

On Quality Metrics of Bounding Volume Hierarchies

Timo Aila

Tero Karras

Samuli Laine

NVIDIA

Abstract

The surface area heuristic (SAH) is widely used as a predictor for ray tracing performance, and as a heuristic to guide the construction of spatial acceleration structures. We investigate how well SAH actually predicts ray tracing performance of a bounding volume hierarchy (BVH), observe that this relationship is far from perfect, and then propose two new metrics that together with SAH almost completely explain the measured performance. Our observations shed light on the increasingly common situation that a supposedly good tree construction algorithm produces trees that are slower to trace than expected. We also note that the trees constructed using greedy top-down algorithms are consistently faster to trace than SAH indicates and are also more SIMD-friendly than competing approaches.

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

Keywords: ray tracing, acceleration structures, quality metrics

1 Introduction

The construction of optimal acceleration structures for ray tracing is believed to be an NP-hard problem [Havran 2000; Ng and Trifonov 2003; Popov et al. 2009], and we need to rely on greedy algorithms and heuristic quality descriptors to be able to build the trees in a reasonable amount of time. By far the most commonly used descriptor is the *surface area heuristic*, SAH [Goldsmith and Salmon 1987; MacDonald and Booth 1990] which corresponds to the expected cost of tracing a non-terminating (long) random ray:

$$\text{SAH} := \frac{1}{A_{\text{root}}} \left(C_{\text{inn}} \sum_{n \in I} A_n + C_{\text{tri}} \sum_{n \in L} T_n A_n \right) = \sum_{n \in N} C_n \frac{A_n}{A_{\text{root}}}, \quad (1)$$

where I , L and N are the sets of inner, leaf, and all nodes, respectively. C_{inn} is the cost of an inner node (two ray-node tests), C_{tri} the cost of a ray-triangle test, and C_n the cost of processing node n (either C_{inn} or $C_{\text{tri}}T_n$). We use $C_{\text{inn}} = 1.2$ and $C_{\text{tri}} = 1.0$, based on the number of instructions needed by each operation on our test platform. A_n denotes the surface area of node n , and T_n the number of triangles in it. A_n/A_{root} is the probability that a long random ray intersects a convex node, provided that it intersects the root node.

It is known that the surface area heuristic correlates surprisingly well with the measured ray tracing performance when one tweaks the parameters of a particular tree builder [Havran 2000], but an increasing amount of evidence suggests that the relation is not nearly

as perfect when trees built using different algorithms are compared, e.g. [Ng and Trifonov 2003; Popov et al. 2009; Bittner et al. 2013]. We analyze this relationship comprehensively by implementing nine different tree builders and quantifying how well SAH correlates with the ray tracing performance in 22 test scenes.

We observe, for example, that there are cases where a bottom-up builder [Walter et al. 2008] leads to lower SAH cost than sweep-based top-down builders [MacDonald and Booth 1990; Stich et al. 2009], but the tracing of incoherent rays is nevertheless 30–60% slower. Furthermore, it turns out that there is a bias favoring top-down builders over other algorithms, and that top-down builders are measurably more SIMD-friendly. Additionally, we note that SAH consistently and significantly underestimates the cost caused by scene rotation.

Discrepancies like this make it difficult to develop new tree construction algorithms that deviate from the standard greedy top-down approach. Motivated by the failure of SAH to sufficiently describe the quality of trees that were constructed differently, we look for additional descriptors that could provide the missing pieces of the explanation. We identify two new descriptors that, together with SAH, explain the measured performance very well in a wide range of scenes on both scalar and SIMD architectures. Although the new descriptors cannot be directly applied to constructing better trees, they do allow us to better understand what makes certain trees good for ray tracing, and to explain the supremacy of top-down builders.

2 Expected cost of a ray

SAH gives the expected cost of tracing a long random ray through the scene. It predicts the ray tracing performance very well as long as the rays are randomly oriented and never start or terminate inside the scene. Path tracing and diffuse inter-reflection are sufficiently close to random rays so that the first assumption holds, but almost all rays start from surfaces and end on surfaces, and therefore the second assumption is on a questionable footing.

For SAH to be proportional to the measured performance with finite random rays as well, one would have to assume that shorter rays simply exhibit a certain percentage of the cost of a long ray. While this is perhaps not unthinkable, in reality the situation is more complicated. The rest of this section will describe two new descriptors that try to quantify the costs that become relevant for finite rays. These will then be used in the next section in addition to SAH to describe the quality of trees.

2.1 End-point overlap (EPO)

If the ray’s end point (origin or hit point) is inside multiple branches in the tree, we have to visit them all. For example, the first identified intersection is often the closest one, but the amount of work needed to verify this depends on how much the branches overlap around surfaces. This overlap-related cost is a quantifiable weakness of the particular tree because every end point could exist within exactly one branch (assuming triangle splitting is allowed). Since we want to use our new metric in addition to SAH, ideally we would want it to be zero when there is no branch overlap anywhere in the tree — e.g., for an octree where triangles have been clipped to leaf nodes.

If we assume that the ray origins and hit points are uniformly distributed on the surfaces, the probability of having such a point inside a node is proportional to the surface area of the triangles inside that node’s *volume* (i.e., not just the triangles in that particular subtree but all clipped triangles that are inside the volume). Then the expected cost of searching a ray’s origin or end point from the tree is

$$\sum_{n \in N} C_n \frac{A(S \cap n)}{A(S)}, \quad (2)$$

where S is the set of all surfaces in the scene. $A(S \cap n)$ is the total area of surfaces inside the bounding volume of node n , which after normalization by $A(S)$ corresponds to the probability that the query point resides inside n .

Now, a large part of this cost is useful payload and already included in the traditional SAH cost. We therefore define a new metric, EPO, that measures the amount of *extra* work caused by the overlap by

$$\text{EPO} := \sum_{n \in N} C_n \frac{A((S \setminus Q(n)) \cap n)}{A(S)}, \quad (3)$$

where $Q(n)$ is the set of surfaces that belong to the subtree of n , and $(S \setminus Q(n)) \cap n$ is the geometry that does not belong to the subtree of n but nevertheless lies within the volume of n . This cost is zero when branches of the tree do not overlap, and lower values are generally better. The geometric interpretation of this metric is that it penalizes node overlap in areas where the ray origins and hit points are likely to be.

