The Edge Volume Heuristic - Robust Triangle Subdivision for Improved BVH Performance

Holger Dammertz[*] Ulm University, Germany

Alexander Keller[†] mental images GmbH, Germany

ABSTRACT

The use of axis-aligned bounding boxes is a basic technique to accelerate geometric algorithms as for example ray tracing. It is a known problem that efficiency suffers, if the axis-aligned bounding volume contains major parts of empty space, which, in the case of ray tracing, causes more ray-object-intersection tests than required. The impact of this problem can be reduced by subdividing triangles at the cost of a larger memory footprint. We present a subdivision algorithm that is designed to generate only very few additional triangle references. Compared to previous approaches the algorithm is numerically robust, and simpler to implement and use. For formerly problematic scenes a speedup of up to a factor of 10 could be achieved, while the number of triangle references increased only by 16%.

Index Terms: I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1 INTRODUCTION

Improving the performance of ray tracing through the use of different acceleration structures has been investigated in detail [6]. Among the most successful acceleration structures is the *k*d-tree [7, 12]. Recent research demonstrated that bounding volume hierarchies (BVH) based on axis-aligned bounding volumes are competitive to or even outperform *k*d-trees [15, 4]. For both kinds of acceleration data structures fast traversal algorithms have been developed [14, 11, 1, 10].

An advantage of a BVH is the small memory footprint compared to a standard *k*d-tree, because each object is only referenced once.

Even though a BVH can be as fast as a *k*d-tree, there are scenes where the resulting performance is far worse. This is a direct consequence of the principle that every object should only be referenced once: Bounding boxes that contain large amounts of empty space increase the number of ray object intersections.

This problem becomes especially apparent for axis-aligned boxes enclosing non-axis-aligned geometry as it results from e.g. rotation: A triangle with a normal along one of the canonical axes has a zero volume axis-aligned bounding box, while any other orientation increases the volume and causes the triangle to be tested against more rays although the probability of hitting the triangle remains the same.

1.1 Previous Work

In the context of ray tracing with bounding volume hierarchies an advanced solution to the above problem has been investigated in [3] and was called early split clipping. The authors present a triangle splitting method based on the surface area of the triangle bounding volume and use an axis-aligned plane to split a triangle into three triangles (resulting from one triangle and one quadrangle). The approach of early split clipping reduces empty space contained in bounding volumes, which in addition reduces overlap and thus improves overall performance.

However, considering surface area also causes triangles to be split that are already tightly packed in a bounding volume (e.g. larger triangles with normals parallel to the canonical axes). In addition the splitting threshold is based on user experimentation per scene and triangles are split even when no speedup can be achieved.

While one might argue that the same benefits can be obtained using a *k*d-tree [14], especially when bounding the memory of the *k*d-tree construction [13], both approaches have to clip triangles against planes, which is a numerically tricky and costly operation.

1.2 Contribution

In the following we address some of these disadvantages by introducing a numerically robust triangle subdivision (in contrast to splitting) algorithm that only subdivides triangles where required thus keeping the biggest advantage of bounding volume hierarchies: The small memory footprint. This heuristic is based on the observation that not all large triangles effect the performance of a SAH [5] based BVH significantly but only the one that can't fit tightly into a bounding box. In addition we provide a simple heuristic to automatically choose the level of subdivision. This is especially important when animations are rendered and the scene configuration changes over time.

2 SUBDIVISION ALGORITHM

In order to improve the performance of bounding volume hierarchies we follow the idea of subdividing geometry and propose the economical edge volume heuristic that only moderately increases the memory footprint.

This new heuristic (see Section 2.1) measures the tightness of the bounding box of each triangle edge and subdivides the triangle until a certain threshold ε_v (see Section 2.2) is met.

