratification by Lloyd-Relaxation



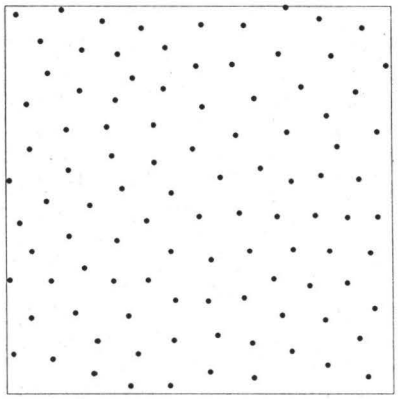
- Iteration 6

Stratification by Lloyd-Relaxation



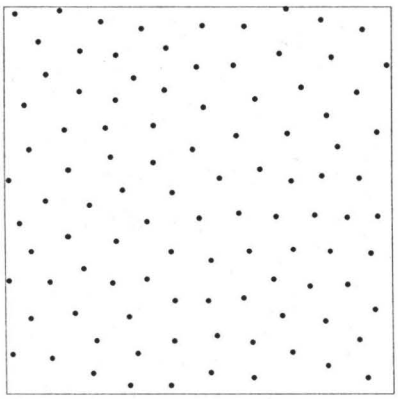
- Iteration 7

Stratification by Lloyd-Relaxation



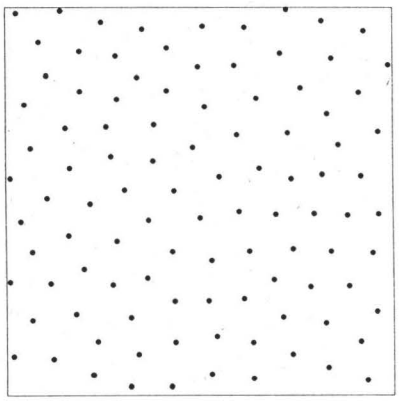
- Iteration 8

Stratification by Lloyd-Relaxation



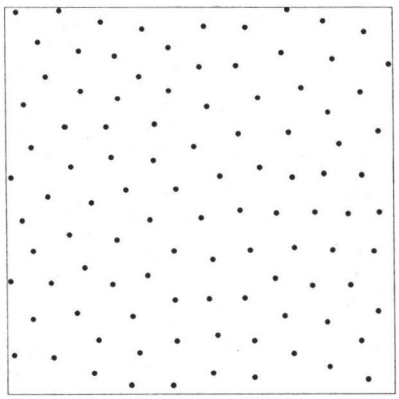
- Iteration 9

Stratification by Lloyd-Relaxation



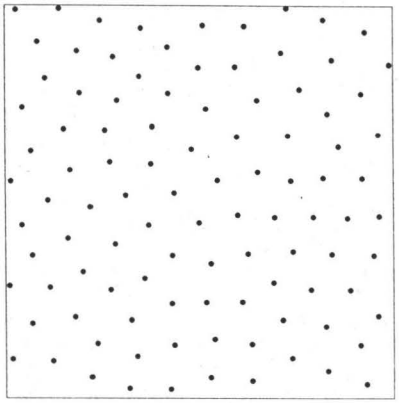
- Iteration 10

Stratification by Lloyd-Relaxation



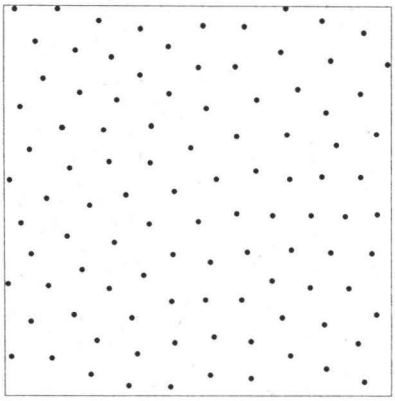
- Iteration 11

Stratification by Lloyd-Relaxation



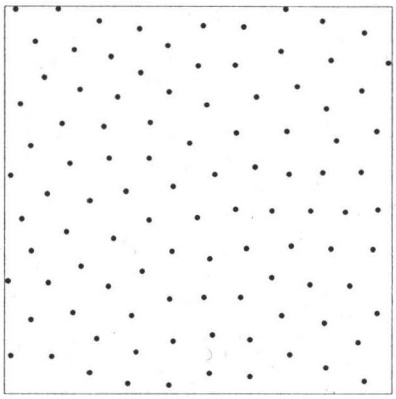
- Iteration 12

Stratification by Lloyd-Relaxation



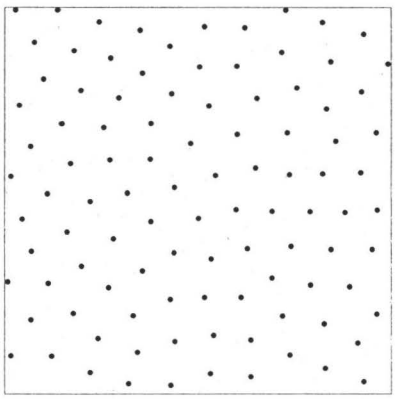
- Iteration 13

Stratification by Lloyd-Relaxation



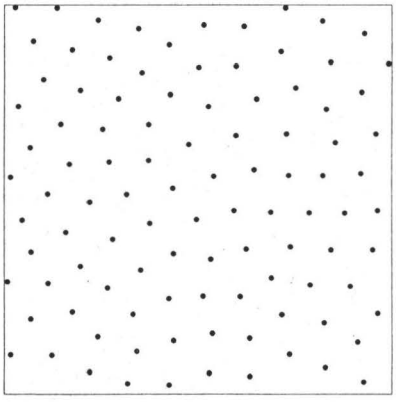
- Iteration 14

Stratification by Lloyd-Relaxation



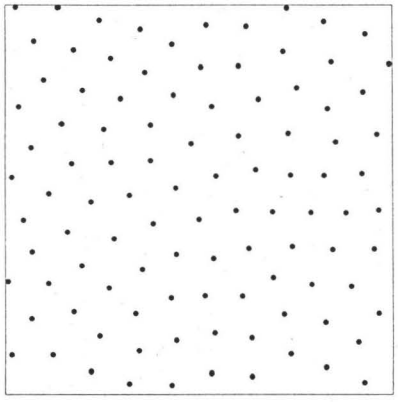
- Iteration 15

Stratification by Lloyd-Relaxation



- Iteration 16

Stratification by Lloyd-Relaxation



- Iteration 17

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

- **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

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

- 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

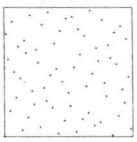
- Radical inverse (van der Corput sequence) in base b

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

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

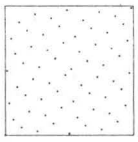
- Halton sequence $x_i := \left(\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i)\right)$ where b_i is the i-th prime number

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



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

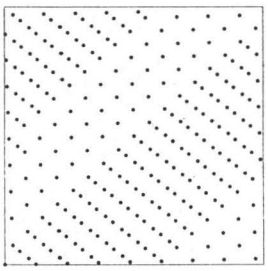
- Can be efficiently computed in contrast to L_∞-norm based discrepancies
- Numerical example: Triangular discrepancy

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

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

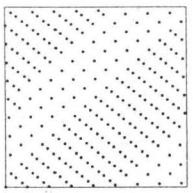
- Scrambled Halton sequence

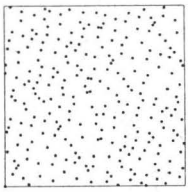
$$x_i := \left(\Phi_{b_1}(i, \sigma_{b_1}), \ldots, \Phi_{b_s}(i, \sigma_{b_s}) \right)$$