While searching for a good overlap metric, we tried many variations of the general idea (e.g. [Stich et al. 2009], [Popov et al. 2009], or using node areas to approximate triangle areas), but EPO was significantly more descriptive than the alternatives.

2.2 Leaf count variability (LCV)

SIMD execution deviates from scalar in that a ray can affect the execution of another, in various ways. Assuming the primary bottleneck is execution [Aila et al. 2012], this has to relate to the variance in the number of inner nodes, leaf nodes, or triangles processed by a ray. We tested this hypothesis by measuring the variances, and noticed that inner nodes and triangles do not seem to play a significant role, but leaf nodes do. This relates to how the GPU trace kernels are structured; there is a warp-wide mode switch between inner and leaf nodes, and the variance in the number of leaf nodes lowers the SIMD utilization of not only the triangle test but also the inner node processing. We therefore define a new metric that computes the standard deviation of the number of leaf nodes N_i intersected by a ray:

$$\text{LCV} := \sqrt{E[N_i^2] - E[N_i]^2}. \quad (4)$$

At present we calculate the expected values by counting the number of leaf nodes encountered by each ray, for the set of rays used in performance measurements. Therefore this metric should be seen as an explanation for the effects incurred by SIMD execution rather than something that can be optimized during construction.

3 Test setup

We perform our measurements on NVIDIA GTX680 using Aila et al.’s publicly available ray tracing kernels [Aila and Laine 2009] and diffuse inter-reflection rays that are incoherent, and thus correspond to at least partially to the assumptions of SAH. In particular, we do not use primary or shadow rays because they are

too view-dependent. We use 22 test scenes and 9 tree building algorithms: fast spatial mean-based LBVH [Lauterbach et al. 2009; Karras 2012], two greedy sweep-based top-down algorithms (BBVH) [MacDonald and Booth 1990] and SBVH [Stich et al. 2009], one bottom-up algorithm (Agglo) [Walter et al. 2008], post-process improvement using iterative reinsertion (Bittner) [Bittner et al. 2013], two methods based on local tree rotations: hill climbing and simulated annealing [Kensler 2008], and treelet restructuring [Karras and Aila 2013] with and without triangle splitting (STreelet and Treelet, respectively). The last five are initialized using LBVH, and all algorithms use our own implementations relying on the default parameters recommended by the authors. We set SBVH’s free parameter to 10^{-5} in all scenes, except that 10^{-4} was needed in HAIRBALL and VEGETATION to avoid running out of memory and 10^{-6} in SANMIGUEL to see any splitting at all. STreelet uses 30% split quota for all scenes, except SANMIGUEL where only 10% were allowed due to memory constraints. With the exception of LBVH, all methods aim for high quality trees. LBVH also differs from the other methods in that its leaf nodes contain always a single triangle, while the others use (more efficient) variable-sized leaf nodes. STreelet and SBVH split the input triangles adaptively, and can therefore produce better trees than is possible for the other methods. We include LBVH, SBVH, and STreelet because it is important that the cost metrics also apply to these scenarios.

Tables 1 and 3 show detailed data for the 22 test scenes. All aggregate results (average, minimum, maximum) refer to the full set of measurements.

We use an unusually large number of test scenes because scene-to-scene variation in behavior can be significant, and we wish to ensure that the conclusions apply as broadly as possible. We focus on commonly used ray tracing test scenes, but augment the list with several city models. CONFERENCE, SIBENIK, CRYTEK-SPONZA, BAR, and SODA are widely used architectural models. ARMADILLO, BUDDHA, DRAGON, BLADE, and MOTOR are finely tessellated objects. MUSTANG and VEYRON are cars. FAIRY and BUBS have widely varying triangle sizes. HAIRBALL, VEGETATION, and POWERPLANT have difficult structure. The four cities, CITY, BABYLONIAN, ARABIC, and ITALIAN show large spatial extents; the last three are from Mitsuba distribution [Jakob 2010]. SANMIGUEL combines architecture with fine geometric detail and vegetation; of our scenes it is probably the most realistic.

The measurements are averages over a scene-dependent number of viewpoints. For individual objects we use two or three viewpoints, for simple architecture four or five, and for bigger scenes up to ten. The viewpoints try to capture all interesting aspects of the scene.

We measure the prediction power of a metric by computing the sample Pearson correlation coefficient between the measured performance x and the metric y :

$$\text{Correlation} := \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \sigma_y}, \quad (5)$$

where $\sigma_x = \sqrt{\sum_i (x_i - \mu_x)^2}$. Although very widely used, Pearson’s correlation coefficient has some weaknesses. In particular, it tends to give seemingly high values even when a particular metric is a poor descriptor; as will be shown below, even 0.90 is not necessarily an indication of success, and we really should aim for 0.99 or higher.

4 Explaining measured performance

Although our goal is to explain the ray casting performance on GPUs, we start by analyzing scalar execution because it is impossible to explain SIMD results properly without first understanding