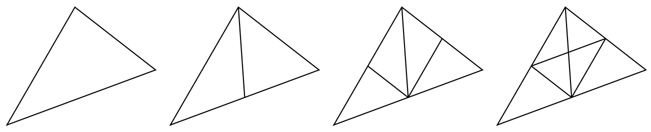


Figure 2: Recursive subdivision of a triangle along the longest edges. Although only edges are considered unaware of topology, the tessellation remains watertight if edge subdivision is symmetric.

2.1 Edge Volume Heuristic

For each edge of a triangle the subdivision algorithm determines its axis-aligned bounding box. The volume of the largest of the three boxes is compared to a volume threshold ε_v. If the volume is larger than the threshold the triangle is subdivided in the middle of this edge, which is easily implemented in a numerically robust

[*]e-mail: holger.dammertz@uni-ulm.de
[†]e-mail: alex@mental.com

IEEE/EG Symposium on Interactive Ray Tracing 2008
9 - 10 August, Los Angeles, California, USA
978-1-4244-2741-3/08/$25.00 ©2008 IEEE

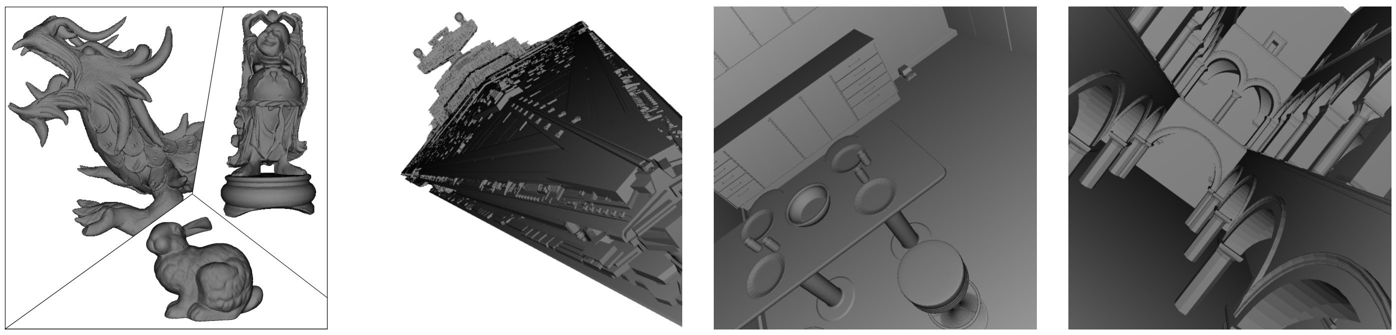


Figure 1: Images of the test scenes used for inspecting the subdivision heuristic. The space ship consists of 1554416 triangles. The kitchen has 110561 triangles and the Sponza Atrium 66454.

manner. The procedure is repeated for the two new triangles until it terminates (see the illustration in Figure 2).

The heuristic guarantees that without any knowledge of topology, identical operations will be performed on shared edges. Consequently the resulting bounding boxes fit without gaps, which overcomes precision issues of clipping and watertight meshes will remain watertight after subdivision. This is true for any symmetric subdivision of edges [9].

Note that using a heuristic not based on edges, like e.g. bounding box surface area, cannot guarantee watertight subdivision: Cracks can occur, because shared edges are not necessarily subdivided in an identical way. In addition a surface area criterion would divide large triangles regardless of the tightness of the bounding box, which is not memory efficient. In fact the edge volume heuristic is economical as it only subdivides triangles with very inefficient bounding boxes.

2.2 Determining the Edge Volume Threshold ε_v

The threshold

$$\varepsilon_v(t) := \frac{V}{2^t}$$

is determined as a fraction of the volume V of the scene bounding box and thus controls the amount of triangles generated by subdivision.

Over a broad range of scenes it turned out that choosing the threshold parameter $t = 14$ as a default value yields good results with respect to increased triangle references and performance. With this threshold value many scenes that already exhibit high ray tracing performance are not subdivided at all or the increase of triangle references is less than 1%. Thus it is safe to rely on a fixed parameter t. But as with any heuristic, specially constructed scenes may break the assumption of course (for example a single thin diagonal triangle). For this kind of scenes the user may have to choose the parameter t by hand.