- Scrambled Hammersley point set

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

- 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

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

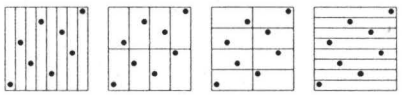
- **Definition:** For two integers $0 \leq t \leq m$, a finite point set of b^m points in s dimensions is called a (t, m, s)-*net in base* b, if every elementary interval of volume $\lambda_s(E) = b^{t-m}$ contains exactly b^t points.
- For (t, m, s)-nets in base b we have

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

  - t is the quality parameter
- **Note:** So far the concept applies to random and deterministic points

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

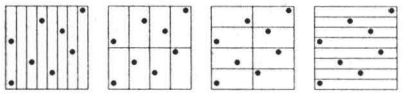
- (t, m, s)-net in base b:
  - Set P_N of $N = b^m$ s-dimensional points of low discrepancy
  - Every *elementary interval* of volume b^{t-m} contains exactly b^t points
- $(0, m, 2)$-net in base $b = 2$
  - Set P_N of $N = 2^m$ 2-dimensional points of low discrepancy
  - Every *elementary interval*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

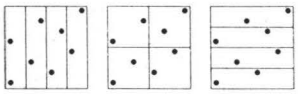
Example of a $(1, 3, 2)$-*Net in Base* $b = 2$

- 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 !

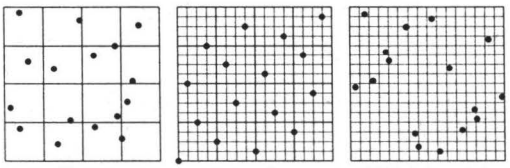
- All elementary volumes of a $(1, 3, 2)$-net in base $b = 2$:



$\lambda_s(E) = b^{t-m} = 2^{1-3} = \frac{1}{4}$ 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
- $(0, 2n, 2)$-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.
- 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

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

- 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
- 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

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

- 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

$$C_2 = I$$

- 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

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

- 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

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

- 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)
- $(0, 2)$-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

