PROGRESSIVE MULTI-JITTERED SAMPLE SEQUENCES

Per Christensen
Joint work with Andrew Kensler and Charlie Kilpatrick
Pixar Animation Studios
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. Rethink!

final frame

interactive animation
OVERVIEW

- Survey + evaluation of existing sample sequences
- 3 new algorithms: generate pj, pmj, pmj02 samples
- More evaluations: pixel sampling, area lights
- Extensions: blue noise, multi-class
- Speed-ups by Matt Pharr
- Higher dimensions, better visual quality
SAMPLE PATTERNS: SETS VS SEQUENCES

Sets:
- finite (fixed size)
- no particular order
- need to know how many samples
- no good for incremental rendering, adaptive sampling

Sequences:
- infinite
- every prefix has a good distribution
- no need to know how many samples
- great for incremental rendering, adaptive sampling
SAMPLE PATTERNS: SETS VS SEQUENCES

Incremental rendering: area light sampling

100 samples from **set** with 400
100 samples from **sequence**
(same render time)
SAMPLE SETS

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

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

- Regular grid
- Jitter
- Multijitter
- Correlated multijitter
- Hammersley
- Larcher-Pillichshammer

[Chiu94] [Kensler13]

Quasi-random ("qmc") sets
SAMPLE SEQUENCES

random  blue noise  Halton  Sobol  [Ahmed17]  [Perrier18]

(best candidate/Poisson disk)  quasi-random sequences  blue noise + stratification
SAMPLE SEQUENCES: RANDOMIZED QUASI-RANDOM

- Halton rot
- Halton scr
- Sobol rot
- Sobol xor scr
- Sobol Owen scr

Cranley-Patterson rotations [Cranley76]
bit-wise exclusive-or [Kollig02]

[Owen97]
FIRST COMPARISON OF 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, …
INITIAL TESTS OF 2D SEQUENCES

- Sample simple discontinuous and smooth functions on $[0,1)^2$
- Known analytical reference values
INITIAL TESTS: DISCONTINUOUS FUNCTIONS

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

reference value: 0.5
INITIAL TESTS: DISCONTINUOUS FUNCTIONS

Disk function: sampling error

bad: $O(N^{-0.5})$
INITIAL TESTS: DISCONTINUOUS FUNCTIONS

Disk function: sampling error

bad: $O(N^{-0.5})$
INITIAL TESTS: DISCONTINUOUS FUNCTIONS

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

okay: $O(N^{-0.75})$
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.
INITIAL TESTS: SMOOTH FUNCTIONS

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

reference value: \( \sim 0.557746 \)
INITIAL TESTS: SMOOTH FUNCTIONS

![Graph showing Gaussian function sampling error with error on the y-axis and samples on the x-axis. The graph includes lines for random, best cand, and \( N^{-0.5} \). The legend indicates that the graph is labeled "Gaussian function: sampling error," and the error scales are shown from \( 1 \times 10^{-5} \) to \( 1 \times 10^{-1} \). The notation \( O(N^{-0.5}) \) is used to denote the error growth.}]
INITIAL TESTS: SMOOTH FUNCTIONS

Gaussian function: sampling error

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

good: $O(N^{-1})$
INITIAL TESTS: SMOOTH FUNCTIONS

Gaussian function: sampling error

- **Bad:** $O(N^{-0.5})$
- **Good:** $O(N^{-1})$
- **Excellent:** $O(N^{-1.5})$
INITIAL TESTS: SMOOTH FUNCTIONS

Bilinear function \( f(x,y) = xy \): similar results

Reference value: 0.25
SUMMARY OF INITIAL TESTS

Owen-scrambled Sobol is best:

• no pathological error for discontinuities at certain angles
• extraordinarily fast convergence for smooth functions
PROGRESSIVE (MULTI)JITTERED SEQUENCES
PROGRESSIVE (MULTI)JITTERING