In Section 3 the impact of varying t is quantified for different bounding volume hierarchy unfriendly scenes.

This simple threshold selection is of course not limited to the edge volume heuristic but can be used (with a different scale) for example for the early split clipping approach.

2.3 Implementation Details

The procedure can be applied either as a preprocess before ray tracing or for on-demand subdivision at the beginning of the tree construction. The first variant is especially useful as it can be used to upgrade any existing ray tracing module without modifying the internals. The second variant is transparent for the user and just produces a different tree during construction.

For each triangle the algorithm performs the recursive subdivision procedure. As the bounding volume hierarchy construction only uses the bounding boxes during construction, it is memory-efficient to output the bounding boxes with the original triangle reference instead and in place of the subdivided triangles.

Usually a BVH contains more than one triangle per leaf. An additional optimization during tree construction is to remove references to the same triangle in each leaf. In our experiments this resulted in a speedup of about 10%.

The scan over the triangles is so efficient, that it even pays off, to have a pass that only counts the number of generated triangles, to allocate memory accordingly, and to scan the data again to generate the bounding boxes.

The remaining problematic cases are overlapping bounding boxes that cannot be separated. This problem is ameliorated by the fact that efficient bounding volume hierarchies usually reference more than one triangle per leaf thus grouping some of the overlapping geometry in one box, which reduces the overall overlap.

3 Results

We apply the subdivision heuristic to a set of quite different scenes in order to verify its versatility (see Figure 1). The benchmarks were performed using an implementation of the QBVH [2] using single thread primary rays on a Intel Core2 Duo 2.4GHz processor.

We first consider four static scenes:

1. The Dragon, Buddha and Bunny scenes are just included to verify that for many small triangles of roughly the same size the heuristic does not add any new triangles in the used parameter range. This kind of scenes do not benefit from subdivision.

2. The space ship scene represents a kind of worst case scenario for classic bounding volume hierarchies: It consists of long thin triangles for the outer hull and many small triangles for details. Additionally this object is rotated by 45 degrees in space. The effect of different threshold parameters t is visualized in Figure 3.

3. The kitchen scene is one frame from the BART animation repository [8], where instead of moving the camera the triangles are transformed.

4. The Sponza atrium scene is a typical architectural model, where many triangles are parallel to the canonical planes. While classic heuristics like the surface area heuristic can build efficient bounding volume hierarchies for such scenes, performance drops dramatically, if geometry is rotated and bounding boxes increase in volume.

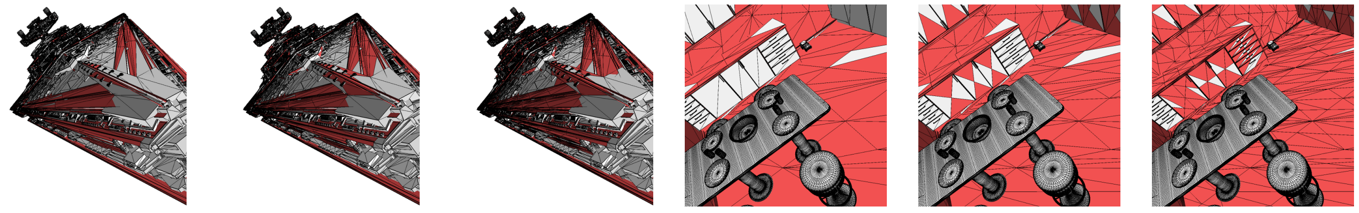


Figure 3: Visualization of the effect of different threshold parameters t, where the subdivided triangles are highlighted in red. Note how the long thin triangles with huge axis-aligned bounding boxes become subdivided, while others that would not hurt bounding volume hierarchy performance remain untouched (for the space ship this is observed best at the center of the images).

