Eurographics Symposium on Rendering 2015
J. Lehtinen and D. Nowrouzezahrai
(Guest Editors)

Volume 34 (2015), Number 4

MBVH Child Node Sorting for Fast Occlusion Test

Shinji Ogaki and Alexandre Derouet-Jourdan

OLM Digital, Inc. / JST, CREST

Figure 1: *Semi-outdoor scene containing1.2 million triangles. The heat views show the summed number of traversed nodes and primitive intersection tests for shadow rays of our algorithm (middle) and SATO (Surface Area Traversal Order) (right).*

Abstract

Optimal BVH layout differs among ray types. To accelerate shadow rays, the use of a specialized traversal order, optionally with an additional data structure has been proposed. In this paper we show how sorting child nodes of MBVH (Multi Bounding Volume Hierarchy) improves the performance of occlusion test without changing the topology of the data structure. We introduce a cost metric suitable for MBVH which takes into account the distribution of representative rays, and prove that the cost can be minimized by sorting child nodes based on a very simple criterion. Our method is very easy to implement and requires only small amounts of storage and preprocessing time for sorting. We also demonstrate how rendering performance can be improved by up to 10% in conjunction with various algorithms.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

Occlusion test plays an important role in ray-based rendering for realistic image synthesis. Closest-hit tests are performed for radiance rays and any-hit tests for occlusion rays. The purpose of the former is to find the closest intersection point from a ray origin whereas the purpose of the latter is to check if there is any object in a given direction within a given length. Therefore, the topology of acceleration data structure and traversal order should be determined using different cost metrics for both ray types. They also should vary depending on a scene to be rendered. More than one BVHs thus needed to achieve the best rendering performance. However, it is preferable to avoid having multiple data structures to store more geometry, textures, etc. It is also very important to minimize the BVH construction cost because interactive feedback is crucial for artists. Optimizing a single BVH for shadow rays ends up degrading the performance of closest-hit test.

We overcome this problem with a very simple idea: sorting child nodes of MBVH using the distribution of representative shadow rays. We only change the order of child nodes and the topological structure of MBVH remains intact. Therefore, the performance of closest-hit test is not affected as the SAH (Surface Area Heuristic) is not changed. Utilizing the dead space of an MBVH node, which is used for cache line alignment, no extra memory is required.

	RTSAH	SRDH	SATO	Ours
BBVH	✓	✓	✓	✓
MBVH	non-trivial	non-trivial	✓	✓
Ray distribution	✗	✓	✗	✓

Table 1: *Our method works for BBVH and MBVH, and also takes into account the distribution of shadow rays. On the other hand, the state of the art techniques RTSAH [IH11], SRDH [FLF12], and SATO [NM14] have limitations.*

2. Related Work

Traversal order has a huge impact on rendering performance. The most common approach for closest-hit test is depth-first traversal order. When a ray hits n child nodes of an MBVH node, their indices (or pointers) are pushed onto a stack. Then the top n elements on the stack are sorted [Áfr13] [WFWB13]. This approach provides a desirable front-to-back traversal order when SBVH [SFD09] is used. When splitting is not used, the closest node can be found by sorting all intersected nodes using minimum heap. This approach is very helpful for heavy ray-primitive intersection tests such as ray-parametric surface intersection test because the numbers of node traversal steps and primitive intersection tests are reduced. For triangle meshes, sorting all intersected nodes exhibits a poor performance and the previous approach is hence preferred. In general sorting takes a large part of closest-hit tests which becomes more noticeable as the SIMD width increases [Áfr13].

Regarding occlusion rays, a better performance can be achieved by using a different traversal order with a specialized data structure. RDH (Ray Distribution Heuristics) [BH09] uses representative rays and builds BVH based on their distribution to accelerate general ray traversal. This method only gives a subtle speedup. The idea is later adopted by SRDH (Shadow Ray Distribution Heuristic) [FLF12] to improve the performance of occlusion test by limiting the use of representative rays to shadow rays. After recording their distribution, another BVH tailored for shadow rays is created based on the SRDH cost metric. The performance gain is relatively large but it comes at the cost of extra memory and BVH construction time. OSAH [VHS12] is somewhat similar to SRDH. This metric takes into account visibility. Exploiting approximated visibility of triangles, a BVH optimized for visibility test is constructed. Ize et al. introduced a cost metric called RTSAH (Ray Termination SAH) [IH11] suitable for occlusion ray traversal. This method does not use representative rays and no additional data structure is built. The traversal order of shadow rays is stored in each node as a boolean flag. SATO (Surface Area Traversal Order) [NM14] also does not require representative rays and simply uses the surface area of the bounding box of each child node to determine which to traverse first. The preprocessing time for NodeSATO is rather fast. However, as reported in the paper it does not necessarily accelerate occlusion tests because the distribution of occlusion rays is not taken into account. The comparison among the state of the art techniques is given in Table 1.