Framework for stochastic sample generation
Three simple algorithms that progressively fill in holes in increasingly fine stratifications
Build on jittered [Cook84] and multijittered [Chiu94] sample sets — but sequences
PROGRESSIVE JITTERED SEQUENCES

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

2x2 → 4x4
PROGRESSIVE JITTERED SEQUENCES

Sample 1: random position
PROGRESSIVE JITTERED SEQUENCES

Sample 2: opposite diagonal square
PROGRESSIVE JITTERED SEQUENCES

Sample 3: one of the empty squares
PROGRESSIVE JITTERED SEQUENCES

Sample 4: last empty square
PROGRESSIVE JITTERED SEQUENCES

Samples 5-8: opposite squares
PROGRESSIVE JITTERED SEQUENCES

Samples 9-12: one of remaining squares
PROGRESSIVE JITTERED SEQUENCES

Samples 13-16: last remaining squares
PROGRESSIVE JITTERED SEQUENCES

And so on …
Simple! Similar to [Dippe85,Kajiya86]
See pseudocode in EGSR 2018 paper
PROGRESSIVE MULTI-JITTERED — PMJ

Stratification goal: squares, rows, and columns

- 4 samples
- 8 samples
- 16 samples
PROGRESSIVE MULTIJITTERED — PMJ

Sample 1: random position
PROGRESSIVE MULTIJITTERED — PMJ

Sample 2: opposite diagonal square
PROGRESSIVE MULTIJITTERED — PMJ

Sample 3: one of the empty squares + empty 1D strips
PROGRESSIVE MULTIJITTERED — PMJ

Sample 4: remaining square + 1D strips
PROGRESSIVE MULTIJITTERED — PMJ

Samples 5-8: opposite squares + empty 1D strips
PROGRESSIVE MULTIJITTERED — PMJ

Samples 9-12: one of remaining squares + empty 1D strips
PROGRESSIVE MULTIJIITERRED — PMJ

Samples 13-16: last remaining squares + empty 1D strips
And so on …
Similar to multijittered sets [Chiu94], but for sequences
Pseudocode in EGSR 2018 paper
PROGRESSIVE MULTIJITTERED (0,2) — PMJ02

Stratification goal: all base-2 elementary intervals

4 samples

8 samples

16 samples
PROGRESSIVE MULTIJITTERED (0,2) — PMJ02

- Very similar to pmj, but reject samples if in elementary interval stratum that is already occupied
- See pseudo-code in EGSR 2018 paper for details
- Speed: 39,000 samples/sec (1 CPU thread)
  - too slow during rendering, so pre-generate tables
SECOND COMPARISON OF SEQUENCES
PIXEL SAMPLING

- Each pixel is a “function” we sample
- Image resolution: 400x300
- Reference images: \(500^2 = 250,000\) jittered samples/pixel
- Each error curve: average of 100 sequences
PIXEL SAMPLING: CHECKERED TEAPOTS

cHECKERED TEAPOTS ON CHECKERED GROUND PLANE
PIXEL SAMPLING: CHECKERED TEAPOTS

![Checkered teapots](image)

**Checkered teapots: pixel sampling rms error**

- random
- best cand
- Perrier rot
- Ahmed
- Halton rot
- Halton scr
- Sobol rot
- Sobol xor
- Sobol Owen

- pj
- pmj
- pmj02
- $N^{0.5}$
- $N^{0.75}$

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

okay: $O(N^{-0.75})$
PIXEL SAMPLING: TEXTURED TEAPOTS

discontinuities due to object edges

smooth (texture filtering)

textured teapots on textured ground plane
PIXEL SAMPLING: TEXTURED TEAPOTS (1)

Textured teapot: pixel sampling rms error

- discontinuous
- bad: \(O(N^{-0.5})\)
- okay: \(O(N^{-0.75})\)
PIXEL SAMPLING: TEXTURED TEAPOTS (2)