- **Numerical Recipes**
  - Sobol' sequence
- **http://www.mcqmc.org/Software.html**
  - Sobol' sequence
  - Faure sequence
  - Niederreiter sequence
- **http://www.multires.caltech.edu/software/libseq/index.html**
  - general package
  - several sequences (Halton, Niederreiter, ...)
- **http://www.dismat.oeaw.ac.at/pirs/niedxing.html**
  - generator matrices for the Niederreiter-Xing sequence

Good Lattice Points: Rank-1 *Lattices*

- **Definition:** A discrete subset

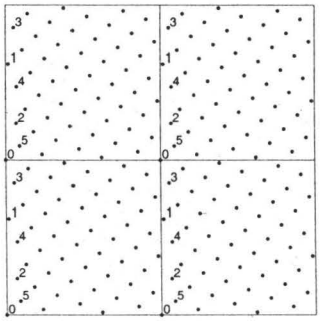
  $$L := P_N + \mathbb{Z}^s \subset \mathbb{R}^s$$

  that is closed under addition and subtraction is called a **lattice**.
- 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 $L \cap [0, 1)^s$

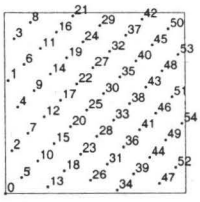


- Low discrepancy
- Much better discrepancy than regular grids

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 $\mathbf{g} = (1, F_{k-1})$ 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

  $$\mathbf{x}_i = \phi_b(i) \cdot \mathbf{g}$$
- 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
  - no completely uniform distribution due to correlation

Monte Carlo and Beyond

- **Day 1**
  - computer graphics and visual effects
  - principles of rendering algorithms
- **Day 2**
  - Monte Carlo integration
  - quasi-Monte Carlo points
- **Day 3**
  - **quasi-Monte Carlo integration**
  - Monte Carlo extensions of quasi-Monte Carlo
- **Day 4**
  - application to computer graphics

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

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:**

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

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

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

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

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

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

- **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 Δ 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)

- 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)

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

- 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

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

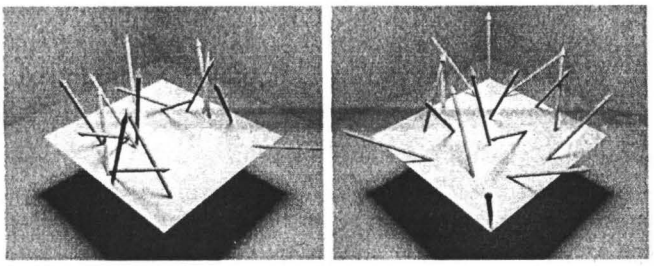
Discrete density approximation by elements of low discrepancy outperforms random sampling !!!

- Generalized Koksma-Hlawka inequality

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

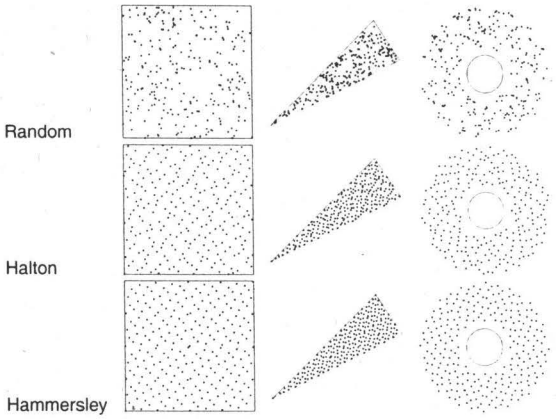
Discrete Density Approximation

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



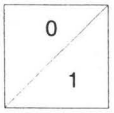
- **Note:** Assigning dimensions is crucial

Discrete Density Approximation



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 $s > 2$
- 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
- 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 $\mathcal{O}\left(N^{-\frac{s+1}{2s}}\right)$.*

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

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

- 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

$$\left|\int_{I^s} \chi_A(x)dx - \frac{1}{N}\sum_{i=0}^{N-1} \chi_A(x_i)\right| \leq \sqrt{cN^{-\frac{s+1}{s}}} \in \mathcal{O}(N^{-\frac{s+1}{2s}}) \qquad \textit{q.e.d.}$$

- **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"
- 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
- 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
- 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

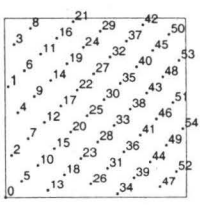
Quasi-Monte Carlo Integration using Lattice Points

- Originally developed for the class $E_\alpha(c)$ with $c > 0, \alpha > 1$, where

$$f \in E_\alpha(c) \Leftrightarrow |\hat{f}(h)| \leq \frac{c}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha} \qquad \bar{h}_j := \max\{1, |h_j|\}, \mathbf{h} \in \mathbb{Z}^s$$

