Bounding Volume Hierarchy Optimization through Agglomerative Treelet Restructuring

Leonardo R. Domingues
Helio Pedrini

Eldorado Research Institute
Institute of Computing - University of Campinas
Brazil

High-Performance Graphics Conference
August 07, 2015

Summary

1. Introduction
2. Related Work
3. Methodology
4. Results
5. Conclusion

1 Introduction

2 Related Work

3 Methodology

4 Results

5 Conclusion

Construction time x structure quality

- Lower construction time is important for:
  - Animated scenes
  - Interactive applications
- Higher quality is important for:
  - Tracing a large number of rays
- GPU methods typically increase speed but reduce quality

Objectives

- Expand on the current state of the art[1]
- Test heuristics to replace exhaustive search
- Reduce construction times further
- Keep quality competitive with the most time demanding algorithms

[1] T. Karras and T. Aila. Fast parallel construction of high-quality bounding volume hierarchies. High-Performance Graphics Conference, pages 89-99. ACM, 2013.

1 Introduction

2 Related Work

3 Methodology

4 Results

5 Conclusion

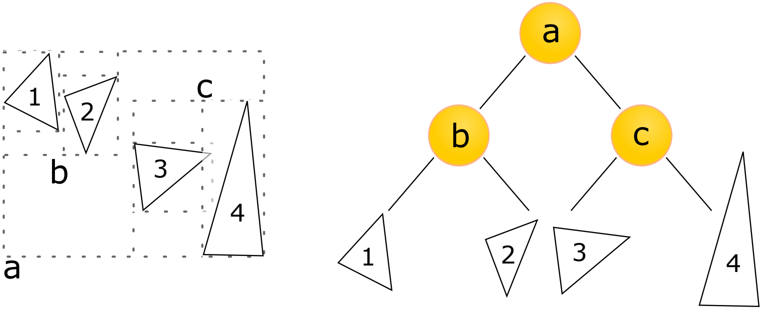


Figure: Bounding Volume Hierarchy

The quality of a BVH can be measured using the Surface Area Heuristic (SAH)

$$\text{SAH} = \frac{1}{A_t}\left(C_i \sum_{n \in I} A(n) + C_t \sum_{n \in L} A(n) N(n)\right) \quad (1)$$

- A_t = surface area of the root node
- $A(n)$ = surface area of node n
- C_i = relative cost for traversing an internal node
- C_t = relative cost for performing ray-triangle intersection
- I = set of internal nodes
- L = set of leaves
- $N(n)$ = number of triangles referenced by leaf n

GPU construction:

- Faster methods
- Lower quality trees
- LBVH[2] is the fastest method:

LBVH

- Sort triangles along the Z curve
- Split the sorted array to create the internal nodes

[2] C. Lauterbach, M. Garland, S. Sengupta, D. P. Luebke, and D. Manocha. Fast BVH construction on GPUs. Computer Graphics Forum, 28(2):375-384, Apr. 2009.

Karras and Aila (2013) is the state of the art on GPU optimization

TRBVH

- Treelets = small neighborhood of nodes
- Bottom-up traversal of the tree
- Form a treelet for each traversed node
- Restructure treelet nodes to find the optimal topology

1 Introduction

2 Related Work

3 Methodology

4 Results

5 Conclusion

- Improvement on TRBVH
- Approximate the search for the optimal treelet structure
  - Greedy
  - Bottom-up
  - Agglomerative

Algorithm 1: RearrangeTreelets

```
for internal node i in BVH do
    treelet ← FormTreelet(i)
    clusters ← treeletLeaves
    while length(clusters) > 1 do
        distances ← []
        foreach pair of clusters (x, y) do
            d ← Dissimilarity(x, y)
            distances ← (d, x, y)
        end
        (m, n) ← FindMinimumDistance(distances)
        o ← MergeClusters(m, n)
        clusters.remove(m)
        clusters.remove(n)
        clusters.add(o)
    end
end
```

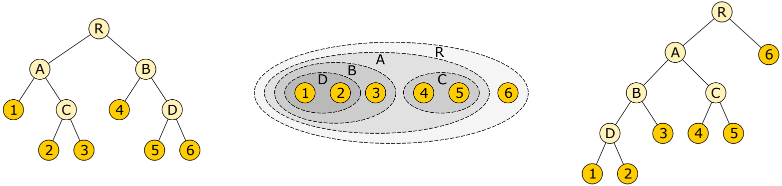


Figure: Agglomerative treelet restructuring

Distance Metric

Surface area of the bounding box containing the two nodes

- Minimizes SAH

Distance Cache

- Cache distances in a triangular matrix
- Borrowed from Gu et al. (2013)[3]

[3] Y. Gu, Y. He, K. Fatahalian, and G. Blelloch. Efficient BVH construction via approximate agglomerative clustering. In Proceedings of the High-Performance Graphics Conference, pages 81-88. ACM, 2013.