Textured groundplane: pixel sampling rms error

- bad: $O(N^{-0.5})$
- good: $O(N^{-1})$
- excellent: $O(N^{-1.5})$
SQUARE AREA LIGHT SAMPLING

penumbra: shadow discontinuities

smooth illum

teapots on ground plane illum by square light source (no pixel sampling)
SQUARE AREA LIGHT SAMPLING (1)

Square light: penumbra sampling rms error

bad: $O(N^{-0.5})$
okay: $O(N^{-0.75})$
SQUARE AREA LIGHT SAMPLING (2)

Square light: full illum sampling rms error

- bad: $O(N^{-0.5})$
- good: $O(N^{-1})$
- excellent: $O(N^{-1.5})$
VARIATIONS AND EXTENSIONS
VARIATIONS AND EXTENSIONS

- Status: up until this point, we’ve only shown that pmj02 samples are as good as Owen-scrambled Sobol
- So what ??
- BUT: within pmj framework we can add blue noise, generate 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:
FOURIER SPECTRA

plain pmj

pmj w/ blue noise
PMJ WITH BLUE NOISE

Not clear whether blue noise reduces error?
But at least the patterns look more pleasing
PMJ WITH INTERLEAVED MULTICLASS SAMPLES

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

- Two classes can provide two independent estimates for each pixel
- Can be useful for adaptive sampling
FASTER SAMPLE GENERATION [PHARR19]

Faster sample generation by better data structure -- keeping track of unoccupied elementary intervals

Reference: Matt Pharr, “Efficient generation of points that satisfy two-dimensional elementary intervals”, JCGT 2019

Speed: 333,000 points / sec (1 CPU thread)
HIGHER DIMENSIONS: 3D, 4D, 5D, …

- For depth-of-field (DOF), motion blur (MB)
- DOF: need 2D samples for pixel pos + 2D for lens pos
- If we just use two pmj02 sequences: correlation
- Better: randomly shuffle sample order of one of the 2D sequences (similar to [Cook84] for sample sets). Avoids correlation
- Even better: carefully shuffle sample order such that 2D+2D points are stratified in 4D. Implementation: swap order of two points and check if that improves 4D stratification; stop when fully stratified.
- MB: similar for 2D pixel pos + 1D time samples
- Combined: 2D+2D+1D table
Better placement of 1st sample/pixel: fully stratify in 4x4 pixel blocks. Similar in spirit to [Georgiev16] “Blue noise dithered sampling”

New, better technique:

• Heitz et al, “A low-discrepancy sampler that distributes Monte Carlo errors as blue noise in screen space” -- this afternoon!
• shuffles and xor-scrambles Sobol samples to improve visual quality for all samples
• we could/should do that with pmj02 samples, too!
CONCLUSION + FUTURE WORK

Two main contributions:
- fresh assessment of existing sample sequences
- framework for stochastic progressive sample generation

Error equal to best quasi-random sequence, but allows blue noise, multiclass, future variations

More info: EGSR 2018 paper + supplemental material

Future: hopefully even more optimal sample sequences?
A “FREEBIE”: FUNCSAMP2D PROGRAM

- C++ program to integrate 2D functions with various sample sequences
- For comparison of error and convergence rates of sequences
- Polished version of program I used for plots in this talk
- Different function classes: discontinuous, continuous, smooth, …
- Available at GitHub: github.com/perchristensen/funcsamp2D
- Feel free to extend it: more functions, higher dimensions, …
ACKNOWLEDGEMENTS

- Alexander Keller for organizing this course
- Colleagues in Pixar’s RenderMan team
- Brent Burley @ Disney: efficient Owen scrambling code
- Matt Pharr @ Nvidea: much faster implementation
- Victor Ostromoukhov, Christophe Hery, Ryusuke Vilemin, Emmanuel Turquin, Andre Mazzone, …
“The generation of random samples is too important to be left to chance”

— R. Coveyou
Thank you!