Performance Comparison of Bounding Volume Hierarchies for GPU Ray Tracing

Daniel Meister
Advanced Micro Devices, Inc.

Jiří Bittner
Czech Technical University in Prague

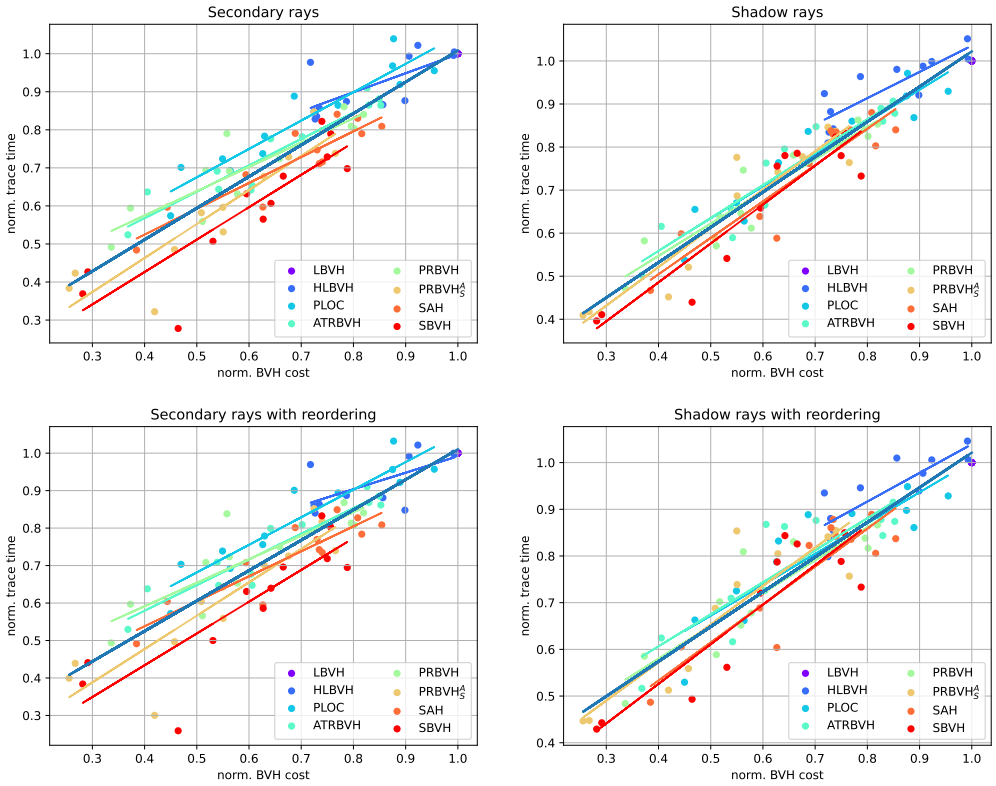

Figure 1. The correlation between BVH costs and tracing times for secondary and shadow rays using binary BVHs. Both the BVH cost and the trace times are normalized by LBVH.

Abstract

Ray tracing is an inherent component of modern rendering algorithms. The bounding volume hierarchy (BVH) is a commonly used acceleration data structure employed in most ray tracing frameworks. Through the last decade, many methods addressing ray tracing with bounding volume hierarchies have been proposed. However, most of these methods were typically

compared to only one or two reference methods. The acceleration provided by a particular BVH depends heavily on its construction algorithm. Even experts in the field dispute which method is the best. To add more insight into this challenge, we empirically compare the most popular methods addressing BVHs in the context of GPU ray tracing. Moreover, we combine the construction algorithms with other enhancements such as spatial splits, ray reordering, and wide BVHs. To estimate how close we are from the best performing BVH, we propose a novel method using global optimization with simulated annealing and combine it with the existing best-performing BVH builders. For the sake of fairness and consistency, all methods are evaluated in a single unified framework, and we make all source code publicly available. We also study the correlation between tracing times and the estimated traversal cost induced by the surface area heuristic (SAH).