3. Child Node Sorting

Similar to SRDH [FLF12], our method consists of two stages. The first stage is to cast representative shadow rays and record which nodes occlude them. Instead of determining the traversal order at run time, N child nodes in all MBVH nodes are sorted so that the number of traversal steps is reduced. After sorting, we continue rendering with the modified MBVH (the second stage). Note that we assume that nodes are traversed in a depth-first order. In the following subsections we elaborate our algorithm.

3.1. Surface Area Heuristic

Before casting representative rays, we build an MBVH. This is done using the SAH cost metric [MB90] given as

$$\begin{aligned} C_{SAH}(node) &= C_t + C_1P_1 + C_2P_2 + \cdots + C_NP_N, \\ C_{SAH}(leaf) &= C_t + N_pC_p, \end{aligned}$$

where C_i is the cost of the i-th child node, N_p the number of primitives in a leaf node, C_p the cost of ray-primitive intersection test, and C_t the cost of node traversal. We set $C_t = C_p = 1.0$. By letting A_{node} and A_i be the surface areas of a node and its i-th child node, the probability of rays intersecting the i-th child node P_i is approximated by A_i/A_{node}. In our rendering system a binary BVH is first built without splitting primitives and then collapsed as in [DHK08] [WBB08].

3.2. Logging

Once an MBVH is constructed, representative shadow rays are cast. When a representative ray hits a leaf node, we increment the counter associated with it. We use the structure given below for an MBVH node like in [Áfr13]. The variable *padding* is added for cache line alignment and left unused. Therefore, the numbers of visits by representative shadow rays are stored in *padding* to effectively use this dead space. If there is no such space, a $4 \times N$ byte array should be allocated per node.

```
template <int N>
struct Node {
  float minBound[3][N];
  float maxBound[3][N];
  int ids[N];
  int padding[N];
};
```

In our practical implementation, a counter of leaf node is only incremented to reduce the number of atomic operations. The value of each counter is later propagated to its parent

Algorithm 1 Sorting MBVH child nodes

input: N-ary tree with Q_i at each node.
output: Optimized cost of the tree. The tree is sorted.
if tree is a leaf **then**
  return $C_{SORT}(leaf)$.
else
  for each child **do**
    $C_{child} \leftarrow$ *recursive call for child*
  end for
  Sort the children so that $\frac{P_{child}}{C_{child}}$ is decreasing.
  return $C_{SORT}(node)$.
end if

node. The resulting performance slightly changes depending on the traversal order used in the logging phase. Unless specified otherwise, we use SATO (NodeSATO).

3.3. Sorting

Child nodes are sorted so that the cost of occlusion tests becomes minimal (see Algorithm 1). Letting Q_i be the probability of a ray being occluded by the i-th child node, the cost of occlusion tests is recursively determined by

$$\begin{aligned} C_{SORT}(node) &= C_t \\ &\quad + \sum_{i=1}^{N} \left(\sum_{j=1}^{i} C_j \right) Q_i \prod_{k=1}^{i-1} (1 - Q_k) \\ &\quad + \left(\sum_{i=1}^{N} C_i \right) \prod_{i=1}^{N} (1 - Q_i), \\ C_{SORT}(leaf) &= C_t + C_p N_p \end{aligned} \tag{1}$$

as illustrated Figure 2, where C_i is the expected cost of the i-th child node. We model it as

$$C_i = P_i \times C_{SORT}(i\text{-th child node}), \tag{2}$$

where P_i is the probability of intersecting the bounding box of the i-th child node. We do not record precisely this probability to minimize the memory footprint and the number of atomic operations. Instead, we approximate it by either $P_i = A_i / A_{node}$ or by the worst case scenario $P_i = 1$. We report the comparison of the two models in the next section.

