Progressive Multi-Jittered Sample Sequences

Per Christensen   Andrew Kensler   Charlie Kilpatrick

Pixar Animation Studios

EGSR 2018, Karlsruhe

PIXAR

Overview

- Motivation
- Survey + evaluation of existing sample sequences
- 3 new algorithms: pj, pmj, pmj02 samples
- More evaluations: pixel sampling, area lights
- Variations: blue noise, multi-class

Motivation

- RenderMan used to be off-line rendering (final movie frames)
- But lately: also interactive rendering for faster feedback: modeling, animation, lighting, …
- This has consequences for sample pattern choices!

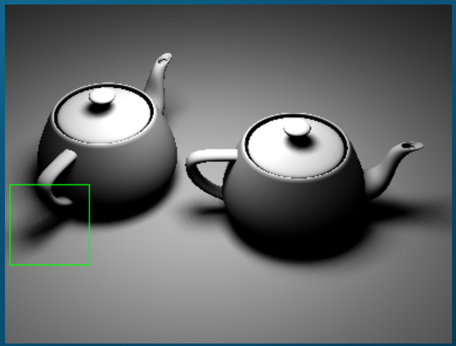
PIXAR

Sample sets vs. sequences

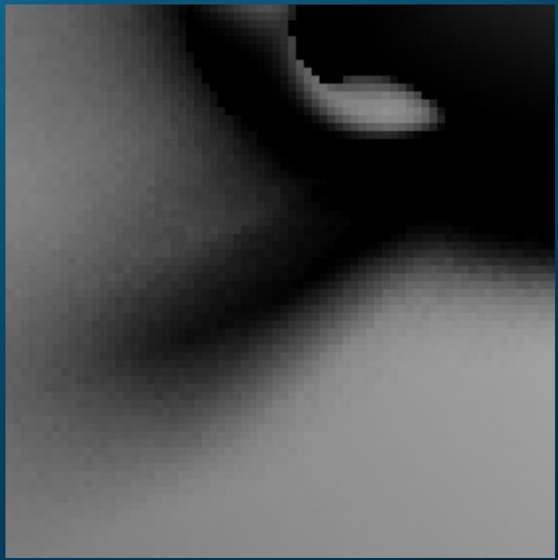
- Finite sets:
  - Need to know how many samples
  - No good for incremental rendering, adaptive sampling
- Infinite sequences:
  - Every prefix has a good distribution
  - No need to know how many samples

Sample sets vs. sequences

- Incremental rendering: area light sampling



100 samples from set with 400

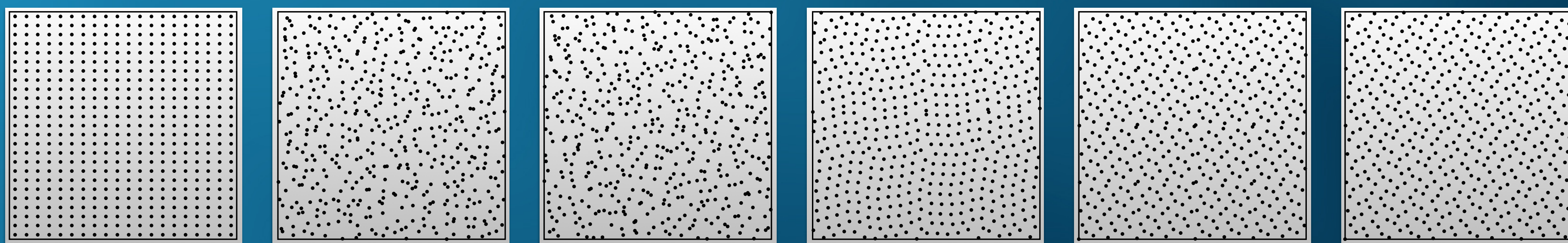


100 samples from sequence

(same render time)

PIXAR

Sample sets

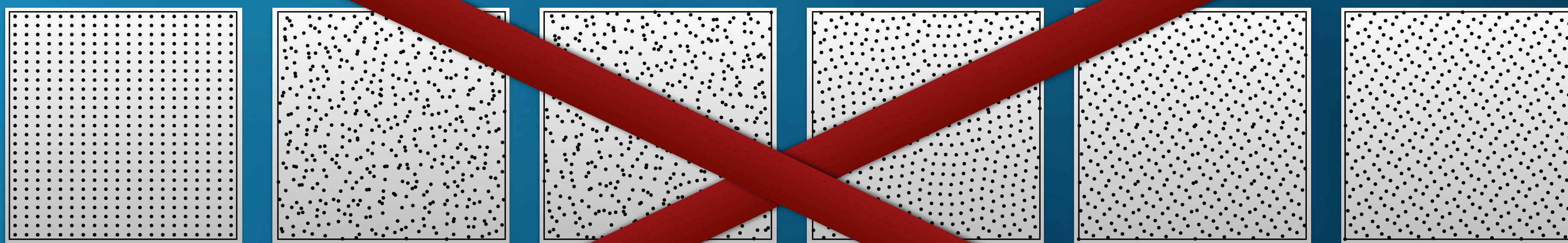


regular grid | jitter | multijitter | correlated multijitter | Hammersley | Larcher-Pillichshammer

[Chiu94]   [Kensler13]   quasi-random (“qmc”) sets

PIXAR

Sample sets



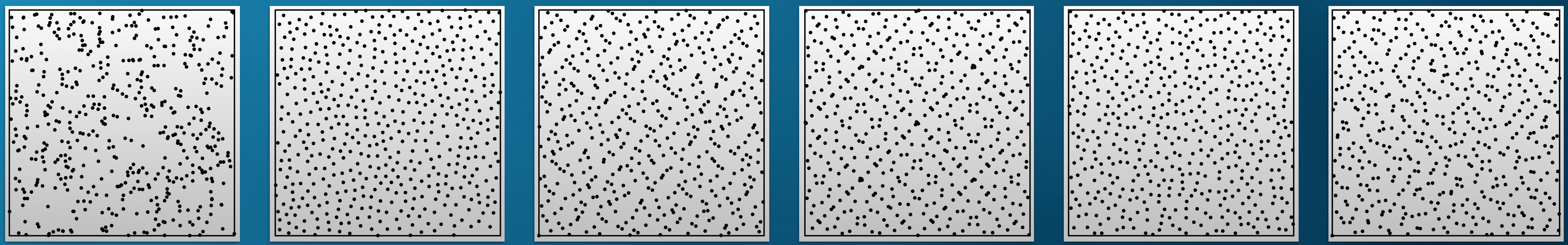
[Chiu94]

[Kensler13]

quasi-rando mc") sets

PIXAR

Sample sequences



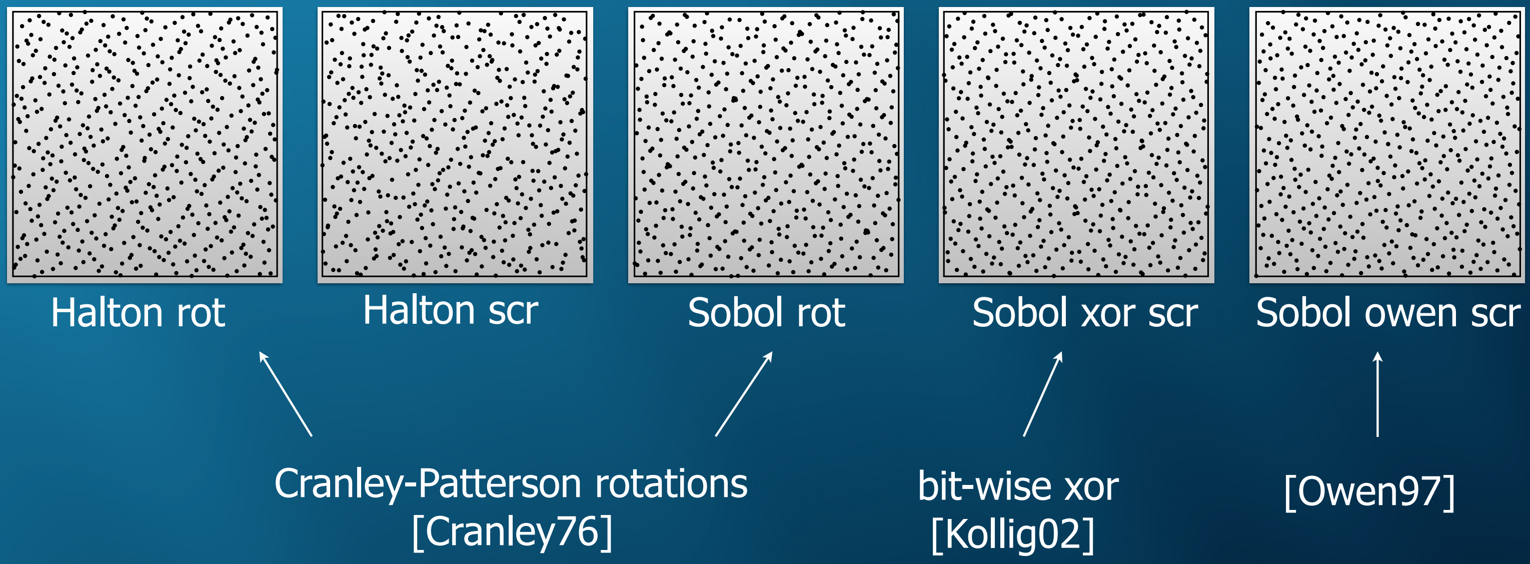
random | blue noise | Halton | Sobol | [Ahmed17] | [Perrier18]

(best candidate/
Poisson disk)

quasi-random sequences

blue noise + stratification

Sample sequences: randomized quasi-random



Comparing sample sequences

- How to measure “best”?
- Definitely not lowest discrepancy — don’t get me started!
- Better:
  - measure error when sampling various functions
  - confirm results in actual rendering: sample pixel positions, area lights, …