- Error bound

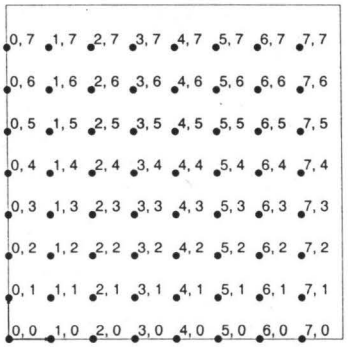
$$\left|\frac{1}{N}\sum_{i=0}^{N-1} f\left(\frac{i}{N}\mathbf{g}\right) - \int_{I^s} f(x)dx\right| \leq \sum_{\mathbf{h}\cdot\mathbf{g}\equiv 0(\bmod N), \mathbf{h}\neq 0} \frac{1}{(\bar{h}_1 \cdots \bar{h}_s)^\alpha}$$



- 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



- $\mathcal{O}(n^s \log n)$ 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

  $\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod N\}$

  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 ⇒ **way to break curse of dimension !**

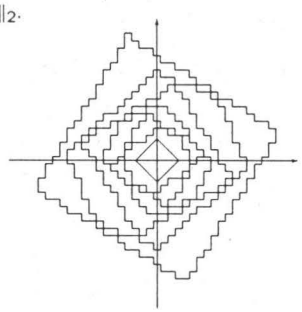
Determining the Wave Vectors

- Many possible choices for

  $\mathbf{k}_m \in Z_m := \{\mathbf{k} \in \mathbb{Z}^s \mid \mathbf{k}^T \cdot \mathbf{g} \equiv m \pmod N\}$

- 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

- **Use whenever you can write the problem as an integral**

Monte Carlo and Beyond

- **Day 1**
  - computer graphics and visual effects
  - principles of rendering algorithms
- **Day 2**
  - Monte Carlo integration
  - quasi-Monte Carlo points
- **Day 3**
  - quasi-Monte Carlo integration
  - **Monte Carlo extensions of quasi-Monte Carlo**
- **Day 4**
  - application to computer graphics

Monte Carlo Extensions of Quasi-Monte Carlo

- Random field synthesis on good lattice points
- Randomized quasi-Monte Carlo integration
- Randomized replications
- Restricted randomized replications

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 $\xi_{r,m}, \xi_{i,m} \sim \mathcal{N}(0,1)$
2. Fourier coefficients by filtering with Philipps spectrum $P_h(k_m)$

$$\tilde{h}_\omega(\mathbf{k}_m, t) = \sqrt{\frac{P_h(k_m)}{2}}\left((\xi_{r,m} + i\xi_{i,m})e^{i\omega(k_m)t} + (\xi_{r,m} - i\xi_{i,m})e^{-i\omega(k_m)t}\right.$$

3. Height field $h_\omega : \mathbb{R}^3 \to \mathbb{R}$ and normals by $\nabla h_\omega : \mathbb{R}^3 \to \mathbb{R}^3$

$$\begin{aligned} h_\omega(\mathbf{x}_n, t) &= \sum_{m=0}^{N-1} \tilde{h}_\omega(\mathbf{k}_m, t) e^{2\pi i m \frac{n}{N}} \\ \nabla h_\omega(\mathbf{x}_n, t) &= \sum_{m=0}^{N-1} 2\pi i \mathbf{k}_m \tilde{h}_\omega(\mathbf{k}_m, t) e^{2\pi i m \frac{n}{N}} \end{aligned}$$

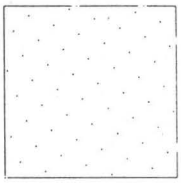
$\Rightarrow \dim \mathbf{x}_n = 2$, but evaluation by **one-dimensional** fast Fourier transform

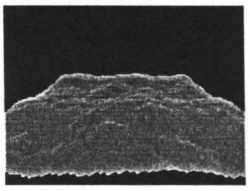
Example: Ocean Waves on Fibonacci Rank-1 Lattices

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

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

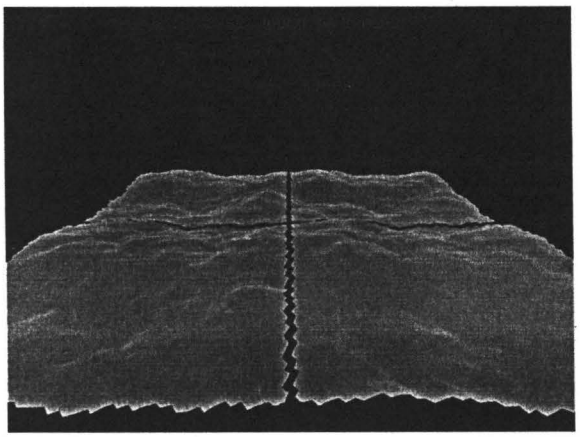
  - Low discrepancy
- Example: $N = F_{10} = 55$, $\mathbf{x}_n := \frac{n}{55}(1, 34)$