This cost metric is inspired by SRDH [FLF12] and provides a fast way of reordering the child nodes of the MBVH. Finding the order of child nodes that minimizes $C_{SORT}(node)$ seems to be infeasible because there are $N!$ ways of ordering. However, it can be shown that sorting in a descending order by Q_i/C_i yields the lowest value of $C_{SORT}(node)$ (Appendix A). The probability Q_i is approximated by $|R_i|/|R|$ $(0 < |R|)$, where $|R_i|$ is the number of representative rays occluded by the i-th child nodes and $|R| = \sum_{i=1}^{N} |R_i|$. If $|R| = 0$, we skip sorting. Setting a large value for C_t favours shallow subtrees. However, the resulting performance is not greatly affected by the value of C_t. We hence omit this term in our implementation.

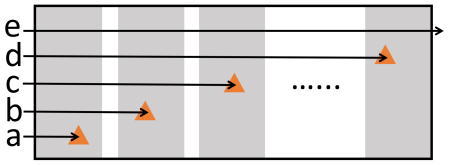


Figure 2: *Cost of occlusion test for one node. The probability to stop in the first child node (a) is Q_1, with a cost of C_1, because we just test the first child node. Stopping in the second child node (b) has a probability of $(1-Q_1)Q_2$, with a cost of C_1+C_2, because we visit the first and the second child nodes. By extension, stopping in the third child node (c) has a probability of $(1-Q_1)(1-Q_2)Q_3$, with a cost of $C_1+C_2+C_3$ and stopping in the last child node (d) has a probability of $(1-Q_1)\dots(1-Q_{N-1})Q_N$ with a cost of $C_1+\dots+C_N$. The probability of a negative occlusion test (e) has a probability of $(1-Q_1)\dots(1-Q_N)$ and a cost of $C_1+\dots+C_N$.*

Note that sorting child nodes has no influence on closest-hit test because we do not change the topology of MBVH. Non-opaque materials can be handled by stochastically incrementing counter based on their opacity.

4. Results

We implemented the presented method in our rendering system as well as in Embree 2.0 [WFWB13]. All measurements are done on a dual Intel Xeon E5-2620 v2 system. The number of node visits, leaf node visits, and triangle intersection tests are denoted by N_B, N_L, and N_T, respectively.

We do not compare our method with SRDH or RTSAH as they are originally developed for binary BVHs and applying them to MBVH is not straightforward. Also speedup comes at the price for the additional memory and building time of a specialized BVH for shadow rays. In production rendering we often need to store a large amount of geometry data and textures, and thus want to avoid having multiple BVHs. BVH construction time is also problematic because users need interactive feedback. We hence choose SATO because of the following reasons: 1) it does not need an extra memory; 2) it can be easily extended for MBVH; 3) its preprocessing cost is almost negligible.

First we measured how the choice of P_i affects performance (Table 2). Apart from the Crytek Sponza scene, approximating P_i by A_i/A_{node} performs better, although the difference is subtle. For the rest of experiments we use $P_i = 1$ to optimize for the worst case scenario.

The comparison with FBTO (Front-to-Back Traversal Order) and SATO (NodeSATO) with a different number of

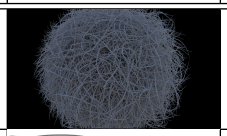
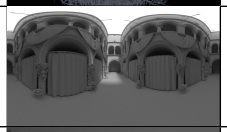
	Scene	P_i	A_i/A_{node}	1
Hairball		N_B/ray	18.25	18.36
		N_L/ray	8.28	8.37
		N_T/ray	33.97	34.22
Crytek Sponza		N_B/ray	7.50	6.55
		N_L/ray	1.60	1.48
		N_T/ray	4.96	4.25
Conference		N_B/ray	7.52	7.51
		N_L/ray	1.93	1.93
		N_T/ray	5.33	5.36
Bedroom		N_B/ray	3.60	3.61
		N_L/ray	1.60	1.63
		N_T/ray	2.75	2.78