1. Introduction

Ray tracing stays at the core of many physically based rendering algorithms. These algorithms model light transport by means of geometric optics, assuming that light travels instantaneously through the medium in straight lines. Ray tracing serves as an underlying engine to find the nearest intersections with scene geometry, to test visibility between two points, or to find all ray-scene intersections. The challenge is that we have to trace a large number of rays to achieve plausible results. Therefore, the scene geometry is arranged into various spatial data structures to accelerate the search for intersections. Modern ray tracing engines almost exclusively employ a bounding volume hierarchy (BVH) as an acceleration data structure as it provides excellent ray tracing performance, scalable construction times, and predictable memory footprints.

Over the last two decades, many different algorithms and methods addressing BVHs have been proposed. Nonetheless, they were tested only in a very limited setting, often against only one or very few reference algorithms. Moreover, using a combination of multiple techniques at once is usually simplified to a claim that individual techniques are orthogonal to each other, which might not be necessarily the truth in practice.

In this paper, we present a performance comparison of the BVH algorithms and techniques with a focus on GPU-based methods. To gain more insight into how close the state-of-the-art techniques are from the optimal BVH for the given scene, we extend and combine the state-of-the-art BVH build methods that provided the best currently known BVHs. In particular, we propose a modification of the parallel reinsertion BVH (PRBVH) method that includes a simulating annealing–based global optimization, and we combine this method with the well-known split BVH (SBVH) method using spatial splits.

We mainly target offline rendering or static content in interactive or real-time applications, where the BVH is reused many times and thus the build time becomes

negligible with respect to trace times. Therefore, our comparison focuses on the ray tracing performance that can be achieved using the particular BVH construction algorithm. To make the evaluation more informative, we also briefly discuss the BVH build times.

We compare the performance of ten different construction algorithms that are either existing well-known algorithms or newly designed modifications and combinations. For each construction algorithm, we evaluate the ray tracing performance of resulting BVHs with different branching factors. We also study the influence of ray reordering on the BVH ray tracing performance. All evaluated methods are implemented in a single unified framework in order to make the comparison as fair as possible using a path tracer as the target rendering algorithm.

There are several works directly related to our paper. Meister et al. [2021] provided a comprehensive overview of bounding volume hierarchies methods. However, an empirical analysis was missing. Aila et al. [2013] provided a performance comparison focusing on the BVH quality metrics and their correlation with the actual tracing times limited to binary BVHs. Vinkler et al. [2016] conducted a performance comparison of bounding volume hierarchies and KD-trees on the GPU.

2. Background

In this section, we briefly describe the methods that we use for our comparison. For details, we refer to the individual papers or a recent survey by Meister et al. [2021].

2.1. Cost Model

We can estimate the quality of a particular BVH in terms of the expected number of operations needed to find the nearest ray-triangle intersection. Using the surface area heuristic (SAH) [Goldsmith and Salmon 1987], we can express the cost function as follows:

$$c(N) = \begin{cases} c_T + \sum_{N_c} \frac{SA(N_c)}{SA(N)} c(N_c) & \text{if } N \text{ is an interior node,} \\ c_I |N| & \text{otherwise,} \end{cases} \tag{1}$$

where $c(N)$ is the cost of a subtree with root N, N_c is a child of node N, $SA(N)$ is a surface area of the bounding box of node N, and $|N|$ is the number of triangles in node N. Constants c_T and c_I express the average cost of the traversal and intersection steps, respectively. We can remove the recurrence by unrolling:

$$c(N) = \frac{1}{SA(N)} \left[c_T \sum_{N_i} SA(N_i) + c_I \sum_{N_l} SA(N_l)|N_l| \right], \tag{2}$$

where N_i and N_l are interior and leaf nodes, respectively. This model only roughly estimates what is happening in reality as it does not take into account many other fac-

tors such as the actual ray distribution, occlusions, underlying hardware architecture, or a particular rendering algorithm. However, most of the construction algorithms minimize this cost function (at least locally) to maximize the actual ray tracing performance. Notice that if we use one triangle per leaf, then the cost function reduces simply to the sum of the surface areas of the bounding boxes in interior nodes, which is a fact that some of the construction algorithms exploit.