Scene	Metric	Builder									Correlation with ns/ray		
		LBVH	BBVH	SBVH	Agglo	Bittner	H.Climb	Anneal.	Treelet	STreelet	SAH	SAH+EPO	SAH+EPO+LCV
 CONFERENCE (282755 tris)	Scalar (ns/ray)	111.36	69.01	56.85	58.41	55.90	68.07	55.16	59.31	55.01	0.992	0.999 $_{\alpha=0.42}$	0.999 $_{\alpha=0.41,\beta=0.03}$
	SIMD (ns/ray)	12.02	5.97	4.16	5.81	5.53	6.15	5.17	5.56	4.42	0.951	0.983 $_{\alpha=0.72}$	0.998 $_{\alpha=0.24,\beta=0.71}$
	SAH	71.89	46.50	38.94	38.17	36.51	42.69	36.24	38.48	39.32			
	EPO	21.68	9.79	4.10	8.27	6.76	11.51	7.07	8.21	2.19			
	LCV	11.66	2.63	1.35	3.25	3.30	2.66	2.40	2.50	1.38			
 FAIRY (174117 tris)	Scalar (ns/ray)	90.74	65.66	65.40	78.25	65.27	68.07	71.07	67.61	68.73	0.672	0.988 $_{\alpha=0.80}$	0.996 $_{\alpha=0.63,\beta=0.21}$
	SIMD (ns/ray)	8.52	4.77	4.65	5.69	4.98	5.05	5.04	5.08	5.09	0.660	0.915 $_{\alpha=0.77}$	0.994 $_{\alpha=0.23,\beta=0.69}$
	SAH	43.54	33.58	34.70	37.31	32.16	33.38	34.32	33.02	43.72			
	EPO	7.72	3.45	2.78	6.59	3.20	3.38	4.75	3.47	1.58			
	LCV	7.15	1.79	1.67	1.86	1.98	1.79	1.79	1.81	1.51			
 BUBS (1888101 tris)	Scalar (ns/ray)	86.36	68.63	53.62	55.10	51.28	58.14	52.52	55.16	57.11	0.947	0.988 $_{\alpha=0.56}$	0.997 $_{\alpha=0.11,\beta=0.73}$
	SIMD (ns/ray)	7.26	5.06	4.32	4.75	4.55	4.91	4.64	4.76	4.73	0.830	0.907 $_{\alpha=0.75}$	0.989 $_{\alpha=0.00,\beta=0.96}$
	SAH	28.72	24.51	17.81	16.28	15.29	17.73	15.51	16.43	21.11			
	EPO	12.26	8.39	2.68	4.24	2.90	3.93	3.56	3.65	1.27			
	LCV	2.82	1.28	1.18	1.27	1.29	1.34	1.21	1.30	1.05			
 SODA (2168983 tris)	Scalar (ns/ray)	130.38	78.43	63.41	120.19	68.35	92.25	77.10	80.26	63.98	0.860	0.988 $_{\alpha=0.88}$	0.988 $_{\alpha=0.88,\beta=0.00}$
	SIMD (ns/ray)	9.49	4.75	3.80	8.09	4.37	5.95	4.88	5.27	3.94	0.878	0.986 $_{\alpha=0.83}$	0.988 $_{\alpha=0.53,\beta=0.41}$
	SAH	115.17	78.26	67.56	82.90	59.39	76.08	63.49	71.04	69.53			
	EPO	32.68	13.75	3.45	28.88	8.33	16.95	13.37	15.28	2.95			
	LCV	8.75	1.78	1.42	2.57	2.21	2.13	1.90	2.03	1.37			
 BABYLONIAN (499036 tris)	Scalar (ns/ray)	162.34	69.74	48.95	71.99	61.54	84.53	64.35	77.82	53.76	0.989	0.994 $_{\alpha=0.49}$	0.996 $_{\alpha=0.07,\beta=0.84}$
	SIMD (ns/ray)	12.50	4.56	3.07	4.92	4.26	5.78	4.32	5.38	3.64	0.987	0.990 $_{\alpha=0.42}$	0.995 $_{\alpha=0.01,\beta=0.93}$
	SAH	93.85	54.97	41.31	50.16	44.89	55.62	45.14	51.25	43.97			
	EPO	52.33	18.97	3.09	17.96	13.15	22.59	14.31	19.04	6.55			
	LCV	5.45	1.75	0.97	2.01	1.88	2.00	1.76	1.95	1.05			
 DRAGON (869928 tris)	Scalar (ns/ray)	95.97	78.74	77.10	96.62	82.37	84.82	93.98	83.54	81.90	0.896	0.996 $_{\alpha=0.61}$	0.996 $_{\alpha=0.30,\beta=0.56}$
	SIMD (ns/ray)	8.94	6.83	6.65	8.49	7.28	7.46	8.15	7.52	7.35	0.959	0.988 $_{\alpha=0.39}$	0.990 $_{\alpha=0.14,\beta=0.73}$
	SAH	71.97	56.74	55.75	65.43	57.26	62.04	63.28	60.41	59.22			
	EPO	6.04	3.29	2.50	11.35	5.02	5.00	9.66	5.00	3.47			
	LCV	2.65	1.79	1.76	1.95	1.88	1.87	1.84	1.85	1.78			
 MOTOR (322284 tris)	Scalar (ns/ray)	124.38	84.18	70.37	102.46	87.26	96.99	90.50	91.16	77.10	0.964	0.994 $_{\alpha=0.77}$	0.995 $_{\alpha=0.65,\beta=0.16}$
	SIMD (ns/ray)	13.55	7.42	5.46	10.95	9.29	9.20	8.20	8.61	6.39	0.926	0.967 $_{\alpha=0.86}$	0.993 $_{\alpha=0.46,\beta=0.50}$
	SAH	101.82	69.70	58.02	74.49	68.71	74.58	69.00	70.57	63.13			
	EPO	24.71	10.28	3.35	17.01	11.33	14.51	14.28	13.21	5.19			
	LCV	12.92	3.00	1.65	11.01	7.25	3.48	2.90	3.09	1.86			
 HAIRBALL (2850000 tris)	Scalar (ns/ray)	287.36	206.19	191.57	288.18	204.50	216.45	279.33	211.42	194.17	0.652	0.992 $_{\alpha=0.98}$	0.994 $_{\alpha=0.74,\beta=0.26}$
	SIMD (ns/ray)	59.56	31.48	29.36	49.65	32.15	33.44	47.69	33.17	31.00	0.816	0.979 $_{\alpha=0.89}$	0.992 $_{\alpha=0.37,\beta=0.63}$
	SAH	664.98	469.00	429.45	479.96	441.95	486.08	482.41	473.90	447.49			
	EPO	60.47	38.79	28.80	66.16	36.40	41.61	60.71	40.31	32.06			
	LCV	22.45	5.83	5.09	7.81	6.15	6.07	7.49	6.04	5.55			
 SANMIGUEL (10483269 tris)	Scalar (ns/ray)	273.97	186.22	140.65	179.86	149.93	184.84	165.29	172.71	137.93	0.818	0.994 $_{\alpha=0.72}$	0.994 $_{\alpha=0.72,\beta=0.00}$
	SIMD (ns/ray)	35.71	19.80	15.22	20.58	17.15	21.23	18.80	20.05	15.85	0.839	0.982 $_{\alpha=0.64}$	0.991 $_{\alpha=0.28,\beta=0.61}$
	SAH	28.37	20.13	18.71	17.06	15.69	18.50	16.27	17.38	19.95			
	EPO	20.59	10.38	5.73	10.66	6.87	10.67	9.19	8.72	2.59			
	LCV	9.18	2.90	2.35	3.67	3.17	3.19	3.01	3.05	2.30			

Table 1: Measured performance for scalar and SIMD for each construction algorithm in ns/ray, along with SAH, EPO, and standard deviation of the number of leaf nodes per ray (LCV), of which the last is relevant only for SIMD execution. The last three columns give correlations for SAH-only, combination of SAH and EPO, and finally a combination of SAH, EPO and LCV.

scalar. We create the scalar kernel by exiting 31 of the 32 lanes of GTX680 immediately after launch. This is a one-line change to the kernels, and allows a direct comparison of scalar and SIMD on the same architecture and same code. We also disable adaptive clocking of the GPU.