Scene	best	average	worst
Space Ship	0.25	0.435	1.0
Kitchen	0.25	0.463	0.992
Sponza (Frame 16)	0.25	0.457	0.99

Table 1: Factors of best, average, and worst surface area reduction resulting from applying the edge volume heuristic for triangle subdivision. The theoretical maximum of 0.25 is achieved in some cases.

A higher threshold parameter t improves performance, but also increases the memory footprint. Both numbers are related in Figure 4, where we show the relative increase in the number of triangles and corresponding rendering time for a range of thresholds parameters t. A clearly consistent improvement over the test scenes can be observed and it is especially interesting that major performance improvements are obtained at already a moderate increase of the number of triangles.

The second test consists of rotating the well known Sponza atrium scene to illustrate the adaptivity of the edge volume heuristic. The scene is first rotated by $90°$, $20°$, and $30°$ around the x-, y-, and z-axis in 32 steps. Second it is rotated another 32 steps to it's final position $-180°$, $0°$, and $90°$ where all large triangles are again axis aligned. Figure 5 shows the performance figures and the number of generated triangle references over the animation for several threshold parameters t. Again the heuristic proves to be reliable: In simple cases no triangles are added. When bounding boxes become inefficient a moderate increase in the number of triangle references avoids the dramatic performance drop.

Subdividing an edge in the middle results in two new bounding boxes that each have $\frac{1}{8}$ of the original volume, because the split edges remain diagonals of their bounding boxes. Since our tree construction is based on the SAH it is interesting to look at the reduction of the triangle's bounding box surface area upon subdivision and corresponding statistics are collected in Table 1.

Even though the factor of the worst area reduction is quite large the average area reduction shows the effectiveness of the heuristic.

4 Conclusion

We introduced an economical heuristic to subdivide triangles such that the amount of empty space in bounding boxes is efficiently reduced. The algorithm is simple, numerically robust, and can be used as a topology unaware preprocess to any renderer. Big performance improvements already result from very moderate additional memory requirements.

While the technique certainly has applications in collision detection and occlusion culling, too, there are two more points of future interest: There are situations, where a global volume threshold ε_v may be not sufficient and a local threshold may perform better. Furthermore there are situations where neither our new heuristic nor the surface area heuristic can reduce overlap. We would like to find criteria to easily identify these situations.

Acknowledgements

The first author would like to thank mental images GmbH for support and funding of this research. The second author would like to thank Henry Moreton for the comments on the symmetric subdivision. We would also like to thank Ansel Hsiao for the space ship model and the Stanford Scanning Repository for the Dragon Buddha and Bunny model.

References

[1] C. Benthin. *Realtime Ray Tracing on Current CPU Architectures*. PhD thesis, Saarland University, 2004.

[2] H. Dammertz, J. Hanika, and A. Keller. Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays. In *Rendering Techniques 2008 (Proc. 19th Eurographics Symposium on Rendering)*, page to appear, 2008.

[3] M. Ernst and G. Greiner. Early split clipping for bounding volume hierarchies. In *Proc. 2007 IEEE/EG Symposium on Interactive Ray Tracing*, pages 73–78, 2007.

[4] M. Geimer. *Interaktives Ray Tracing*. PhD thesis, Koblenz-Landau University, Germany, 2006.

[5] J. Goldsmith and J. Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics & Applications*, 7(5):14–20, 1987.

[6] V. Havran. *Heuristic Ray Shooting Algorithms*. PhD thesis, Czech Technical University, 2001.

[7] V. Havran and J. Bittner. On improving kd-trees for ray shooting. *Journal of WSCG*, 10(1):209–216, 2002.

[8] J. Lext, U. Assarsson, and T. Möller. BART: A benchmark for animated ray tracing. Technical report, Chalmers University of Technology, 2000.

[9] H. Moreton. Watertight tessellation using forward differencing. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, pages 25–32, 2001.

[10] A. Reshetov. Faster ray packets - triangle intersection through vertex culling. In *Proc. 2007 IEEE/EG Symposium on Interactive Ray Tracing*, pages 105–112, 2007.