Modern hardware architectures are equipped with SIMD units that allow the processing of multiple intersections tests at once, for example, in wide BVHs, where a single ray is tested against multiple bounding boxes. Triangles in leaves are processed in a similar fashion, and thus it is desirable to have the number of triangles aligned with the SIMD width as the number of executed intersection tests is equal to the SIMD width. Motivated by this fact, Wald et al. [2008] proposed a modified version of the cost model:

$$c(N) = \frac{1}{SA(N)} \left[c_T \sum_{N_i} SA(N_i) + c_I \sum_{N_l} SA(N_l) k \left\lceil \frac{|N_l|}{k} \right\rceil \right], \tag{3}$$

where k is the branching factor or the SIMD width. The modified cost mode penalizes leaves if their sizes are not aligned to k. Notice that we multiply the term by k to make c_T independent of k.

2.2. Construction

We first briefly describe the BVH construction algorithms evaluated in this paper.

Linear BVH *Linear BVH* (LBVH) is a very fast algorithm that reduces the problem of the BVH construction to the sorting problem. The idea is to sort triangles along a space-filling curve such as the Morton curve [Morton 1966]. The Morton curve subdivides space into a uniform grid, where the order along the curve is given by the Morton codes such that each Morton code corresponds to one cell of the grid. There is a simple mapping between cell coordinates and Morton codes by interleaving successive bits of cell coordinates. To construct a BVH over triangles, we approximate each triangle by a single point (e.g., the centroid of its bounding box), project it to the corresponding cell to compute its Morton code, and sort the triangles using the Morton codes as sorting keys. The Morton curve encodes an implicit binary BVH constructed by spatial median splits, where the most significant bit corresponds to the top-most split and so on. Karras [2012] proposed an algorithm that constructs the topology of this BVH in a single kernel launch, while the bounding boxes are computed separately in an additional kernel launch in a bottom-up fashion using atomic counters to prevent race conditions.

HLBVH LBVH is very fast, but it does not take into account the surface area heuristic, and thus the resulting BVHs are of rather inferior quality. To address this issue,

several researchers proposed to combine LBVH with SAH, which is known as *hierarchical LBVH* (HLBVH) [Pantaleoni and Luebke 2010; Garanzha et al. 2011]. Garanzha et al. [2011] proposed to construct bottom levels by LBVH and then top levels in a top-down manner by binning. To reduce the workload for binning, the algorithm constructs the bottom levels by LBVH using fewer significant bits of Morton codes. Root nodes of bottom-level trees are further fed to binning using more significant bits as bin indices. Top levels are constructed level by level using double buffering, where each level requires a separate kernel launch.

Parallel locally ordered clustering *Parallel locally ordered clustering* (PLOC) [Meister and Bittner 2018a] is a parallel bottom-up construction algorithm based on agglomerative clustering. The algorithm starts from the individual triangles by considering them as clusters. Then, in each iteration, it merges multiple cluster pairs in parallel, repeating this until only one cluster remains, which corresponds to the root node. The algorithm employs the Morton curve to find the nearest neighbors efficiently.

Treelet restructuring *Treelet restructuring* (TRBVH) is a parallel optimization method that improves the quality of an existing BVH [Karras and Aila 2013]. The algorithm starts from the leaves and proceeds up to the root until a given treelet size is reached. Then, the treelets are restructured in order to locally minimize the BVH cost induced by a particular treelet. After the treelet is restructured, the algorithm continues one level above and repeats the process. To avoid race conditions, a node is only processed by the second thread (in the case of binary BVHs) using atomic counters to be sure that both children were already processed. Karras and Aila [2013] restructured the treelet in an optimal way regarding the cost function using a dynamic programming approach. Later, Domingues and Pedrini [2015] showed that better results could be achieved using agglomerative clustering (agglomerative treelet restructuring, ATRBVH), which is simple and thus allows for restructuring large treelets. This approach provides BVHs of very good quality in reasonable times.

