Babeş-Bolyai University
Cluj-Napoca

Faculty of Mathematics and Computer Science
Department of Computer Science

High-Performance Ray Tracing on Modern Parallel Processors

Summary of the PhD Thesis

Author:
Attila T. Áfra

Supervisor:
Prof. Dr. Horia F. Pop

Cluj-Napoca
2013

List of Publications

ISI Journal Papers

ÁFRA A. T., SZIRMAY-KALOS L.: Stackless Multi-BVH traversal for CPU, MIC and GPU ray tracing. *Computer Graphics Forum* (2013). To appear.

ÁFRA A. T.: Interactive ray tracing of large models using voxel hierarchies. *Computer Graphics Forum 31*, 1 (2012), 75–88. Presented at *Eurographics 2013* (Girona, Spain, 2013). Independent citations: 2 [KSY13, SW13].

International Conference Papers

ÁFRA A. T.: Incoherent ray tracing without acceleration structures. In *Eurographics 2012 - Short Papers* (Cagliari, Sardinia, Italy, 2012), Eurographics Association, pp. 97–100. Independent citations: 3 [NIDN13, VBHH13, CPJ13].

ÁFRA A. T.: Improving BVH ray tracing speed using the AVX instruction set. In *Eurographics 2011 - Posters* (Llandudno, UK, 2011), Eurographics Association, pp. 27–28.

Technical Reports

ÁFRA A. T.: *Faster Incoherent Ray Traversal Using 8-Wide AVX Instructions*. Tech. rep., Babeş-Bolyai University, Cluj-Napoca, Romania, Aug. 2013.

Research Grants

Grant **OTKA K-104476** of the Hungarian Scientific Research Fund: *Physics Simulation and Inverse Problem Solution on Massively Parallel Systems*.

Contents

Contents of the Thesis

Summary

Keywords: computer graphics, realistic image synthesis, ray tracing, parallel algorithms

1 Introduction

One of the most fundamental problems in computer graphics is to generate realistic or stylized images of three-dimensional virtual scenes. This process is called *rendering* or *image synthesis*. Rendering has numerous important applications in a wide variety of domains (e.g., computer-aided design, architecture, medical visualization, films, games).

In many cases, the rendered images must be as high-quality and as photorealistic as possible. *Ray tracing* [Gla89, SM03] is a powerful and elegant rendering algorithm that achieves this by simulating the interactions of *light rays* with the objects in the scene (see Figure 1).

Ray tracing is an inherently *parallel* task as the rays of light can be traced independently from each other. This is a very useful property since processors are becoming more and more parallel, but fully exploiting the available parallelism is a challenging problem. Another major issue is the amount of required memory to render an image.

In this thesis, we present a collection of novel high-performance ray tracing algorithms that address the problems mentioned above. These algorithms improve the computational efficiency and reduce the memory requirements of advanced ray tracing based methods on modern parallel processor (CPU, MIC, and GPU) architectures.

Ray Tracing

Ray tracing generates images by constructing *light transport paths* that connect pixels of the image plane with light sources in the virtual scene. The basis of physically based image synthesis is the *rendering equation* [Kaj86, ICG86]. Solving the rendering equation is commonly done with *Monte Carlo ray tracing* methods [Szi08] (e.g., *path tracing* [Kaj86]).

A fundamental operation in ray tracing is *ray shooting*, the objective of which is to find the closest intersection of a ray with the scene. The efficiency of ray shooting is one of the

Figure 1: Example of a photorealistic image rendered with ray tracing. *Source: "Greek Vases" by Florin Mocanu.*

key factors that determine the overall performance of a ray tracing renderer. Thus, we have chosen ray shooting as the central topic of our research.

Modern Parallel Processors

Most modern processor architectures are highly parallel and are able to exploit application parallelism at multiple levels. In this thesis, we focus on three novel processor architectures: Intel Sandy/Ivy Bridge [Int12a] (CPU), Intel Knights Corner [Int13b, Int12b] (MIC), and NVIDIA Kepler GK110 [Nvi12b, Nvi12a] (GPU).

Contributions of This Thesis

This thesis has the following main contributions:

1. The AVX instruction set introduced with the Intel Sandy Bridge architecture has doubled the peak floating point processing power of x86 CPUs. However, efficiently utilizing the 8-wide SIMD units for ray tracing is a difficult problem. We propose AVX-optimized ray traversal algorithms for both coherent and incoherent rays that provide higher performance than state-of-the-art SSE-based approaches. We use binary and 8-way branching BVHs as acceleration structures. We have measured improvements of up to 74% for coherent rays and up to 25% for incoherent rays. [Áfr11, Áfr13]

2. Extracting hidden coherence from random ray distributions requires the processing of very large ray batches [PKGH97, ENSB13]. Most hierarchical ray traversal algorithms maintain a stack, which prohibitively increases the memory requirements of tracing many rays in parallel. Consequently, the size of the ray batches must be relatively small, which leads to suboptimal coherence, and thus performance. The solution is to use

stackless approaches instead, but for some efficient acceleration structures like the *multi bounding volume hierarchy* (Multi-BVH or MBVH), no such algorithms have been proposed so far. We present a stackless traversal algorithm for 4-way and binary MBVHs, which replaces the regular traversal stack with a small *bitstack*. The bitstack encodes the per-level traversal state using *skip codes*. This reduces the total traversal state size by about 22–51×. We demonstrate that our approach has low computational overhead (9–31%) on the latest CPU, MIC, and GPU architectures. [ÁS13]

3. Rendering massive models consisting of hundreds of millions or even billions of primitives has many challenges. Such scenes usually exceed the size of the available memory, in which case special *out-of-core* rendering methods [YGKM08] must be used. Unfortunately, those usually have severe limitations in interactivity, visual quality, and shading complexity. We propose a new ray tracing based out-of-core rendering method, which supports accurate shadows and indirect illumination, and runs at interactive speeds on multi-core CPUs. Its key components are: an out-of-core hierarchical model representation that stores triangles and *level-of-detail* (LOD) voxels, an efficient memory management method with asynchronous I/O, and a LOD-based kd-tree traversal algorithm suitable for many ray types. Our renderer has a unique set of features, combining the advantages of previous state-of-the-art approaches. [Áfr12b]

4. Standard ray tracing methods build an acceleration structure for the scene to render, which must be updated or rebuilt every time the geometry changes. This makes rendering dynamic scenes with ray tracing much more difficult than with rasterization. A recently introduced approach called *divide-and-conquer (DAC) ray tracing* [KW11] eliminates the need to maintain an acceleration structure while providing competitive performance. However, very little research has been done on the implementation of this approach on parallel architectures. Another unexplored area is taking advantage of the actual ray distribution to perform more efficient primitive partitioning, which is not possible with prebuilt acceleration structures. We introduce an efficient DAC ray traversal algorithm optimized for incoherent rays and SIMD processing (using SSE and AVX instructions). In addition to these optimizations, we propose *adaptive partitioning* based on the active ray/primitive ratio, which reduces the partitioning overhead. We demonstrate that our approach outperforms the previous state of the art by up to 2.2×. [Áfr12a]