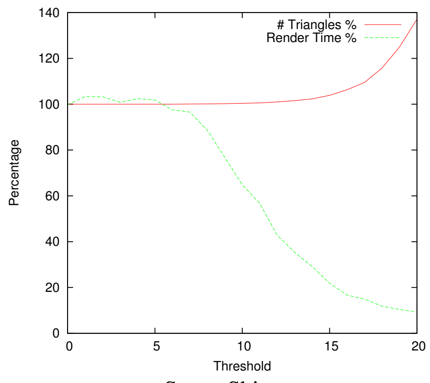
[11] A. Reshetov, A. Soupikov, and J. Hurley. Multi-level ray tracing algorithm. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, pages 1176–1185, 2005.

[12] M. Shevtsov, A. Soupikov, and A. Kapustin. Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. In *Computer Graphics Forum (Proc. Eurographics 2007)*, pages 395–404, 2007.

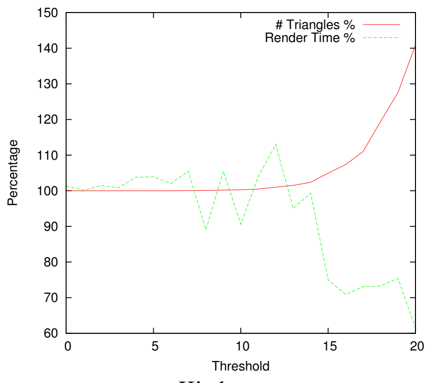
[13] C. Wächter and A. Keller. Terminating spatial partition hierarchies by a priory bounding memory. In *Proc. 2007 IEEE/EG Symposium on Interactive Ray Tracing*, pages 41–46, 2007.

[14] I. Wald. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Saarland University, 2004.

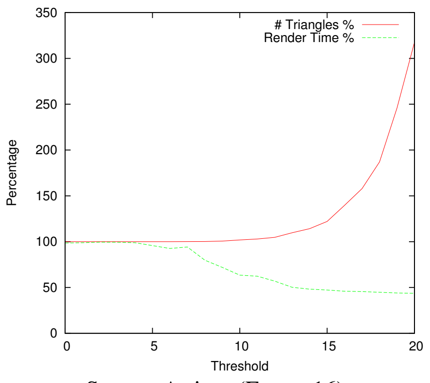
[15] I. Wald, S. Boulos, and P. Shirley. Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Transactions on Graphics*, 26(1), 2007.