Parallel reinsertion *Parallel reinsertion* (PRBVH) [Meister and Bittner 2018b] is an optimization algorithm that iteratively modifies the BVH topology by local updates in order to decrease the global cost. The algorithm iteratively removes whole subtrees and inserts them into new positions while tracking the cost reduction for each such update. Thus, the cost function is systematically optimized as we know how exactly each update will influence the global cost, which is not the case of other methods that optimize the cost function only locally. The key observation is that we do not need to remove the node to find a new position. This fact allows us to search for new positions for all nodes in parallel, starting from original positions using simple state logic with parent links. To prevent topological conflicts, we employ atomic locks, prioritizing nodes with higher cost reduction. The algorithm always admits updates that reduce

the global cost, which guarantees the global cost reduction. However, the algorithm may get stuck in a local minimum. Therefore, we propose to combine the algorithm with simulated annealing, which may also admit updates that temporally increase the global cost based on the stochastic decision to prevent getting stuck in local minima.

We use the simulated annealing in a similar manner as Kensler [2008] proposed for the rotation-based optimization. The acceptance probability is given by the Boltzmann factor:

$$P(\Delta d, T) = \begin{cases} \min(e^{-\Delta d/T}, 1) & T > 0, \\ 0 & T = 0, \end{cases} \tag{4}$$

where Δd is the difference in surface area of the proposed change and T is the current temperature. The temperature is a clamped sine function:

$$T(i) = \max\left(0, -\sin\left(\frac{2\pi i}{f}\right)\right) T_{\text{max}} \frac{I - i}{I}, \tag{5}$$

where is f is the frequency of the sine function, T_{max} is the hottest temperature allowed, i is the current iteration, and I is the total number of iterations.

Nonetheless, in the case of reinsertion, there are two factors that make it more complicated. The first factor is that the search space is huge, unlike the tree rotations, where the space is limited to the constant number of rotations. Similarly, that is why the method uses search space pruning [Meister and Bittner 2018b], which should also be stochastic, as the acceptance. However, we would search practically the whole space for high temperatures, which is not desirable. This effect is amplified through multiple iterations as the original position used for pruning gets worse in each iteration during the heating phase. Thus, we use probability P (Equation (4)) clamped to P_{pruning} (a parameter of typically small value). Note that for the pruning we keep the best value found so far not necessarily corresponding to the last accepted value, which may also be negative.

The second factor is the parallel processing, which causes two more issues. First, in the original algorithm [Meister and Bittner 2018b], there are two locking schemes to prevent race conditions: aggressive strategy (lock only nodes that are modified) and conservative strategy (lock the modified nodes and all nodes between them). The aggressive strategy is more efficient as it allows to perform more reinsertions in parallel as fewer nodes are locked. Nevertheless, this approach is not applicable for simulated annealing because inserting two subtrees into each other may result in a cycle, disconnecting the tree into two separate parts. In the case of hill climbing, these cases will not happen as they are automatically rejected because they do not provide any cost reduction. We resolve this issue by using the conservative strategy when the temperature is positive. Second, we need to resolve the conflicts between nodes. Prioritizing nodes with higher cost reductions would cause the deterministic reinsertions (with positive cost reduction) to discard most of the stochastic reinsertions (with negative

cost reduction), resulting in practically no change, assuming that the BVH is already optimized. During the heating phase, we basically want to worsen the BVH to be able to improve it later when the temperature reaches zero again. Therefore, during the heating phase, we prioritize nodes with negative cost reduction.

Spatial splits A BVH may adapt poorly to overlapping triangles with nonuniform sizes or long thin diagonal triangles. We can relax the restriction that each triangle is referenced only once, allowing us to achieve tighter bounding boxes at the cost of a higher number of triangle references in leaves. SBVH [Stich et al. 2009] is a top-down construction algorithm that allows, besides standard objects, also spatial splits in the same manner as in the KD-tree construction, selecting one that minimizes the BVH cost. This algorithm provides BVHs of the highest quality at the cost of slower construction and more difficult parallelization. Unlike standard BVHs, BVHs with spatial splits cannot be easily updated by refitting the bounding volumes, which might cause issues in handling dynamic scenes.

2.3. Traversal

Traversal on the GPU might be challenging due to warp divergence and incoherent memory accesses. Aila and Laine [2009] proposed a stack-based traversal algorithm with persistent warps and dynamic fetch. The algorithm spawns an optimal number of warps that fetch rays from the global queue. To prevent warp divergence, the traversal is divided into two independent loops processing interior and leaf nodes separately (i.e., the *while-while* traversal). If all active threads in the warp reached leaf nodes, then the algorithm switches to the second while loop and tests the triangles in the leaf nodes. Dynamic fetch allows to fetch new rays if a certain number of threads are inactive to keep parallel resources occupied enough.