2 Coherent Ray Packet Traversal Using the AVX Instruction Set

Introduction

Ray packet algorithms [WSBW01, WBS07, ORM08] enable the fast tracing of *coherent* rays, which is especially important for real-time ray tracing solutions. One major source of perfor-

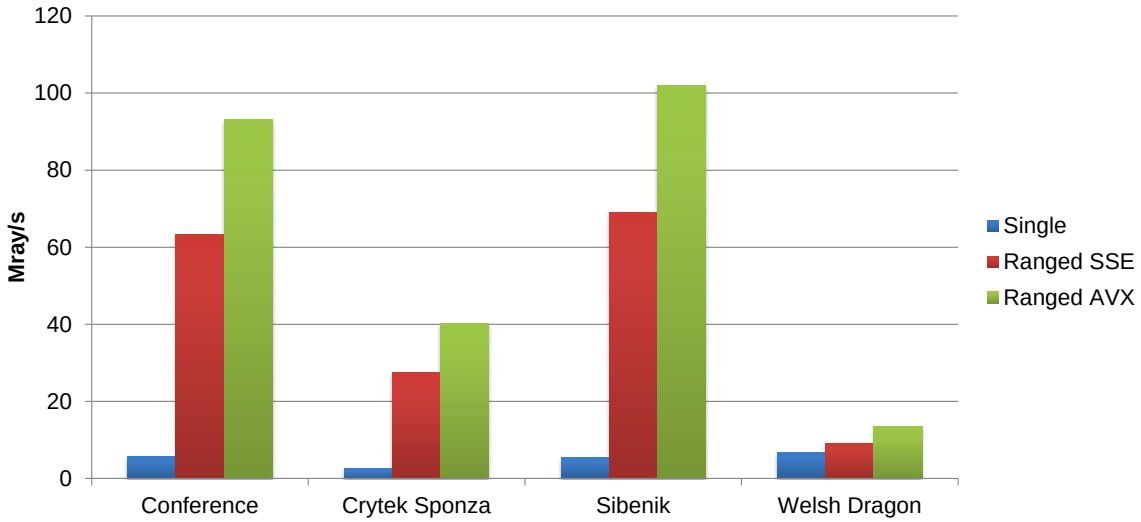


Figure 2: Single and ranged (implemented with SSE and AVX) traversal performance for primary rays in Mray/s.

mance improvement is the use of SIMD operations provided by the CPU.

Until recently, popular CPU architectures, like the x86, were able to operate on up to 4 single-precision floating-point numbers simultaneously. This has changed with the introduction of the 256-bit AVX (Advanced Vector Extensions) instruction set [Int13a], which has doubled the SIMD width of SSE, AltiVec, NEON, etc.

We have optimized two BVH packet traversal algorithms for AVX [Áfr11]: *ranged traversal* [WBS07] and *partition traversal* [ORM08].

AVX Ray Packet Tracing

The smallest ray primitive of both the ranged and partition traversal algorithms is the *SIMD ray*, which consists of multiple rays that are traced together throughout the entire algorithm. In 4-wide SIMD implementations, a SIMD ray usually contains 2×2 rays, thus, nearby rays are packed together to maximize coherence. For AVX, we employ 4×2 SIMD rays.

Ray packets can be quite large, commonly having a size of 256 or even 1024 rays, therefore, it is important to store the ray data in a cache-friendly way. This can be achieved by using an *array-of-structures-of-arrays* (AoSoA) layout.

The performance of ray packet tracing can be improved by applying frustum culling in addition to SIMD ray techniques. We use interval arithmetic (IA) [WBS07] for culling nodes, and corner rays for culling triangles, as described in [BWS06].

Results

All tests were run on a system with an Intel Core i5-2400 processor. The rendering resolution was set to 1024×768 pixels.

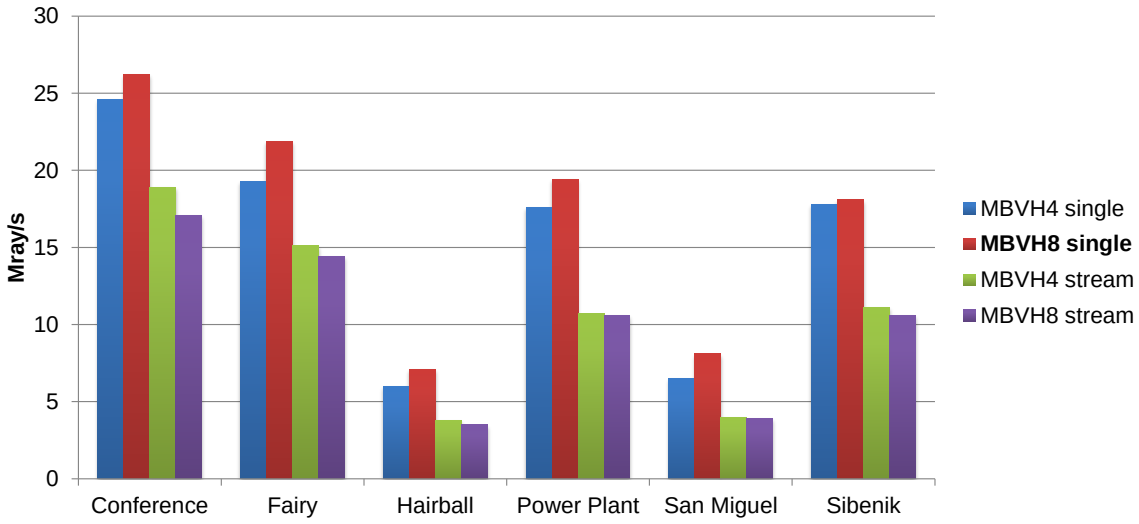


Figure 3: 8-bounce diffuse ray traversal performance in Mray/s.

Figure 2 shows the performance results for primary rays. AVX provides a speedup, compared to SSE, of at least roughly 50% in most cases. This sublinear increase is due to larger SIMD rays with lower utilization and non-SIMD parts of the algorithm.

3 Incoherent Ray Traversal Using the AVX Instruction Set

Introduction

In this chapter, we propose an AVX-optimized single-ray traversal algorithm for the MBVH acceleration structure [Áfr13]. The MBVH enables high SIMD utilization for single-ray traversal; therefore, it is a good choice for *incoherent* ray tracing. Our approach offers higher path tracing performance than previous SSE-based methods for a wide variety of test cases.

Ray Traversal Algorithm

In our algorithm, we trace rays in batches (e.g., 256 rays), but we trace them individually. Similarly to the binary BVH traversal algorithm [WBS07], we traverse the N-way MBVH with depth-first ordered traversal, which needs a traversal stack. We use the intersection distances provided by the box test algorithm to determine the traversal order of the intersected child nodes. A straightforward way to achieve this is to do horizontal SIMD sorting [FAN07], which unfortunately is quite costly and has suboptimal SIMD efficiency.

If there are less than N intersected nodes, SIMD sorting is even less efficient because it always sorts N values. For about 90% of the valid multi-node intersections, only 1–3 children are hit. A single hit is the most likely (40–60%). Therefore, SIMD sorting almost always works at a very low efficiency, especially for wide branching factors.

