Improving BVH Ray Tracing Speed Using the AVX Instruction Set

Attila T. Áfra

Budapest University of Technology and Economics, Hungary
Babeș-Bolyai University, Cluj-Napoca, Romania

Contact: attila.afra@gmail.com

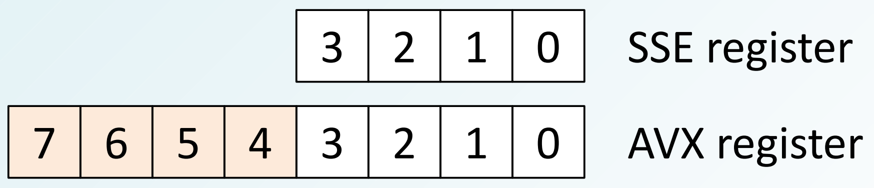
Introduction

Ray packet algorithms enable the efficient tracing of coherent rays, which is especially important for real-time ray tracing solutions. One major source of performance improvement is the use of SIMD operations provided by the CPU. For example, the SSE instruction set enables the intersection of 4 rays with an acceleration structure node or a primitive approximately at the cost of 1. However, advanced packet-based approaches also have remarkable algorithmic benefits which are independent of SIMD.

In this poster, we present our approach to optimizing coherent BVH ray packet tracing for the new AVX instruction set, which has, amongst others, 8-wide SIMD operations on 32-bit floating-point numbers. We have measured an average speedup of about 50% compared to our SSE4.1 implementation, on an Intel Sandy Bridge processor.

AVX Instruction Set

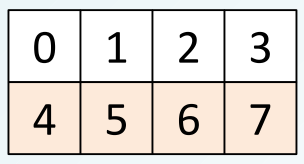
Until recently, popular CPU architectures, like x86, were able to operate on up to 4 single-precision floating-point numbers at the same time. This has changed with the introduction of the 256-bit AVX (Advanced Vector Extensions) instruction set, which has double the SIMD width of SSE, AltiVec, NEON,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



Ray Packet Traversal

We have optimized two BVH packet traversal algorithms for AVX: **ranged traversal** and **partition traversal**. The smallest ray primitive of both these algorithms is the **SIMD ray**, which consists of multiple rays that are traced together throughout the entire algorithm. This enables efficient parallel intersection of rays with nodes and triangles. 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**.

Ranged traversal is very sensitive to the order of the rays. Partition traversal is more robust in this area. In order to maximize speed, SIMD rays are stored in Morton-order.

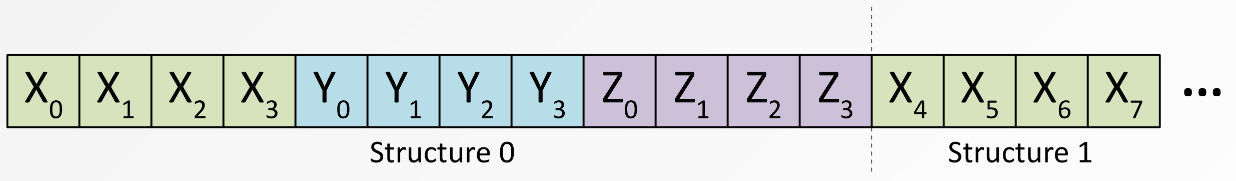


AVX 4×2 SIMD ray pixel layout

AoSoA Ray Layout

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, which, by grouping together the data belonging to a SIMD ray, combines the SIMD-friendliness of SoA (structure of arrays) with the locality of AoS (array of structures).

Example (3D vector, 4-wide SIMD):


X_0 X_1 X_2 X_3 Y_0 Y_1 Y_2 Y_3 Z_0 Z_1 Z_2 Z_3 X_4 X_5 X_6 X_7 ...
Structure 0
Structure 1

Frustum Culling

The performance of ray packet tracing can be improved by applying frustum culling in addition to SIMD ray methods. Our implementation uses **interval arithmetic** (IA) for culling nodes, and **corner rays** for culling triangles. Neither of these techniques take advantage of AVX. Our box test is entirely scalar, and the triangle test does not map well to wide SIMD execution because there are only 4 corner rays.