Table 1 gives results for nine scenes, diffuse ray measurements for scalar and SIMD, along with the SAH, EPO and LCV costs, and the computed correlations. The ray tracing performance is shown as nanoseconds per ray (ns/ray), because the more typical MRays/s would have an inverse relationship with SAH and EPO. We will defer further discussion about LCV to Section 4.2 because it is not relevant for scalar execution.

4.1 Scalar performance

Although SAH describes the performance reasonably well in several scenes, including CONFERENCE and BABYLONIAN, there are also surprises. For example, in SANMIGUEL Agglo has clearly lower SAH cost than SBVH but 30% worse scalar performance, Treelet has 13% lower SAH cost than STreelet but takes 25% longer to trace, and Bittner and Simulated annealing have almost 20% lower SAH cost than SBVH or STreelet but are still slower to trace. In HAIRBALL all of the builders except for LBVH have similar SAH cost, but the performance varies up to 50%, without an obvious relation to SAH cost. In SODA BBVH and Agglo have similar SAH costs but Agglo takes 55% longer to render. Figure 1 further illustrates these issues.

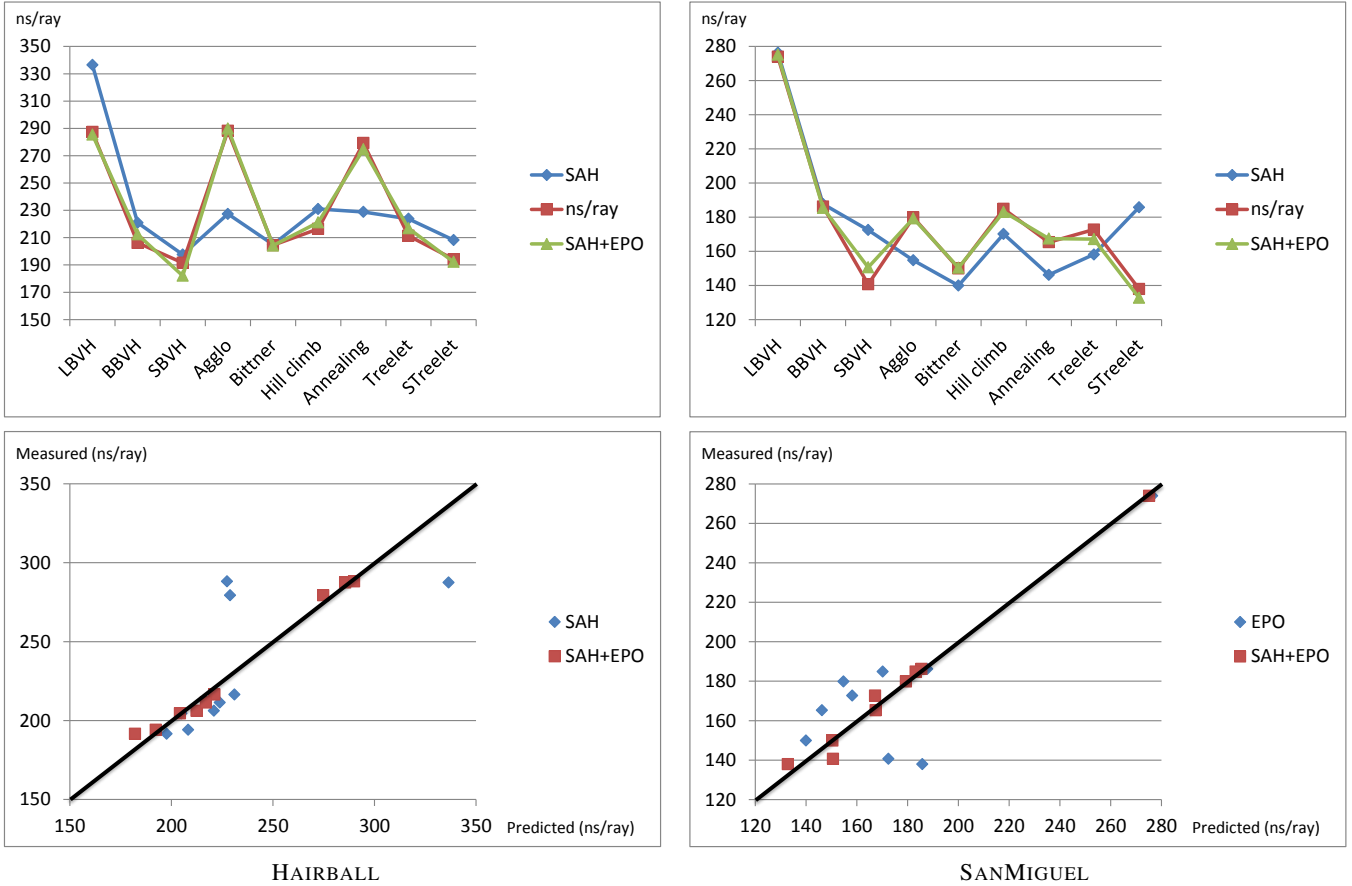


Figure 1: Measured scalar performance for HAIRBALL and SANMIGUEL, along with the predictions from SAH and the combination of SAH and EPO. All three have been normalized to the scale of performance (ns/ray). The top row visualises the values for each algorithm, and as can be seen SAH significantly underestimates the performance differences in HAIRBALL and gets the relative ordering badly wrong in SANMIGUEL, whereas the combination of SAH and EPO closely tracks the measurements in both examples. The bottom graphs plot the predictions against measurements. Ideally all of the samples would lie on the black line.