Table 2: *Number of node visits (N_B), leaf node visits (N_L), and triangle intersection tests (N_T) per shadow ray after sorting with different bounding box intersection probability approximations, $P_i = A_i/A_{node}$ and $P_i = 1$. SATO is used for the logging phase. The results show that the choice of P_i does not have a great influence on the general performance.*

child nodes in our rendering system is given in Table 5. For this experiment about 5% of a total ray budget is used as representative rays. Our method reduces the traversal steps by up to 60% compared to SATO. Although it is very small, a performance gain is still obtained for fine objects such as hair strands as illustrated in Table 5. Our method is more effective when the number of child nodes is greater. The traversal order used in the logging phase clearly affects the number of node/leaf visits and triangle intersection tests. However, it depends on scene and we could not conclude which one is consistently better.

The results with Embree are shown in Table 7. We use 10% of total ray budget as representative rays. We obtain around 10% speedup for the Crytek Sponza scene when object splitting is used for MBVH construction. When using spatial splitting, the performance gain is smaller. We believe that this is a consequence of not accounting for children's bounding box misses in the cost model. We made the assumption that a ray entering a node will intersect all the children's bounding boxes ($P_i = 1$). The result shows that this assumption is reasonable for object splitting but not accurate enough for spatial splitting where the bounding boxes are tighter.

The atomic operations used in the logging phase have a slight overhead. We therefore implemented two occlusion test kernels: one for representative rays and the other for normal occlusion rays. Our sorting is slower compared to the preprocessing time of SATO but runs at a reasonable speed (about a few hundred milliseconds without parallelizing for the San Miguel scene). We have not observed any major performance drop for all the scenes we tested.

5. Discussion

In this section we discuss when it is suitable to use our method with additional experiments.

5.1. Bidirectional Algorithms

Crytek Sponza		SATO	Ours
	N_B	8.01×10^7	7.91×10^7
	N_L	0.82×10^7	0.75×10^7
	N_T	1.33×10^7	1.21×10^7

Table 3: *Application of our traversal order in light tracing. The numbers of node visits (N_B), leaf node visits (N_L), and triangle intersection tests (N_T) before (with SATO) and after sorting. 10 million photon paths are used in the logging phase. Sorting the nodes slightly reduces traversal steps.*

Our algorithm is not limited to shadow rays and can also be used for any occlusion tests. Bidirectional light transport algorithms require many visibility tests to connect eye subpath vertices and light subpath vertices. We cannot simply use a small portion of them as representative rays because their distribution is too uniform in general. Therefore, we recommend limiting the use to the paths between light subpath vertices and eye. We implemented light tracing in our rendering system and tested with the Crytek Sponza scene (Table 3). The number of triangle intersection tests is reduced by roughly 10%. Our method works well when a camera is located inside of a scene unlike [VHS12]. Fast preprocessing makes our method suitable for multi-pass algorithms.

5.2. Multiple Lights and Environment Maps

If a scene contains multiple light sources, sorting child nodes per light might be beneficial since the optimal ordering depends on lighting in a scene. We rendered the car scene with two directional lights. When the MBVH is updated per light, the total number of traversal steps is slightly reduced (Table 4). These results suggest that sorted deferred shading [ENSB13] could benefit from our algorithm by sorting shadow rays by their associated light sources.

When using an environment map, light comes from every direction. Importance sampling should be used because it does not only reduce noise but also helps yielding ray coherence. We also experimented to see how environment maps affect the performance (Table 6). The number of traversal steps is reduced by about 10-20% for all images. To achieve further speedup, images could be segmented into regions

Light 1+2	Light1	Light2		Light 1	Light 2	Sum	Whole
			N_B	2.84×10^8	1.61×10^8	4.45×10^8	4.74×10^8
			N_L	0.62×10^8	0.34×10^8	0.96×10^8	1.04×10^8
			N_T	1.02×10^8	0.55×10^8	1.57×10^8	1.67×10^8

Table 4: *Comparison of reordering per light against reordering for all lights. The scene contains two directional lights and 1024 samples per pixel are used for each light. In this case, the total number of tests performed for the reordering per light, written in the second to last column is noticeably smaller to the number of tests performed for the reordering for all lights.*

that have the same amount of energy and processed one by one. The optimal segmentation is out of the scope of this paper.

