Vol. 14, No. 2: 33–43

Coherent Path Tracing

Iman Sadeghi, Bin Chen, and Henrik Wann Jensen
University of California, San Diego

Abstract. Packet tracing is a popular and efficient method for accelerating ray tracing. However, packet traversal techniques become inefficient when they are applied to path tracing since the secondary rays are incoherent. In this paper, we present a simple technique for improving the coherency of secondary rays. This technique uses the same sequence of random numbers for generating secondary rays for all the pixels in each sample. This improves the efficiency of the packet tracing algorithm but creates structured noise patterns in the image. We propose an interleaved sampling scheme that reduces the correlation in the noise and makes it virtually imperceptible in the final image. Coherent path tracing is unbiased, simple to implement, and outperforms standard path tracing with packet tracing, while producing images with similar RMS error values.

1. Introduction

Ray tracing is a widely used algorithm for realistic image synthesis. It has traditionally been used for offline rendering, but recent developments in hardware and algorithms have enabled interactive ray tracing [Parker et al. 99, Wald et al. 01, Lauterbach et al. 06, Wald et al. 07]. One important algorithmic improvement to ray tracing is the use of *ray packets* [Wald et al. 01]. Ray packets enable the use of SIMD hardware by tracing several rays with similar origins and directions through the acceleration data structure at the same time. In addition to allowing the use of SIMD instructions, the use of ray packets also reduces the traversal cost and memory bandwidth.

1086-7651/09 $0.50 per page

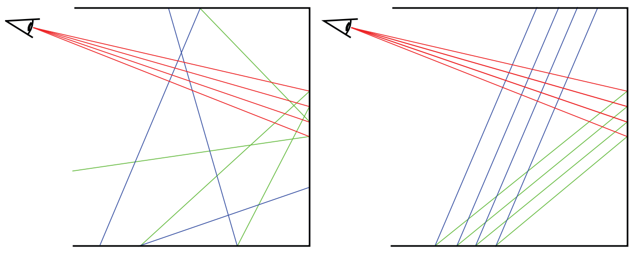

Figure 1. Primary, first bounce, and second bounce rays are drawn red, green, and blue, respectively. In standard path tracing (left), secondary bounces are generated randomly and lack coherency. In coherent path tracing (right), secondary rays are generated with the same sequence of random numbers and remain coherent, as long as they intersect geometries with similar normal directions.

Ray packets require similar rays (coherent rays) to operate efficiently. They work well for nearby primary rays, since these rays often traverse similar parts of the acceleration data structure. It is also possible to generate coherent packets of shadow rays, and combined with instant radiosity [Keller 97], it is even possible to simulate global illumination at interactive rates [Wald et al. 02].

Unfortunately, it is not yet possible to use ray packets effectively with general Monte Carlo ray tracing algorithms such as path tracing [Kajiya 86]. The primary reason is that the rays in path tracing become randomized (incoherent) after the first few diffuse or glossy reflections [Reshetov 06].

In this paper we present a simple technique for improving the performance of ray packets in the context of path tracing. We force secondary rays to become coherent by using the same sequence of random numbers for all rays in a packet. This causes the reflected rays to use similar sample directions even if they originate at a diffuse surface (see Figure 1). The naive implementation of this approach suffers from coherent noise patterns, which can be undesirable if the goal is to visualize intermediate results. To alleviate this problem we propose the use of an interleaved sampling pattern that effectively scrambles the coherent random sequences across the image plane. This reduces the correlation in the noise patterns at the cost of a small loss in coherence in the traced ray packets.

2. Algorithm

2.1. Standard Path Tracing (SPT)

In path tracing, ray paths are traced through each pixel. Rays going through pixels are called primary rays, while rays resulting from the intersection of a ray with an object are called secondary rays. The directions of secondary rays are determined by the BRDF at the intersection point and are generated through random sampling. We can think of random sampling as using a sequence of random numbers $\vec{u} = (u_1, \ldots, u_n)$ per path. Path tracing computes the radiance, $L(x)$, of each pixel using the following integral of a multidimensional function f:

$$L(x) = \int \ldots \int f(x, u_1, \ldots, u_n)\, du_1 \ldots\, du_n = \int f(x, \vec{u})\, d\vec{u}, \tag{1}$$