We sort the nodes with a scalar method optimized for a low number of hits, which was introduced in the Intel Embree ray tracer for 4-wide MBVH ray traversal [Ern11]. This is significantly faster than SIMD sorting. The main idea of the method is to use specialized implementations of sorting for the most frequent numbers of hits. We extend the original algorithm to handle wider trees by implementing the following cases: 1, 2, 3, 4, and 5–N hits.

Results

We compared our method with MBVH4 single-ray traversal and also with MBVH RS traversal, which, unlike the other methods, is able to extract hidden coherence from rays. Our benchmark system had an Intel Core i7-3770 processor. We used SSE for the MBVH4 traversal methods and AVX for the MBVH8 ones.

The performance results in million rays per second for 8-bounce diffuse rays are shown in Figure 3. Our method, MBVH8 single traversal implemented with AVX, yields a speedup of 2–25% relative to MBVH4 for the tested ray types and scenes. Also, it is faster than MBVH RS in all our tests, including primary ray benchmarks.

4 Stackless Multi-BVH Traversal for CPU, MIC, and GPU Ray Tracing

Introduction

Ray traversal algorithms can be divided into two main categories: *stack-based* and *stackless* algorithms. Using a stack for the traversal is typically the most straightforward and efficient approach. However, if many rays are traced in parallel, the storage and bandwidth costs of maintaining a full stack for each ray can be very high.

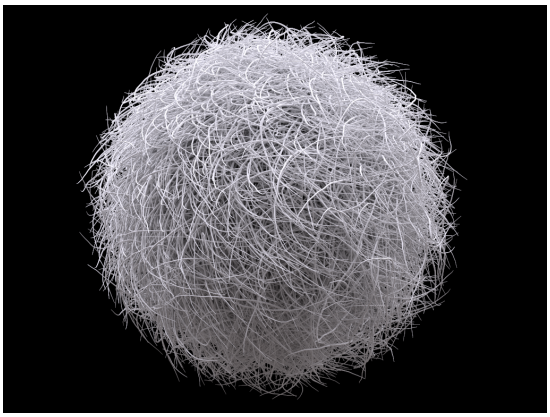
In this chapter, we propose a new efficient stackless ray traversal algorithm for MBVHs that supports distance-based ordered traversal without restarts [ÁS13]. We add parent and sibling pointers to the tree without necessarily increasing the memory footprint, and we replace the regular stack with a compact *bitstack*, an integer that fits into one or two machine registers. In the bitstack we store *skip codes* that indicate which siblings of a node must be traversed.

Two variations of the algorithm are presented: one variation for 4-way branching MBVHs (MBVH4) and one for binary BVHs having two child boxes per node (MBVH2). The MBVH4 is primarily used on CPUs with 4-wide or 8-wide SIMD, and also on the recent Intel MIC architecture with 16-wide SIMD. On the other hand, the MBVH2 is the preferred choice on current NVIDIA GPUs [AL09, ALK12]. We have optimized our method and evaluated its performance for all these hardware platforms.

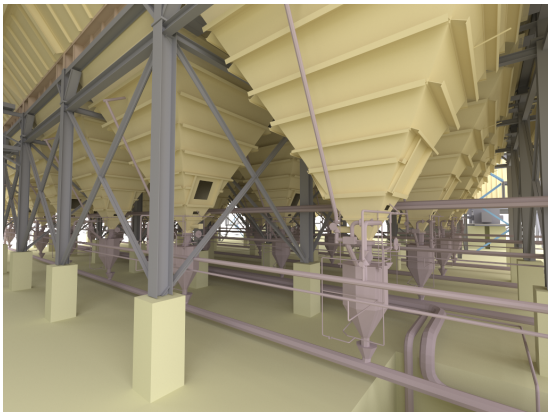
(a) CONFERENCE (282K triangles)

(b) CRYTEK SPONZA (262K triangles)

(c) FAIRY (174K triangles)



(d) HAIRBALL (2.9M triangles)



(e) POWER PLANT (12.7M triangles)

(f) SAN MIGUEL (10.5M triangles)

Figure 4: Test scenes used for the performance measurements of the ray traversal algorithms. The images were rendered using simple 8-bounce diffuse path tracing.

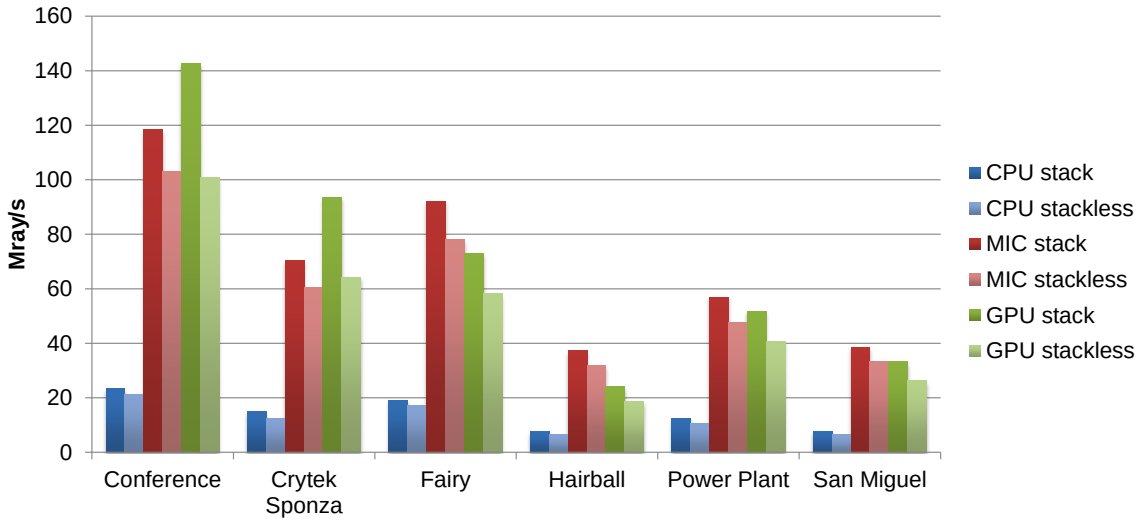


Figure 5: Stack-based and stackless traversal performance for 8-bounce diffuse path tracing.

Algorithm Overview

Our algorithm replaces the stack pop of standard stack-based approaches with backtracking in the tree from the current node. The purpose of this operation is to find the next unprocessed node, which is a sibling of either the current node or one of its ancestors. To be able to ascend in the tree, we add a parent pointer to each node. We also store pointers to the siblings for accessing them without taking a round trip to the parent.

The backtracking is guided by a bitmask that encodes which part of the N-way tree needs to be traversed. It stores $N-1$ bits for each visited tree level (except the root level), and is updated similarly to a stack, using bitwise push and pop operations. Hence, we call this special bitmask a *bitstack*. The per-level values in the bitstack are *skip codes*. These indicate which siblings of the most recently visited node on the respective level must be skipped.

Results

We evaluated the performance of our stackless traversal algorithms and the corresponding stack-based ones using a simple but highly optimized diffuse path tracer on all three architectures: Intel Core i7-3770 (CPU), Intel Xeon Phi SE10P (MIC), and NVIDIA Tesla K20c (GPU).

The performance results are shown in Figure 5. Our stackless algorithms, similarly to previous methods, are somewhat slower than the reference stack-based ones when used for ordinary ray tracing; however, they maintain about 22–51× smaller traversal states. For our test scenes (Figure 4), stackless traversal is slower by 9–17% on the CPU, 13–16% on the MIC, and 20–31% on the GPU.