5.3. Memory Consumption

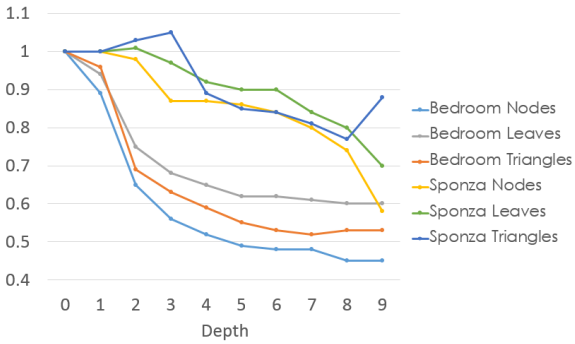


Figure 3: *Number of node visits, leaf node visits, and triangle intersection tests for our reordering relatively to the number of tests without reordering given the depth of reordering. Depth = 0 means no reordering, Depth = 1 means reordering only the child nodes of the root, etc. The results were obtained with an OBVH (N = 8) using SATO in the logging phase. The number of tests decreases as depth increases allowing trade-off between memory and computation time.*

One drawback of our algorithm is that a $4 \times N$-byte array has to be added to each node when padding is not available. The memory consumption can be reduced by allocating counters for the upper-level nodes. To see if this is possible, we measured how the number of traversal steps is influenced by the depth range in the logging and sorting phases (Figure 3). Although they are not monotonically decreasing, the traversal steps quickly drop as the depth increases. This means we can trade-off performance with memory. Not sorting child nodes at all depth levels also leads to faster preprocessing.

6. Conclusion and Future Work

We presented a very simple technique for fast occlusion test. This technique provides more than 10% traversal step reduction and we obtained up to 10% speedup for Embree.

Our algorithm can be readily implemented in a ray tracing based pipeline because occlusion tests can be accelerated by only sorting MBVH child nodes. Once child nodes are sorted, we do not need extra computation to determine traversal order at run time. We believe this property is very useful to take advantage of increasingly wider SIMD units. If padding is not used, a small memory footprint is necessary. However, this can be mitigated by allocating counters only for MBVH nodes at shallow levels. Reducing memory footprint might also be helpful to handle instancing.

For future work, we would like to apply our method to dynamic scenes using a fast BVH construction algorithm such as LBVH [LGS*09], and improve the logging and sorting phases further, for example, by removing atomic operations. Our algorithm is orthogonal to advanced traversal techniques such as DRST (Dynamic Ray Streaming Traversal) [BAM14]. It could be possible to accelerate occlusion tests even further using DRST. We would also like to find better approximations of the probabilities used in the cost model to improve performance especially for MBVH built with spatial splits. Another interesting research avenue is to apply this technique to other fields, for instance, collision detection.

Acknowledgements

We thank anonymous reviewers for their very helpful comments and suggestions. The Bedroom scene was modeled by David Vacek and is available from the Lighting Challenges (http://3dRender.com). The room model of the Car scene is created by Yoichi Kimura. The San Miguel scene was modeled by Guillermo M. Leal Llaguno of Evolucien Visual. This scene is available from the McGuire Graphics Data site (http://graphics.cs.williams.edu/data/meshes.xml). The Crytek Sponza scene was created by Frank Meinl. All environment maps used in the paper are available from the sIBL Archive (http://www.hdrlabs.com/sibl/archive.html). We would like to thank Ken Anjyo for simplifying the proof, and Toshiya Hachisuka for his insightful comment that the proof can be extended for the

whole tree. This work was supported by Japan Science and Technology Agency, CREST.

References

[Áfr13] ÁFRA A. T.: *Faster Incoherent Ray Traversal Using 8-Wide AVX Instructions*. Tech. rep., Babeş-Bolyai University, Cluj-Napoca, Romania, Aug. 2013. 2

[BAM14] BARRINGER R., AKENINE-MÖLLER T.: Dynamic ray stream traversal. *ACM Trans. Graph. 33*, 4 (July 2014), 151:1–151:9. 5

[BH09] BITTNER J., HAVRAN V.: RDH: Ray Distribution Heuristics for Construction of Spatial Data Structures. In *25th Spring Conference on Computer Graphics (SCCG 2009)* (Budmerice, Slovakia, May 2009), Hauser H., (Ed.), ACM SIGGRAPH and EUROGRAPHICS, ACM, pp. 61–67. 2