where $f(x, u_1, \ldots, u_n) = f(x, \vec{u})$ is the contribution of each sample to $L(x)$ using $\vec{u}$ as the random numbers. Path tracing integrates f by using Monte Carlo sampling to compute $L(x_i)$ for each pixel x_i:

$$L(x_i) = \frac{1}{N} \sum_{k=1}^{N} f(x_i, \vec{u_{ik}}), \tag{2}$$

where $\vec{u_{ik}}$ is a vector of random numbers for the ith pixel and the kth sample from a total of N samples.

Path tracing can utilize packet tracing by placing a number of neighboring rays into a ray packet and tracing ray packets into the scene. We refer to this method as *standard path tracing (SPT).*

2.2. Coherent Path Tracing (CPT)

The *coherent path tracing (CPT)* algorithm uses the same vector $\vec{u_k}$ of random numbers for each pixel x_i and each sample k:

$$L(x_i) = \frac{1}{N} \sum_{k=1}^{N} f(x_i, \vec{u_k}), \tag{3}$$

Since the shared random number sequence $\vec{u_k}$ has a uniform random distribution, CPT will produce the correct and unbiased result for any single pixel similar to the SPT method. It uses the method of dependent tests to evaluate the per-pixel integrals, which increases the coherency of secondary rays and thereby the efficiency of packet tracing.

This efficiency gain, however, comes at the cost of structured noise (see first row of Figure 2). This type of noise is more noticeable than the random noise of SPT (third row of Figure 2). It is not an issue in terms of the quality of the final converged image, but if intermediate results are used for visualization then this structured noise is undesirable.

2.3. Interleaved Coherent Path Tracing (ICPT)

To improve the visual quality of the rendered results, we propose a randomized interleaved sampling scheme similar to [Keller and Heidrich 01], which we call *interleaved coherent path tracing (ICPT)* (see second row of Figure 2).

If the interleaving patterns are not randomized (like the one presented in [Keller and Heidrich 01]) then the rendered results will suffer from structured noise patterns. To address this, we divide the image into blocks of 4×4 pixels and partition each block into 4 subsets. For each subset, we use the same sequence of random numbers. This results in four different random sequences per 8×8 blocks of pixels, and therefore four coherent ray packets of 4×4 pixels (see Figure 3).

For partitioning, we use Latin Hypercube sampling [McKay et al. 79], also known as N-rooks sampling [Shirley 91], which places a unique sample sequence in each row and column as shown in Figure 3. We randomize the partitioning patterns for every sample and for every block of 4×4 pixels.

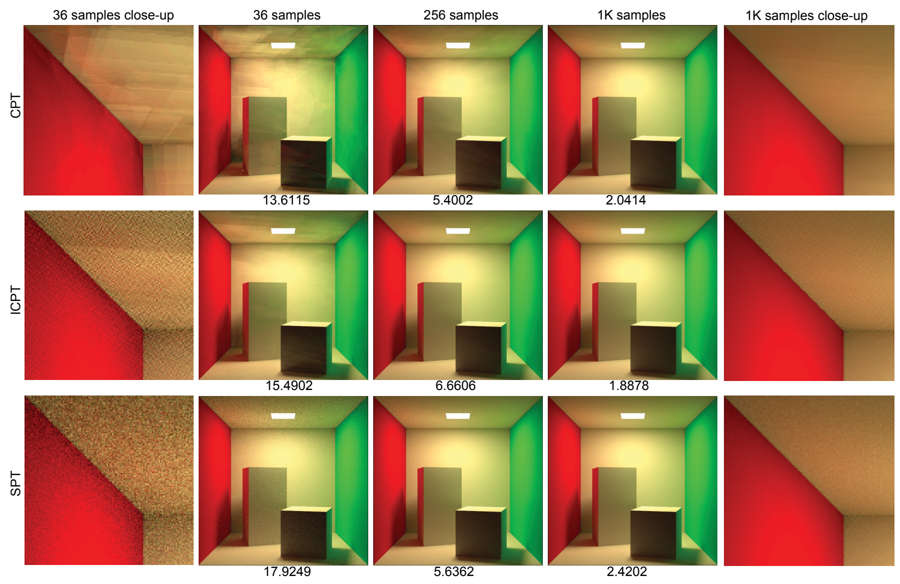

Figure 2. Diffuse Cornell box scene rendered with different number of samples using CPT (top row), ICPT (middle row), and SPT (bottom row). The numbers below each image indicate the RMS error values (the reference image was rendered using SPT with 16K samples/pixel). The far-right and far-left columns are close-ups of the rendered images and show the different noise patterns generated with each method.

