BVH Construction, Update, and Traversal for Ray Tracing

Nicolas Feltman

October 14, 2011

1 Background

1.1 Acceleration Structures for Ray Tracing

The naive way to trace a ray is to test it against each primitive in the scene and report the intersection closest to the source of the ray. This is of course $O(n)$ on the number of primitive and would be completely infeasible for all but the smallest scenes. To overcome this complexity, the scene is carefully split into several smaller continuous parts, the union of which comprise the whole scene. Thus, rays are only tested against the primitives in a part of the scene should it be determined that the ray enters that part of the scene at all. Havran's PhD thesis [3] provides a good early comparison of various splitting schemes, although the landscape has changed considerably since them.

This project will focus on methods involving bounding volume hierarchies (henceforth referred to as BVHs). In a BVH, the set of objects in a scene are partitioned into two or more subsets, the exact number of which is called the branching factor, and a bounding volume is constructed around each subset. Each subvolume may be further split into smaller parts, thus forming a tree with groups of primitives at the leaves. There is no requirement that sub-volumes of a BVH not overlap, although minimizing this overlap is a goal in the construction of BVHs. Most BVH implementations use axis-aligned bounding boxes for bounding volumes, as the rest of this project will assume. Although initially found to be lacking in [3], BVHs have been an area of intense research. Their relative permissive invariant not only makes them well-suited for applications in animation, but also allows for easy implementation of some packet-traversal algorithms [7].

1.2 BVH Traversal

The basic observation behind a BVH is that a ray will only intersect a primitive object if it intersects every bounding box containing that object in the hierarchy. Thus, if a ray does not intersect the bounding box describing a particular node in the hierarchy, all objects under that node can be discarded. For a single ray, this implies a simple traversal algorithm. Starting at the root of the BVH, test for intersection with the bounding box of the node. If one exists, recur for every subnode and pick the closest of the childrens' intersections. If the bounding box is missed, or if every child node is missed, then return a miss for the node. At leaves, simply test for intersection with every contained primitive.

1.2.1 Subnode Test Order

This optimization makes use of the fact that an intersection in the near part of the ray obviates the need for further tests of the rest of the ray. If the children of a node do not overlap, which is rare, they can be tested in front-to-back order from the point-of-view of the ray, so that the first intersection will be the true intersection for that node. If the children do overlap, then there are two algorithmic options. The simpler option is to test the children nodes depth-first, skipping subnodes only when their bounding boxes lie entirely behind the closest intersection so far. This benefits from a simple *a priori* ordering, as described in [7]. The more complicated option is to traverse all

nodes in front-first order, which requires maintaining an active-node list, as described in [6].

1.2.2 Packets and SIMD

Packet tracing, originally explored for kd-trees, has been shown to be very effective in the BVH setting as well, and it is somewhat easier to implement packet methods for BVHs than for kd-trees because of the more relaxed invariants on BVHs. Although packet tests allow for very easy SIMD implementation and promote memory access locality, packet-based methods also admit some powerful algorithmic improvements, espcially speculative traversal. In [7], 70% to 95% of packet intersection tests were resolved either with early hit or early miss tests, without resorting to a full SIMD test of every ray. The early hit test, satisfied when the first ray in the packet is shown to hit the bounding box in question, short circuits tests for the rest of the rays in the packet, and causes the entire packet to be further intersected with child nodes. The early miss test, applied only if the early hit fails, is a conservative frstrum interval test used to reject entire packets. Both of these tests benefit from tight coherence of the rays in the packet.

When ray cohorence breaks down, as it tends to do with not-strictly-specular bounces, packet tracing methods lose their advantage. This can be mitigated with reordering methods ([2]) to recover coherence, or packets can be abandoned altogether at later stages in the trace. MultiBVH trees ([6]) take advantage of SIMD by using a branching factor equal to SIMD width, along with strict front-to-back traversal. They additionally have the somewhat unexpected bonus of low memory footprint.

1.2.3 Cotraversal

We will breifly attempt to motivate the advantages of early-hit tests in packet traversal. To our knowledge, this precise formulation is novel. Consider a BVH tree containing node $\mathcal{A}$, which has the child node $\mathcal{B}$, which in turn has the child $\mathcal{C}$. By the BVH invariant, this means that $bbox(\mathcal{C}) \subset bbox(\mathcal{B}) \subset bbox(\mathcal{A})$. One well-recognized impication of this is that rays which do not intersect $bbox(\mathcal{A})$ cannot intersect $bbox(\mathcal{B})$ or $bbox(\mathcal{C})$. Indeed this fact is what allows us to trace the scene quickly by ignoring whole subtrees. It also has a corollary: rays which intersect $bbox(\mathcal{C})$ must also intersect $bbox(\mathcal{B})$. Thus, if we should for one reason or another be very sure that a ray will make it to $\mathcal{C}$, we can speculatively skip the test for $\mathcal{B}$, gambling the cost of a box test. This hunch might be provided by the notion that several rays will sometimes take largely the same path in a tree; we call this phenomenon *ray cotraversal*. Packet techniques work well even in absense of SIMD because they exploit natural cotraversal between one ray in a coherent packet and the other rays of that packet. Obviously, as coherence deteriorates after bounces, so does cotraversal. We imagine there might be other ways to detect and exploit such cotraversal between rays in order to skip or collapse levels of the tree.