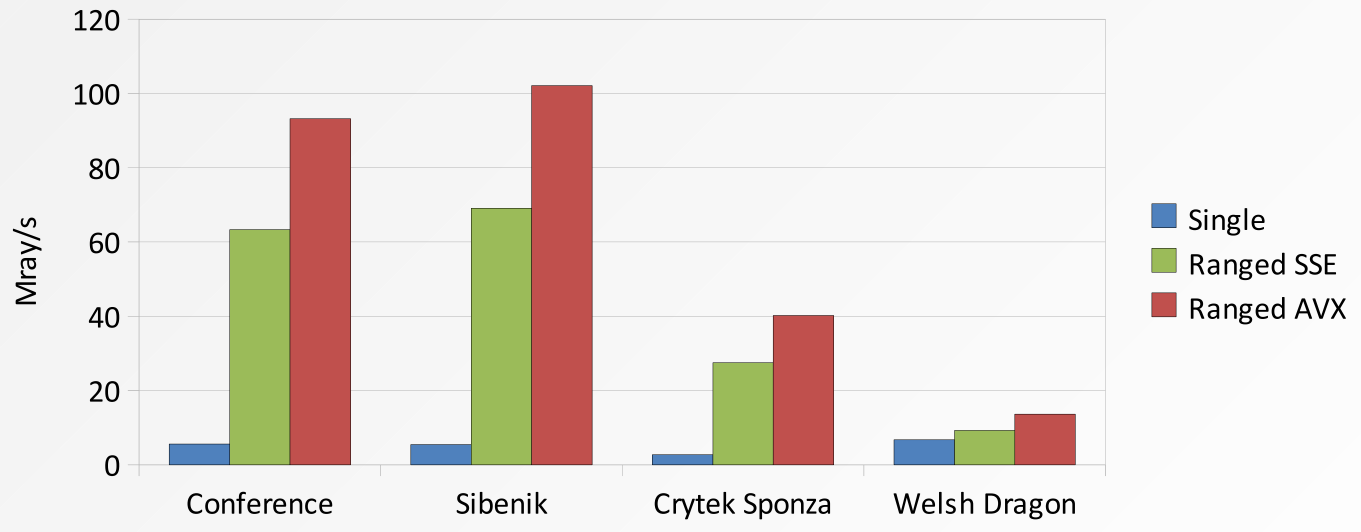
Results

The algorithms were implemented in C++ using AVX and SSE4.1 intrinsic functions. The code was compiled for 64 bits with Visual C++ 2010. All tests were run on a system with an **Intel Core i5-2400** processor (4 cores, 4 threads, 3.1 GHz) and Windows 7 SP1 64-bit. The rendering resolution was set to 1024×768 pixels.

For all models except the highly tessellated Welsh Dragon, ranged traversal with frustum culling is the fastest approach. **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.

We will continue our research with a primary focus on incoherent rays.

Ranged Traversal Performance for Primary Rays



Acknowledgements

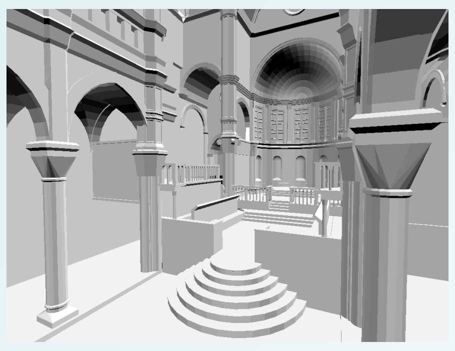
This work has been supported by the TeraTomo project of the National Office for Research and Technology, OTKA K-719922, and by TÁMOP-4.2.1/B-09/1/KMR-2010-0002. The Welsh Dragon model was released by Bangor University, UK, for Eurographics 2011.

References

- 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.
- OVERBECK R., RAMAMOORTHI R., MARK W. R.: Large ray packets for real-time whitted ray tracing. In *IEEE/EG Symposium on Interactive Ray Tracing 2008 (2008)*.
- WALD I., BOULOS S., SHIRLEY P.: Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies. *ACM Transactions on Graphics 26*, 1 (2007).

Benchmark Models

Conference (283K tris)



Sibenik (80K tris)

Crytek Sponza (279K tris)

Welsh Dragon (2.2M tris)

Table of Traversal Performance for Primary Rays

Model	Single	Ranged			Ranged w/o culling			Partition			Partition w/o culling		
	Mray/s	SSE Mray/s	AVX Mray/s	+ %	SSE Mray/s	AVX Mray/s	+ %	SSE Mray/s	AVX Mray/s	+ %	SSE Mray/s	AVX Mray/s	+ %
Conference	5.6	63.3	**93.2**	47.1	44.6	76.1	70.7	35.3	52.3	48.2	29.8	48.3	62.3
Sibenik	5.4	69.1	**102.1**	47.8	47.4	82.6	74.3	33.0	50.8	53.8	28.8	47.6	64.9
Crytek Sponza	2.7	27.5	**40.2**	46.3	20.2	34.3	70.1	16.1	23.2	44.2	14.0	22.4	60.3
Welsh Dragon	6.7	9.2	13.6	48.4	10.0	15.3	53.3	13.8	15.5	12.4	19.0	**22.9**	20.7

Measurements were executed for single ray traversal, ranged, and partition traversal, with and without frustum culling. The timings do not include ray generation and shading. The best value for each model is highlighted, and the speedups caused by AVX over SSE4.1 are also listed. The CPU used was an **Intel Core i5-2400**, and the resolution was 1024×768.