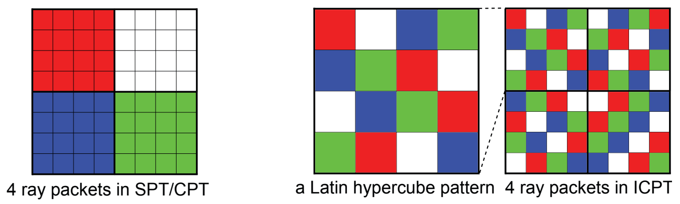

Figure 3. Each color represents the pixels contained in a packet that will use the same sequence of random numbers. The left image illustrates the standard way of grouping primary rays into ray packets of size 4 × 4, and the right image demonstrates our randomized interleaved ray packets, which consist of 4 randomized Latin hypercube patterns. Notice that both cases need 4 sequences of random numbers for an 8 × 8 block of pixels.

This way we can eliminate the correlated noise in the CPT method. Furthermore, as shown in Figure 2, we obtain similar RMS error values using both CPT and ICPT compared to the SPT method.

3. Results

In this section, we present our rendered images and measurements for five different scenes with different geometric complexities. Our implementation is based on the packet-tracing method with bounding volume hierarchies (BVH) [Wald et al. 07]. For SIMD instructions, we use SSE intrinsics. The traversal algorithm is a standard masked traversal with early hit tests (but without inactive ray filtering) [Wald et al. 07]. All results are measured on a single core of a Desktop PC with Intel 2.83 GHz Core 2 Quad Processors and 3.25 GB of RAM. Table 1 shows the performance of our packet-tracing implementation using eye rays without shading for the scenes we have tested. Figures 2, 4 and 5 show the rendered images for our test scenes.

Scene	Triangles	Rays/Sec
Diffuse Cornell Box	36	4.56 M
Diffuse Sponza	76148	1.21 M
Glossy Bunnies	208996	1.14 M
Diffuse Conference Room	282759	1.28 M
Glossy Buddha	1087416	0.86 M

Table 1. Geometric complexity of the five different scenes used in our measurements and the performance of our packet-tracing implementation for tracing eye rays without shading for a packet size of 8 × 8 pixels and 1 sample per pixel.

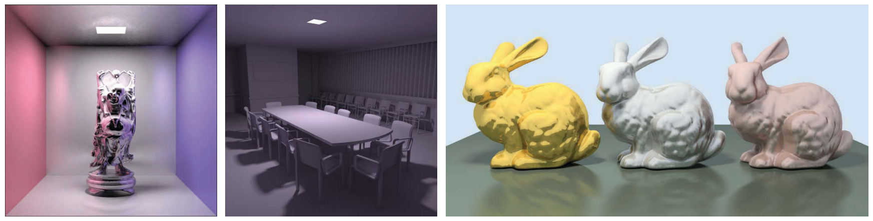

Figure 4. Test scenes rendered using ICPT: a glossy Buddha in a glossy Cornell box (left), a diffuse conference room scene (middle), and glossy bunnies with different glossiness with 1K samples per pixel (right).

We use a linear congruential random number generator for all of our results. We also tested an XOR random number generator, and it produced similar results in terms of both quality and performance. Since many pixels will use the same sequence of random numbers, we pre-generate and tabulate the random numbers.

