Computers & Graphics 62 (2017) 41–52

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on Graphics Interface 2016

Construction of bounding volume hierarchies with SAH cost approximation on temporary subtrees

Dominik Wodniok[a,b,*], Michael Goesele[a,b]

[a] TU Darmstadt, Rundeturmstraße 12, 64283 Darmstadt, Germany
[b] Graduate School of Computational Engineering, Dolivostraße 15, 64293 Darmstadt, Germany

A R T I C L E I N F O

Article history:
Received 1 September 2016
Received in revised form
24 November 2016
Accepted 3 December 2016
Available online 10 December 2016

Keywords:
Ray tracing
Bounding volume hierarchy

A B S T R A C T

Advances in research on quality metrics for bounding volume hierarchies (BVHs) have shown that greedy top-down SAH builders construct BVHs with superior traversal performance despite the fact that the resulting SAH values are higher than those created by more sophisticated builders. Motivated by this observation we examine three construction algorithms that use recursive SAH values of temporarily constructed SAH-built BVHs to guide the construction and perform an extensive evaluation. The resulting BVHs achieve up to 37% better trace performance for primary rays and up to 32% better trace performance for secondary diffuse rays compared to standard plane sweeping without applying spatial splits. Allowing spatial splits still achieves up to 18% resp. 15% better performance. While our approach is not suitable for real-time BVH construction, we show that the proposed algorithm has subquadratic computational complexity in the number of primitives, which renders it usable in practical applications. Additionally, we describe an approach for improving the forecasting abilities of the SAH-EPO ray tracing performance predictor which also increases its relevance for primary rays.

1. Introduction

Ray tracing is an important computational primitive used in different algorithms including collision detection, line-of-sight computations, ray tracing-based sound propagation, and most prominently light transport algorithms. An efficient ray tracing implementation needs to rely on an acceleration structure. The most common ray tracing acceleration structures are kd-trees and bounding volume hierarchies (BVHs), with BVHs having gained popularity in the last decade. The reasons for this are the much smaller and controllable memory footprint of BVHs, a more efficient empty space cut-off, faster construction, and a simple update procedure of the structure for animated data, while at the same time offering similar ray tracing performance as kd-trees.

BVHs give best ray tracing performance when constructed with the surface area heuristic (SAH) [1]. State-of-the-art SAH-based BVH builders are the greedy top-down plane-sweeping algorithm from MacDonald and Booth [2] and the extension of this algorithm with spatial splits proposed by Stich et al. [3]. More sophisticated algorithms have been developed (see the summary in Aila et al. [4]) that produce higher quality BVHs with respect to SAH. But the improvements do not translate well to actual measured performance and can in fact even decrease performance. Aila et al. [4] identified geometry that overlaps bounds of subtrees to which it does not belong as a second major factor and proposed the end-point-overlap metric (EPO) to measure this effect. They also revealed the unique characteristic of greedy top-down SAH builders, that they not only optimize SAH but also implicitly minimize EPO, which explains why they perform so well in practice.

To the best of our knowledge no approach has been proposed to date, which directly takes advantage of this implicit correlation of SAH and EPO for greedy top-down builders to construct better BVHs. We examine the possibility to improve EPO further by using recursive SAH evaluation on temporarily built BVHs as an accurate prediction for the SAH cost of subtrees during construction. Further, we reason why the temporary BVHs themselves have to be constructed with SAH to gain any benefit and propose an algorithm that can construct BVHs with recursive SAH in $O(N\log^2 N)$. Due to the computational complexity the algorithm is mainly suitable for static scenes and global illumination algorithms.

Our main contributions are as follows:

- a BVH construction algorithm that produces BVHs with better average performance than state-of-the-art methods
- a complexity analysis of our algorithm that reveals subquadratic runtime in the number of primitives.

* Corresponding author at: TU Darmstadt, Rundeturmstraße 12, 64283 Darmstadt, Germany.
E-mail address: dominik.wodniok@gris.informatik.tu-darmstadt.de (D. Wodniok).

http://dx.doi.org/10.1016/j.cag.2016.12.003
0097-8493/© 2016 Elsevier Ltd. All rights reserved.

- a spatial split-based algorithm, which applies temporary spatial splits to push quality of BVHs even further
- an approach for reducing the forecasting error of the ray tracing performance predictor from Aila et al. [4] which also enables more accurate predictions for primary rays and
- a comparison with a related algorithm proposed by Popov et al. [5] and a hybrid of their algorithm with ours.

This paper is an extended version of an earlier conference paper by Wodniok and Goesele [6] and adds the last three contributions.

2. Background and related work

The strategy chosen for BVH construction has a tremendous influence on ray tracing performance. State-of-the-art construction approaches use the surface area heuristic (SAH) originally introduced by Goldsmith and Salmo [1]. It provides an approximation for the expected cost of traversing a given kd-tree or BVH. Underlying assumptions are that rays originate at infinity, have a uniform ray direction distribution, and do not terminate on intersection. The first and second assumption allow to compute the geometrical conditional probability p_n of intersecting the convex bounding volume of a node n with a random ray given that the ray hits the convex bounds of the surrounding parent node $P(n)$ of n as the surface area ratios of their bounds. Combined with implementation dependent constants c_t for traversal step costs and c_i for primitive intersection test cost the expected traversal cost for a tree node n can be recursively computed as:

$$c(n) = \begin{cases} c_t + p_l c(l) + p_r c(r) & \text{inner node} \\ |n| c_i & \text{leaf node} \end{cases} \tag{1}$$

Here, l and r are the left and right child of n in case of an inner node, and $|n|$ is the number of primitives belonging to n. Evaluating c for the tree root yields the expected cost of the whole tree.