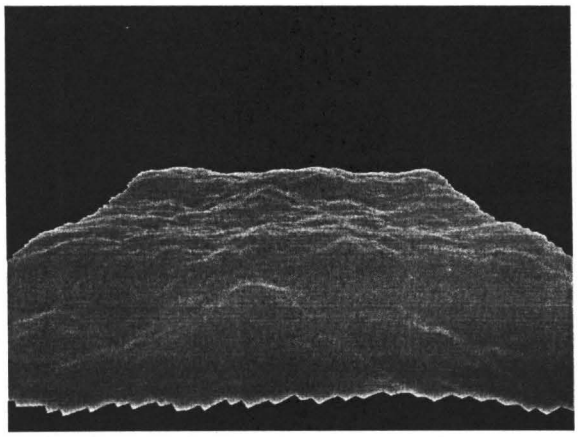


- Barycentric interpolation on periodic Delauney triangulation

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

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

  - 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

- 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

- 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

$$X_{k,i}^{(j)} := A_i^{(j)} + U_k^{(j)} \bmod 1 \text{ for } 1 \leq j \leq s$$

- 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 $H = H_h$
- 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

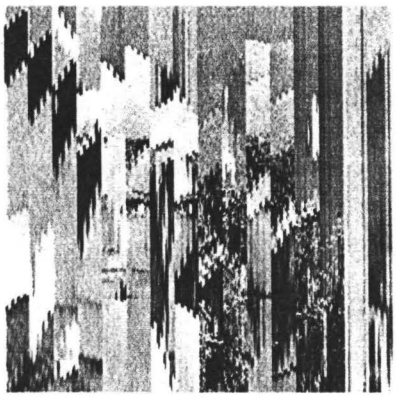
- Bit 2 of x

Replication by Scrambling

- Bit 3 of x

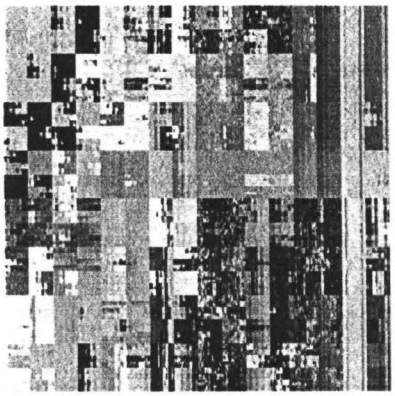
Replication by Scrambling

- 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

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

- 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 onc
- 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* $b = 2$