3.1. Relative Performance

Figure 6 compares the relative performance of SPT, CPT and ICPT for up to 4 bounces (the effect of each bounce is shown in Figure 5). It can be seen that CPT and ICPT outperform SPT for all test scenes even after a few bounces. Furthermore, CPT outperforms ICPT since the rays in each packet are closely grouped and therefore more coherent. For primary rays (i.e., bounce 0), CPT slightly outperforms SPT because shadow rays are coherent and we have to generate fewer random numbers. In contrast, ICPT performs slightly worse than SPT, since the primary rays in each ray packet are less coherent. Also, there is a small overhead for interleaving the samples. As the number of bounces increases, the relative performance of CPT/ICPT over SPT decreases. The reason is that secondary rays lose a bit of coherency after each bounce and the difference between the coherent methods and SPT becomes smaller after several bounces.

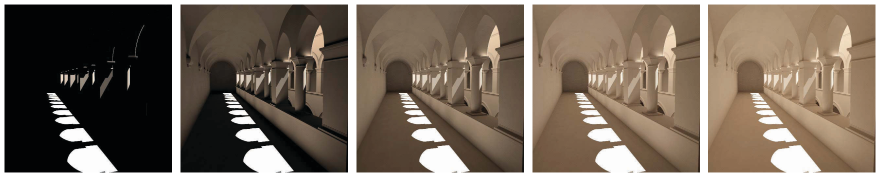

Figure 5. ICPT used for rendering our diffuse Sponza scene with 0, 1, 2, 3, and 4 bounces of indirect illumination with 4K samples per pixels. Indirect illumination is essential for rendering this scene.

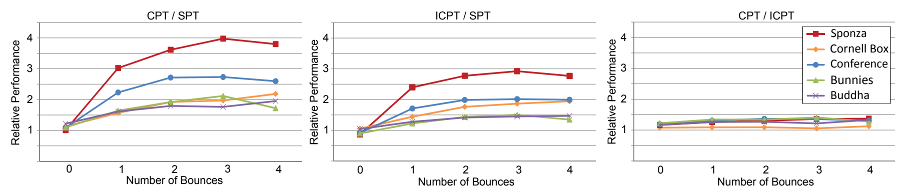

Figure 6. Relative performance of CPT over SPT (left), ICPT over SPT (middle), and CPT over ICPT (right) for fixed packet size of 4×4 pixels and 64 samples per pixel. CPT outperforms ICPT, and ICPT outperforms SPT.

We have used a packet size of 4×4 pixels for all measurements. Larger packet sizes result in more performance gains for CPT and ICPT compared to SPT. For example, when using 16×16 ray packets, CPT performs 6.2 times faster than the SPT method for the Sponza scene with 3 bounces of indirect light.

As noted in Figure 6, the Sponza scene results in the largest performance gain for the coherent methods. We investigated the case of the Sponza scene and found out that it has the highest number of BVH traversal steps per ray. The high traversal-step count makes ray tracing the Sponza scene more cache and memory intensive. Table 2 shows a measurement of the cache performance for Sponza. It can be seen that CPT has a higher cache utilization than the SPT method, and bus utilization measurements indicate that SPT require more memory accesses than CPT. Similar results can be expected from complex scenes in general.

	Bounces	**L1 Miss**	**L2 Miss**	**Bus Util.**
CPT/SPT	1	0.7/1.1	0.1/0.4	6.91/13.92
	2	0.8/1.3	0.2/0.5	8.46/16.91
	3	0.8/1.3	0.2/0.5	9.46/17.99

Table 2. Cache performance measured in percentages for the Sponza scene for CPT and SPT. CPT has lower L1 and L2 cache miss rates and bus utilization. All measurements are done for a packet size of 4×4 pixels and 64 samples per pixel using Intel® VTune™ for Performance Analyzer 9.0 evaluation edition.

3.2. SIMD Utilization

Figure 7 summarizes the SIMD utilization of SPT, CPT, and ICPT. SIMD utilization is much higher in CPT and ICPT compared to SPT even after a few bounces. For primary rays, CPT and SPT have the same SIMD utilization, which is higher than the utilization of ICPT. As previously mentioned, this is