The state-of-the-art greedy top-down plane-sweeping construction from MacDonald and Booth [2] locally applies an approximation of Eq. (1) when splitting a node. Several candidate partitions are generated and rated with c under the assumption that the newly generated children stay leaves. That is we compute:

$$c_{split} = c_t + p_l |l| c_i + p_r |r| c_i \tag{2}$$

The partition with smallest c_{split} is chosen and construction recursively proceeds with the children. The recursion terminates as soon as the smallest c_{split} is higher than the cost for creating a leaf node. Partitions are typically generated by sweeping axis aligned planes through every dimension and checking on which side the bounding volume centroids of primitives fall. With this approach, only planes which contain bounding volume centroids are relevant.

Though the assumptions underlying SAH generally do not apply in practice, SAH guided construction empirically produces the best performing BVHs to date. Unfortunately, SAH-based construction is also the most expensive. Wald et al. [7] introduced an $O(n\log(n))$ algorithm for SAH-based BVH construction. This is achieved by replacing the sorting step with a $O(n)$ binning step adapted from binned kd-tree construction by Popov et al. [8]. For this a limited number of equidistant split planes with respect to the bounds of the primitives are computed. Consecutive split planes define bins to which primitives are distributed. A best split candidate can then be computed from the binning results in linear time. With a sufficient number of bins hierarchy quality is practically identical to full sweep construction. Fabianowski et al. [9] changed the SAH assumption of infinitely far away ray origins to origins uniformly distributed in the scene bounds. On average ray tracing performance increases of 3.5% have been reported.

2.1. Fast construction

Lauterbach et al. [10] proposed three GPU-based BVH construction algorithms with different trade-offs between tree quality and construction time: The median split-based linear BVH (LBVH) algorithm is fast but has poor tree quality. The second algorithm is a parallel approach for full binned-SAH BVH construction (see Wald [11]) with high tree quality but slower construction. The third algorithm, a hybrid of the former two, strikes a balance: Upper levels are constructed according to the highly parallel first algorithm while the remaining levels expose enough parallelism to be efficiently constructed according to the second one. Pantaleoni and Luebke [12], and Garanzha et al. [13] proposed much faster implementations for the median split and the hybrid algorithm called hierarchical LBVH (HLBVH) which allow real-time rebuilds for scenes with up to 2 million triangles. An important change to the hybrid algorithm is, that LBVH is used to build the lower levels of the tree first. The roots of the subtrees themselves are then used for binned top-down SAH BVH construction. Thus the expensive part of the algorithm is performed on much less input elements and tree quality is improved in the important upper levels.

Ganestam al. [14] proposed a hybrid algorithm called *BONSAI*. Similar to the original hybrid algorithm from Lauterbach et al. [10] it first performs a spatial-median split partitioning on the input to produce sufficiently small chunks of spatially coherent primitives. Then for each chunk top-down plane-sweeping SAH-based BVHs are constructed in parallel. Finally, in the spirit of HLBVH an SAH-based top-level BVH is constructed on the chunks. Quality of the final hierarchy on average is identical to full sweep-based construction but construction time is much lower.

Bittner et al. [15] presented the first incremental BVH construction algorithm which can produce high quality BVHs. This was previously not considered possible. While average hierarchy quality is higher than for a full sweep construction, the actual average measured performance is slightly lower. Nonetheless, construction is faster than top-down sweep construction.

As dynamic scenes are not in focus of our work we only give a very brief overview on related algorithms. One approach is to simply construct a new BVH each frame as fast as possible. This was the purpose of LBVH from Lauterbach et al. [10] and derived work (Pantaleoni and Luebke [12], and Garanzha et al. [13]). For state-of-the-art in refitting-based approaches we refer to the algorithm from Yin et al. [16] and the references therein.

2.2. Higher quality BVHs

Stich et al.'s [3] offline spatial split BVH (SBVH) algorithm drastically improves tree quality for scenes with widely varying degree of tessellation. Their key idea is to either use spatial splits or object partitioning depending on which of them yields a better SAH value. When searching for a node split, the best spatial split is determined in addition to the best object split. Spatial splits are only applied when considered beneficial. To date no efficient GPU implementation of this algorithm has been presented. Karras and Aila [17] proposed an approximate but real-time construction algorithm for GPUs that takes any BVH (i.e., LBVH) as input and performs local optimizations on small node subsets (treelets) w.r.t. SAH. They also present a triangle pre-splitting heuristic with strong focus on producing splits, which are likely to be beneficial for tree quality. Resulting trees achieve about 90% of SBVH tree quality. Ganestam et al. [18] present an alternative triangle pre-splitting algorithm, which can optionally directly be integrated into the clustering phase of the BONSAI algorithm. They report traversal performance improvements to be similar to Karras and Aila [17] while producing less duplicate triangles.