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.

References