Space Ship

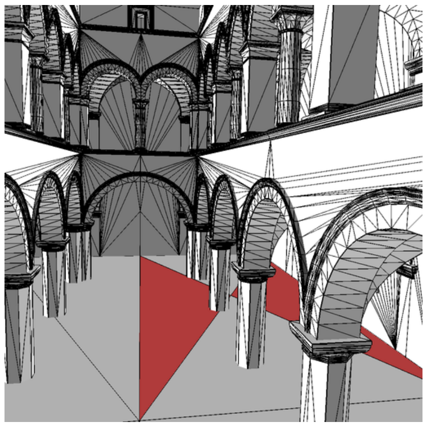


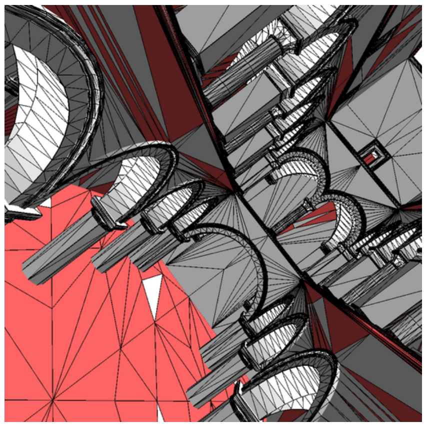
Kitchen

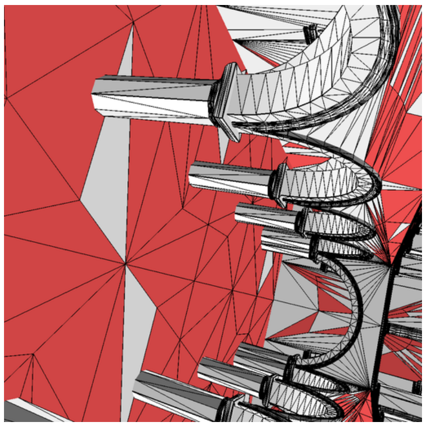


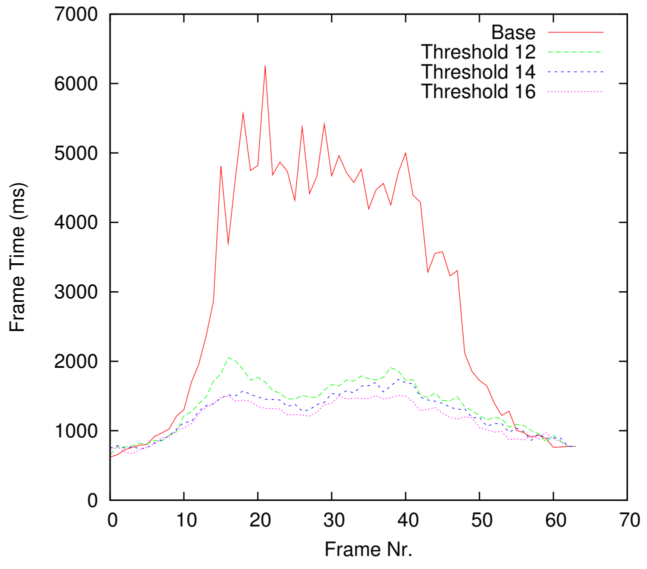
Sponza Atrium (Frame 16)

Figure 4: Relative performance for three example scenes. As expected render time improves until it asymptotically reaches saturation and the number of triangles increases with an asymptotically exponential behavior. The consistent behavior over quite different scenes clearly shows dramatic performance improvements at already very moderate increase in the number of triangles.

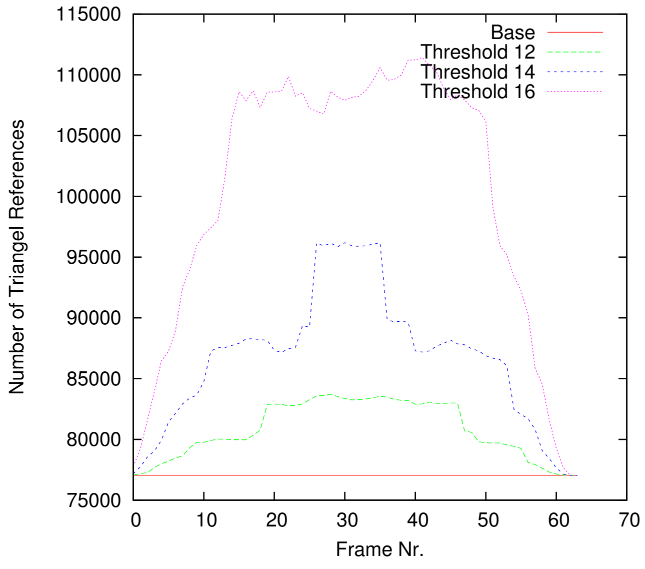

Frame 1


Frame 16


Frame 32



Performance



Triangle References

Figure 5: The Sponza atrium under rotation. The top row shows 3 example frames out of the total 64 frames, where subdivided triangles are highlighted in red. The more the architectural model becomes rotated, the more bounding boxes of previously axis-aligned geometry become inefficient, which is reliably avoided by the subdivision heuristic (threshold parameter $t = 14$). The graphs in the bottom row show how the frame time is improved and how many triangles are added by the subdivision heuristic for the unsubdivided geometry (Base) and three subdivision thresholds over the 64 frames of the animation.