5 Interactive Ray Tracing of Large Models Using Voxel Hierarchies

Introduction

This chapter presents a new massive model rendering method based on ray tracing [Áfr12b], efficiently combining the advantages and techniques of different existing approaches.

Several testing examples demonstrate that our method works effectively for different types of complex models, achieving interactive frame rates on a quad-core desktop PC. It supports a wide variety of ray traced shading algorithms, which include direct lighting with shadows, ambient occlusion, and global illumination (see Figure 6).

Method Overview

We first construct a *hierarchical out-of-core data structure*, which contains, in a compressed format, the original triangles and several LOD levels consisting of voxels.

Thanks to the hierarchical LOD mechanism, it is possible to render huge data sets that cannot be completely loaded into the system memory. During rendering, we load the necessary details *asynchronously*, thus, there is no stuttering due to insufficient available data.

We organize all primitives (i.e., the triangles and voxels) into a *kd-tree*. This out-of-core kd-tree has a dual purpose in our approach: it speeds up the ray intersections with the triangles, and stores the voxel hierarchy.

A subset of the kd-tree nodes contain a single *LOD voxel*, which is a primitive rendered as an axis-aligned box. It roughly approximates the original primitives stored in the subtree of the corresponding node and holds *shading attributes* (e.g., normal, color) per box face.

The entire kd-tree is decomposed into *treelets*, which are grouped into equally sized *blocks*. In order to reduce storage requirements, the blocks are encoded using a lossless data compression algorithm.

We employ a custom, purely software-based *memory manager*, which is responsible for the loading of the blocks required by the renderer.

The tight integration of the LOD levels with the acceleration structure enables an efficient model representation and ray traversal algorithm. By using LOD voxels, significantly higher frame rates can be achieved, with minimal loss of image quality. We provide fast LOD error metrics for primary, shadow, ambient occlusion, and diffuse interreflection rays.

Results

All benchmarks were performed on a desktop PC with an Intel Core i7-2600 CPU, 8 GB RAM, an NVIDIA GeForce GTX 560 Ti GPU, and two 7200 RPM hard disks in RAID 0 setup.

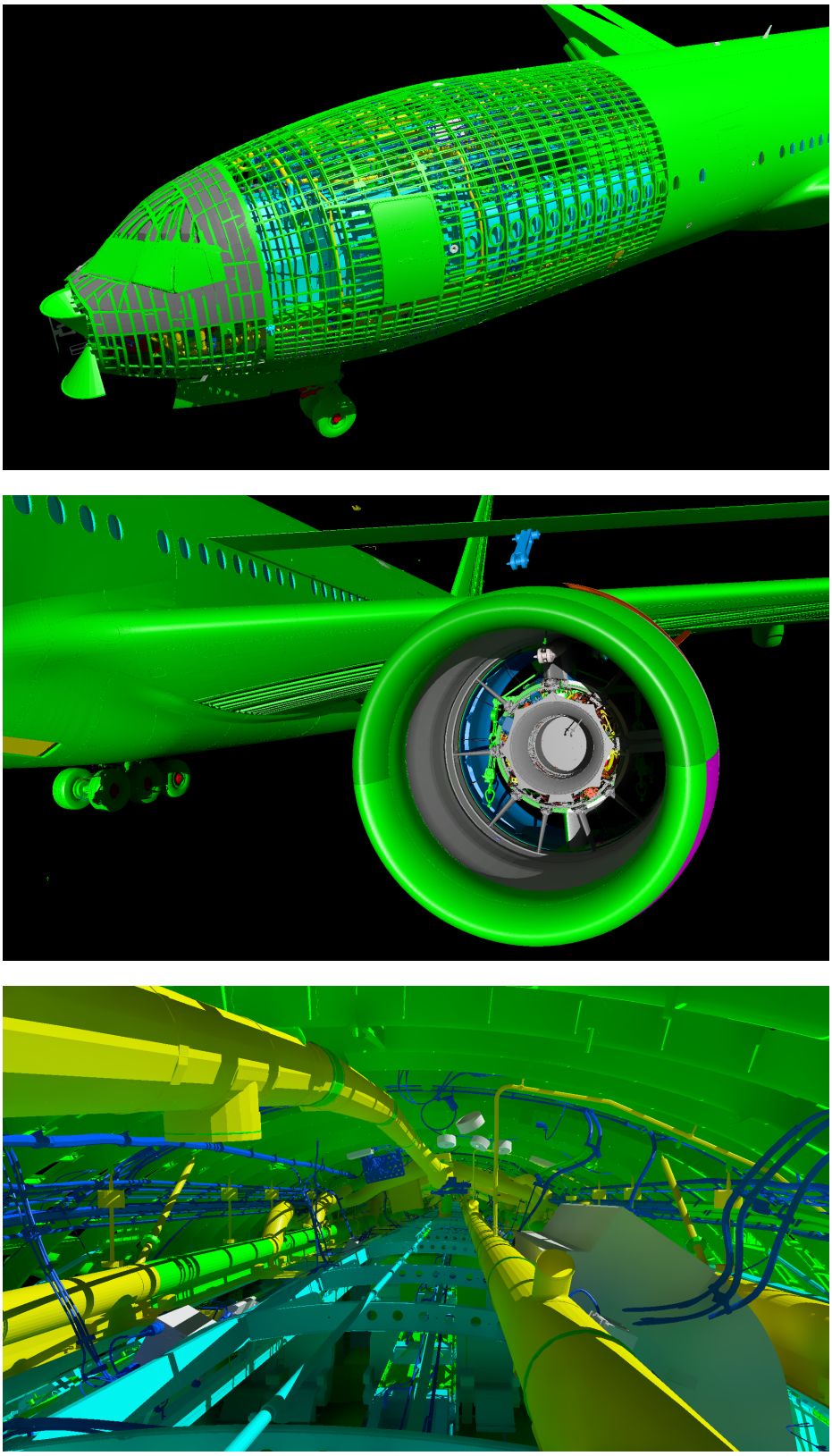


Figure 6: The BOEING 777 model (337M triangles) rendered interactively with shadows and one-bounce indirect illumination.

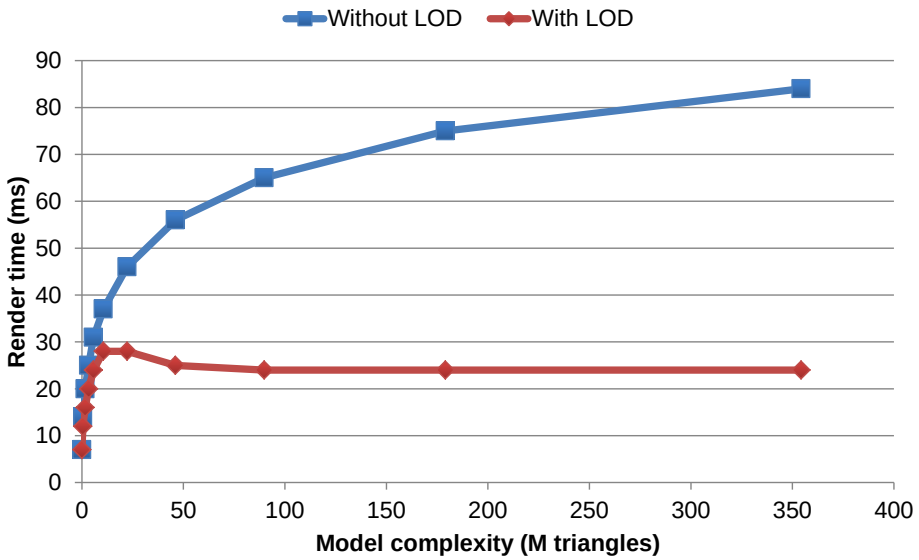


Figure 7: The scaling of ray casting performance with model complexity for MANDELBULB.