PIXAR

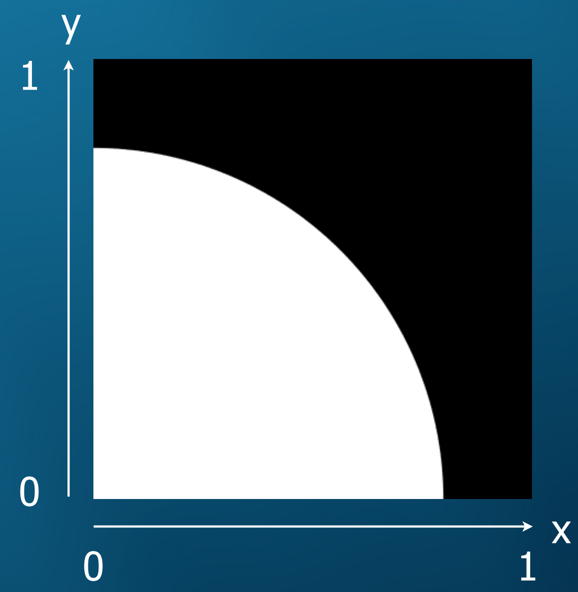
Initial tests of sequences

- Sample simple discontinuous and smooth functions
- Known analytical reference value

PIXAR

Initial tests: discontinuous functions

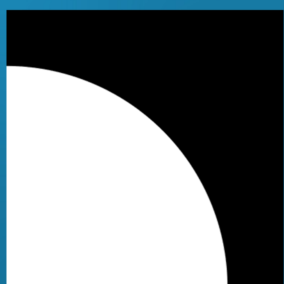
- Disk function: $f(x,y) = 1$ if $x^2 + y^2 < 2/\pi$, 0 otherwise

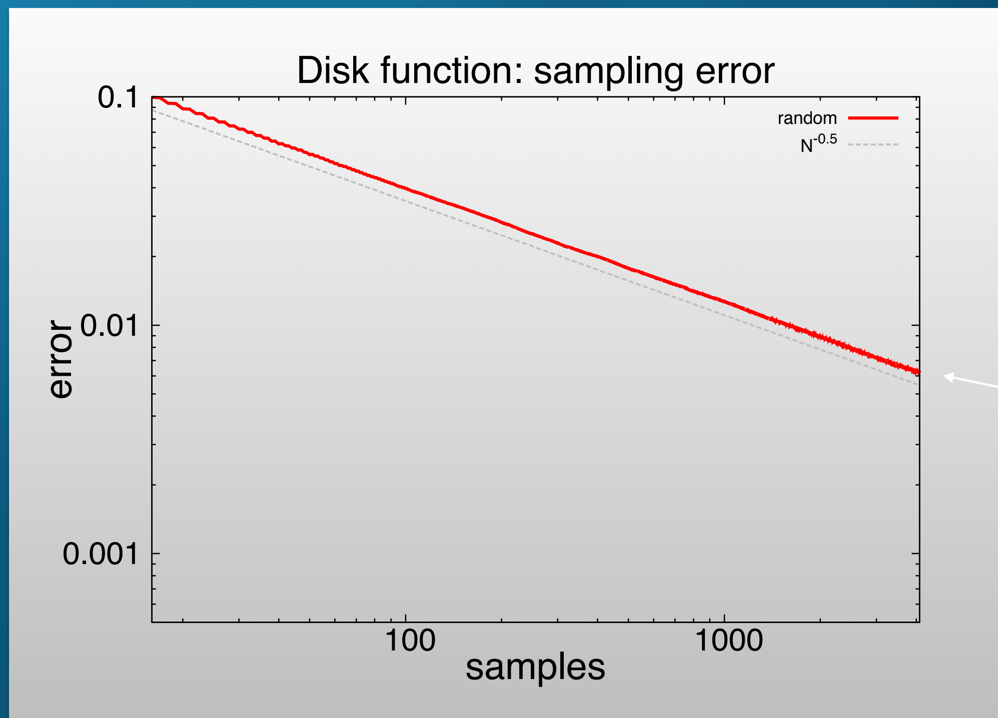


Reference value: 0.5

PIXAR

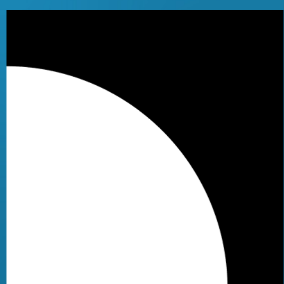
Initial tests: discontinuous functions

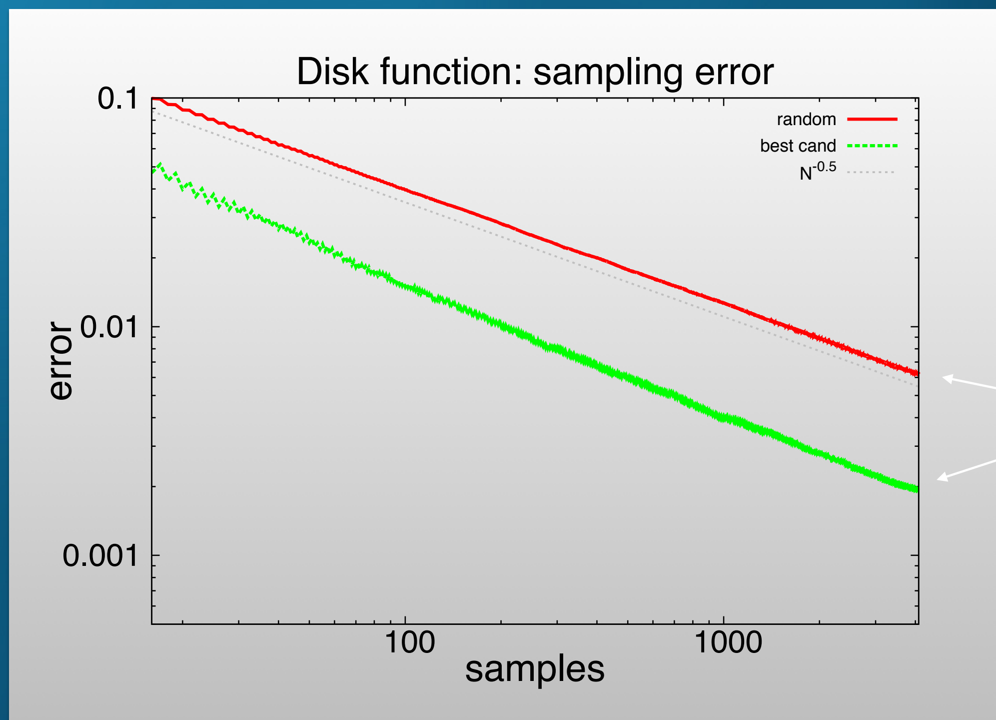




bad: $O(N^{-0.5})$

Initial tests: discontinuous functions

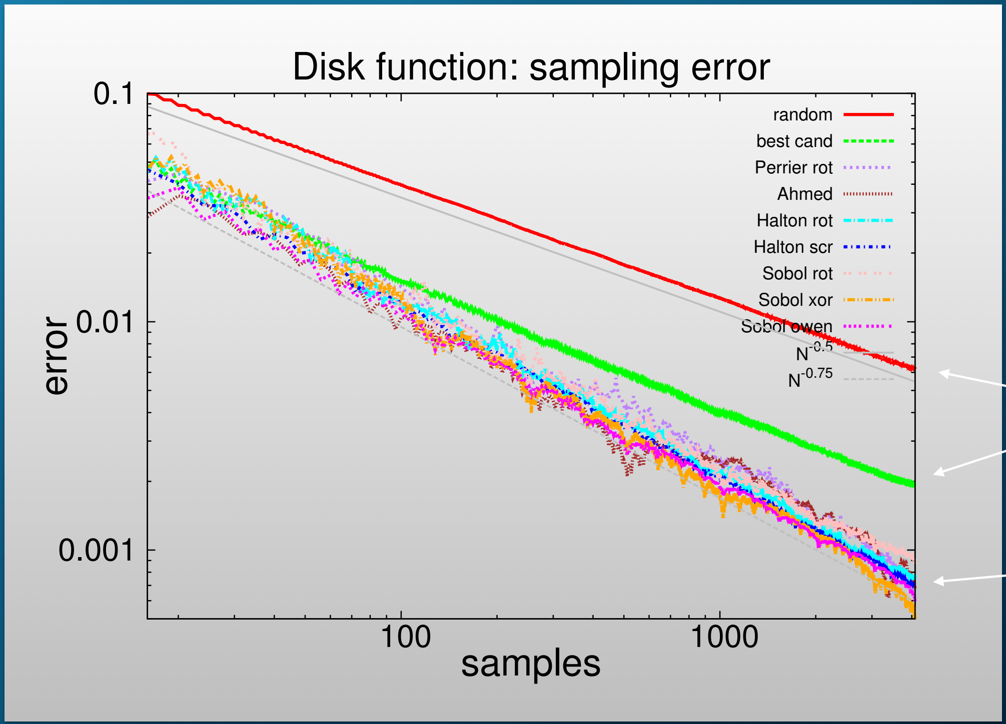




bad: $O(N^{-0.5})$

PIXAR

Initial tests: discontinuous functions



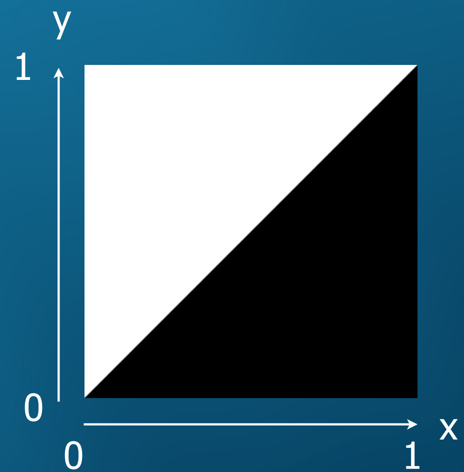
bad: $O(N^{-0.5})$

okay: $O(N^{-0.75})$

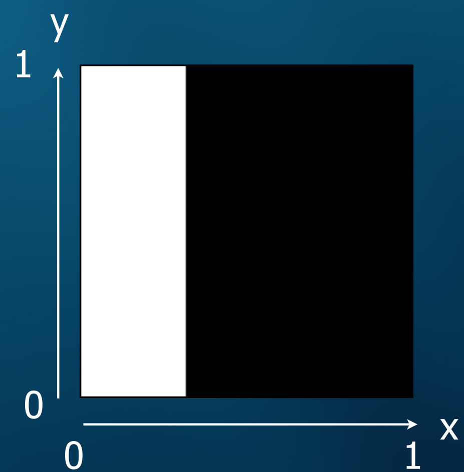
PIXAR

Initial tests: discontinuous functions

- Similar tests for triangle function and step function shows high error for Sobol rot and Sobol xor, and Ahmed and Perrier