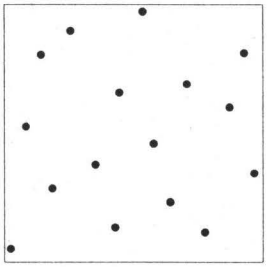
- $N = 2^m$ points $A = \{A_0, \ldots, A_{N-1}\}$
- The components correspond to the inverse permutations $\sigma_j^{-1}(i) = N \cdot A_i^{(j)}$
  - e.g. Hammersley: $\sigma_0^{-1}(i) = 2^m \cdot \frac{i}{N}$ and $\sigma_1^{-1}(i) = 2^m \cdot \Phi_2(i)$
- Random permutations on $\mathbb{Z}_2$ are random bit flips and can be vectorized
  - i.e. applying a path of permutation means XORing the bit vector of bit permutations
- 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;
}
```

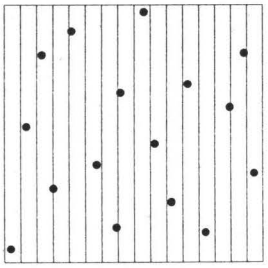
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



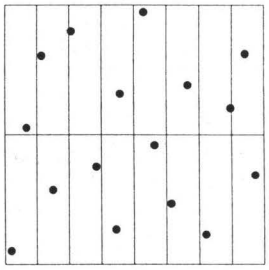
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Ne*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



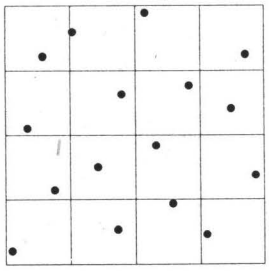
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



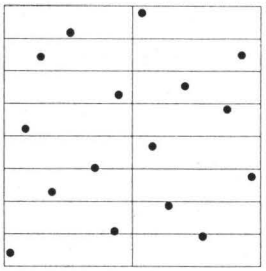
Example: Instance of a Randomly Scrambled $(0, 4, 2)$*-Ne*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



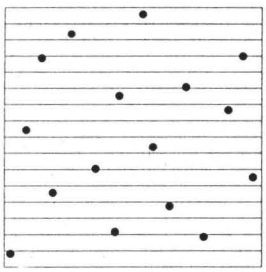
Example: Instance of a Randomly Scrambled $(0,4,2)$*-Net*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



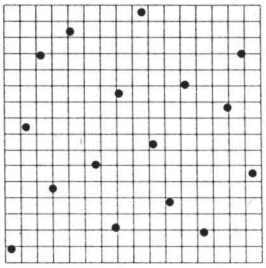
Example: Instance of a Randomly Scrambled $(0,4,2)$*-Ne*

- Random scrambling preserves the net properties
- Uniformly random, Stratified, Latin Hypercube sample, and even more...



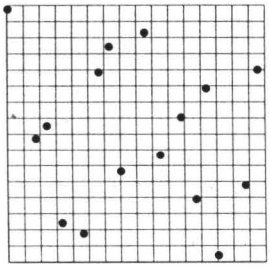
Example: Instance of a Randomly Scrambled $(0,4,2)$*-Net*

- 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

$$\frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y_i) \approx \int_{I^{s_1}}\int_{I^{s_2}} f(x,y)dxdy \approx \frac{1}{Ns}\sum_{i=0}^{N-1}\sum_{j=0}^{s-1} f(x_i,y_{i,j})$$

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 $z_j \in I^{s_2}$

$$\begin{aligned} R_j : I^{s_2} &\to I^{s_2} \\ y &\mapsto (y+z_j) \bmod 1 \end{aligned} \quad \Rightarrow \quad \int_{I^{s_2}} g(y)dy = \int_{I^{s_2}} g(R_j(y))dy$$

- 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
- 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

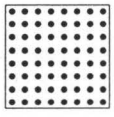
- **Day 1**
  - computer graphics and visual effects
  - principles of rendering algorithms
- **Day 2**
  - Monte Carlo integration
  - quasi-Monte Carlo points
- **Day 3**
  - quasi-Monte Carlo integration
  - Monte Carlo extensions of quasi-Monte Carlo
- **Day 4**
  - **application to computer graphics**

Applications to Computer Graphics

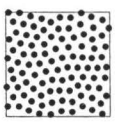
- Interleaved sampling
  - interleaved method of dependent tests
- Volume rendering
  - dependent splitting by restricted Cranley-Patterson rotations
