Compact, Fast and Robust Grids for Ray Tracing

Ares Lagae & Philip Dutré*
Department of Computer Science
Katholieke Universiteit Leuven

	Armadillo	Atrium	Dragon	Conference	Buddha	Cruiser	Asian Dragon	Thai Statue	Lucy	Nature	David
										
scene statistics											
#tri's	345 K	560 K	871 K	988 K	1.09 M	3.64 M	7.22 M	10.0 M	28.1 M	41.4 M	56.2 M
memory	11.9 MB	19.2 MB	29.9 MB	33.9 MB	37.3 MB	125 MB	248 MB	343 MB	963 MB	1.39 GB	1.89 GB
compact grid statistics											
build time	0.05 s	0.08 s	0.11 s	0.12 s	0.14 s	0.39 s	0.80 s	1.17 s	3.15 s	9.12 s	5.81 s
render time	0.90 s	1.93 s	0.80 s	2.28 s	0.58 s	2.49 s	1.43 s	1.55 s	1.90 s	10.75 s	1.74 s
time to image	0.95 s	2.01 s	0.91 s	2.40 s	0.72 s	2.89 s	2.23 s	2.72 s	5.05 s	19.87 s	7.55 s
memory	8.32 MB	15.1 MB	21.9 MB	26.1 MB	27.5 MB	84.3 MB	155.3 MB	222.5 MB	606 MB	2.01 GB	1.17 GB
hashed grid statistics											
build time	0.07 s	0.10 s	0.19 s	0.19 s	0.22 s	0.72 s	1.22 s	1.76 s	4.77 s	21.23 s	8.92 s
render time	0.89 s	1.86 s	0.79 s	2.22 s	0.57 s	2.52 s	1.25 s	1.43 s	1.53 s	10.07 s	1.29 s
time to image	0.96 s	1.97 s	0.98 s	2.41 s	0.79 s	3.25 s	2.47 s	3.18 s	6.30 s	31.30 s	10.21 s
memory	3.49 MB	8.16 MB	10.0 MB	13.3 MB	12.9 MB	35.0 MB	50.8 MB	78.8 MB	199 MB	1.52 GB	379 MB

Figure 1: *Build time, render time, time to image, and memory usage for the compact grid method and hashed grid method presented in this work, for various scenes. Images were rendered using simple shading, at a resolution of 1024×1024. Timings were obtained on a 3 GHz Intel Xeon X5365 CPU.*

Abstract

Ray tracing is becoming more and more the method of choice for both offline global illumination simulations as well as interactive visualizations. Because intersecting a ray with all objects in a scene is usually very expensive, almost all ray tracers rely on acceleration structures, trading preprocessing time and memory for faster ray-object intersections.

The uniform grid was one of the first proposed acceleration structures. Over time, several other acceleration structures, such as bounding volume hierarchies and kd-trees, have been introduced. For static scenes, kd-trees are by many considered the best acceleration structure. Uniform grids usually perform worse than kd-trees, mainly because they are not adaptive. For dynamic scenes however, there is no consensus. The acceleration structure has to be rebuilt every frame, and rather than minimizing render time, the time to image, the sum of the build time and the render time, has to be minimized. Building a grid can be done in linear time, while other popular acceleration structures require super linear time. For dynamic scenes, a shorter build time can compensate for a longer render time. Therefore, a grid can result in a shorter time to image than other acceleration structures that are usually considered superior.

Algorithms are typically CPU-bound or memory-bound. The execution time of an algorithm that is CPU-bound mainly depends on the speed of the CPU, while the execution time of an algorithm that is memory-bound mainly depends on the access speed of the memory. Memory-bound algorithms can be made significantly faster just by reducing the memory footprint of the data they work on. Building a grid is memory-bound, while rendering is CPU-bound. Therefore, reducing the memory footprint of a grid can results in shorter build times.

*e-mail: {ares.lagae,philip.dutre}@cs.kuleuven.be

Uniform grids were used in one of the first systems for interactive ray tracing. Recent work on grids for ray tracing concentrated on fast traversal, parallelizing the build process, and choosing the grid size. In this work, we present two efficient methods for representing and building a grid.

The *compact grid* method consist of a static data structure for representing a grid with minimal memory requirements, more specifically exactly one index per grid cell and exactly one index per object reference, and an algorithm for building that data structure in linear time, that does not require additional memory.

The *hashed grid* method consists of a static data structure for representing a grid that reduces memory requirements even further, by using perfect hashing based on row displacement compression, and a fast algorithm for building that data structure.

Figure 1 shows several results. For example, the time to image and memory usage for the 1.89 GB *David* model is respectively 10.21 s and 379.94 MB using the hashed grid method. We show that the compact grid method and the hashed grid methods are more efficient in both time and space than traditional methods based on linked lists and dynamic arrays. We also investigate parallell grid building and rendering, we compare with other acceleration structures using the recent *bwfirt* benchmark, and we present a more robust grid traversal algorithm. We show that, for applications where time to image or memory usage is important, such as interactive ray tracing and rendering large models, the grid acceleration structure is an attractive alternative.

References

LAGAE, A., AND DUTRÉ, P. 2008. Compact, fast and robust grids for ray tracing. *Computer Graphics Forum (Proceedings of the 19th Eurographics Symposium on Rendering) 27*, 8.