Reference value: 0.5

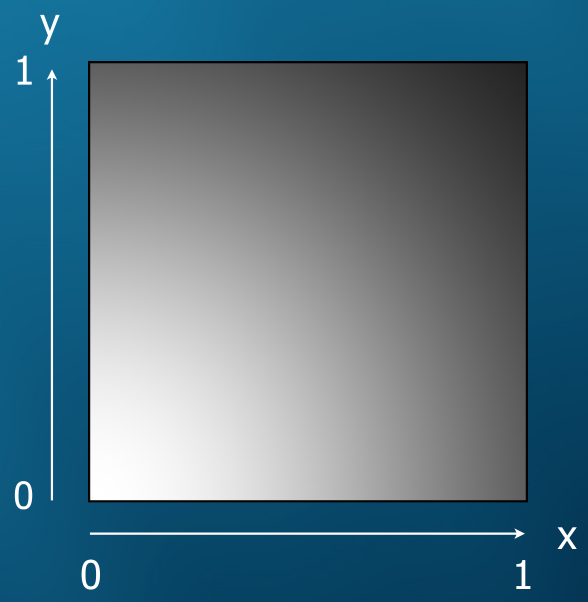


Reference value: 1/pi

PIXAR

Initial tests: smooth functions

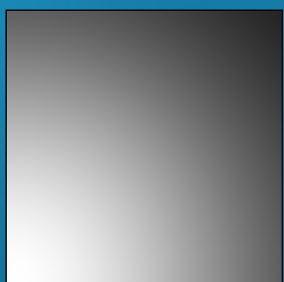
- 2D Gaussian function: $f(x,y) = \exp(-x^2 - y^2)$

