R-LODs: Fast LOD-Based Ray Tracing of Massive Models

Sung-Eui Yoon Christian Lauterbach Dinesh Manocha
http://gamma.cs.unc.edu/RAY

Introduction: In recent years, there has been a renewed interest in real-time ray tracing for interactive applications. This is due to many factors: firstly, processor speed has continued to rise at exponential rates as predicted by Moore's Law and is approaching the raw computational power needed for interactive ray tracing. Secondly, ray tracing algorithms can be highly parallelized on shared memory and distributed memory systems. Therefore, the current hardware trend towards desktop systems with multi-core CPUs and programmable GPUs can be used to accelerate ray tracing. Finally, recent algorithmic improvements that exploit ray coherence can achieve a significant improvement in rendering time [Reshetov et al. 2005; Wald 2004].

Our goal is to perform interactive ray tracing of massive models consisting of tens or hundreds of millions of triangles on current desktop systems. Such gigabyte-sized models are the result of advances in model acquisition, computer-aided design (CAD), and simulation technologies. Their complexity makes interactive visualization and walk-throughs a challenging task. Ray tracing has an important property in the context of rendering massive models: its asymptotic performance is logarithmic in the number of primitives for a given resolution. This is due to the use of hierarchical data structures such as bounding volume hierarchies or kd-trees. The asymptotic complexity makes ray tracing an attractive choice, especially for rendering of massive models.

However, we found that the logarithmic growth continues only as long as the system has sufficient memory to store the entire model and hierarchical data structures. As models grow much larger, the size of the hierarchical structure also increases linearly and the underlying ray tracer performs its computations in an out-of-core manner. A major trend in computing hardware has been the increasing gap between processor speed and memory speed. Moreover, disk I/O accesses are in general more than three orders of magnitude slower than main memory accesses. Because of these gaps, hardware advances are not expected to provide an efficient solution to the problem of ray tracing massive models.

Our approach: We propose a new algorithm to accelerate ray tracing of massive models using geometric levels-of-detail (LODs). Our approach computes simple and drastic simplifications, denoted *R-LODs*, of polygonal models. A R-LOD consists of a plane with material attributes (e.g. color), which is a drastic simplification of the descendant triangles contained in an inner node of the kd-tree. Each R-LOD is also associated with a surface deviation error which is used to quantify the projected screen-space error at runtime. The R-LODs have a compact representation and are tightly integrated with the kd-tree. We present a simple and efficient LOD error metric to bound the error for primary and secondary rays. Additionally, we use techniques based on spatial coherence and cache-oblivious layouts [Yoon and Manocha 2006] to improve the performance of our LOD based ray tracing algorithm. R-LODs also alleviate the temporal aliasing that can arise during ray tracing of highly tessellated models.

We have implemented and tested our system on two machines running Windows XP 32-bit and 64-bit with two dual-core Xeon CPUs and have evaluated its performance on three different models: a CAD double eagle tanker (81M triangles), a forest model (32M triangles), and a scanned St. Matthew model (128M triangles). The performance gain of our LOD based ray tracer is proportional to the reduction in the number of intersection tests and, more importantly, the working set size (See Fig. 2). The frame rate improvement varies from 2 times on models with small working set size to almost 20–50 times on models with very large working set size. Furthermore, we are able to render most of these models at 5–12 frames a second with primary rays and 1–8 frames a second when we include reflections and shadow rays. These results are shown in the video. These performances are measured at 512×512 with 2×2 super-sampling in 64-bit machine with two Xeon CPUs.

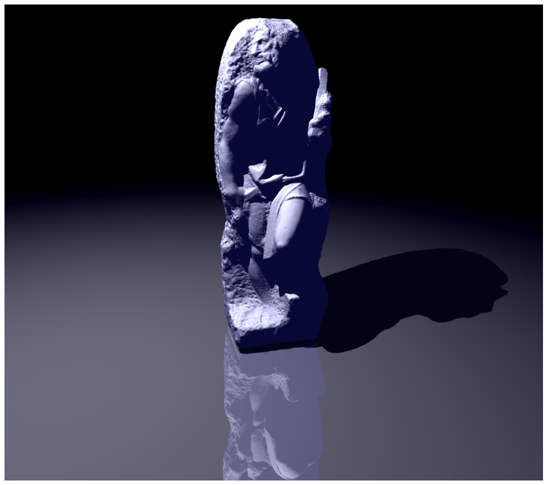

Figure 1: St. Matthew Model: *We use our LOD-based algorithm to accelerate ray tracing of St. Matthew model with shadows and reflections. We ray trace the* $128M$ *triangle model at* 512×512 *resolution with pixels-of-error (PoE)* $= 4$ *and* 2×2 *anti-aliasing. We are able to achieve* $2 - 3$ *frames per second on two dual-core Xeon processor workstation with 4GB of memory. We observe* $2 - 20$ *times increase in the frame rate due to R-LODs with very little loss in image quality.*

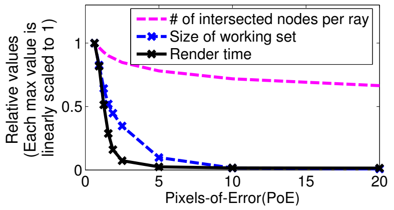

Figure 2: **Performance variation as a function of PoE**: *We measure the rendering time, average number of processed node per ray, and size of working set during rendering St. Matthew model with different PoE values. All these values are shown in a scale-invariant manner by linearly scaling their maximum values to* 1*. The performance of our LOD-based ray tracer drastically decreases as we linearly increase the PoE.*

Advantages of our approach: Our ray tracing algorithm offers the following *benefits*:

1. **Simplicity:** R-LODs are very easy to implement and their representation has small runtime overhead. Our algorithm maintains the simplicity, coherence, and performance of the kd-tree data structure.
2. **Front size:** R-LODs reduce the size of the front traversed in the kd-tree. This results in fewer ray intersection tests and decreases the size of the working set.
3. **Coherence:** R-LODs make memory accesses more coherent and reduce the number of L1/L2 cache misses and page faults. Furthermore, they can also improve the performance of spatial coherence techniques.
4. **Interactivity:** The LOD based ray tracer provides a framework for interactive ray tracing due to the fact that we can trade off image quality for improved frame rate.
5. **Generality:** Our algorithm is applicable to a wide variety of polygonal models, including scanned and CAD models.

References

RESHETOV, A., SOUPIKOV, A., AND HURLEY, J. 2005. Multi-level ray tracing algorithm. *ACM Trans. Graph. 24*, 3, 1176–1185.

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

YOON, S.-E., AND MANOCHA, D. 2006. Cache-Efficient Layouts of Bounding Volume Hierarchies. In *Computer Graphics Forum (Eurographics)*. Conditionally accepted.