Lier et al. [2018] proposed a stack-based traversal algorithm for wide BVHs. The idea is to process a single node by multiple threads in a similar manner as nodes are processed on the CPU using SIMD processing.

2.4. Ray Reordering

Ray reordering is a technique that regroups the rays in order to increase control flow, increase cache-hit ratios, and decrease warp divergence [Meister et al. 2020]. Using ray reordering only pays off if the performance gain outweighs additional overhead. Ray reordering is typically implemented by sorting the rays along the Morton curve. The crucial part is how the ray as a 5D entity is encoded into a Morton code, which is not as straightforward as for 3D points.

3. Implementation

In this section, we provide implementations details of the evaluated methods. Most of the tested methods have publicly available implementations. However, to provide a fair comparison, we have to use the same settings for all methods, which might be difficult in different frameworks. Thus, we created a single unified framework with publicly available implementations of the algorithm integrated into it. Our framework is based on Aila's ray tracing framework [Aila and Laine 2009].

Aila's framework contains high-performance traversal kernels for binary BVHs. To support wide BVHs, we adopted the publicly available implementations of the method proposed by Lier et al. [2018]. In the original version of these kernels, the leaf size is limited to the branching factor of the BVHs. We modified the kernels to support arbitrary leaf sizes.

For LBVH [Karras 2012] and HLBVH [Garanzha et al. 2011], we use our implementation. For PLOC [Meister and Bittner 2018a], ATRBVH [Domingues and Pedrini 2015], and PRBVH [Meister and Bittner 2018b], we integrated publicly available implementations into our framework. For SBVH [Stich et al. 2009], we use the implementation that is available in Aila's framework [Aila and Laine 2009]. We implemented the collapsing algorithm of Ylitie et al. [2017] to convert binary BVHs to wide ones with adaptive leaf sizes while minimizing the BVH cost (Equation (3)). The data layout (i.e., how nodes are stored in the memory) has a significant impact on the actual ray tracing performance. Various methods provide BVHs in different data layouts. Hence, to provide a fair comparison, we transform each BVH to the breadth-first-search (BFS) layout prior to the actual ray tracing. The nodes are stored in the *array-of-structures* fashion, where each node contains bounding boxes of children and child indices, taking $32k$ bytes per node for k-ary BVHs.

We implemented a wavefront path tracing with next event estimation on top of our ray tracing framework. We can optionally use ray reordering to sort secondary and shadow rays. In the case of sorting, we access the rays indirectly through sorted indices in the traversal kernels to minimize the overhead caused by the actual reordering of the rays [Meister et al. 2020]. We use the compacted Aila's method to compute the Morton codes. We employ the radix sort algorithm by Merrill and Grimshaw [2011] for both construction algorithms and ray reordering.

4. Results and Discussion

We evaluated selected methods on 12 scenes of various complexity (see Figure 2). The path tracer uses 32 samples per pixel with the next event estimation using two shadow rays per hit with the maximum recursion depth 8. We do not employ Russian roulette and so in closed interior scenes the maximum recursion depth is often reached; in exterior scenes, the light paths are significantly shorter. All images were

rendered in 1024×768 resolution. The reported numbers are averaged over three different camera views to reduce the influence of view dependency. We evaluated all methods with and without ray reordering. The reordering overhead is included in trace times for the sake of fairness; both the traversal and the reordering are implemented on the same platform, and thus we expect that similar performance results could be obtained using hardware accelerated units for reordering and tracing. All experiments were executed on the AMD Ryzen 9 5950X CPU and AMD Radeon 6800 XT GPU and implemented using Heterogeneous-Computing Interface for Portability (HIP) as part of ROCm 4.4.