We have selected test models from different application domains: POWER PLANT (12M triangles), ASIAN DRAGON (7M triangles), LUCY (28M triangles), MPI V1.0 (73M triangles), BOEING 777 (337M triangles), and MANDELBULB (354M triangles).

The scaling of the ray casting performance with the number of triangles is illustrated in Figure 7. Notice that without LOD, the render time increases logarithmically, as expected from employing a kd-tree as an acceleration structure. However, if we enable the use of LOD voxels, the performance becomes nearly constant after a certain point.

We demonstrate that even the most complex models from the test suite can be rendered at interactive speeds with our approach.

6 Incoherent Ray Tracing Without Acceleration Structures

Introduction

A ray tracer typically consists of two main parts: ray traversal and acceleration structure building. Keller and Wächter [KW11] recently proposed a largely different and elegant approach called *divide-and-conquer ray tracing*, which does not require an acceleration structure.

In this chapter, we propose a new DAC traversal algorithm [Áfr12a] based on the core method by Keller et al. Our approach is generally more efficient than Mora's method [Mor11], and it exploits the AVX instruction set. We have optimized our method for incoherent rays.

Ray Filtering

The filtering can be executed in-place by rearranging the ray list to create an *active* and an *inactive* partition. A ray is specified using a point of origin, a direction vector, an interval

$[0, t_{\max}]$ defining a line segment, and an ID. The total size of a ray is 32 bytes, which means that it fits into a single AVX register or two SSE registers.

We avoid caching problems by simply reordering the rays in the original array. Rays can be quickly copied in blocks of 32 (with AVX) or 16 bytes (with SSE).

We simultaneously intersect 4 rays when using SSE and 8 rays when using AVX. Before doing so, the ray data, which consists of 8 values per ray, must be rearranged into SoA (structure-of-arrays) format.

Triangle Partitioning

Triangle partitioning divides a list of triangles into two disjoint sublists. We use two different partitioning methods: *middle partitioning* and *SAH partitioning*.

The partitioning algorithms do not process the triangles themselves, but only their AABBs (axis-aligned bounding boxes), which we precompute. In contrast with ray filtering, we manage a triangle ID array instead of directly reordering the AABBs.

Always partitioning with the SAH does not necessarily lead to the highest possible ray tracing performance. We solve this problem by adaptively deciding between SAH and middle partitioning. In each partitioning step, the ratio of the number of active rays and current triangles is checked against a predefined threshold (e.g, 1–2).

Triangle Intersection

In our method, a special triangle representation is used to save memory space and bandwidth. Similarly to the ray filtering routine, multiple rays are intersected with the current triangle using SIMD.

Ordered Traversal

For primary rays, front-to-back traversal has a significant positive impact on the ray tracing speed. However, the improvement is small for incoherent rays. We determine the traversal order with the very cheap approach from the packet tracer by Wald et al. [WBS07].

Results

The benchmarks were run on two different systems: on an Intel Core i7-960 with 24 GB RAM (triple channel), and on an Intel Core i7-2600 with 8 GB RAM (dual channel). We tested the algorithms using a 1-bounce and an 8-bounce Monte Carlo path tracer with diffuse reflections.

Our method is quite competitive to a highly optimized static ray tracer that uses the MBVH acceleration structure [Ern11]. For example, MBVH is only 12% faster for the 8-bounce path tracing of the CONFERENCE scene, on a single thread of the i7-960. However, the difference is greater on multiple threads, especially for HAIRBALL, where our method is 4× slower.

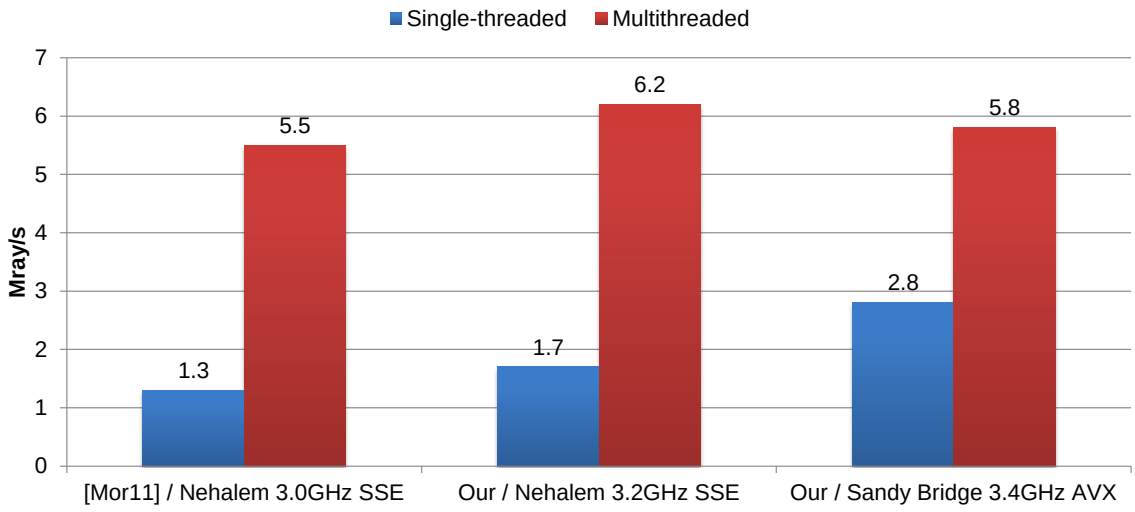


Figure 8: 8-bounce diffuse ray traversal performance in Mray/s for the CONFERENCE scene using Mora's [Mor11] versus our algorithm.

Compared to Mora's method, our approach filters rays more efficiently, uses higher quality object partitioning, exploits wider SIMD, and is optimized for incoherent rays (see Figure 8).

7 Conclusions

In this thesis, we investigated high-performance ray tracing on modern parallel processor architectures. Specifically, we proposed methods that better exploit the available parallelism and memory of latest-generation hardware, and enable superior image quality and interactivity than previous approaches. We provided efficient solutions for both offline and real-time rendering.

Possible future research directions include: MBVH4 ray traversal on GPUs, general purpose out-of-core ray tracing, higher quality voxel-based LOD, and DAC ray tracing on massively parallel processors.

Bibliography

[Áfr10] ÁFRA A. T.: Interactive out-of-core ray casting of massive triangular models with voxel-based LODs. In *Proceedings of the 5th Hungarian Conference on Computer Graphics and Geometry* (Budapest, Hungary, 2010), pp. 4–11.