[DHK08] DAMMERTZ H., HANIKA J., KELLER A.: Shallow bounding volume hierarchies for fast simd ray tracing of incoherent rays. In *Proceedings of the Nineteenth Eurographics Conference on Rendering* (2008), EGSR '08, pp. 1225–1233. 2

[ENSB13] EISENACHER C., NICHOLS G., SELLE A., BURLEY B.: Sorted Deferred Shading for Production Path Tracing. *Computer Graphics Forum 32*, 4 (2013), 125–132. 4

[FLF12] FELTMAN N., LEE M., FATAHALIAN K.: SRDH: Specializing BVH Construction and Traversal Order Using Representative Shadow Ray Sets. In *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics* (2012), Dachsbacher C., Munkberg J., Pantaleoni J., (Eds.), The Eurographics Association. 2, 3

[IH11] IZE T., HANSEN C.: RTSAH Traversal Order for Occlusion Rays. In *Computer Graphics Forum (Proceedings of Eurographics 2011)* (2011), vol. 30, pp. 297–305. 2

[LGS*09] LAUTERBACH C., GARLAND M., SENGUPTA S., LUEBKE D., MANOCHA D.: Fast BVH Construction on GPUs. *Computer Graphics Forum* (2009). 5

[MB90] MACDONALD D. J., BOOTH K. S.: Heuristics for ray tracing using space subdivision. *Vis. Comput. 6*, 3 (May 1990), 153–166. URL: http://dx.doi.org/10.1007/BF01911006, doi:10.1007/BF01911006. 2

[NM14] NAH J.-H., MANOCHA D.: SATO: Surface Area Traversal Order for Shadow Ray Tracing. *Computer Graphics Forum 33*, 6 (2014), 167–177. 2

[SFD09] STICH M., FRIEDRICH H., DIETRICH A.: Spatial splits in bounding volume hierarchies. In *Proc. High-Performance Graphics 2009* (2009). 2

[VHS12] VINKLER M., HAVRAN V., SOCHOR J.: Technical section: Visibility driven bvh build up algorithm for ray tracing. *Comput. Graph. 36*, 4 (June 2012), 283–296. 2, 4

[WBB08] WALD I., BENTHIN C., BOULOS S.: Getting rid of packets - efficient simd single-ray traversal using multi-branching bvhs. In *IEEE Symposium on Interactive Ray Tracing, 2008. RT 2008* (2008), IEEE, pp. 49–57. 2

[WFWB13] WOOP S., FENG L., WALD I., BENTHIN C.: Embree ray tracing kernels for cpus and the xeon phi architecture. In *ACM SIGGRAPH 2013 Talks* (2013), SIGGRAPH '13, pp. 44:1–44:1. 2, 3

Appendix A: Minimizing the occlusion test cost for one node

Let's consider a node with N children, each one associated with a probability or intersecting a ray Q_i and visitation cost C_i. Suppose that the cost of the parent node C_{SORT} defined eq (1) is minimal. Let's consider $1 \leq I < N$ and the same node, with the children I and $I+1$ swapped. The new probabilities Q' and costs C' follow

$$\forall i \in [\![1,N]\!], \quad Q'_i, C'_i = \begin{cases} Q_{I+1}, C_{I+1} & \text{if } i = I \\ Q_I, C_I & \text{if } i = I+1 \\ Q_i, C_i & \text{otherwise.} \end{cases}$$

Denoting the new cost C'_{SORT}, the cost difference is

$$\begin{aligned} C'_{SORT} - C_{SORT} = \tau((\sigma + C_{I+1})Q_{I+1} - (\sigma + C_I)Q_I \\ + (\sigma + C_I + C_{I+1})(1 - Q_{I+1})Q_I \\ - (\sigma + C_I + C_{I+1})(1 - Q_I)Q_{I+1}), \end{aligned}$$

where

$$\sigma = \sum_{j=1}^{I-1} C_j$$

and

$$\tau = \prod_{k=1}^{I-1} (1 - Q_k).$$

Both are positive, independent of (Q_l, C_l) and (Q_{l+1}, C_{l+1}). After reduction, we have