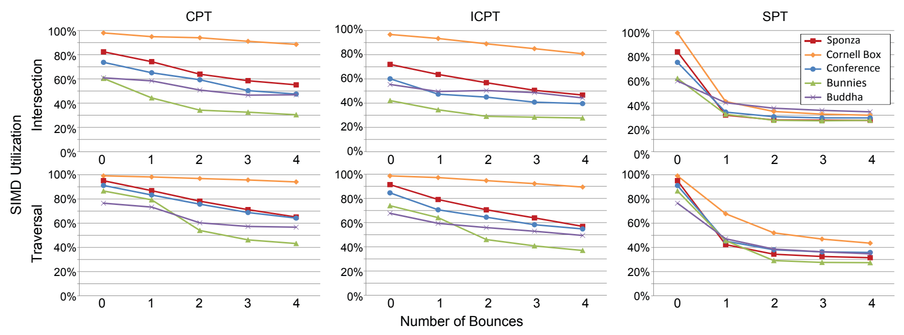

Figure 7. SIMD utilization plots for ray/triangle intersection computations (top row) and BVH traversal computations (bottom row) for different scenes. The horizontal axes are the number of bounces and the vertical axes are the SIMD utilization values. All measurements are done for a packet size of 4×4 pixels and 64 samples per pixel. CPT and ICPT have higher utilizations compared to SPT.

due to the primary rays of ICPT being less coherent than CPT and SPT as a result of interleaving. Additionally, SIMD utilization of CPT/ICPT is scene dependent. In general, scenes with large flat surfaces will have higher SIMD utilization. We get similar results for wider SIMD widths, which is important for both GPUs and architectures like Larrabee [Seiler et al. 08].

3.3. Coherence Measurements

Figure 8 shows our measurements of traversal and intersection coherence [Månsson et al. 07]. As expected, CPT has the highest coherency and SPT has the lowest, while ICPT loses a bit of coherency due to interleaving. The coherence of primary rays is roughly the same for all methods.

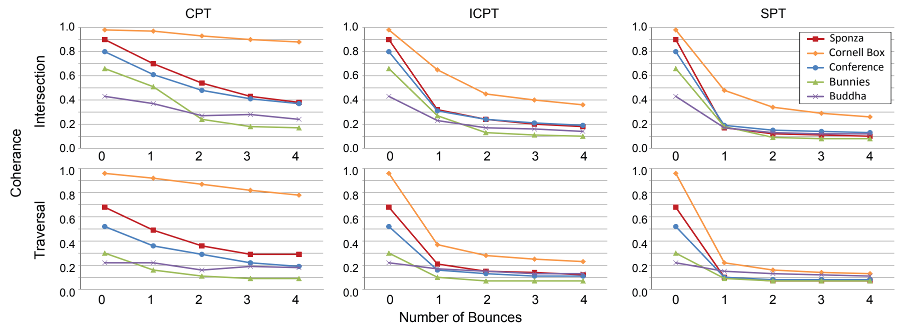

Figure 8. Normalized coherence-measurement plots for different scenes for a packet size of 4×4 pixels and 64 samples per pixel. The horizontal axes are the number of bounces. CPT and ICPT have higher coherency compared to SPT.

4. Discussion

Our results show that using the same sequence of random numbers for groups of rays is an effective technique for increasing the coherence and thereby performance of packet-based ray tracing. In most cases, CPT and ICPT can generate coherent secondary rays and therefore outperform SPT. The use of CPT and ICPT is most effective in scenes that contain smooth slowly changing geometry (e.g., the Cornell Box scene) since the secondary rays remain coherent after a number of bounces. CPT and ICPT also perform well in complex scenes with a high cache and memory bandwidth (e.g., the Sponza scene) since the bandwidth requirement is reduced by the coherent access patterns. Packet-based methods are not as efficient in scenes with pixel-sized geometry, and CPT and ICPT do not improve the performance in those situations.

CPT is at least as fast as SPT; in fact, it always performs slightly better than SPT, since all pixels will use the same sequence of random numbers and it computes fewer random numbers. This, however, comes at the cost of structured noise.

ICPT improves the quality of the rendered results by sacrificing some of the coherency. The primary rays of ICPT are less coherent than SPT and CPT, and this affects the performance negatively. However, for most scenes, ICPT outperforms SPT due to the improved coherency for secondary rays.