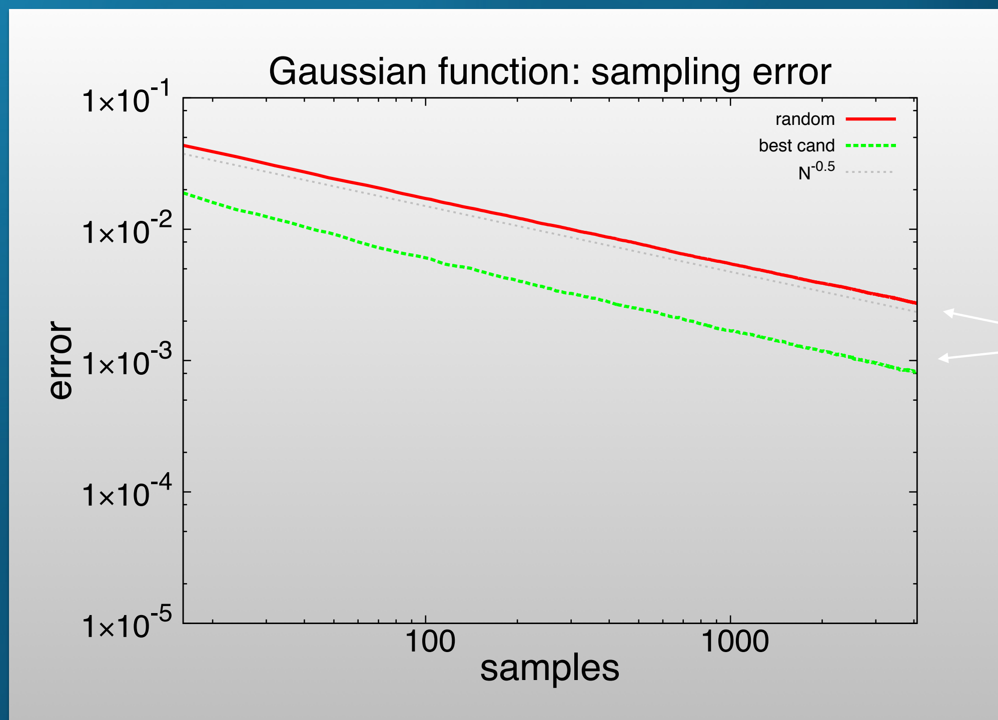


Reference value: ~0.557746

PIXAR

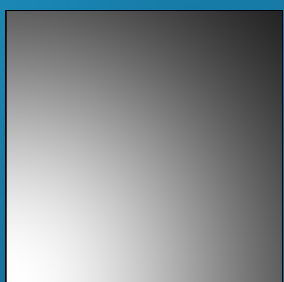
Initial tests: smooth functions

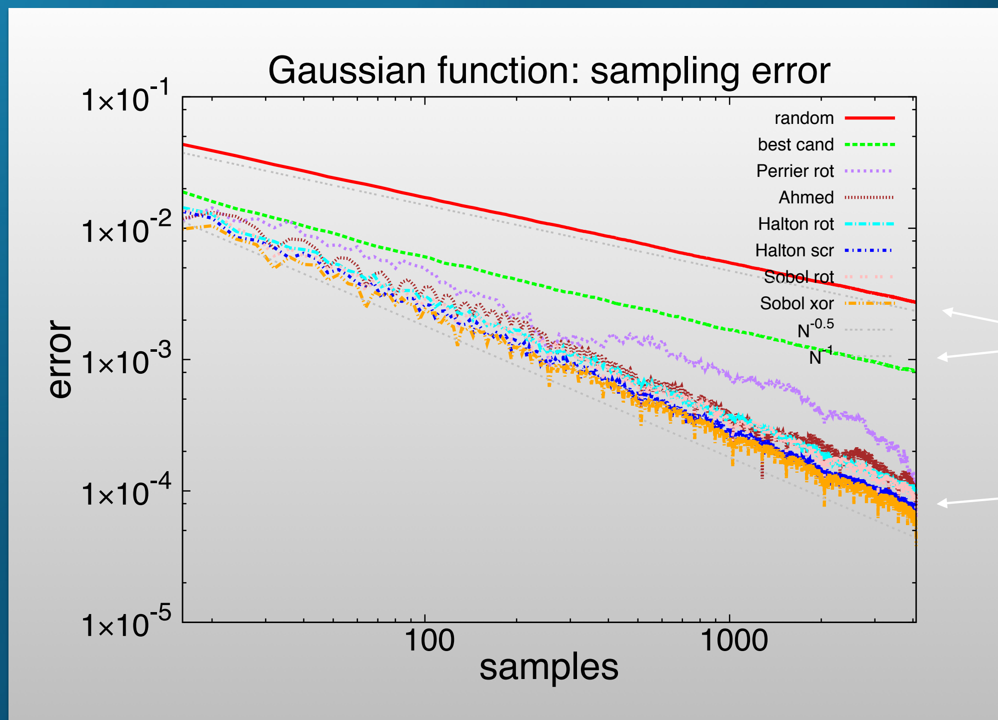




bad: $O(N^{-0.5})$

Initial tests: smooth functions



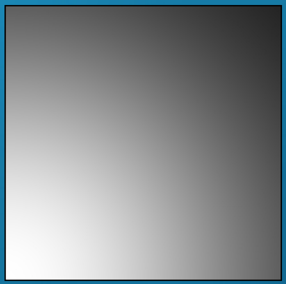


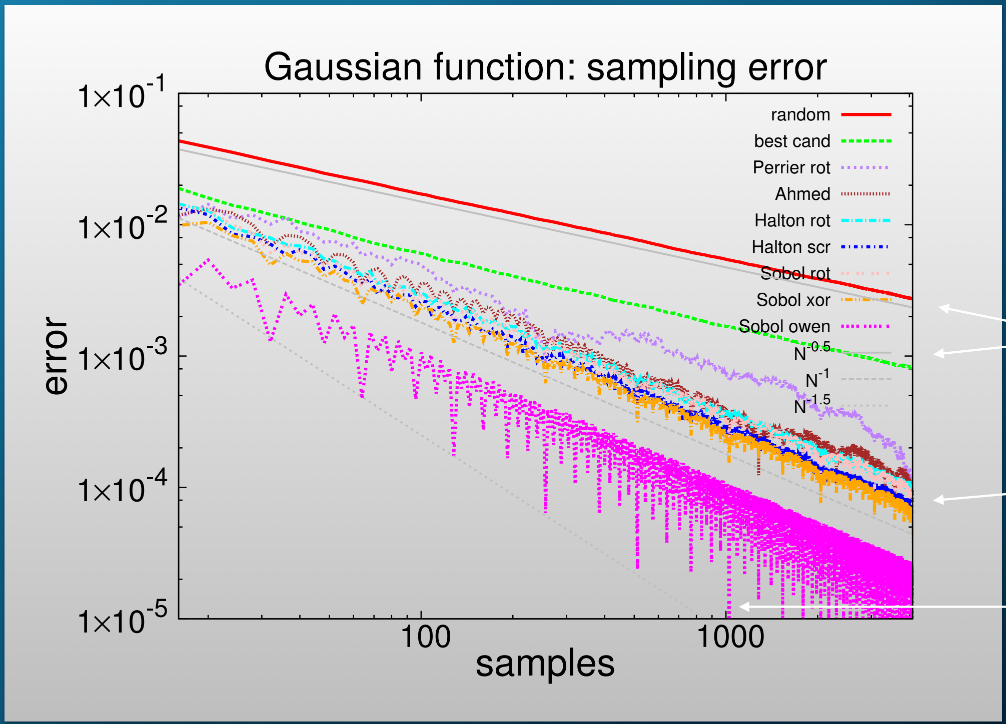
bad: $O(N^{-0.5})$

good: $O(N^{-1})$

PIXAR

Initial tests: smooth functions





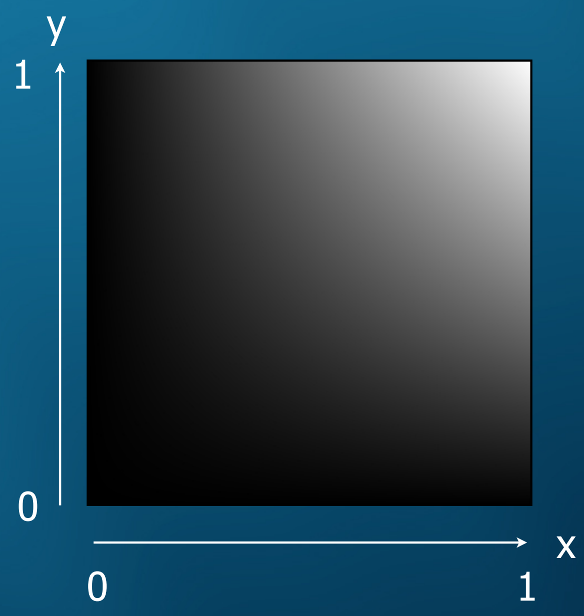
bad: $O(N^{-0.5})$

good: $O(N^{-1})$

excellent: $O(N^{-1.5})$

Initial tests: smooth functions

- Bilinear function f(x,y) = xy: same results



Reference value: 0.25

PIXAR

Summary of initial tests

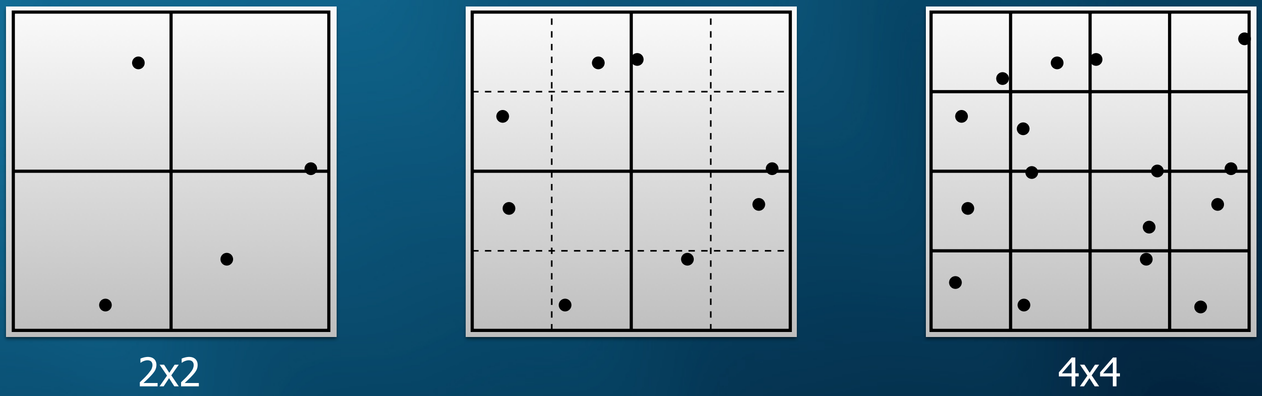
- Owen-scrambled Sobol sequence is best:
  - no pathological error for discontinuities at certain angles
  - extraordinarily fast convergence for smooth functions

Progressive (multi)jittering

- New framework for stochastic sample generation
- Three simple algorithms that progressively fill in holes in increasingly fine stratifications

Progressive jittered sequences — pj

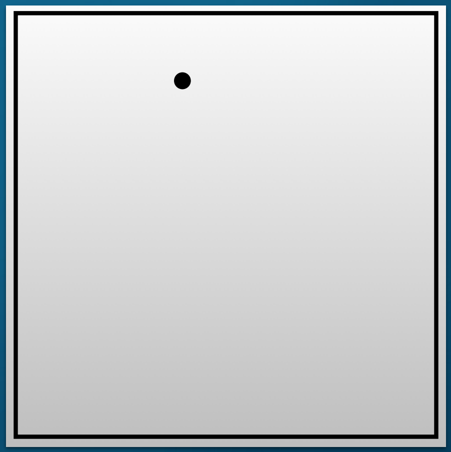
- No multi-jitter
- Stratification goal: increasingly small squares



PIXAR

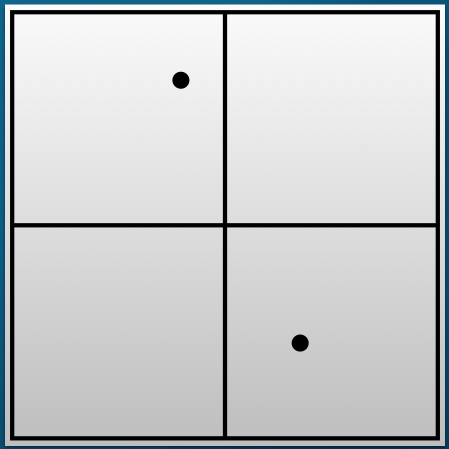
Progressive jittered sequences — pj

- Sample 1: random position



Progressive jittered sequences — pj

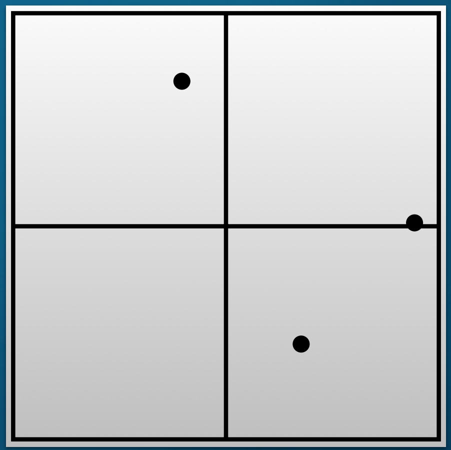
- Sample 2: opposite diagonal



PIXAR

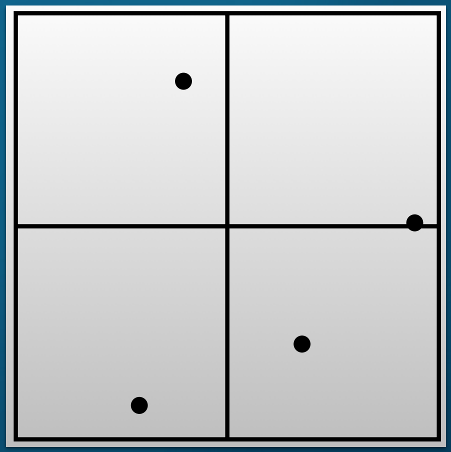
Progressive jittered sequences — pj

- Sample 3: one of the two empty squares



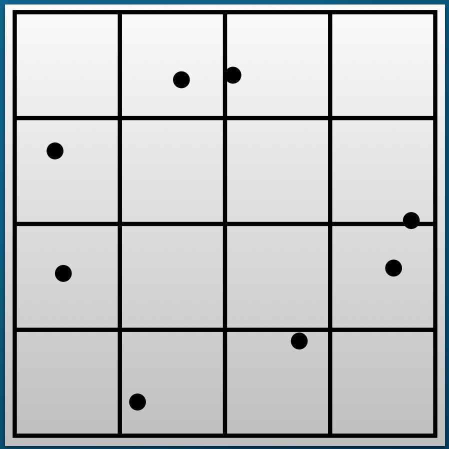
Progressive jittered sequences — pj

- Sample 4: last remaining square



Progressive jittered sequences — pj

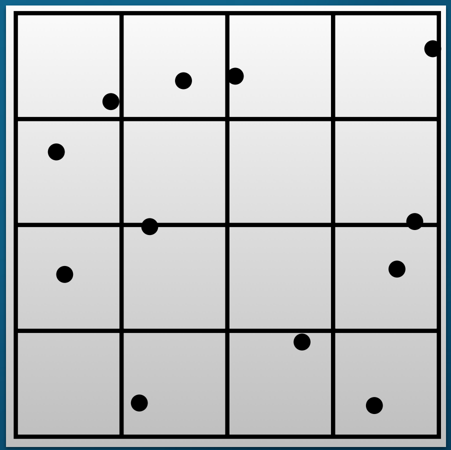
- Samples 5-8: opposite squares



PIXAR

Progressive jittered sequences — pj

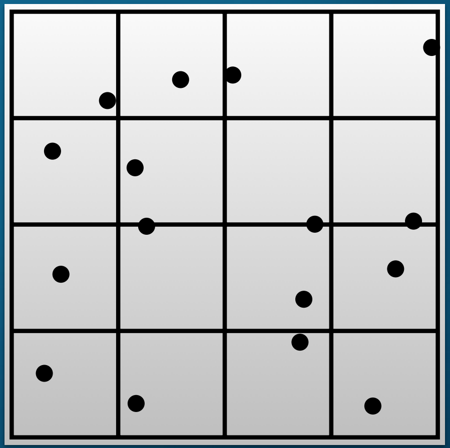
- Samples 9-12: one of remaining squares