[Áfr11] ÁFRA A. T.: Improving BVH ray tracing speed using the AVX instruction set. In *Eurographics 2011 - Posters* (Llandudno, UK, 2011), Eurographics Association, pp. 27–28.

[Áfr12a] ÁFRA A. T.: Incoherent ray tracing without acceleration structures. In *Eurographics 2012 - Short Papers* (Cagliari, Sardinia, Italy, 2012), Eurographics Association, pp. 97–100.

[Áfr12b] ÁFRA A. T.: Interactive ray tracing of large models using voxel hierarchies. *Computer Graphics Forum 31*, 1 (2012), 75–88.

[Áfr13] ÁFRA A. T.: *Faster Incoherent Ray Traversal Using 8-Wide AVX Instructions*. Tech. rep., Babeş-Bolyai University, Cluj-Napoca, Romania, Aug. 2013.

[AK90] ARVO J., KIRK D.: Particle transport and image synthesis. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 63–66.

[AK10] AILA T., KARRAS T.: Architecture considerations for tracing incoherent rays. In *Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, 2010), HPG '10, Eurographics Association, pp. 113–122.

[AL09] AILA T., LAINE S.: Understanding the efficiency of ray traversal on GPUs. In *Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), HPG '09, ACM, pp. 145–149.

[ALK12] AILA T., LAINE S., KARRAS T.: *Understanding the Efficiency of Ray Traversal on GPUs – Kepler and Fermi Addendum*. NVIDIA Technical Report NVR-2012-02, NVIDIA Corporation, June 2012.

[ÁS13] ÁFRA A. T., SZIRMAY-KALOS L.: Stackless Multi-BVH traversal for CPU, MIC and GPU ray tracing. *Computer Graphics Forum* (2013). To appear.

[BA13] BARRINGER R., AKENINE-MÖLLER T.: Dynamic stackless binary tree traversal. *Journal of Computer Graphics Techniques (JCGT) 2*, 2 (March 2013), 38–49.

[Bat68] BATCHER K. E.: Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference* (New York, NY, USA, 1968), AFIPS '68 (Spring), ACM, pp. 307–314.

[BBS*09] BUDGE B., BERNARDIN T., STUART J. A., SENGUPTA S., JOY K. I., OWENS J. D.: Out-of-core data management for path tracing on hybrid resources. *Computer Graphics Forum 28*, 2 (2009), 385–396.

[Ben06] BENTHIN C.: *Realtime Ray Tracing on Current CPU Architectures*. PhD thesis, Computer Graphics Group, Saarland University, 2006.

[Bik12] BIKKER J.: *Ray Tracing in Real-Time Games*. PhD thesis, Delft University of Technology, 2012.

[BWB08] BOULOS S., WALD I., BENTHIN C.: Adaptive ray packet reordering. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2008* (Los Alamitos, CA, USA, 2008), IEEE Computer Society, pp. 131–138.

[BWS06] BOULOS S., WALD I., SHIRLEY P.: *Geometric and Arithmetic Culling Methods for Entire Ray Packets*. Tech. Rep. UUCS-06-010, School of Computing, University of Utah, 2006.

[BWW*12] BENTHIN C., WALD I., WOOP S., ERNST M., MARK W.: Combining single and packet-ray tracing for arbitrary ray distributions on the Intel MIC architecture. *IEEE Transactions on Visualization and Computer Graphics 18*, 9 (September 2012), 1438–1448.

[CCC87] COOK R. L., CARPENTER L., CATMULL E.: The Reyes image rendering architecture. *SIGGRAPH '87 21*, 4 (Aug. 1987), 95–102.

[CKL*10] CHOI B., KOMURAVELLI R., LU V., SUNG H., BOCCHINO R. L., ADVE S. V., HART J. C.: Parallel SAH k-D tree construction. In *Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, 2010), HPG '10, Eurographics Association, pp. 77–86.

[CNLE09] CRASSIN C., NEYRET F., LEFEBVRE S., EISEMANN E.: GigaVoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 15–22.

[CPJ13] COSTA V., PEREIRA J. M., JORGE J. A.: Compressed grids for GPU ray tracing of large models. In *WSCG 2013 - Poster Proceedings* (Plzen, Czech Republic, 2013), Vaclav Skala - Union Agency, pp. 29–32.

[Dam11] DAMMERTZ H.: *Acceleration Methods for Ray Tracing based Global Illumination*. PhD thesis, Ulm University, 2011.

[DHK08] DAMMERTZ H., HANIKA J., KELLER A.: Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays. *Computer Graphics Forum 27*, 4 (2008), 1225–1233.

[EG08] ERNST M., GREINER G.: Multi bounding volume hierarchies. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2008* (2008), pp. 35–40.

[ENSB13] EISENACHER C., NICHOLS G., SELLE A., BURLEY B.: Sorted deferred shading for production path tracing. *Computer Graphics Forum 32*, 4 (2013), 125–132.

[Ern11] ERNST M.: Embree: Photo-realistic ray tracing kernels. *SIGGRAPH 2011 Talk* (2011).

[FAN07] FURTAK T., AMARAL J. N., NIEWIADOMSKI R.: Using SIMD registers and instructions to enable instruction-level parallelism in sorting algorithms. In *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures* (New York, NY, USA, 2007), SPAA '07, ACM, pp. 348–357.

[Fra04] FRASER K.: *Practical lock-freedom*. Tech. Rep. UCAM-CL-TR-579, University of Cambridge, Computer Laboratory, Feb. 2004.

[FS88] FUSSELL D. S., SUBRAMANIAN K. R.: *Fast Ray Tracing Using K-d Trees*. Tech. rep., University of Texas, Austin, TX, USA, 1988.

[FS05] FOLEY T., SUGERMAN J.: KD-tree acceleration structures for a GPU raytracer. In *Proceedings of the ACM SIGGRAPH/Eurographics Conference on Graphics Hardware* (New York, NY, USA, 2005), HWWS '05, ACM, pp. 15–22.

[FTI86] FUJIMOTO A., TANAKA T., IWATA K.: ARTS: Accelerated ray-tracing system. *Computer Graphics and Applications, IEEE 6*, 4 (1986), 16–26.

[GKDS12] GEORGIEV I., KŘIVÁNEK J., DAVIDOVIČ T., SLUSALLEK P.: Light transport simulation with vertex connection and merging. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 192:1–192:10.

[Gla89] GLASSNER A. S. (Ed.): *An Introduction to Ray Tracing*. Academic Press Ltd., London, UK, 1989.

[GM04] GOBBETTI E., MARTON F.: Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Computers & Graphics 28*, 6 (2004), 815–826.

[GM05] GOBBETTI E., MARTON F.: Far Voxels – a multiresolution framework for interactive rendering of huge complex 3D models on commodity graphics platforms. *ACM Transactions on Graphics 24*, 3 (August 2005), 878–885. Proc. SIGGRAPH 2005.

[GPM11] GARANZHA K., PANTALEONI J., MCALLISTER D.: Simpler and faster HLBVH with work queues. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics* (New York, NY, USA, 2011), HPG '11, ACM, pp. 59–64.

[GR08] GRIBBLE C. P., RAMANI K.: Coherent ray tracing via stream filtering. In *2008 IEEE/Eurographics Symposium on Interactive Ray Tracing* (August 2008), pp. 59–66.