When measured in our 22 scenes, the average correlation coefficient between SAH and performance is 0.915, and perhaps more importantly the lowest value is a dismal 0.652. If we allow a combination of SAH and EPO

$$(1 - \alpha)SAH + \alpha EPO; \quad 0 \leq \alpha \leq 1, \quad (6)$$

where α is scene-dependent, all of the scenes are explained very well, with the correlation coefficient averaging 0.994, and even the lowest correlation rises to 0.988. For example, the discrepancy in SODA is explained by Agglo having twice the EPO of BBVH, and in HAIRBALL the performance appears to be almost completely determined by EPO. The optimal scene-dependent α varies significantly, from 0.05 to 0.98, with an average of 0.59.

The optimal combination weight α is necessarily scene-dependent because in some scenes the rays traverse through (almost) the entire scene and in some others they are very short. Also, with individual objects most of the secondary rays fail to hit anything, whereas in indoor scenes all of them will terminate. As a rule of thumb, EPO is more important when rays traverse a relatively small part of the scene; we have not experimented with automatic estimation of α yet. Results with fixed α are given in Section 4.3.

A remaining uncertainty is whether the inaccuracy of SAH is purely random or if there is perhaps a systematic bias? To investigate this

we compute the average scalar performance relative to the SAH cost ($(ns/ray)/SAH$). We select the de facto standard BBVH as the reference point. It turns out that this metric is 10% below BBVH for SBVH and STreetlet, but 5–10% higher for all non-top-down methods, i.e., top-down methods are faster than they should be. In other words, SAH systematically underestimates the performance of greedy sweep-based top-down builders.¹ Since the correlations for a linear combination of SAH and EPO are very high, these builders must implicitly optimize EPO when it matters. We will return to this topic in Section 5.

4.2 SIMD performance

The correlation coefficient between SIMD performance and SAH averages 0.933, and the lowest correlation is 0.660. The scene-dependent combination of SAH and EPO pushes the average to 0.979 and the lowest correlation to 0.907, with an average α of 0.50. Although the numbers represent a significant improvement compared to SAH-only, particularly the minimum is still much lower than for scalar, and certain discrepancies remain. For example, in CONFERENCE Agglo has 18% lower SAH than BBVH, and 15% lower EPO and scalar ns/ray, but for some reason the SIMD

¹Similar behavior has been previously documented by Popov et al. [2009] when investigating a more exhaustive top-down builder.

		Correlation with ns/ray		
		SAH	SAH+EPO	SAH+EPO+LCV
Scalar	Average	0.915	0.994	0.996
	Minimum	0.652	0.988	0.988
SIMD	Average	0.933	0.979	0.993
	Minimum	0.660	0.907	0.988

Table 2: Aggregate results from our 22 test scenes for SAH-only, combination of SAH and EPO, and finally a combination of SAH, EPO and LCV.

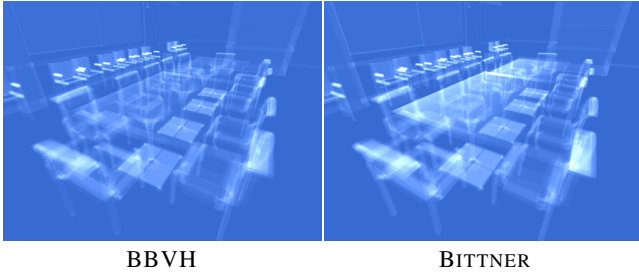


Figure 2: Visualization of the number of leaf nodes visited by non-terminating primary rays in CONFERENCE. Note that this example uses primary rays for illustration purposes only; the actual measurements use diffuse inter-reflection rays. As can be seen, BBVH has a more even cost across the image, especially on the tabletop, where Bittner has a surprising gradient. Agglo also creates the gradient, but Simulated annealing does not, so clearly it is possible to get low SAH cost and low variance in the number of visited leaf nodes simultaneously in this scene.

performance is very similar. Bittner also seems every bit as good as Simulated annealing, but clearly loses in practice.

This raises the question whether the remaining discrepancies are random or if there is perhaps a systematic bias again? It is certainly possible that some tree building algorithms generate trees that are inherently more parallel-friendly. To the best of our knowledge, this has never been investigated before. If we divide the respective scalar and SIMD timings in the 22 scenes — which is accurate because they both run the same code on same hardware — we get an average speedup factor of ~ 13 for BBVH and SBVH, ~ 12 for Agglo, Bittner, Hill climbing, Simulated annealing and Treelets, and 10.6 for LBVH. It is therefore clear that *greedy top-down builders consistently create trees that are more SIMD-friendly* than the other methods. We know that this cannot be explained by SAH or EPO, and therefore it has to be a new, SIMD-specific phenomenon.

Our LCV metric is designed to explain this discrepancy, and the last column in Table 1 shows the achievable per-scene correlations taking SAH, EPO, and LCV into account:

$$(1 - \alpha - \beta)SAH + \alpha EPO + \beta LCV; \quad 0 \leq \alpha, \beta \leq 1, \alpha + \beta \leq 1. \quad (7)$$

This pushes the average correlation coefficient to 0.993 and the minimum to 0.988. The aggregate results are collected in Table 2, and Figure 2 gives an example of the practical meaning of LCV.

4.3 Without scene-specific parameters

Interestingly, LCV also seems to improve the average scalar correlation coefficient slightly, by 0.001, but that is probably an illusion: the scene-dependent choice of (α, β) is able to extract a small

amount of seemingly useful information from what is basically inconsequential randomness for scalar execution. Of the 22 scenes, the only clear outliers are FAIRY and BUBS that gain about 0.008. The values of β may seem large in Tables 1 and 3, but that is mainly because the magnitude of LCV is generally much lower than SAH or EPO.

In an additional experiment we disallowed the scene-specific variation of α and β . When the same, fixed α and β are used for all scenes, the average correlation is 0.980 for scalar ($\alpha = 0.71$) and 0.981 for SIMD ($\alpha = 0.14, \beta = 0.76$). Although clearly lower than the correlations achievable using scene-specific parameters (0.994 and 0.993), this is still a clear improvement over SAH alone (0.915 and 0.933).