PIXAR

Progressive jittered sequences — pj

- Samples 13-16: last remaining squares



PIXAR

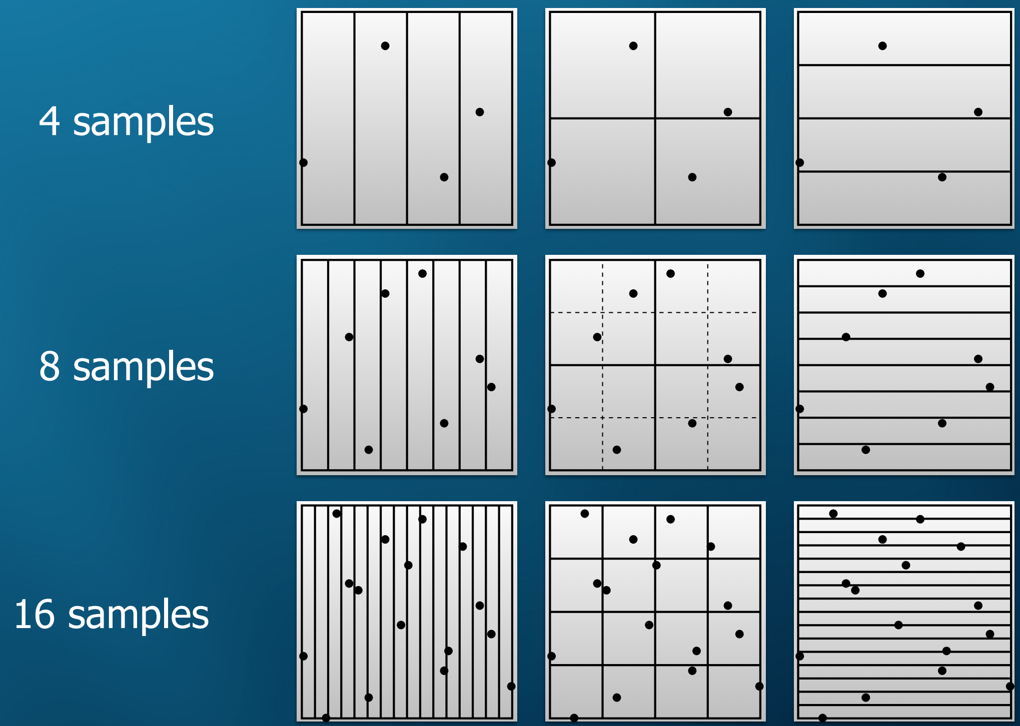
Progressive jittered sequences — pj

- And so on …
- Simple! Similar to [Dippe85,Kajiya86]
- See pseudo-code in supplemental material
- Speed: 170M samples/sec (C++, single core)
  - for comparison: drand48() speed: 73M samples/sec

PIXAR

Progressive multijittered — pmj

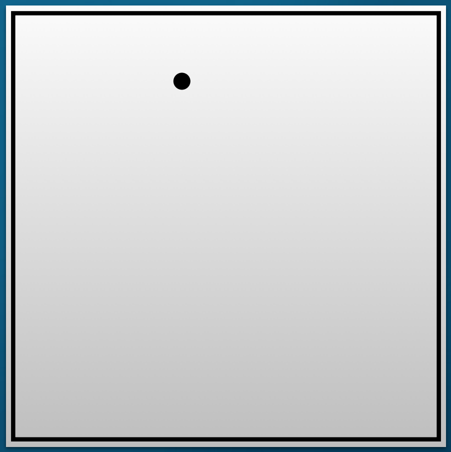
- Stratification goal: squares, rows, and columns



PIXAR

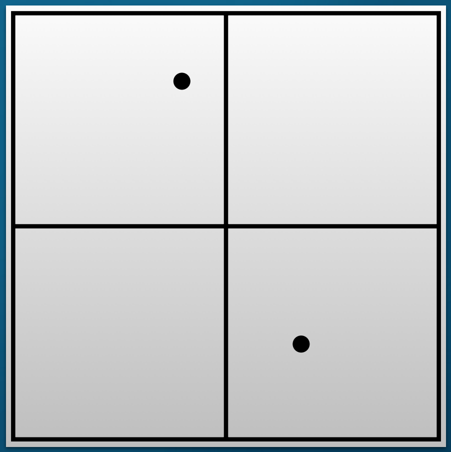
Progressive multijittered — pmj

- Sample 1: random position



Progressive multijittered — pmj

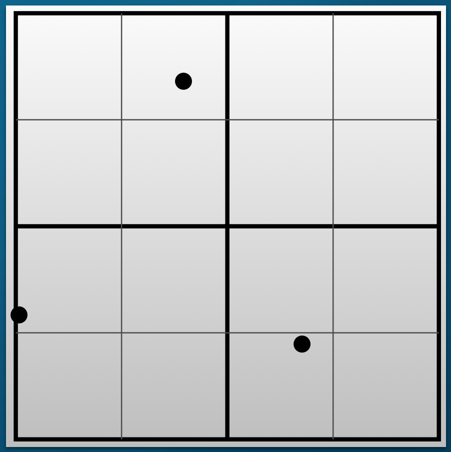
- Sample 2: opposite diagonal



PIXAR

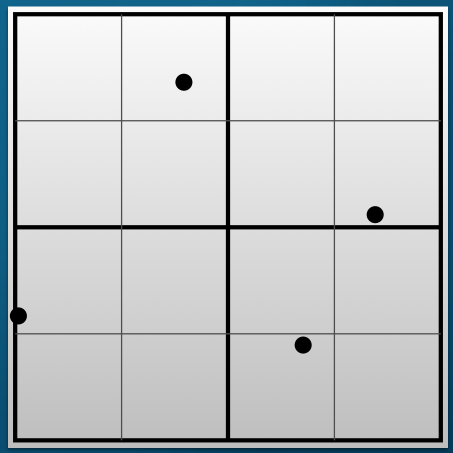
Progressive multijittered — pmj

- Sample 3: one of the two empty squares + empty 1D strips



Progressive multijittered — pmj

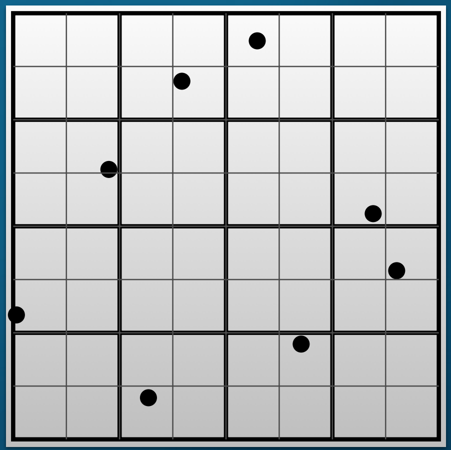
- Sample 4: last remaining square + 1D strips



PIXAR

Progressive multijittered — pmj

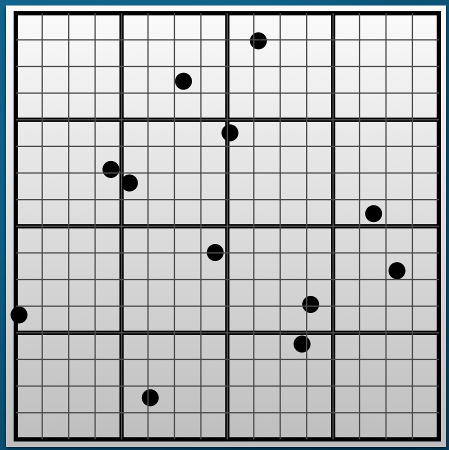
- Samples 5-8: opposite squares (+ empty 1D strips)



PIXAR

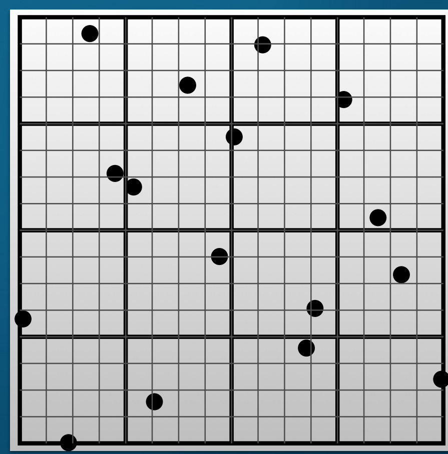
Progressive multijittered — pmj

- Samples 9-12: one of remaining squares (+ empty 1D strips)