[GS87] GOLDSMITH J., SALMON J.: Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications 7*, 5 (May 1987), 14–20.

[Hav00] HAVRAN V.: *Heuristic Ray Shooting Algorithms*. PhD thesis, Czech Technical University in Prague, November 2000.

[HBŽ98] HAVRAN V., BITTNER J., ŽÁRA J.: Ray tracing with rope trees. In *Proceedings of SCCG'98 (Spring Conference on Computer Graphics)* (Budmerice, Slovak Republic, Apr. 1998), pp. 130–139.

[HDW*11] HAPALA M., DAVIDOVIČ T., WALD I., HAVRAN V., SLUSALLEK P.: Efficient stack-less BVH traversal for ray tracing. In *Proceedings of the 27th Spring Conference on Computer Graphics* (New York, NY, USA, 2011), SCCG '11, ACM, pp. 7–12.

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. In *ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), SIGGRAPH Asia '08, ACM, pp. 130:1–130:8.

[HP11] HENNESSY J. L., PATTERSON D. A.: *Computer Architecture: A Quantitative Approach*, 5th ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[HPJ12] HACHISUKA T., PANTALEONI J., JENSEN H. W.: A path space extension for robust light transport simulation. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 191:1–191:10.

[HSHH07] HORN D. R., SUGERMAN J., HOUSTON M., HANRAHAN P.: Interactive k-D tree GPU raytracing. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2007), I3D '07, ACM, pp. 167–174.

[HZDS09] HAVRAN V., ZAJAC J., DRAHOKOUPIL J., SEIDEL H.-P.: *MPI Informatics Building Model as Data for Your Research*. Research Report MPI-I-2009-4-004, MPI Informatik, Saarbruecken, Germany, December 2009.

[ICG86] IMMEL D. S., COHEN M. F., GREENBERG D. P.: A radiosity method for non-diffuse environments. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 133–142.

[Int12a] INTEL: *Intel 64 and IA-32 Architectures Optimization Reference Manual*, April 2012.

[Int12b] INTEL: *Knights Corner Instruction Set Reference Manual*, August 2012.

[Int13a] INTEL: *Intel 64 and IA-32 Architectures Software Developer's Manual*, June 2013.

[Int13b] INTEL: *Intel Xeon Phi System Software Developer's Guide*, June 2013.

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (London, UK, 1996), Springer-Verlag, pp. 21–30.

[Jen01] JENSEN H. W.: *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[JR13] JEFFERS J., REINDERS J.: *Intel Xeon Phi Coprocessor High Performance Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2013.

[KA13] KARRAS T., AILA T.: Fast parallel construction of high-quality bounding volume hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference* (New York, NY, USA, 2013), HPG '13, ACM, pp. 89–99.

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 143–150.

[KH01] KELLER A., HEIDRICH W.: Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (London, UK, UK, 2001), Springer-Verlag, pp. 269–276.

[KIS*12] KOPTA D., IZE T., SPJUT J., BRUNVAND E., DAVIS A., KENSLER A.: Fast, effective BVH updates for animated scenes. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2012), I3D '12, ACM, pp. 197–204.

[KK86] KAY T. L., KAJIYA J. T.: Ray tracing complex scenes. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 269–278.

[KSAC02] KELEMEN C., SZIRMAY-KALOS L., ANTAL G., CSONKA F.: A simple and robust mutation strategy for the Metropolis light transport algorithm. *Computer Graphics Forum 21*, 3 (2002), 531–540.

[KSS*13] KOPTA D., SHKURKO K., SPJUT J., BRUNVAND E., DAVIS A.: An energy and bandwidth efficient ray tracing architecture. In *Proceedings of the 5th High-Performance Graphics Conference* (New York, NY, USA, 2013), HPG '13, ACM, pp. 121–128.

[KSY13] KIM T.-J., SUN X., YOON S.-E.: T-ReX: Interactive global illumination of massive models on heterogeneous computing resources. *IEEE Transactions on Visualization and Computer Graphics* (2013). To appear.

[KW11] KELLER A., WÄCHTER C.: Efficient ray tracing without auxiliary acceleration data structure. *High-Performance Graphics 2011 (Poster)* (2011).

[KZ11] KNAUS C., ZWICKER M.: Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics 30*, 3 (May 2011), 25:1–25:13.

[Laf96] LAFORTUNE E.: *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*. PhD thesis, Katholieke Universiteit Leuven, 1996.

[Lai10] LAINE S.: Restart trail for stackless BVH traversal. In *Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, 2010), HPG '10, Eurographics Association, pp. 107–111.

[LK10] LAINE S., KARRAS T.: Efficient sparse voxel octrees. In *Proceedings of ACM SIGGRAPH 2010 Symposium on Interactive 3D Graphics and Games* (2010), ACM Press, pp. 55–63.

[LYTM08] LAUTERBACH C., YOON S.-E., TANG M., MANOCHA D.: ReduceM: Interactive and memory efficient ray tracing of large models. *Computer Graphics Forum 27*, 4 (2008), 1313–1321.

[MB90] MACDONALD D. J., BOOTH K. S.: Heuristics for ray tracing using space subdivision. *The Visual Computer 6*, 3 (May 1990), 153–166.

[Mor66] MORTON G. M.: *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*. Tech. Rep. Ottawa, Ontario, Canada, IBM Ltd., 1966.

[Mor11] MORA B.: Naive ray-tracing: A divide-and-conquer approach. *ACM Transactions on Graphics 30* (October 2011), 117:1–117:12.

[MT97] MÖLLER T., TRUMBORE B.: Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools 2*, 1 (1997), 21–28.

[NFLM07] NAVRÁTIL P. A., FUSSELL D. S., LIN C., MARK W. R.: Dynamic ray scheduling to improve ray coherence and bandwidth utilization. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing* (Washington, DC, USA, 2007), RT '07, IEEE Computer Society, pp. 95–104.

[NIDN13] NABATA K., IWASAKI K., DOBASHI Y., NISHITA T.: Efficient divide-and-conquer ray tracing using ray sampling. In *Proceedings of the 5th High-Performance Graphics Conference* (New York, NY, USA, 2013), HPG '13, ACM, pp. 129–135.

[Nvi12a] NVIDIA: *CUDA C Programming Guide*, October 2012.

[Nvi12b] NVIDIA: *NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110*. Whitepaper, NVIDIA Corporation, 2012.

[ORM08] OVERBECK R., RAMAMOORTHI R., MARK W. R.: Large ray packets for real-time whitted ray tracing. In *IEEE/Eurographics Symposium on Interactive Ray Tracing 2008* (2008), pp. 41–48.

[PFHA10] PANTALEONI J., FASCIONE L., HILL M., AILA T.: PantaRay: fast ray-traced occlusion caching of massive scenes. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), ACM, pp. 37:1–37:10.

[PGSS07] POPOV S., GÜNTHER J., SEIDEL H.-P., SLUSALLEK P.: Stackless kd-tree traversal for high performance GPU ray tracing. *Computer Graphics Forum 26*, 3 (2007), 415–424.

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010.

[PKGH97] PHARR M., KOLB C., GERSHBEIN R., HANRAHAN P.: Rendering complex scenes with memory-coherent ray tracing. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 101–108.