$$C'_{SORT} - C_{SORT} = \tau(C_{I+1}Q_I - C_I Q_{I+1}).$$

We assumed C_{SORT} to be minimal. Thus we have

$$C'_{SORT} - C_{SORT} \geq 0.$$

Since τ is positive, necessarily,

$$C_{I+1}Q_I - C_I Q_{I+1} \geq 0,$$

which leads to

$$\frac{Q_I}{C_I} \geq \frac{Q_{I+1}}{C_{I+1}}.$$

Therefore, if C_{SORT} is minimal, the sequence $(\frac{Q_i}{C_i})$ is decreasing. Since C_{SORT} can only take a finite number of values over the permutations of (Q_i, C_i), we deduce that the minimal value is obtained when the sequence $(\frac{Q_I}{C_I})$ is ordered in decreasing order.

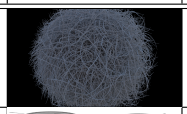
			BBVH (N=2)				QBVH (N=4)				OBVH (N=8)			
			FBTO		SATO		FBTO		SATO		FBTO		SATO	
Hairball		N_B/ray	55.51	53.40	56.62	54.64	27.62	26.99	28.77	27.58	18.67	18.01	18.93	18.36
		N_L/ray	5.04	4.52	4.45	4.49	6.81	6.12	5.79	6.11	9.47	8.48	8.45	8.37
		N_T/ray	46.06	40.75	40.61	40.10	42.30	36.91	35.96	37.22	39.72	34.57	35.22	34.22
Crytek Sponza		N_B/ray	30.94	25.50	31.15	25.55	14.96	11.55	12.50	11.56	13.75	6.85	11.20	6.55
		N_L/ray	1.37	1.22	1.37	1.23	1.53	1.27	1.38	1.26	2.53	1.52	2.09	1.48
		N_T/ray	3.86	3.43	4.00	3.45	2.79	2.19	2.65	2.30	4.07	2.39	4.84	4.25
Conference		N_B/ray	24.16	23.64	23.99	23.57	11.78	11.44	11.63	11.45	7.91	7.60	7.76	7.51
		N_L/ray	1.71	1.73	1.79	1.72	1.84	1.80	1.85	1.79	1.97	1.94	1.95	1.93
		N_T/ray	6.78	6.46	6.71	6.60	6.11	5.59	5.79	5.58	5.69	5.13	5.77	5.36
Bedroom		N_B/ray	19.34	15.92	16.12	15.34	10.28	6.72	8.56	6.38	9.44	3.67	8.02	3.61
		N_L/ray	1.52	1.27	1.29	1.24	2.20	1.50	1.89	1.49	3.42	1.63	2.73	1.63
		N_T/ray	5.54	3.77	4.04	3.58	6.04	3.10	4.79	3.06	7.61	2.69	5.22	2.78

Table 5: *Impact of the traversal order used in the logging phase for MBVH with different number of child nodes, measured by the number of node visits (N_B), leaf node visits (N_L), and triangle intersection tests (N_T) per shadow ray. Path tracing with multiple importance sampling is used to render images.*

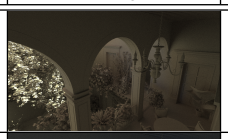
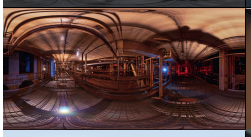
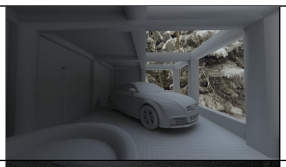
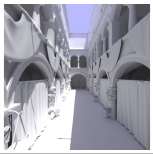
Environment map	Car		SATO	Ours	San Miguel		SATO	Ours
		N_B/ray	8.29	7.10		N_B/ray	7.85	5.69
		N_L/ray	1.90	1.63		N_L/ray	2.00	1.59
		N_T/ray	3.22	2.70		N_T/ray	3.15	2.63
		N_B/ray	8.44	7.49		N_B/ray	7.47	6.25
		N_L/ray	2.00	1.77		N_L/ray	1.86	1.62
		N_T/ray	3.44	3.03		N_T/ray	2.99	2.84
		N_B/ray	7.62	6.54		N_B/ray	6.51	5.42
		N_L/ray	1.86	1.52		N_L/ray	1.73	1.51
		N_T/ray	3.22	2.52		N_T/ray	2.84	2.65
		N_B/ray	8.07	7.33		N_B/ray	6.48	5.37
		N_L/ray	1.95	1.75		N_L/ray	1.72	1.52
		N_T/ray	3.42	2.95		N_T/ray	2.82	2.68