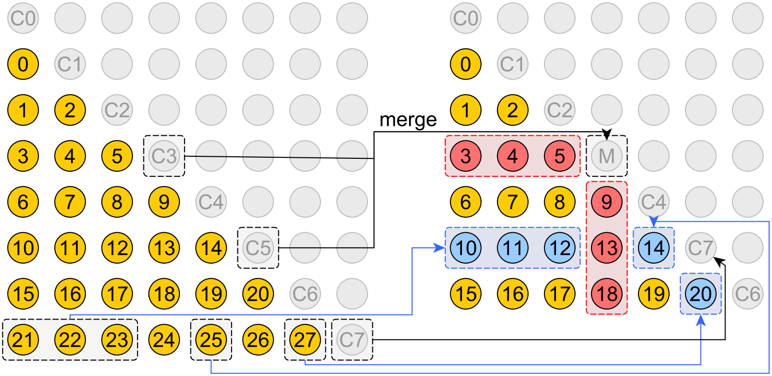


Figure: Updating the distance matrix

- Store modifications in a list
- Check if the new structure will reduce the tree SAH
- Only apply changes that improve the tree SAH

- Collapse the tree
- More than one triangle per node
- Compare cost of the subtree with cost of the collapsed subtree

Cost of the collapsed tree

$$c = C_t A(n) N(n) \tag{2}$$

- C_t = Relative cost of ray-triangle intersection
- $A(n)$ = Surface area of node n
- $N(n)$ = Number of triangles contained in the subtree

1 Introduction

2 Related Work

3 Methodology

4 Results

5 Conclusion

Compared methods

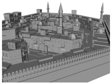
- LBVH
- TRBVH
- ATRBVH

- TRBVH is not publicly available
- Build times for our TRBVH are 3x higher than reported by the authors
    - GTX 770 x GTX Titan
    - Low-level optimizations
- TRBVH and ATRBVH share a large similarity

- NVIDIA GTX 770
- 16 scenes
- Aila et al. (2009)[4](2012)[5] traversal framework
- LBVH as basis for optimization
- Treelet size of 9 for ATRBVH
- All trees are collapsed as post-processing

[4] T. Aila and S. Laine. Understanding the efficiency of ray traversal on GPUs. In Proceedings of the High-Performance Graphics Conference, pages 145-149. ACM, 2009.

[5] T. Aila, S. Laine, and T. Karras. Understanding the efficiency of ray traversal on GPUs - Kepler and Fermi addendum. NVIDIA Technical Report NVR-2012-02, NVIDIA Corporation, June 2012.

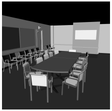


Arabic (412K)

Method	Mrays/s	Time (ms)	SAH	Relative (%)
LBVH	41.73	10.61	127.15	65.27
TRBVH	63.93	56.75	77.89	100.00
ATRBVH	61.50	38.44	77.45	96.20

Buddha (1.1M)

Method	Mrays/s	Time (ms)	SAH	Relative (%)
LBVH	75.31	23.53	89.14	85.11
TRBVH	88.49	134.64	70.38	100.00
ATRBVH	87.67	90.33	70.73	99.07

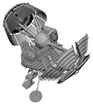


Conference (282K)

Method	Mrays/s	Time (ms)	SAH	Relative (%)
LBVH	98.14	8.59	65.53	71.45
TRBVH	137.36	39.25	39.53	100.00
ATRBVH	139.67	27.76	38.93	101.68

Dragon (870K)

Method	Mrays/s	Time (ms)	SAH	Relative (%)
LBVH	84.13	19.64	75.50	89.19
TRBVH	94.33	105.61	62.04	100.00
ATRBVH	94.23	72.21	62.10	99.89



Time Machine (4.7M)

Method	Mrays/s	Time (ms)	SAH	Relative (%)
LBVH	9.43	95.29	308.61	86.91
TRBVH	10.85	583.55	248.99	100.00
ATRBVH	11.21	415.50	246.78	103.32

Welsh Dragon (2.2M)

Method	Mrays/s	Time (ms)	SAH	Relative (%)
LBVH	58.59	40.92	90.11	89.23
TRBVH	65.66	256.53	73.51	100.00
ATRBVH	65.18	176.62	74.38	99.27

Table: Test results

Method	Performance (%)	Time (%)
LBVH	80.8	20.3
TRBVH	100	100
ATRBVH	99.7	69.5

Table: Results averaged over all test scenes

1 Introduction

2 Related Work

3 Methodology

4 Results

5 Conclusion

ATRBVH

- 30% faster than TRBVH
- Virtually same quality (99.7%)
- Implementation publicly available[6]

[6] https://github.com/leonardo-domingues/atrbvh

- Treelet sizes > 32
- Dynamically adjust treelet size
- Test with triangle splitting

leonardo.domingues@eldorado.org.br
helio@ic.unicamp.br

Bounding Volume Hierarchy Optimization through Agglomerative Treelet Restructuring

Leonardo R. Domingues
Helio Pedrini

Eldorado Research Institute
Institute of Computing - University of Campinas
Brazil

High-Performance Graphics Conference
August 07, 2015