Progressive multijittered — pmj

- Samples 13-16: last remaining squares + 1D strips



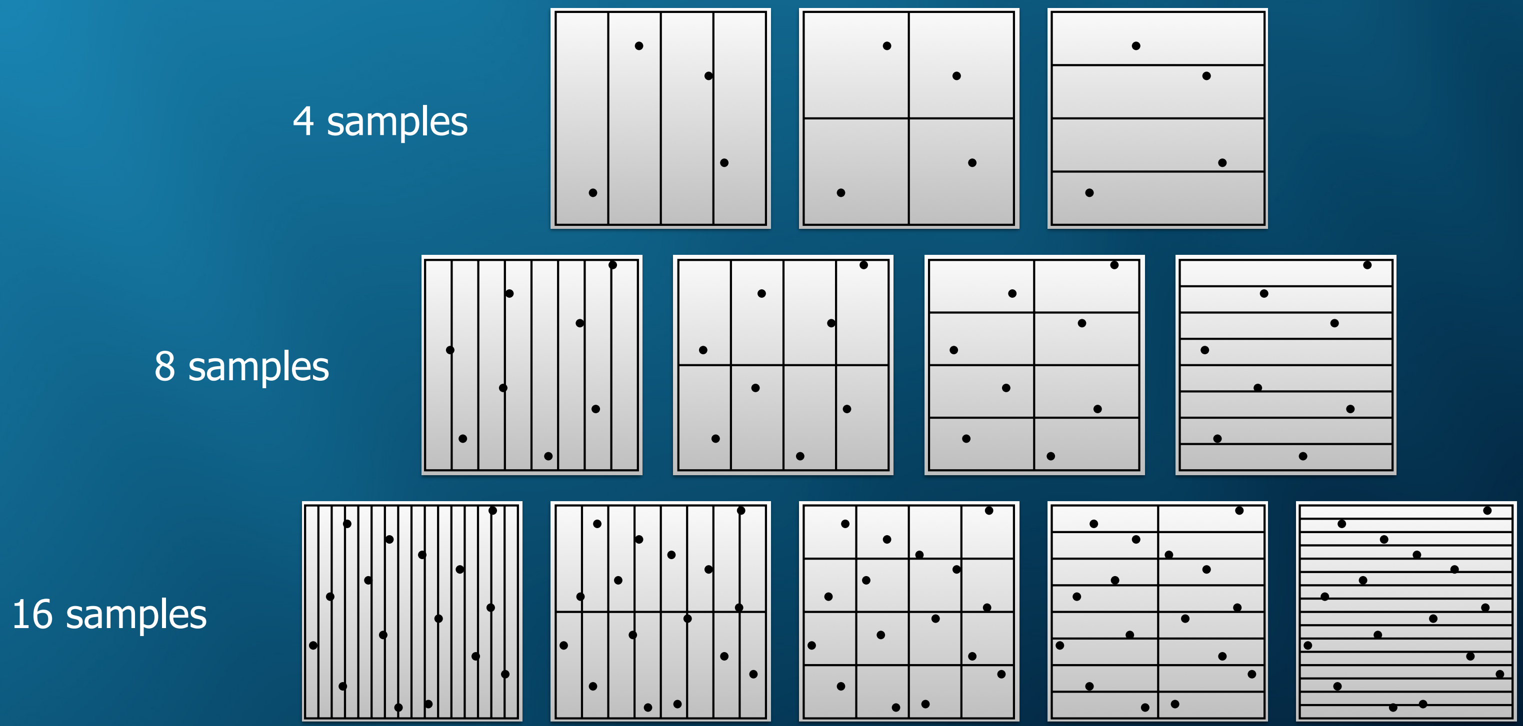
Progressive multijittered — pmj

- And so on …
- See pseudo-code in supplemental material
- Speed: 11M samples/sec
  - for comparison: Owen-scrambled Sobol: 7M samples/sec

PIXAR

Progressive multijittered (0,2): pmj02

- Stratification goal: all base 2 elementary intervals



PIXAR

Progressive multijittered (0,2): pmj02

- Very similar to pmj, but reject samples if in elementary interval stratum that is already occupied
- See pseudo-code for details
- Speed: 39,000 samples/sec
  - too slow during rendering, so pre-generate tables

PIXAR

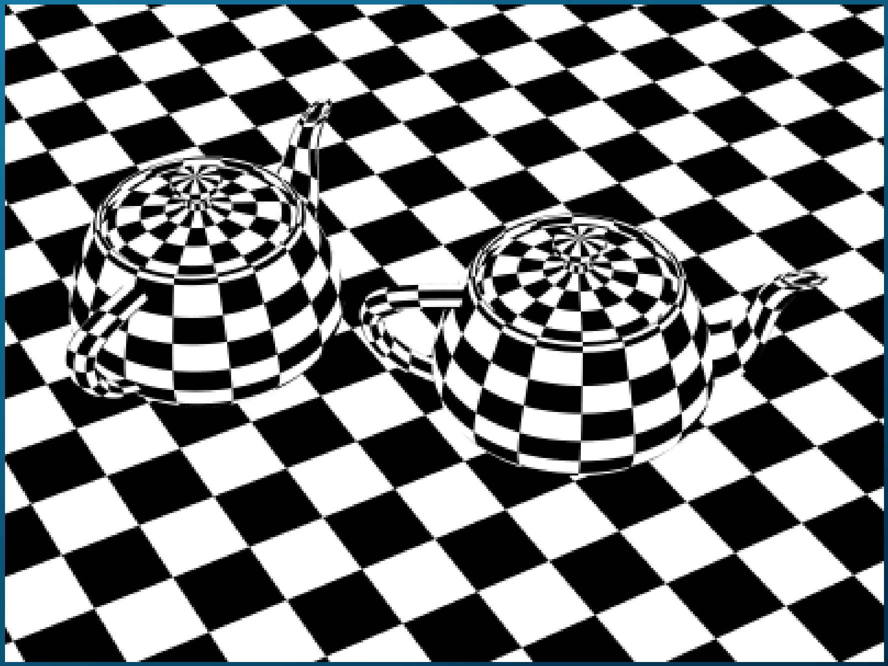
Second comparison of sequences

Pixel sampling

- Each pixel is a "function" that we sample
- Image resolution: 400x300
- Reference images: 500^2 = 250,000 jittered samples / pixel
- Each error curve: average of 100 sequences

PIXAR

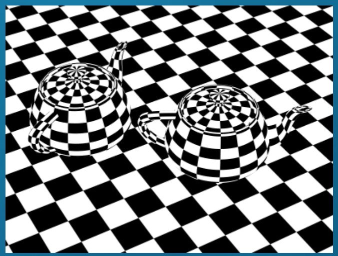
Pixel sampling: checkered teapots

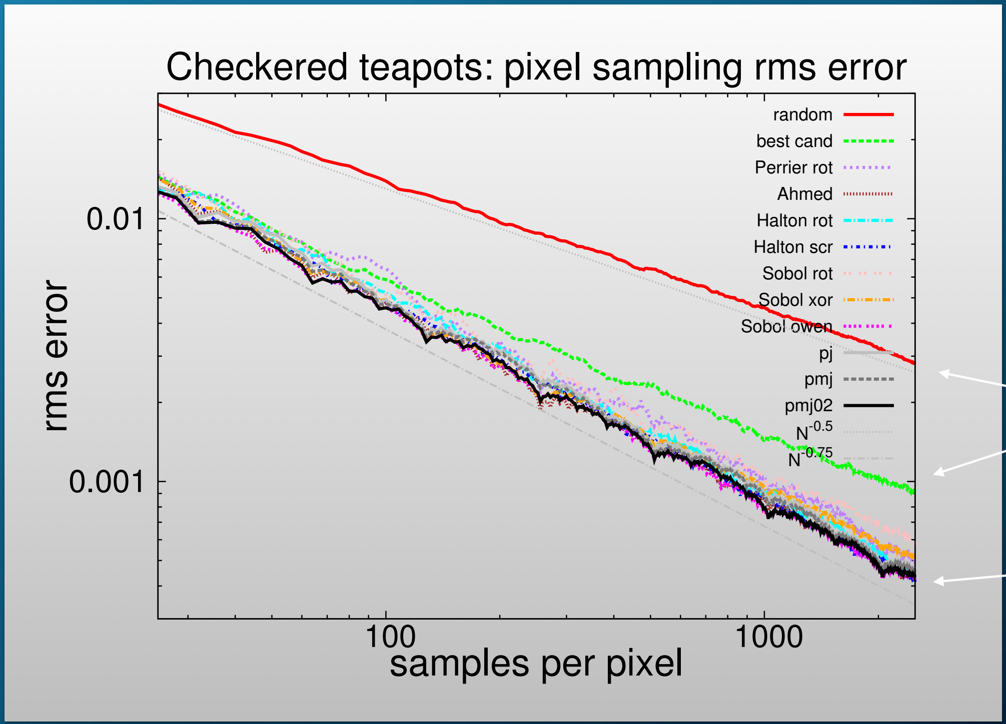


Checkered teapots on checkered ground plane

PIXAR

Pixel sampling: checkered teapots





bad: $O(N^{-0.5})$

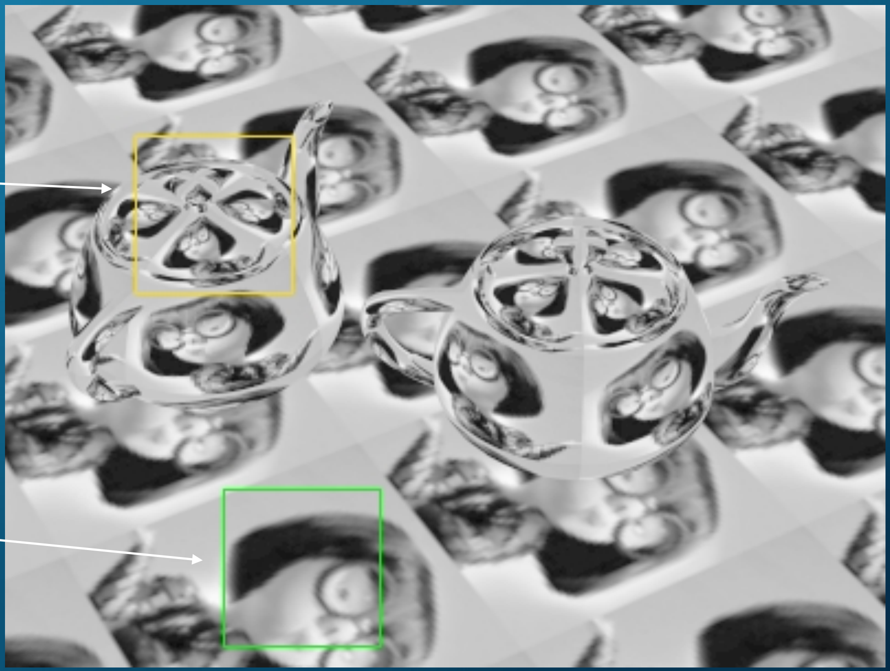
okay: $O(N^{-0.75})$

PIXAR

Pixel sampling: textured teapots

discontinuities due to object edges

smooth (texture filtering)

