LOW-DISCREPANCY BLUE NOISE SAMPLING

Abdalla G. M. Ahmed
King Abdullah University of Science and Technology

Co-authored with
Hélène Perrier, David Coeurjolly, Victor Ostromoukhov, Jianwei Guo, Dongming Yan, Hui Huang, and Oliver Deussen
OVERVIEW

- Motivation and Concept
  - Monte Carlo Integration
  - Discrepancy
  - Blue Noise vs Low Discrepancy
  - Combining the two properties
  - Target-Matching Algorithm

- Low-Discrepancy Blue Noise Sampler
  - Performance
  - Limitations
MONTE CARLO INTEGRATION

- We want to compute the area of the circle given the area of the box.
- In practice, the box could be a light source, and the circle is an object that hides (occludes) it.
- We randomly generate sample points in the box.
- The percentage of the points inside the circle estimates the area of the circle relative to the box.
Images are synthesized from samples
Rays are cast through image samples and traced to the scene
End-points of rays are traced to sample points on light sources to compute shadows
Millions of samples, efficiency is crucial!
- Start discrepancy: rectangles anchored to origin.
- Star discrepancy is easier to compute, and can be associated with points, hence visualized as a map.
White noise has excessively-large discrepancy, hence produces too much noise.
- Regular grid has a regular pattern of discrepancy, leading to aliased manifestation of error; e.g. bands of shadow.
- We could do better than random: use evenly distributed point sets.
- Blue noise: even density, but otherwise random.
- Low-discrepancy: mathematically-computed, optimized for LD.
- Which sampling pattern would you prefer?
- Even in this room, some people might argue for blue noise, and others argue for low discrepancy.
- I may argue for BN for low sampling rates, and LD for high rates.
But we no longer have to ask this question, because we managed to have BN and LD in the same point set?
- We may proceed by choosing a LD set/sequence, the deterministic side, and optimize it for BN.
- Existing LD constructions are topologically unstable: different neighbors for the same sample point, depending on the taken number of samples.
To combine blue noise and low discrepancy patterns we bring them to a common ground using stratification; that is, we start with a stratified LD and stratified blue noise sets, and try to align their points in each stratum.
- We solved the topology problem with existing LD constructions by inventing a new LD construction.
- Consider a stratified point set like this: a regular lattice of cells, each cell hosting a single point.
- Look at the sequence of horizontal offsets over each and every column, and vertical offsets over each and every row.
- We proved that if all these sequences are LD sequences, then the point set is a LD point set.
- Use LD sequences to supply the offsets. This one uses van der Corput sequence
- Same neighbors for same index.
- Infinite point set, slice and scale as needed!
- We can’t just move the points, we have to preserve the LD sequence property along rows and columns.
- Optimization is possible via rearrangement.
- Reordering small chunks of entries in a LD sequence has little impact on discrepancy.
AXIS-WISE 2D REARRANGEMENT DEMO
- Take a stratified LD set as a reference.
- Reorder coordinates in small blocks of LD set to match the order of coordinates in the reference.
- Store the new order.
- Efficient storage: 1 byte per point.
LDBN SAMPLER
- This is the discrepancy order for point sets, less than sequences.
- Other profiles might be possible, as long as they can stratified.
- Fastest ever! For comparison, AA~Patterns ~100 MPPS, Polyhexes ~10 MPPS.
- Very small, superseded only by penrose tilings. AA~Patterns 64KB
DISCREPANCY
- The spectrum is almost identical (provable).
- The crosshair characterizes LD and Latinized point sets.
IMAGE SAMPLING (ZONE PLATE)

\[ \sin(x^2 + y^2) \]

- 1 Sample per pixel
- Mitchell Filter

Ground Truth

Sobol

BNOT

Scrambled Sobol

LDBN
void generate(int n) {
    double inv = 1.0 / n;
    unsigned mask = t - 1;
    unsigned shift = log2(t);
    int i = 0;
    for (unsigned Y = 0; Y < n; Y++) {
        for (unsigned X = 0; X < n; X++) {
            unsigned index = ((Y & mask) << shift) + (X & mask);
            double x = phi( (Y & 0xffffff0) + (lut[index] & 0xf) );
            double y = phi( (X & 0xffffff0) + (lut[index] >> 4) );
            s[i].x = inv * (X + x);
            s[i++].y = inv * (Y + y);
        }
    }
}
- Website is mirrored in University of Konstanz and University of Lyon.
- You’ll find very rigorous comparisons made by my colleague Helene.
LIMITATIONS

- No adaptivity
- Only 2D
- Reference point set has to be stratified
Thanks