[Rah13] RAHMAN R.: *Intel Xeon Phi Coprocessor Architecture and Tools: The Guide for Application Developers*. Apress Media LLC, New York, NY, USA, 2013.

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), pp. 343–352.

[RL01] RUSINKIEWICZ S., LEVOY M.: Streaming QSplat: A viewer for networked visualization of large, dense models. In *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), ACM, pp. 63–68.

[RSH05] RESHETOV A., SOUPIKOV A., HURLEY J.: Multi-level ray tracing algorithm. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 1176–1185.

[RW80] RUBIN S. M., WHITTED T.: A 3-dimensional representation for fast rendering of complex scenes. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1980), SIGGRAPH '80, ACM, pp. 110–116.

[SCS*08] SEILER L., CARMEAN D., SPRANGLE E., FORSYTH T., ABRASH M., DUBEY P., JUNKINS S., LAKE A., SUGERMAN J., CAVIN R., ESPASA R., GROCHOWSKI E., JUAN T., HANRAHAN P.: Larrabee: a many-core x86 architecture for visual computing. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 18:1–18:15.

[SE10] SEGOVIA B., ERNST M.: Memory efficient ray tracing with hierarchical mesh quantization. In *Graphics Interface 2010* (2010), pp. 153–160.

[SFD09] STICH M., FRIEDRICH H., DIETRICH A.: Spatial splits in bounding volume hierarchies. In *Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), HPG '09, ACM, pp. 7–13.

[SGNS07] SLOAN P.-P., GOVINDARAJU N. K., NOWROUZEZAHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2007), PG '07, IEEE Computer Society, pp. 97–105.

[SHBS02] SZIRMAY-KALOS L., HAVRAN V., BENEDEK B., SZÉCSI L.: On the efficiency of ray-shooting acceleration schemes. In *Proceedings of Spring Conference on Computer Graphics (SCCG)* (2002), pp. 97–106.

[SM98] SZIRMAY-KALOS L., MÁRTON G.: Worst-case versus average-case complexity of ray-shooting. *Journal of Computing 61*, 2 (1998), 103–131.

[SM03] SHIRLEY P., MORLEY R. K.: *Realistic Ray Tracing*, 2 ed. A. K. Peters, Ltd., Natick, MA, USA, 2003.

[Smi98] SMITS B.: Efficiency issues for ray tracing. *Journal of Graphics Tools 3*, 2 (Feb. 1998), 1–14.

[SSK07] SHEVTSOV M., SOUPIKOV A., KAPUSTIN A.: Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. *Computer Graphics Forum 26* (2007), 395–404.

[SSS08] SZIRMAY-KALOS L., SZÉCSI L., SBERT M.: *GPU-Based Techniques for Global Illumination Effects*. Morgan and Claypool Publishers, San Rafael, USA, 2008.

[SW13] SOMERS B., WOOD Z. J.: FlexRender: A distributed rendering architecture for ray tracing huge scenes on commodity hardware. In *GRAPP & IVAPP 2013* (2013), Coquillart S., Andújar C., Laramee R. S., Kerren A., Braz J., (Eds.), SciTePress, pp. 152–164.

[Szi08] SZIRMAY-KALOS L.: *Monte-Carlo Methods in Global Illumination — Photo-realistic Rendering with Randomization*. VDM, Verlag Dr. Müller, Saarbrücken, 2008.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), IEEE Computer Society, pp. 839–846.

[TMG09] TORRES R., MARTÍN P. J., GAVILANES A.: Ray casting using a roped BVH with CUDA. In *Proceedings of the 25th Spring Conference on Computer Graphics* (New York, NY, USA, 2009), SCCG '09, ACM, pp. 95–102.

[Tsa09] TSAKOK J. A.: Faster incoherent rays: Multi-BVH ray stream tracing. In *Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), HPG '09, ACM, pp. 151–158.

[VBHH13] VINKLER M., BITTNER J., HAVRAN V., HAPALA M.: Massively parallel hierarchical scene processing with applications in rendering. *Computer Graphics Forum* (2013). To appear.

[Vea97] VEACH E.: *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1997.

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 65–76.

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004.

[Wal07] WALD I.: On fast construction of SAH based bounding volume hierarchies. In *Proceedings of the 2007 Eurographics/IEEE Symposium on Interactive Ray Tracing* (2007), pp. 33–40.

[WBB08] WALD I., BENTHIN C., BOULOS S.: Getting rid of packets – efficient SIMD single-ray traversal using multi-branching BVHs. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2008* (2008), pp. 49–57.

[WBS07] WALD I., BOULOS S., SHIRLEY P.: Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Transactions on Graphics 26*, 1 (2007).

[WDS04] WALD I., DIETRICH A., SLUSALLEK P.: An interactive out-of-core rendering framework for visualizing massively complex models. In *EGSR04: 15th Eurographics Symposium on Rendering* (Norrköping, Sweden, 2004), Eurographics Association, pp. 81–92.

[WGBK07] WALD I., GRIBBLE C. P., BOULOS S., KENSLER A.: *SIMD Ray Stream Tracing - SIMD Ray Traversal with Generalized Ray Packets and On-the-fly Re-Ordering*. Tech. Rep. UUSCI-2007-012, SCI Institute, University of Utah, 2007.

[WH06] WALD I., HAVRAN V.: On building fast kd-trees for ray tracing, and on doing that in O(N log N). In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing* (2006), pp. 61–69.

[WMG*09] WALD I., MARK W. R., GÜNTHER J., BOULOS S., IZE T., HUNT W., PARKER S. G., SHIRLEY P.: State of the art in ray tracing animated scenes. *Computer Graphics Forum 28*, 6 (2009), 1691–1722.

[WRG07] WAGNER G. N., RAPOSO A., GATTASS M.: An anti-aliasing technique for voxel-based massive model visualization strategies. In *Proceedings of the 3rd International Conference on Advances in Visual Computing - Volume Part I* (Berlin, Heidelberg, 2007), ISVC'07, Springer-Verlag, pp. 288–297.

[WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive rendering with coherent ray tracing. *Computer Graphics Forum (Proceedings of EUROGRAPHICS) 20*, 3 (2001), 153–164.

[YGKM08] YOON S.-E., GOBBETTI E., KASIK D., MANOCHA D.: *Real-Time Massive Model Rendering*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2008.

[YLM06] YOON S.-E., LAUTERBACH C., MANOCHA D.: R-LODs: Fast LOD-based ray tracing of massive models. *The Visual Computer: International Journal of Computer Graphics 22*, 9 (2006), 772–784.

[YLPM05] YOON S.-E., LINDSTROM P., PASCUCCI V., MANOCHA D.: Cache-oblivious mesh layouts. *ACM Transactions on Graphics 24* (July 2005), 886–893.

[YM06] YOON S.-E., MANOCHA D.: Cache-efficient layouts of bounding volume hierarchies. *Computer Graphics Forum 25*, 3 (2006), 507–516.

[YSGM04] YOON S.-E., SALOMON B., GAYLE R., MANOCHA D.: Quick-VDR: Interactive view-dependent rendering of massive models. *IEEE Visualization* (2004), 131–138.

[ZL77] ZIV J., LEMPEL A.: A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory 23* (1977), 337–343.