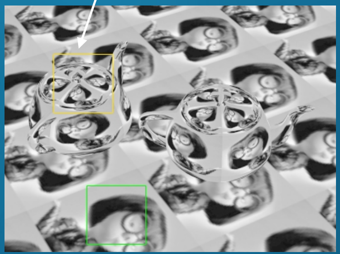


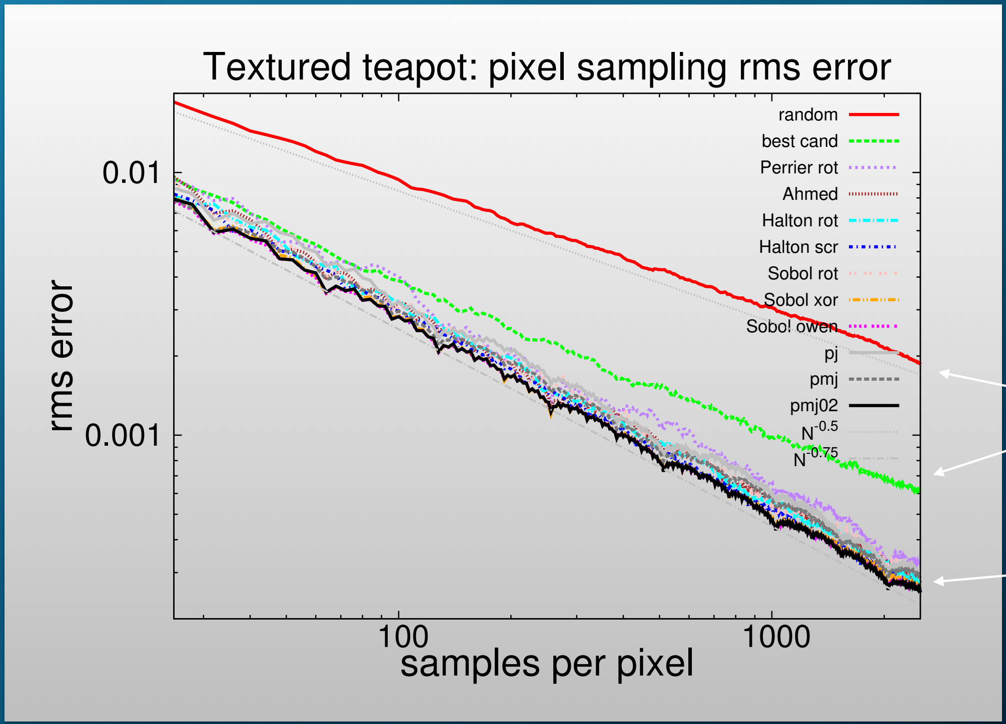
Textured teapots on textured ground plane

PIXAR

Pixel sampling: textured teapots (1)

discontinuous



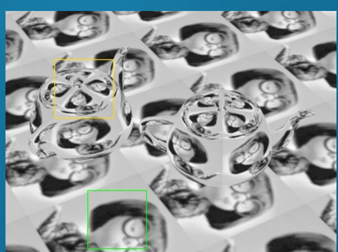


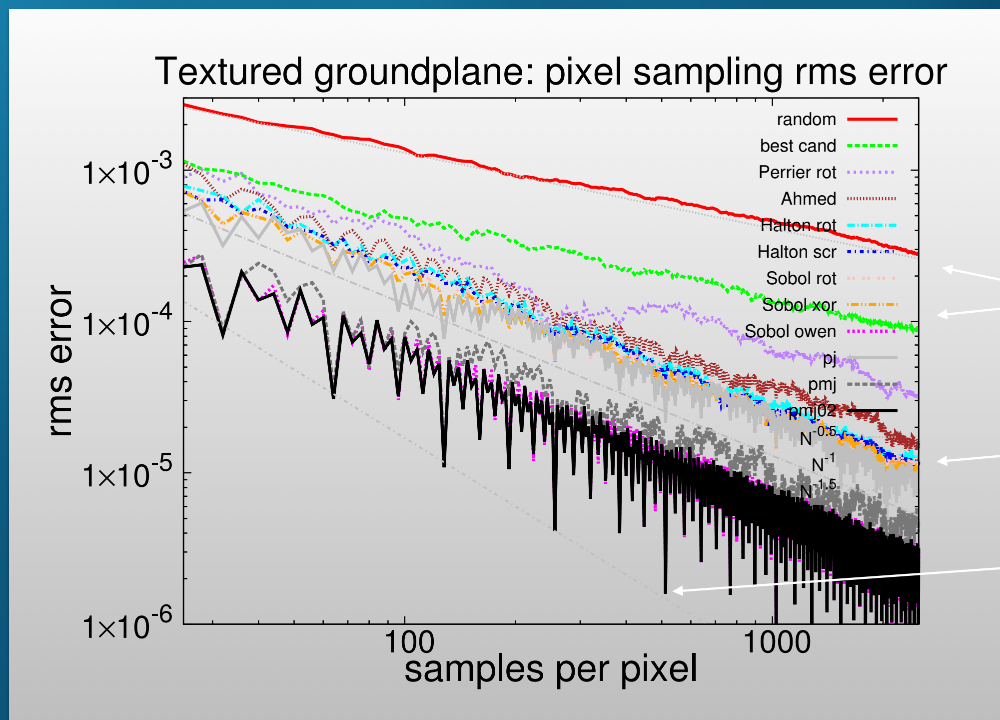
bad: $O(N^{-0.5})$

okay: $O(N^{-0.75})$

PIXAR

Pixel sampling: textured teapots (2)

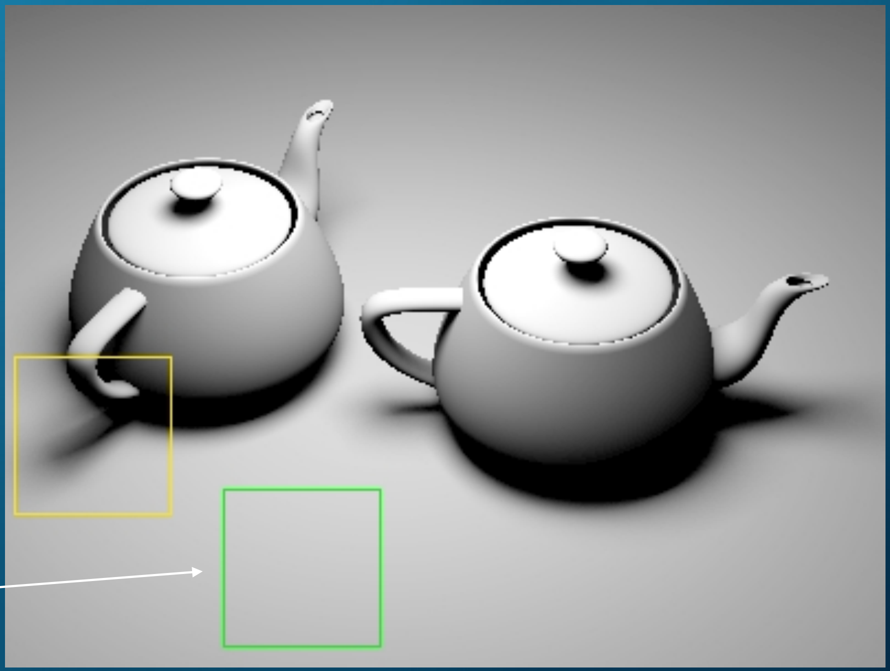




Square area light sampling

penumbra: shadow discontinuities

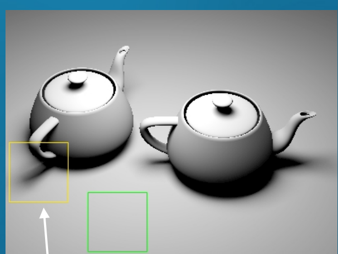
smooth illum



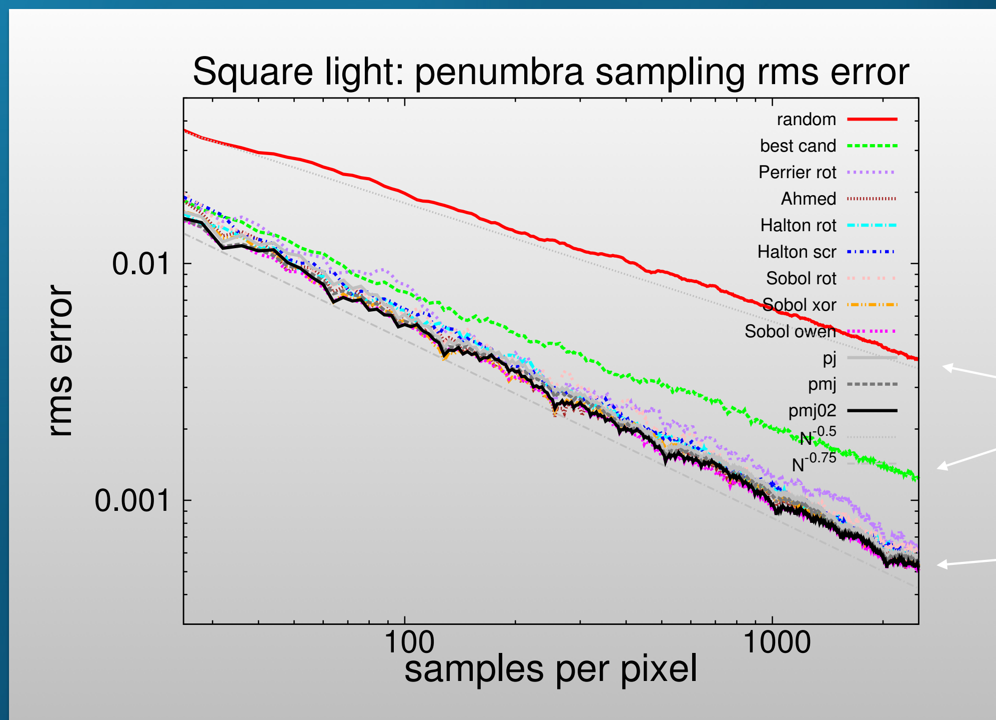
Teapots on ground plane illum by square light source
(no pixel sampling)