- Bidirectional path tracing
  - padded replications sampling for cheap high-dimensional samples
- Distribution ray tracing
  - strictly deterministic
  - dependent splitting by restricted Cranley-Patterson rotations
- **Note:** Discontinuous integrands and high dimension !!!

Sampling

- Regular grids

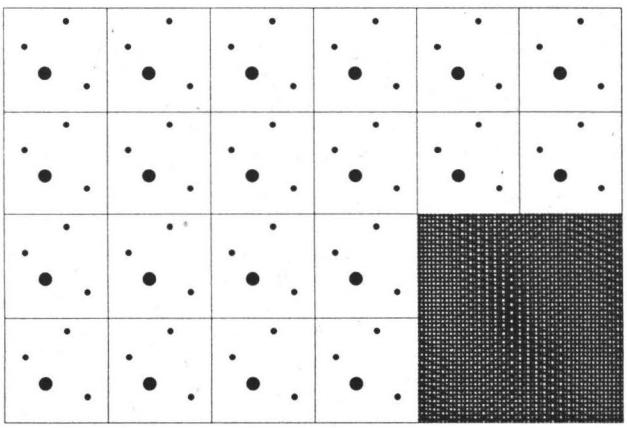


  + fast rasterizers
  - aliasing potential
  - slow convergence
- Irregular patterns

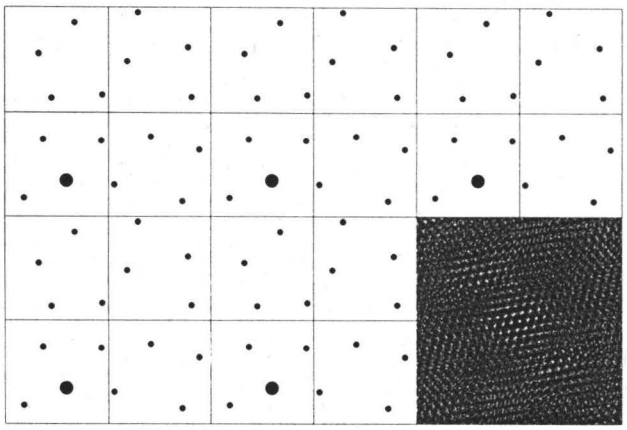


  - no rasterizers
  + optimal spectral properties
  + low discrepancy

Accumulation Buffer



Interleaved Sampling



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

$$g_P(y) = \int_{[0,1)^s} f(x,y)dx \approx \frac{1}{N}\sum_{i=0}^{N-1} f(x_i,y)$$

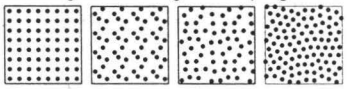
  - Interleaved sampling

$$g_P(y) = \int_{[0,1)^s} \chi_P(x) f'(x,y)dx \approx \frac{1}{N'}\sum_{i=0}^{N'-1} \chi_P(x'_i) f'(x'_i,y)$$

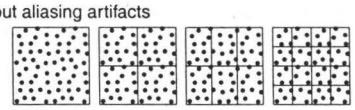
- Exploit intrinsic high coherence
  - new hardware
  - new software parallelization paradigm

Sampling Patterns for Interleaved Sampling

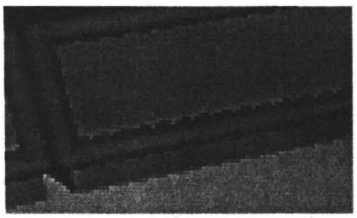
- Precomputed Max-Lloyd relaxation points as basis pattern
  - periodically tile seamlessly
  - blue noise spectral characteristics (minimum distance property)
  - low discrepancy (correlated)
  - for arbitrary problem dimension
- Size N' of irregular basis pattern
  - blend between regular and irregular sampling



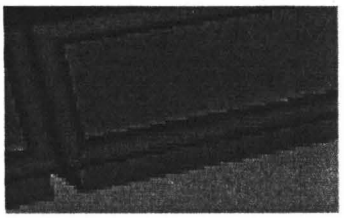
- Choice of interleaving ratio by χ_P
  - spread out aliasing artifacts



Application: Antialiasing



Accumulation Buffer

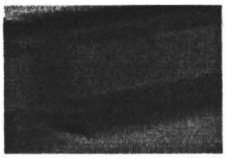


Interleaved Sampling

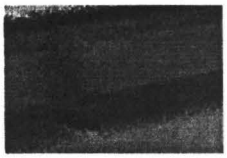
- Reduced aliasing at only 4 samples per pixel
  - artifacts spread out
  - artifacts from repetition, **not** from deterministic sampling
- Simple to implement by hardware (current and future)

Application: Motion Blur

Accumulation Buffer



Interleaved Sampling



Uncorrelated Sampling

- Artifacts replaced by noise at 16 samples per pixel
- 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

$$\int_{I^3} f(x,y,z)dxdydz$$

  - 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

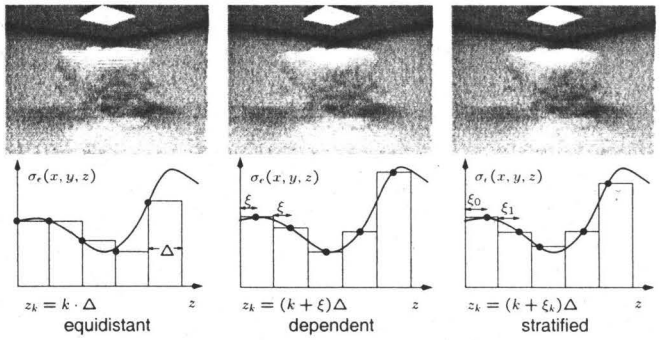
Remember...

- Discontinuous integrand mainly in x, y
- 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)

⇒ **Never use one-dimensional stratified Monte Carlo !!!**

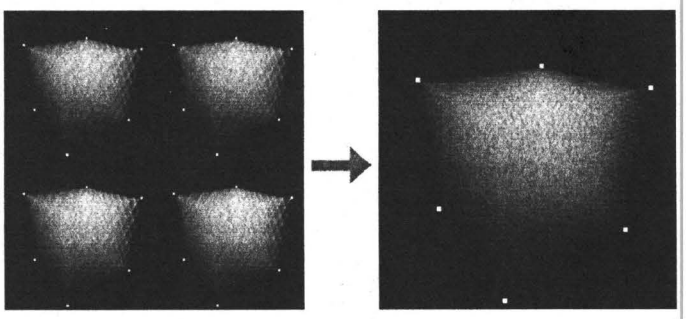
⇒ **Use randomized quasi-Monte Carlo instead**

Volume Rendering



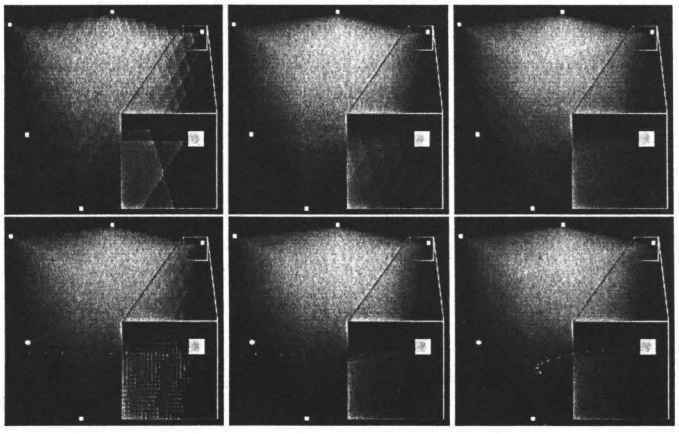
- Dependent sampling saves ~20% rendering time wasted for random numbers