Table 6: *Comparison of SATO and our order traversal for different environment maps with respect to the number of node visits (N_B), leaf node visits (N_L), and triangle intersection tests (N_T) per shadow ray. All images are rendered using OBVH (N=8). Path tracing with multiple importance sampling is used. The number of tests for finding an occlusion is noticeably reduced using our traversal order.*

Object Splitting

Number of triangles per leaf node			1 triangle		4 triangles		8 triangles	
			bvh8	Ours	bvh8	Ours	bvh8	Ours
Car		N_B/ray	7.79	7.40	7.19	6.90	6.84	6.50
		N_L/ray	1.77	1.59	1.45	1.35	1.46	1.49
		N_T/ray	2.59	2.30	1.79	1.62	1.76	1.79
		mrps	22.52	22.68	24.37	24.41	23.26	23.54
		time[s]	48.2	47.9	44.6	44.6	46.6	46.0
Bedroom		N_B/ray	7.41	4.54	6.82	4.25	6.42	4.19
		N_L/ray	2.52	1.85	1.82	1.46	1.84	1.49
		N_T/ray	3.95	2.73	2.17	1.57	2.16	1.60
		mrps	24.85	26.37	27.86	29.54	26.97	27.97
		time[s]	51.1	48.2	45.6	42.9	47.1	45.4
Crytek Sponza		N_B/ray	12.80	9.46	11.68	8.74	11.16	8.39
		N_L/ray	2.20	1.56	2.16	1.78	2.13	1.72
		N_T/ray	3.51	2.68	3.52	3.11	3.00	2.45
		mrps	18.63	20.63	19.16	20.73	18.98	20.58
		time[s]	72.1	65.1	70.1	64.8	70.7	65.2

Spatial Splitting

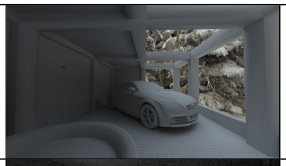
Number of triangles per leaf node			1 triangle		4 triangles		8 triangles	
			bvh8	Ours	bvh8	Ours	bvh8	Ours
Car		N_B/ray	8.11	8.01	7.38	7.30	7.04	7.03
		N_L/ray	1.73	1.58	1.41	1.36	1.40	1.36
		N_T/ray	2.33	2.09	1.68	1.61	1.60	1.54
		mrps	22.50	22.92	24.40	24.48	24.01	24.12
		time[s]	48.4	47.8	44.6	44.5	45.3	45.1
Bedroom		N_B/ray	5.79	5.29	5.25	4.89	4.89	4.52
		N_L/ray	1.81	1.66	1.55	1.49	1.58	1.52
		N_T/ray	2.29	1.99	1.74	1.63	2.01	1.87
		mrps	27.74	28.72	29.60	30.06	28.61	28.54
		time[s]	45.8	44.4	42.9	42.4	44.3	44.5
Crytek Sponza		N_B/ray	8.42	7.80	7.40	6.82	6.89	6.35
		N_L/ray	1.68	1.53	1.47	1.41	1.53	1.47
		N_T/ray	2.14	1.95	1.93	1.86	1.94	1.85
		mrps	23.67	24.43	25.59	26.12	24.88	25.58
		time[s]	56.8	55.0	52.5	51.5	54.0	52.5

Table 7: *Comparison of OBVH from Embree 2.0 (bvh8) and our traversal order with respect to the number of node visits (N_B), leaf node visits (N_L), and triangle intersection tests (N_T) per shadow ray. We also compare the number of rays per second (mrps) as well as the whole rendering time. The number of required tests for finding an occlusion is reduced up to 38% using our traversal order, which translates to a speedup of up to 9% of the total rendering time for object splitting. The number of required tests for finding an occlusion is only slightly reduced for spatial splitting due to the rough approximation $P_i = 1$.*