Cache-Friendly Micro-Jittered Sampling

Arthur Dufay* † ‡, Pascal Lecocq* †, Romain Pacanowski ‡, Jean-Eudes Marvie †, Xavier Granier ‡

† Technicolor Research & Innovation ‡ LP2N (UMR 5298: IOGS - Univ. Bordeaux - CNRS) - Inria, F33400 Talence

1 Motivation

Monte-Carlo integration techniques for global illumination are popular on GPUs thanks to their massive parallel architecture, but efficient implementation remains challenging. The use of randomly decorrelated low-discrepancy sequences in the path-tracing algorithm allows faster visual convergence. However, the parallel tracing of incoherent rays often results in poor memory cache utilization, reducing the ray bandwidth efficiency. Interleaved sampling [Keller et al. 2001] partially solves this problem, by using a small set of distributions split in coherent ray-tracing passes, but the solution is prone to structured noise. On the other hand, ray-reordering methods [Pharr et al. 1997] group stochastic rays into coherent ray packets but their implementation add an additional sorting cost on the GPU [Moon et al. 2010] [Garanzha and Loop 2010].

We introduce a micro-jittering technique for faster multi-dimensional Monte-Carlo integration in ray-based rendering engines. Our method improves ray coherency between GPU threads using a slightly altered low-discrepancy sequence rather than using ray-reordering methods. Compatible with any low-discrepancy sequence and independent of the importance sampling strategy, our method achieves comparable visual quality with classic decorrelation methods, like Cranley-Patterson rotation [Kollig and Keller 2002], while reducing rendering times in all scenarios.

Keywords: path-tracing, ray coherence, sampling, jittering

Concepts: Computing methodologies → Ray tracing;

2 Micro-jittered sampling

Our approach leverages the coherency naturally present in 3D scenes with constrained Cranley-Patterson rotation. Cranley-Patterson rotation decorrelates samples by adding a random offset in each dimension of a single random sequence. However, the parallel evaluation of the i-th ray sample within a block of threads often results in random access to 3D space regions, causing memory cache misses.

Core idea. Because close pixels have higher probability to share the same visible surface characteristics, parallel tracing of nearly similar rays on first bounces tends to follow similar paths in space, increasing thread data coherency. Based on this simple observation, our micro-jittering strategy consists in restricting Cranley-Patterson rotation to micro-jitters within a small volume of size \( \mu \) using a signed random uniform distribution \( \xi \) (eq. 1).

\[
y_i = x_i + \mu_{(s eq, N)} \xi, \quad \xi \in [-0.5, 0.5]
\]

To prevent the apparition of structured noise, or any bias, \( \mu \) is adaptively chosen to ensure that the de-correlated sequences uniformly draw samples within the unit hypercube (see Figure 2). This depends on a constant \( K \) representative of the star-discrepancy of the random sequence, and \( N \), the sampling count.

\[
\mu_{(s eq, N)} = K D_{st}^* N^{-1/2}
\]

As the result, each indexed sample in each de-correlated sequence orbits around its original position, increasing the probability to address the same region of space while preserving (mostly) the discrepancy property of the sequence.

Our method shares some similarities with the uniform jitter sampling method described by [Ramamoorthi et al. 2012] and devised for area-light visibility computations. However our approach is somewhat different. We focus on performance rather than visibility and preserve the noise property by micro-jitter random sequences in a domain based on their discrepancy characteristics.
Figure 2: Illustration of the micro-jittered approach: 128 samples (blue dots) of a two-dimensional Halton sequence (a) de-correlated by micro-jittering $1 \times$ (red dots) and $150 \times$ (c,d) within a small area of size $\mu$ (b). Setting $\mu$ too small may result in a bad coverage of the de-correlated integration domain (c). To prevent any bias or structured noise, we adaptively choose $\mu$ according to the sampling count and the star-discrepancy of the sequence to evenly cover the domain.

3 Results

We implemented the micro-jittering approach in our in-house GPU path-tracer and in a modified version of Blender Cycles (GPU and CPU). We compared our approach against Cranley-Patterson rotation method in terms of visual quality and rendering performances. Comparisons were done on 3D scenes of various complexity, several low-discrepancy sequences and different sampling count. In all scenarios, our method delivers superior rendering performances while providing comparable visual quality. As the sampling rate increases, our method performs faster, converging to coherent path-tracing performances without its inherent flaws. In some scenarios, our method can achieve speed-up factors up to $2 \times$. We also successfully applied our method to screen-space rendering techniques. In a ray-marched SSAO implementation, our micro-jittering approach also improves rendering times without altering the visual quality.

Our method is simple, easy to implement and can drastically reduce rendering times for high-demanding stochastic ray-tracing applications with random memory accesses.

4 Acknowledgments

We would like to thank Gaël Sourimant for its implementation and experimentations of the micro-jittering method in a SSAO context. We also thank the anonymous reviewers for their valuable comments and feedback.

Scene credits: the San-Miguel scene in teaser-left was modeled by Guillermo M. Leal Llaguno. The London NH museum in middle was modeled by Alvaro Luna Bautista and Joel Andersdon. The interior 3D model from figure 3 comes from Chocofur store: store.chocofur.com.

References