We used six construction methods for our experiments: LBVH, HLBVH, PLOC, ATRBVH, PRBVH, and SBVH. We used 60-bit Morton code for LBVH, HLBVH, and PLOC. For HLBVH, we used 15 bits for the SAH splits. For PLOC, we used a search radius of 100. For ATRBVH, we used 20 iterations and LBVH for the initial construction. For PRBVH, we used four configurations: PRBVH (LBVH as the base BVH with hill climbing), PRBVH^{A} (LBVH as the base BVH with simulated annealing), PRBVH_{S} (SBVH as the base BVH with hill climbing), and $\text{PRBVH}_{\text{S}}^{\text{A}}$ (SBVH as the base BVH with simulated annealing). We used $f = 50$, $T_{\text{max}} = 1$, $I = 3000$, and $P_{\text{pruning}} = 0.01$ for the simulated annealing. For SBVH, we used 128 bins for spatial splitting. We can disable spatial splits, which is equivalent to the standard full sweep top-down construction (denoted by SAH). We used $c_T = 3$ and $c_I = 2$ for the BVH cost computation and optimization. These values were established by experimental evaluation as maximizing the trace performance on the target platform. We evaluated all methods using three different branching factors: 2 (binary BVHs), 4 (quaternary BVHs), and 8 (octal BVHs). (These branching factors are supported by the existing traversal kernels.)

We present a compact summary of the results in Tables 1 and 2, where the values are averaged over all scenes. A detailed overview of our results is in Tables 3, 4, and 5 for branching factors 2, 4, and 8, respectively. For each method, we report the BVH cost and trace performance for secondary and shadow rays with and without ray reordering.

4.1. BVH Cost

$\text{PRBVH}_{\text{S}}^{\text{A}}$ achieves the lowest BVH cost across all configurations, showing that SBVH can be optimized even further (about 7% on average for binary BVHs). We can observe that the improvement of the BVH cost achieved by simulated annealing is only marginal. Therefore, we assume that the resulting BVHs are close to the optimal SAH cost for the given scene. Interestingly, PRBVH and PRBVH^{A} provide binary BVHs of similar costs as SBVH. SAH, which was for a long time considered as a reference solution, provides BVHs with relatively high costs on average, higher than PLOC and ATRBVH.

Br. Factor	Rays	LBVH	HLBVH	PLOC	ATRBVH	PRBVH	PRBVH^{A}	PRBVH$_{S}$	PRBVH$_{S}^{A}$	SAH	SBVH
Without Reordering											
2	Secondary	1.00	1.10	1.25	1.39	1.45	1.48	1.85	1.84	1.43	**1.86**
	Shadow	1.00	1.08	1.32	1.39	1.47	1.47	**1.66**	1.64	1.40	1.63
4	Secondary	1.04	1.16	1.31	1.39	1.49	1.50	**1.76**	1.74	1.41	1.73
	Shadow	0.92	1.02	1.19	1.24	1.32	1.32	**1.40**	1.39	1.23	1.36
8	Secondary	0.98	1.08	1.21	1.31	1.40	1.42	**1.58**	1.57	1.30	1.54
	Shadow	0.86	0.94	1.07	1.12	1.17	1.18	**1.23**	1.22	1.10	1.19
With Reordering											
2	Secondary	1.09	1.20	1.36	1.50	1.56	1.58	1.98	1.96	1.54	**2.01**
	Shadow	1.23	1.33	1.58	1.64	1.75	1.72	**1.89**	1.87	1.66	1.87
4	Secondary	1.17	1.30	1.44	1.53	1.62	1.63	**1.90**	1.88	1.54	1.88
	Shadow	1.06	1.16	1.33	1.38	1.45	1.45	**1.51**	1.50	1.37	1.48
8	Secondary	1.08	1.19	1.31	1.41	1.50	1.52	**1.68**	1.65	1.41	1.64
	Shadow	0.92	0.99	1.12	1.16	1.22	1.22	**1.26**	1.25	1.15	1.22

Table 1. Relative ray tracing performance of BVH builds using the tested construction algorithms. The relative trace speed for tracing secondary and shadow rays are shown with and without ray sorting using BVHs of different branching factors. The trace speed is normalized by the speed of the binary LBVH and averaged over all tested scenes.