If the goal is to render noise-free images with a high number of samples, then the best algorithm is CPT. The structured noise will vanish as the number of samples increases, and the performance is better than ICPT. However, ICPT is effective at hiding the structured noise and it can be used in interactive applications when intermediate results are displayed.

Acknowledgments. We would like to thank Toshiya Hachisuka and the reviewers for suggesting several improvements to the paper. Furthermore, we would like to thank Wojciech Jarosz, Will Chang, and Krystle de Mesa for proofreading the paper.

References

[Kajiya 86] James T. Kajiya. "The Rendering Equation." *Computer Graphics (Proc. SIGGRAPH '86)* 20:4 (1986), 143–150.

[Keller and Heidrich 01] Alexander Keller and Wolfgang Heidrich. "Interleaved Sampling." In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pp. 269–276. London: Springer-Verlag, 2001.

[Keller 97] Alexander Keller. "Instant Radiosity." In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 49–56. New York: ACM Press/Addison-Wesley Publishing Co., 1997.

[Lauterbach et al. 06] C. Lauterbach, S.-E. Yoon, D. Tuft, and D. Manocha. "RT-DEFORM: Interactive Ray Tracing of Dynamic Scenes using BVHs." In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, pp. 39–46. Los Alamitos, CA: IEEE Press, 2006.

[McKay et al. 79] M. D. McKay, R. J. Beckman, and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code." *Technometrics* 21:2 (1979), 239–245.

[Månsson et al. 07] Erik Månsson, Jacob Munkberg, and Tomas Akenine-Möller. "Deep Coherent Ray Tracing." In *Proceedings of the 2007 Eurographics/IEEE Symposium on Interactive Ray Tracing*, pp. 79–85. Los Alamitos, CA: IEEE Press, 2007.

[Parker et al. 99] Steven Parker, William Martin, Peter-Pike J. Sloan, Peter Shirley, Brian Smits, and Charles Hansen. "Interactive ray tracing." In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D Graphics*, pp. 119–126. New York: ACM Press, 1999.

[Reshetov 06] A. Reshetov. "Omnidirectional Ray Tracing Traversal Algorithm for Kd-Trees." In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, pp. 57–60. Los Alamitos, CA: IEEE Press, 2006.

[Seiler et al. 08] Larry Seiler, Doug Carmean, Eric Sprangle, Tom Forsyth, Michael Abrash, Pradeep Dubey, Stephen Junkins, Adam Lake, Jeremy Sugerman, Robert Cavin, Roger Espasa, Ed Grochowski, Toni Juan, and Pat Hanrahan. "Larrabee: A Many-Core x86 Architecture for Visual Computing." In *ACM Transactions on Graphics (Proc. SIGGRAPH 2008)* 27:3 (2008), 1–15.

[Shirley 91] Peter S. Shirley. "Physically Based Lighting Calculations for Computer Graphics." Ph.D. thesis, Champaign, IL, 1991.

[Wald et al. 01] Ingo Wald, Carsten Benthin, Markus Wagner, and Philipp Slusallek. "Interactive Rendering with Coherent Ray Tracing." In *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001* 20:3 (2001), 153–164.

[Wald et al. 02] Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. "Interactive Global Illumination Using Fast Ray Tracing." In *EGRW '02: Proceedings of the 13th Eurographics Workshop on Rendering*, pp. 15–24. Aire-la-Ville, Switzerland: Eurographics Association, 2002.

[Wald et al. 07] Ingo Wald, Solomon Boulos, and Peter Shirley. "Ray Tracing Deformable Scenes Using Dynamic Bounding Volume Hierarchies." *ACM Transactions on Graphics* 26:1 (2007), Article 6.

Web Information:

http://jgt.akpeters.com/papers/Sadeghietal09/

http://graphics.ucsd.edu/~iman/CPT

Iman Sadeghi, UC San Diego, 9500 Gilman Dr., La Jolla, CA 92093
(iman@graphics.ucsd.edu)

Bin Chen, UC San Diego, 9500 Gilman Dr., La Jolla, CA 92093
(b6chen@ucsd.edu)

Henrik Wann Jensen, UC San Diego, 9500 Gilman Dr., La Jolla, CA 92093
(henrik@cs.ucsd.edu)

Received May 5, 2009; accepted September 11, 2009.