4.4 Rotated scenes

When we rotate the scenes and the rays 45 degrees around the (1,1,1)-axis, the average SIMD execution time increases 168% (90% for SBVH and STreelet, 250% for Agglo, and $\sim 180\%$ for the rest). The penalty ranges from negligible in HAIRBALL to a very severe $10\times$ in CITY. At the same time the average SAH cost increases by less than 10% for SBVH and STreelet and 50% for the other methods, thus substantially and consistently underestimating the true cost incurred by the rotation. However, the rotation causes EPO to increase by 290% on the average. If we take the average α (0.50) from Section 4.2, our composite estimate gives a penalty of 165%, which agrees closely with the measured performance. The key insight here is that scene rotation is much worse for finite rays than for long rays.

5 Supremacy of top-down builders

Greedy top-down SAH builders (e.g. BBVH [MacDonald and Booth 1990]) start from the root node that initially contains all triangles, and recursively divide the set of triangles into two parts. At each step the set of triangles is divided as if this was the last subdivision we were allowed to do. This process maximizes the amount of worst-case triangle cost saved at every inner node

$$\frac{1}{A_{\text{root}}} C_{\text{tri}}(A_B T_B - (A_L T_L + A_R T_R)), \quad (8)$$

and thus minimizes the worst-case triangle cost that remains after a certain number of nodes. Here B is the current node and L and R are the two child nodes.²

Although it is not clear that this algorithm actually optimizes SAH (Equation 1) because it focuses exclusively on the worst-case triangle cost and views the rest of the tree as an emergent phenomenon, some other properties are more readily apparent. The optimization of worst-case triangle cost biases the resulting trees towards balanced, and at every step the two child nodes are well separated or have limited overlap.

Measurements confirm that this overlap-avoidance leads to a well-minimized EPO cost: the average relative to LBVH is 0.45 for Bittner, 0.46 for BBVH, 0.81 for simulated annealing, and 1.07 for Agglo. This means that the expected amount of extra work at end points is more than twice as high for Agglo than for BBVH.

The other new cost function, LCV, is also well optimized; BBVH has the lowest average relative to LBVH, with Bittner 16% and Ag-

²Since the current node is a constant, this can equivalently be written as minimizing $C_{\text{tri}}(A_L T_L + A_R T_R)$, which reveals the more typical interpretation that this approximates the subtree costs with $C_{\text{tri}} A_L T_L$ and $C_{\text{tri}} A_R T_R$.

glo 27% higher than BBVH. We believe that this is the case because the worst-case optimization and the recursive use of splitting planes lead to non-overlapping leaf nodes whose spatial sizes are more uniform than in the other methods.

While we cannot know the global optimum of SAH cost, we can approximate it as the minimum achieved by any of the builders (except SBVH and STreelet that split triangles and thus have a different minimum). This reveals that BBVH leads to near-optimal results with uniformly tessellated objects such as DRAGON, but on the average its SAH cost is at least 16% higher than the optimum (and 15% higher than Bittner), and in BUBS it is as much as 60% above the optimum. It therefore seems that BBVH is much better at optimizing the two new metrics (EPO and LCV) than SAH, for which it was designed.

6 Discussion

The goal of this paper was to find an explanation for the measured performance, and to better explain what makes certain trees good for ray tracing. This knowledge allows one to quantify what went wrong when the expected performance gain fails to materialize. The next step could be to utilize this knowledge in constructing better trees. This will require further effort and approximations because the two new metrics are much harder to compute than SAH: EPO needs a scene-specific weight for best results and LCV is currently estimated via sampling.

Since the three descriptors seem to explain the performance so thoroughly, it could be that there are no other significant factors. Since our tree builders emit the nodes into different memory layouts — a fact realized only very late in the project — the theory predicts that memory layout is not a significant factor on our test platform. We verified this by randomly permuting the nodes in memory, and this affected the performance by less than 1%. This strongly contrasts with the conventional wisdom that memory layout is a key to good performance; apparently that is not true on current GPUs after all.

Several authors have recently augmented SAH with visibility information in order to accelerate shadow rays [Ize and Hansen 2011; Vinkler et al. 2012; Feltman et al. 2012]. It could be an interesting future effort to investigate how well their modified equations describe the performance outside the special case of shadow rays. It would also be interesting to test how well our metrics describe the performance of shadow rays that do not need to find the closest intersection. It seems possible that EPO plays a smaller role there, but that is yet to be verified.

Our observations suggest that it is important to quote the actual measurements when evaluating a new tree construction algorithm, instead of blindly relying on SAH as a proxy for ray tracing performance. It is also important to use a sufficient number of test scenes (10+) because scene-dependent variation in tree quality can be significant.

Acknowledgements

Peter Shirley, David McAllister, Jaakko Lehtinen, and anonymous reviewers for comments and additional references. Anat Grynberg and Greg Ward for CONFERENCE, University of Utah for FAIRY, Marko Dabrovic for SIBENIK, Ryan Vance for BUBS, UNC for POWERPLANT, Samuli Laine for HAIRBALL and VEGETATION, Guillermo Leal Laguno for SANMIGUEL, Johnathan Good for ARABIC, BABYLONIAN and ITALIAN, Stanford Computer Graphics Laboratory for ARMADILLO, BUDDHA and DRAGON, Cornell University for BAR, Georgia Institute of Technology for BLADE.