1.3 BVH Construction

The efficacy of a BVH in expediting trace computations is a product of both its traversal and its construction. Perfect construction of BVHs to minimize some cost metric is of course computationally infeasible. Nonetheless, it is fairly easy to make a very good tree in practice. The initial methods of bottom-up tree construction have largely given way to greedy top-down methods, in which the pool of objects is recursively split into smaller parts.

One such greedy top-down method utilizes the surface area heuristic (henceforth "SAH"). Noting that the number of randomly-distributed rays which enter a box is proportional to the surface area of that box, the SAH seeks to minimize $S_L C_L + S_R C_R$ on each level, where S_L and S_R are the surface areas of the left and right bounding boxes, and C_L and C_R are the cost of the associated subtrees. Since this splitting is done in a top-down order, the true cost of the subtrees is not known and so some function of the number of primitives is used instead. More details can be found in [4, 7].

While there has been considerable effort in constructing BVHs quickly according to greedy-SAH, little effort has been directed towards improving either the greedy part or the SAH. One such method, introduced in [4], involves iteratively tightening an initial

greedy build via subsequent tree rotations, overcoming the limitations of a greedy build alone. These rotations are selected to strictly minimize the SAH-based cost function. To overcome local minima, a simulated annealing process can be applied, at the cost of considerable upfront runtime. Although this method is quite clever, it only produces a modest 0% to 18% reduction in trace time over greedy-SAH alone. This fact, along with some of the experiments in [4] seem to suggest that SAH cost might not always be the best predictor of trace time.

There is also some recent work in utilizing knowledge about distribution of rays in the scene to form a better cost heuristic than the SAH. Bittner et al. [1] look at replacing the SAH with a ray density heuristic (RDH), a measure of how many rays (from a representative subset) pass through the potential bounding boxes, but they keep the greedy top-down build strategy. They find only modest improvements in kd-tree rendering times.

1.4 BVH Updates for Animations

One of the key obstacles in ray-tracing of animated scenes is keeping the acceleration structure updated after every frame of an animation. The simplest solution is to just build a brand new BVH each frame, but this technique is expensive to begin with, and does not scale well to large scenes, even assuming perfect SIMD utilization [5]. A more subtle method is to carefully update the BVH each frame. The first manifestation of this idea is deformable BVHs, in which each bounding box is refit every frame to contain all of its children. Since the topology of the BVH is not changed, it is possible for bounding boxes to stretch out in some circumstances, which hurts performance. One of the better early implementations of this concept can be found in [7].

Rotation BVHs, first implemented in [5], overcome the stretching issue of deformable BVHs by also including an afformentioned tree rotation update step, which can modify the topology of the tree to split stretched-out nodes. The authors note that tree rotations are much more effective at improving poor BVHs than tightening up already good ones. The imlpementation is very friendly to memory and produces high quality trees extremely cheaply. This BVH update technique may be fertile ground for futher research because it is not bound by the greedy top-down build strategy.

References

[1] Jiří Bittner and Vlastimil Havran. Rdh: ray distribution heuristics for construction of spatial data structures. In *Proceedings of the 2009 Spring Conference on Computer Graphics*, SCCG '09, pages 51–58, New York, NY, USA, 2009. ACM.

[2] Solomon Boulos, Ingo Wald, and Carsten Benthin. Adaptive ray packet reordering. *Symposium on Interactive Ray Tracing*, 0:131–138, 2008.

[3] Vlastimil Havran. *Heuristic Ray Shooting Algorithms.* Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.

[4] Andrew Kensler. Tree Rotations for Improving Bounding Volume Hierarchies. In *Proceedings of the 2008 IEEE Symposium on Interactive Ray Tracing*, pages 73–76, Aug 2008.

[5] Daniel Kopta, Andrew Kensler, Thiago Ize, Josef Spjut, Erik Brunvand, and Al Davis. Fast, effective bvh updates for dynamic ray-traced scenes using tree rotations. Technical Report UUCS-11-002, University of Utah, July 2011.

[6] Ingo Wald, Carsten Benthin, and Solomon Boulos. Getting Rid of Packets – Efficient SIMD Single-Ray Traversal using Multi-Branching BVHs. In *Proceedings of IEEE Symposium on Interactive Ray Tracing 2008*, 2008.

[7] Ingo Wald, Solomon Boulos, and Peter Shirley. Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Trans. Graph.*, 26, January 2007.