- Equidistant, i.e. correlated, samples have lower discrepancy that stratified samples
- Combine with interleaved sampling: Coherent ray marching

Application: Volume Rendering

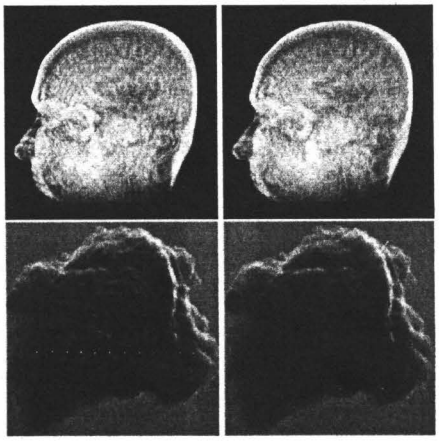


- Much improved depth antialiasing (unbiased)
- Simply interleaving images
  - **coherent** ray marching

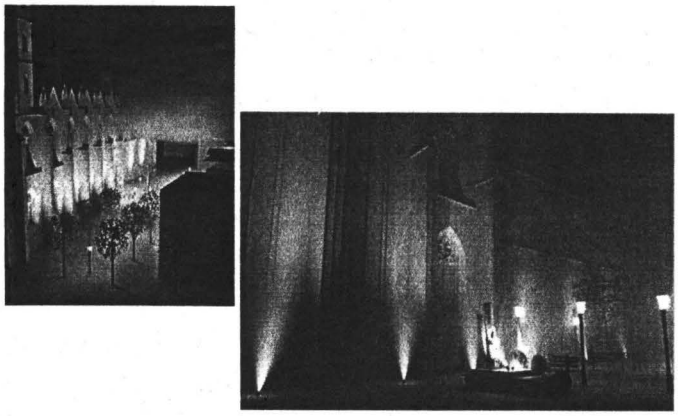
Application: Volume Rendering



Application: Volume Rendering



Application: Volume Rendering



The Global Illumination Problem

- Three-point form of the light transport equation

$$L(y \to z) = L_e(y \to z) + \int_S L(x \to y) f_s(x \to y \to z) G(x \leftrightarrow y) dA(x)$$

Measurement equation

$$I_j = \int_{S \times S} W_e^{(j)}(x \to y) L(x \to y) G(x \leftrightarrow y) dA(y) dA(x)$$

$\Rightarrow$ Path integral formulation

$$I_j = \sum_{k=1}^{\infty} \int_{\mathcal{P}_k} f_j(\bar{x}) d\mu_k(\bar{x}) = \int_{\mathcal{P}} f_j(\bar{x}) d\mu(\bar{x})$$

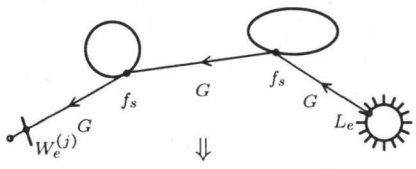
- Bidirectional path tracing
  - Multiple importance sampling for quasi-Monte Carlo integration
  - How much is sacrificed by randomized quasi-Monte Carlo integration ?
  - Adapt to two-dimensional structure of integral equation
    - padded replications sampling

Path Integral Formulation

- Path space and path measure

$$\mathcal{P}_k = \{\bar{x} = x_0 x_1 \dots x_k \mid x_i \in S\} \qquad d\mu_k(\bar{x}) = \prod_{i=0}^{k} dA(x_i)$$

- 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

$$p_1, p_2, \dots, p_N \; : \; D \to \mathbb{R}_0^+$$

- Heuristic

$$w_1, w_2, \dots, w_N \; : \; D \to \mathbb{R}_0^+$$

  - $\sum_{i=1}^{N} w_i(x) = 1$ 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

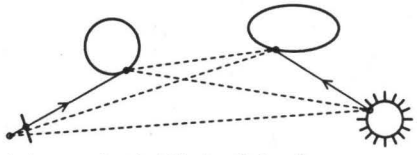
$$\int_D f(x) dx \approx \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{N} w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})} \qquad \text{where} \quad x_{i,j} \sim p_i$$

- Example: Balance heuristic

$$w_i(x) := \frac{p_i(x)}{\sum_{\ell=1}^{N} p_\ell(x)} \qquad \Rightarrow \qquad \int_D f(x) dx \approx \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{N} \frac{f(x_{i,j})}{\sum_{\ell=1}^{N} p_\ell(x_{i,j})}$$

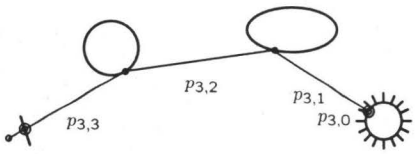
Bidirectional Path Tracing

- Generation of path space samples



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 $A = \{a_1, a_2, \dots, a_m\}$ where $a_i \in I^s$

  r randomized replications of A

$$\begin{matrix} x_{1,1} \dots & x_{i,1} & \dots x_{m,1} \\ \vdots & \vdots & \\ x_{1,j} \dots & x_{i,j} & \dots x_{m,j} \\ \vdots & \vdots & \\ x_{1,r} \dots & x_{i,r} & \dots x_{m,r} \end{matrix}$$

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