References

- AILA, T., AND LAINE, S. 2009. Understanding the efficiency of ray traversal on gpus. In *Proc. High Performance Graphics*, 145–149.
- AILA, T., LAINE, S., AND KARRAS, T. 2012. Understanding the efficiency of ray traversal on gpus – Kepler and Fermi addendum. Tech. Rep. NVR-2012-02, NVIDIA.
- BITTNER, J., HAPALA, M., AND HAVRAN, F. 2013. Fast insertion-based optimization of bounding volume hierarchies. *Computer Graphics Forum* 32, 1, 85–100.
- FELTMAN, N., LEE, M., AND FATAHALIAN, K. 2012. SRDH: specializing BVH construction and traversal order using representative shadow ray sets. In *Proc. High Performance Graphics*, 49–55.
- GOLDSMITH, J., AND SALMON, J. 1987. Automatic creation of object hierarchies for ray tracing. *IEEE Comput. Graph. Appl.* 7, 5, 14–20.
- HAVRAN, V. 2000. *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague.
- IZE, T., AND HANSEN, C. 2011. RTSAH traversal order for occlusion rays. *Comp. Graph. Forum* 30, 2, 297–305.
- JAKOB, W., 2010. Mitsuba renderer. <http://www.mitsuba-renderer.org>.
- KARRAS, T., AND AILA, T. 2013. Fast parallel construction of high-quality bounding volume hierarchies. In *Proc. High-Performance Graphics*.
- KARRAS, T. 2012. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. In *Proc. High-Performance Graphics*, 33–37.
- KENSLER, A. 2008. Tree rotations for improving bounding volume hierarchies. In *Proc. IEEE Symposium on Interactive Ray Tracing*, 73–76.
- LAUTERBACH, C., GARLAND, M., SENGUPTA, S., LUEBKE, D., AND MANOCHA, D. 2009. Fast BVH construction on GPUs. *Computer Graphics Forum* 28, 2, 375–384.
- MACDONALD, D. J., AND BOOTH, K. S. 1990. Heuristics for ray tracing using space subdivision. *Vis. Comput.* 6, 3, 153–166.
- NG, K., AND TRIFONOV, B. 2003. Automatic bounding volume hierarchy generation using stochastic search methods. In *Proc. Mini-Workshop on Stochastic Search Algorithms*.
- POPOV, S., GEORGIEV, I., DIMOV, R., AND SLUSALLEK, P. 2009. Object partitioning considered harmful: space subdivision for BVHs. In *Proc. High Performance Graphics*, 15–22.
- STICH, M., FRIEDRICH, H., AND DIETRICH, A. 2009. Spatial splits in bounding volume hierarchies. In *Proc. High-Performance Graphics*, 7–13.
- VINKLER, M., HAVRAN, V., AND SOCHOR, J. 2012. Visibility driven BVH build up algorithm for ray tracing. *Computers & Graphics* 36, 4, 283–296.
- WALTER, B., BALA, K., KULKARNI, M., AND PINGALI, K. 2008. Fast agglomerative clustering for rendering. In *Proc. IEEE Symposium on Interactive Ray Tracing*, 81–86.














