LITERATURE REVIEW: Optimizing Raytracing Algorithm Using CUDA

Sivakumaran Gowthaman
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
sivakumarangowthaman@cmail.carleton.ca

October 14, 2020

1 Introduction

While modern graphics cards (GPUs) allow for general computation in a parallel manner, one of the most prominent applications for a GPU is image synthesis. This is thanks to the inherent parallel nature of ray tracing and other global illumination algorithms – the decomposition of images into pixels provides a natural way of creating individual tasks for many parallel processors. Unlike the GPUs a one decade ago, modern ones allow us full programmability similar to general CPUs, while the streaming computation model has its own specific issues. This has to be taken into account when adopting the data structures and traversal algorithms for ray tracing on a GPU architecture. In this literature review three formerly published techniques that implement ray tracing with spatial data structures on a GPU are compared. These are uniform grids, kd-trees, and bounding volume hierarchies. While the algorithms were successfully mapped to a GPU, their performance have not been carefully compared on a current programmable GPU architecture as a common implementation framework was not available.

2 Literature Review

All three data structures of interest: uniform grids, kd-trees, and BVHs will be discussed here. At this point of time, I believe that the implementation of these three data structure in a GPU is within my ability. It might change as the project progress.

2.1 Uniform grids

The first ray tracing algorithm mapped fully on a GPU has been published by Purcell et al. uses a uniform grid. Their implementation mapped the computation by means of shaders while their data resided in a texture. In a concurrent work Carr et al. present the architecture of a software ray tracer on a GPU with a focus on ray-triangle intersection with predefined BVH hierarchy. The mapping of both mentioned approaches had been influenced by architectural limitations. Kalojanov and Slusallek presented the algorithm for parallel construction of uniform grids on a GPU.

2.2 Kd-trees

A stack on a GPU with a low level of programmability was studied by Ernst et al. and used for stack-based kd-tree traversal algorithm. Foley and Sugerman presented two algorithms for kd-tree traversal without a stack. Their first algorithm called kd-restart is in fact the algorithm published by Kaplan. The second stack-less algorithm called kd-backtrack requires the storage of the bounding box and link nodes to its parent for every node of a tree, which significantly increases the memory footprint and hence it decreases performance. Both presented algorithms increase the number of nodes traversed compared to stack-based traversal algorithms. Another paper by Horn et al. addresses the lack of local memory to implement the stack much more efficiently. They propose the use of a push-down and short stack which can avoid most of the restarts of a traversal from the root node. This is possible as ray tracing with the kd-tree traverses only a few leaves on average. In concurrent work Popov et al. suggest to use the augmentation of a data structure by neighbor links among the nodes of a kd-tree. They even exceed the performance of CPU-based ray tracers while they achieve comparable performance as in. Further, Zhou et al. proposed the algorithm for kdtree construction on a GPU. This method yields the performance of kd-tree construction comparable to CPU based algorithms for kd-tree construction. This can be used for dynamic scenes up to 200,000 triangles to yield interactive performance.

2.3 Bounding Volume Hierarchies

Bounding volume hierarchies (BVHs) were also successfully implemented on a GPU. Thrane and Simonsen infact compare kd-trees, uniform grids, and bounding volume hierarchies implemented on a GPU (hardware of year 2005). They conclude the performance of BVHs is low, however higher than the performance of other two data structures when no ray packets are used. Carr et al. implemented a variant of BVHs in combination with geometry images. Günther et al. use ray packets and yield interactive performance comparable or exceeding CPU-based implementation, but only for primary and shadow rays. Lauterbach et al. present an algorithm for fast BVH construction on a GPU, where they report performance comparable to kdtrees only for one scene. Torres et al. published an algorithm for stack-less BVH traversal, where the use of stack is replaced by ropes connecting the nodes of a BVH in a sibling order.

3 References

[1] Aila, T., and Laine, S.. Understanding the Efficiency of Ray Traversal on GPUs. In Proceedings of High-Performance Graphics 2009, pages 145–150, New York, NY, USA, 2009. ACM.

[2] Amanatides, J., and Woo, A. A fast voxel traversal algorithm for ray tracing. In G. Marechal, editor, Eurographics '87, pages 3–10. North- Holland, August 1987.

[3] Carr, N.A., Hall, J.D., and Hart, J.C. The ray engine. In HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, pages 37–46, Aire-la- Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[4] Carr, N.A., Hoberock, J., Crane, K., and Hart, J.C. Fast GPU ray tracing of dynamic meshes using geometry images. In GI '06: Proceedings of Graphics Interface 2006, pages 203–209, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[5] Ernst, M., Vogelgsang, C., and Greiner, G.Stack implementation on programmable

graphics hardware. In Vision Modeling and Visualization 2004, pages 255–262, 2004.
[6] Foley, T., and Sugerman, J. KD-tree acceleration structures for a GPU raytracer. In HWWS '05: Proceedings of the ACM SIGGRAPH/ EUROGRAPHICS conference on Graphics hardware, pages 15–22, New York, NY, USA, 2005. ACM.
[7] Günther, J., Popov, S., Seidel, H.-P., and Slusallek, P. Realtime Ray Tracing on GPU with BVH-based Packet Traversal. In Proceedings of the IEEE/Eurographics Symposium on Interactive Ray Tracing 2007, pages 113–118, September 2007.
[8] Havran, V., P ˇrikryl, J., and Purgathofer, W.Statistical Comparison of Ray-Shooting Efficiency Schemes. Technical Report TR-186-2-00-14, Institute of Computer Graphics, Vienna University of Technology, Favoritenstrasse 9/186, A-1040 Vienna, Austria, May 2000.
[9] Havran, V. About the Relation between Spatial Subdivisions and Object Hierarchies Used in Ray Tracing. In 23rd Spring Conference on Computer Graphics (SCCG 2007), pages 55–60, Budmerice, Slovakia, May 2007.
[10] Horn, D.R., Sugerman, J., Houston, M., and Hanrahan, P. Interactive k-D Tree GPU Raytracing.
[11]Alejandro Segovia, Xiaoming Li, Guang Gao. Iterative Layer-Based Raytracing on CUDA. IEEE 28th International Performance Computing and Communications Conference 2009.
[12]Santiago Cioli, Gonzalo Ordeix, Eduardo Fernández, Martin Pedemonte, Pablo Ezzatti. Improving the performance of a ray tracing algorithm using a gpu. International Conference of the Chilean Computer Science Society 2010.
[13]Martin Zlatuška, Vlastimil Havran. Ray Tracing on a GPU with CUDA – Comparative Study of Three Algorithms. Communication Papers Proceedings: 18th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS, p. 69-76.WSCG 2010.
[14]Sayed Ahmadreza Razian, Hossein MahvashMohammadi. Optimizing Raytracing Algorithm Using CUDA. Emerging Science Journal 2017.
[15]Xiaozi Guo, Juan Zhang, Mingquan Zhou. Fast Parallel Bounding Volume Hierarchy Construction. Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC) 2020.
[16]Daniel Meister, Jakub Bokšanský, Michael Guthe, and Jiří Bittner. On Ray Reordering Techniques for Faster GPU Ray Tracing. In Symposium on Interactive 3D Graphics and Games (I3D '20), May 5–7, 2020, San Francisco, CA, USA. ACM, New York, NY, USA, 9 pages.2020.