PIXAR

Square area light sampling (1)



discontinuous

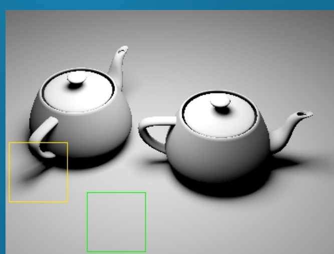


bad: $O(N^{-0.5})$

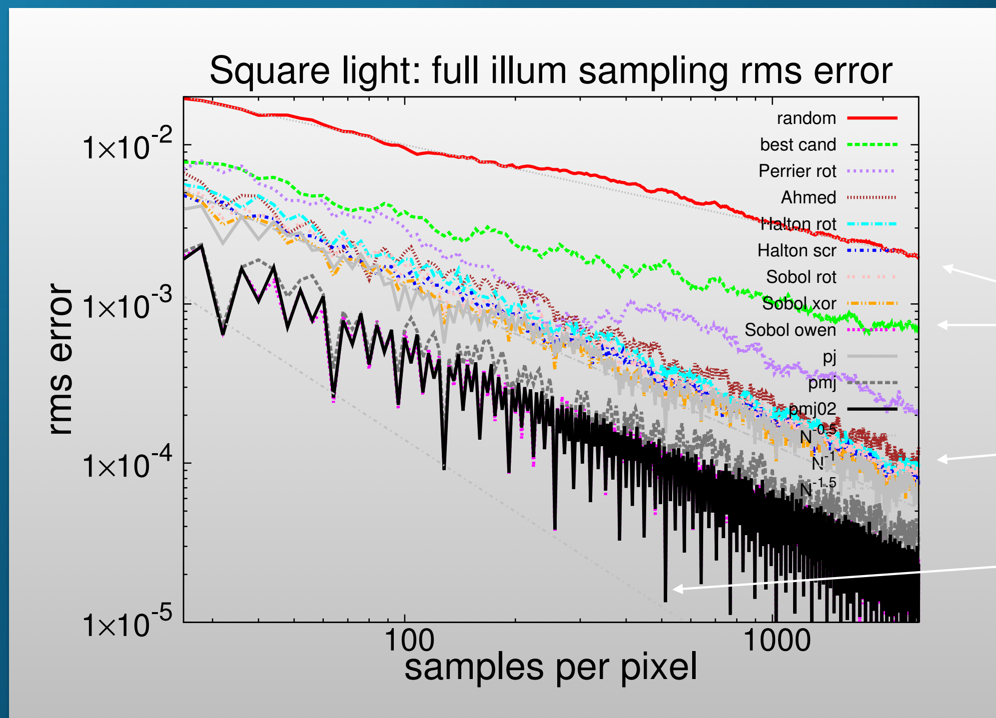
okay: $O(N^{-0.75})$

PIXAR

Square area light sampling (2)



smooth



bad: $O(N^{-0.5})$

good: $O(N^{-1})$

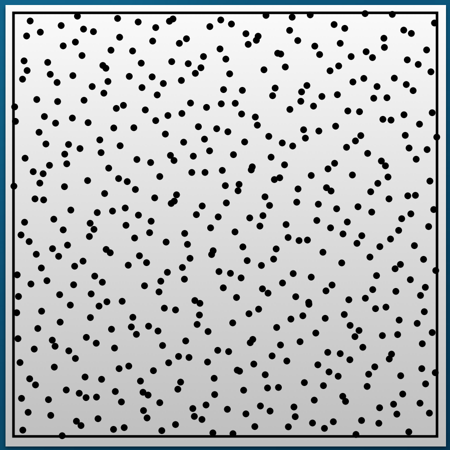
excellent: $O(N^{-1.5})$

Variations and extensions

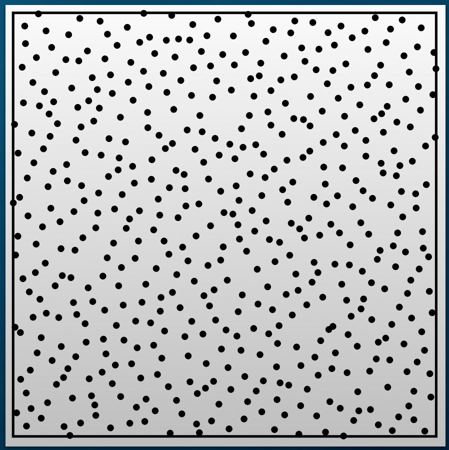
- Status: up until this point we have only shown that pmj02 is as good as Owen-scrambled Sobol
- So what ??
- BUT: within pmj framework we can add blue noise, generate interleaved multi-class samples, …

Pmj with blue noise

- Simple variation: when generating a new pj/pmj/pmj02 sample, generate N candidate points and pick the one that’s most distant from previous samples
- For example:



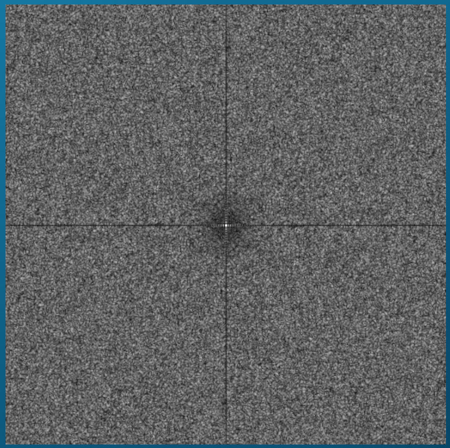
plain pmj



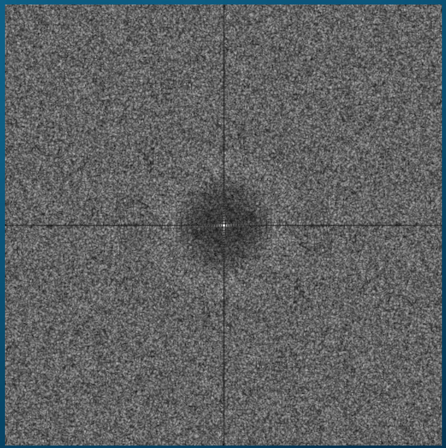
pmj w/ blue noise

PIXAR

Fourier spectra



plain pmj



pmj w/ blue noise

PIXAR

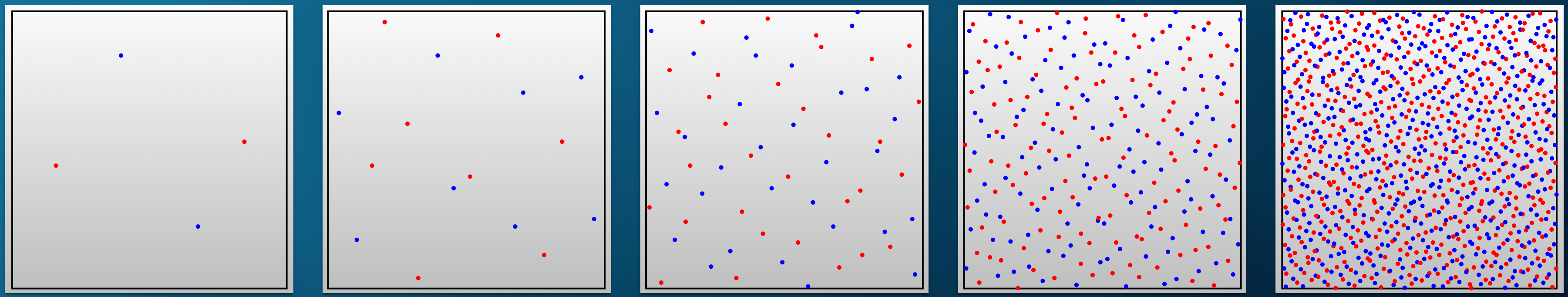
Pmj with blue noise

- Not clear whether blue noise reduces error
- But at least the patterns look more pleasing

PIXAR

Pmj w/ interleaved multiclass samples

- pj/pmj/pmj02 samples can be divided into two classes on the fly. Each class almost as well stratified as full sequence.
- For example:



4 16 64 256 1024

PIXAR

Pmj w/ interleaved multiclass samples

- Two classes can provide two independent estimates for each pixel
- Useful for adaptive sampling (work in progress)

PIXAR

Supplemental material

- Pseudo-code
- More tests: different error metric, Gaussian pixel filter, rectangular area light. (Disk light in separate tech report)
- Comparing sample sets vs sequences (for non-incremental)
- Discussion of discrepancy

PIXAR

Conclusion + future work

- Two contributions: fresh assessment of existing sample sequences, new framework for sample generation
- Error equal to best quasi-random sequence, but allows blue noise, future variations
- Future work: better pmj02 samples, faster generation
- Hopefully even more optimal sample sequences ??

PIXAR

Acknowledgements

- Colleagues in Pixar's RenderMan team
- Brent Burley: Owen scrambling code
- Victor Ostromoukhov, Matt Pharr, Alexander Keller, Christophe Hery, Ryusuke Villemin, Emmanuel Turquin, Andre Mazzone, …

PIXAR

"The generation of random numbers is too important to be left to chance"

— R. Coveyou

PIXAR