Scene	Metric	Builder									Correlation with ns/ray		
		LBVH	BBVH	SBVH	Agglo	Bittner	H.Climb	Anneal.	Treelet	STreelet	SAH	SAH+EPO	SAH+EPO+LCV
	Scalar (ns/ray)	108.81	84.32	73.48	97.94	78.13	84.53	85.69	82.10	82.44	0.925	0.991 $_{\alpha=0.76}$	0.991 $_{\alpha=0.76,\beta=0.00}$
	SIMD (ns/ray)	9.50	6.11	5.22	7.11	5.78	6.20	6.15	6.00	6.11	0.975	0.988 $_{\alpha=0.45}$	0.996 $_{\alpha=0.13,\beta=0.78}$
	SAH	69.71	53.82	47.45	55.48	47.84	52.44	49.41	50.57	52.40			
	EPO	10.88	5.01	1.68	9.69	4.13	5.89	6.06	5.21	3.07			
	LCV	3.48	1.46	1.30	1.69	1.58	1.58	1.52	1.56	1.42			
	Scalar (ns/ray)	203.67	127.71	89.45	121.07	92.25	125.47	106.16	118.34	91.66	0.956	0.995 $_{\alpha=0.84}$	0.995 $_{\alpha=0.66,\beta=0.23}$
	SIMD (ns/ray)	16.58	8.92	6.44	9.03	7.02	9.18	8.03	8.96	6.97	0.961	0.985 $_{\alpha=0.75}$	0.993 $_{\alpha=0.19,\beta=0.77}$
	SAH	127.45	84.29	70.87	72.96	65.72	78.36	68.45	74.69	73.25			
	EPO	27.01	13.21	3.56	10.36	5.88	12.58	7.83	10.06	2.00			
	LCV	8.21	2.55	1.84	2.78	2.52	2.79	2.70	2.72	1.68			
	Scalar (ns/ray)	130.38	97.66	75.41	84.82	76.63	87.95	76.28	79.68	77.58	0.960	0.992 $_{\alpha=0.56}$	0.992 $_{\alpha=0.56,\beta=0.00}$
	SIMD (ns/ray)	12.77	7.29	5.64	6.97	6.54	7.00	5.98	6.46	6.08	0.949	0.987 $_{\alpha=0.60}$	0.999 $_{\alpha=0.12,\beta=0.77}$
	SAH	62.07	42.78	37.90	38.47	33.60	39.50	33.79	36.91	40.50			
	EPO	20.19	9.82	2.80	9.55	6.41	9.78	6.88	7.79	2.72			
	LCV	5.91	1.87	1.33	2.19	2.36	1.79	1.85	1.77	1.41			
	Scalar (ns/ray)	210.53	123.61	76.98	130.21	107.76	126.10	110.86	120.05	100.00	0.985	0.998 $_{\alpha=0.59}$	0.998 $_{\alpha=0.58,\beta=0.15}$
	SIMD (ns/ray)	24.27	10.77	6.01	12.67	9.71	11.01	9.76	10.60	8.64	0.990	0.993 $_{\alpha=0.31}$	0.998 $_{\alpha=0.34,\beta=0.48}$
	SAH	46.24	27.53	20.82	27.95	23.70	28.03	24.29	26.10	24.47			
	EPO	27.78	12.10	3.60	14.89	10.03	15.15	11.45	12.61	8.33			
	LCV	17.46	4.02	1.74	4.72	4.70	3.57	3.36	3.60	2.25			
	Scalar (ns/ray)	233.10	98.62	71.48	99.11	81.77	113.12	88.03	99.30	84.32	0.993	0.999 $_{\alpha=0.48}$	0.999 $_{\alpha=0.35,\beta=0.21}$
	SIMD (ns/ray)	19.57	6.77	4.88	7.85	6.81	8.06	6.68	7.37	6.01	0.982	0.991 $_{\alpha=0.61}$	0.997 $_{\alpha=0.17,\beta=0.61}$
	SAH	75.17	39.50	32.89	35.99	33.25	42.18	33.57	36.77	35.51			
	EPO	40.11	11.66	3.96	13.29	8.97	16.57	11.57	12.68	6.13			
	LCV	6.20	1.61	1.22	4.33	3.00	1.70	1.75	1.71	1.27			
	Scalar (ns/ray)	240.38	114.16	74.07	113.25	97.09	133.87	99.30	117.23	84.60	0.993	0.994 $_{\alpha=0.43}$	0.995 $_{\alpha=0.25,\beta=0.56}$
	SIMD (ns/ray)	22.99	8.38	5.30	8.98	7.92	10.30	7.79	9.42	6.60	0.985	0.985 $_{\alpha=0.07}$	0.995 $_{\alpha=0.07,\beta=0.87}$
	SAH	132.57	80.41	54.79	72.51	67.80	80.20	66.79	74.34	59.23			
	EPO	48.76	19.09	3.97	18.17	14.61	23.64	15.33	19.06	6.66			
	LCV	11.04	2.92	1.62	3.59	3.56	3.25	2.77	3.01	2.00			
	Scalar (ns/ray)	153.37	69.49	43.86	69.93	55.87	78.00	61.73	66.80	48.26	0.993	0.994 $_{\alpha=0.26}$	0.996 $_{\alpha=0.05,\beta=0.77}$
	SIMD (ns/ray)	10.70	4.38	2.71	4.73	3.71	5.07	4.00	4.49	3.14	0.989	0.990 $_{\alpha=0.27}$	0.995 $_{\alpha=0.03,\beta=0.85}$
	SAH	104.08	62.20	43.07	55.24	49.97	62.06	50.78	57.10	47.41			
	EPO	45.00	19.15	2.76	18.74	13.05	22.59	14.91	19.17	6.15			
	LCV	5.55	1.88	0.99	2.33	2.37	1.96	1.91	1.96	1.19			
	Scalar (ns/ray)	78.86	58.96	53.16	59.56	55.01	62.50	55.87	59.67	57.54	0.988	0.999 $_{\alpha=0.50}$	0.999 $_{\alpha=0.45,\beta=0.10}$
	SIMD (ns/ray)	5.60	3.73	3.39	3.97	3.72	4.16	3.64	4.06	3.86	0.978	0.989 $_{\alpha=0.51}$	0.993 $_{\alpha=0.07,\beta=0.82}$
	SAH	63.04	45.39	41.97	45.23	41.71	47.72	42.77	45.51	46.38			
	EPO	13.52	6.39	2.35	7.52	5.09	7.93	6.12	7.01	4.37			
	LCV	3.15	1.69	1.47	1.91	1.89	1.90	1.74	1.86	1.59			
	Scalar (ns/ray)	122.10	97.85	90.01	128.70	97.47	100.30	114.29	99.01	94.16	0.794	0.991 $_{\alpha=0.84}$	0.991 $_{\alpha=0.84,\beta=0.00}$
	SIMD (ns/ray)	13.98	8.87	7.79	12.13	8.98	9.26	10.38	9.27	8.87	0.959	0.989 $_{\alpha=0.49}$	0.992 $_{\alpha=0.21,\beta=0.69}$
	SAH	133.17	105.25	98.30	114.65	99.75	106.83	109.14	103.77	104.39			
	EPO	11.10	6.29	2.85	16.12	6.79	7.55	13.05	7.18	3.76			
	LCV	5.34	2.06	1.83	2.58	2.15	2.15	2.23	2.12	1.84			
	Scalar (ns/ray)	92.34	75.87	75.82	89.61	80.26	81.43	88.11	81.83	81.50	0.922	0.994 $_{\alpha=0.44}$	0.996 $_{\alpha=0.14,\beta=0.76}$
	SIMD (ns/ray)	7.86	6.02	6.05	7.23	6.42	6.55	7.07	6.69	6.65	0.966	0.986 $_{\alpha=0.26}$	0.989 $_{\alpha=0.07,\beta=0.87}$
	SAH	43.75	33.25	33.23	38.48	34.35	36.61	37.49	35.25	35.65			
	EPO	3.07	1.43	1.40	7.84	3.20	2.86	7.04	3.31	2.64			
	LCV	1.84	1.25	1.25	1.32	1.29	1.30	1.28	1.32	1.30			
	Scalar (ns/ray)	108.81	91.66	88.97	110.62	94.25	96.15	105.71	95.79	92.94	0.905	0.994 $_{\alpha=0.67}$	0.995 $_{\alpha=0.22,\beta=0.72}$
	SIMD (ns/ray)	10.68	8.32	8.00	10.10	8.49	8.86	9.58	9.01	8.75	0.979	0.991 $_{\alpha=0.32}$	0.991 $_{\alpha=0.18,\beta=0.50}$
	SAH	81.65	64.48	62.05	74.19	64.58	69.46	71.67	67.91	65.80			
	EPO	6.83	3.84	2.31	11.89	5.04	5.42	9.85	5.66	3.50			
	LCV	2.14	1.42	1.36	1.57	1.49	1.48	1.44	1.48	1.39			
	Scalar (ns/ray)	89.61	75.41	75.41	103.41	78.68	80.13	93.37	79.68	81.57	0.936	0.999 $_{\alpha=0.65}$	0.999 $_{\alpha=0.43,\beta=0.37}$
	SIMD (ns/ray)	8.33	6.25	6.25	8.66	6.55	6.65	7.76	6.81	7.21	0.989	0.990 $_{\alpha=0.10}$	0.990 $_{\alpha=0.10,\beta=0.00}$
	SAH	94.05	73.38	73.39	96.43	75.41	79.45	86.99	77.64	80.32			
	EPO	2.30	0.15	0.15	14.48	1.64	1.51	8.96	1.62	1.78			
	LCV	1.83	1.05	1.05	1.16	1.08	1.09	1.12	1.09	1.09			
	Scalar (ns/ray)	460.83	278.55	141.64	287.36	274.73	281.69	277.78	273.97	138.12	0.998	0.998 $_{\alpha=0.05}$	1.000 $_{\alpha=0.62,\beta=0.38}$
	SIMD (ns/ray)	67.11	32.47	17.45	35.09	33.78	34.60	33.44	34.69	18.06	0.980	0.980 $_{\alpha=0.00}$	0.999 $_{\alpha=0.35,\beta=0.65}$
	SAH	175.61	119.73	75.16	119.64	111.91	120.49	117.31	118.45	73.30			
	EPO	191.95	113.75	13.89	114.99	105.47	112.74	112.29	111.31	9.86			
	LCV	134.14	24.09	3.56	23.86	27.37	25.17	22.28	24.55	3.72			

Table 3: Measurements for the rest of the test scenes.