Scene	LBVH	HLBVH	PLOC	ATRBVH	PRBVH	PRBVH^{A}	PRBVH$_{S}$	PRBVH$_{S}^{A}$	SAH	SBVH
Sibenik	2.0	7.5	17.2	8.2	22.0	3843	2811	7269	281	2801
Crytek Sponza	2.3	7.9	20.9	12.8	52.0	11218	6494	19427	1266	6467
Conference	2.6	7.9	21.2	15.2	107	25215	4347	30759	1561	4271
Gallery	3.9	11.0	22.0	21.3	99.0	37018	20904	63757	4915	20851
Happy Buddha	4.2	11.3	26.6	22.9	123	40294	19062	62096	5494	19004
Sodahall	7.3	15.9	35.2	50.1	453	101822	29896	140533	12024	29575
Hairball	8.7	17.4	55.1	60.7	543	154483	219852	682155	14049	218225
Crown	15.5	26.1	66.5	110	799	223908	51998	292668	30168	51623
Pompeii	17.5	31.7	63.5	145	3500	325587	74168	421387	31482	73344
San Miguel	24.4	38.5	96.3	183	2309	531512	78568	552505	51540	77480
Vienna	27.1	43.2	82.7	207	4254	536393	77383	612130	53257	76493
Powerplant	36.7	53.0	129	292	5752	837940	147898	907441	88119	146337
Avg. build time	1.0	2.3	5.6	6.6	76.1	13720	5237	22134	1444	5196

Table 2. Build times in milliseconds for binary BVHs. The last line shows the average build times; the times are normalized prior to averaging by a value given by LBVH.

4.2. Trace Speed

The results show that PRBVH$_{S}$, PRBVH$_{S}^{A}$, and SBVH achieve the highest trace performance. In most cases, PRBVH$_{S}$ performs better than SBVH with the exception of secondary rays with binary BVHs, which achieves slightly higher trace speed than PRBVH$_{S}$ and PRBVH$_{S}^{A}$. The results indicate that PRBVH can improve an input SBVH in some cases. Interestingly, PRBVH$_{S}^{A}$ is slightly worse than PRBVH$_{S}$. An explanation could be that the simulated annealing breaks optimized top-level splits of the SBVH (see Section 4.3).

We can observe that the trace performance drops significantly with increasing branching factor for both secondary and shadow rays. This behavior is the most pronounced for primary rays that exhibit the highest coherence (the tables do not show numbers for primary rays, as these constitute a small fraction of the render-

Scene	Statistic	LBVH	HLBVH	PLOC	ATRBVH	PRBVH	PRBVH^{A}	PRBVH$_{S}$	PRBVH$_{S}^{A}$	SAH	SBVH
	BVH cost	177 (1.00)	139 (0.79)	136 (0.77)	124 (0.70)	116 (0.66)	114 (0.65)	112 (0.63)	**111 (0.63)**	129 (0.73)	118 (0.67)
Sibenik	Secondary rays	261 (1.00)	299 (1.14)	302 (1.16)	334 (1.28)	361 (1.38)	364 (1.39)	**389 (1.49)**	386 (1.48)	349 (1.34)	385 (1.47)
# triangles	Shadow rays	604 (1.00)	627 (1.04)	702 (1.16)	712 (1.18)	773 (1.28)	803 (1.33)	**814 (1.35)**	814 (1.35)	721 (1.19)	769 (1.27)
75k	Second. rays reord.	311 (1.19)	351 (1.34)	348 (1.33)	385 (1.47)	415 (1.59)	418 (1.60)	**449 (1.72)**	443 (1.70)	405 (1.55)	447 (1.71)
	Shadow rays reord.	891 (1.47)	941 (1.56)	1000 (1.66)	1016 (1.68)	1073 (1.78)	**1107 (1.83)**	1103 (1.83)	1107 (1.83)	1035 (1.71)	1078 (1.79)
	BVH cost	268 (1.00)	195 (0.73)	168 (0.63)	162 (0.60)	155 (0.58)	153 (0.57)	149 (0.56)	**148 (0.55)**	198 (0.74)	168 (0.63)
Crytek Sponza	Secondary rays	159 (1.00)	190 (1.20)	215 (1.36)	247 (1.56)	251 (1.58)	278 (1.75)	298 (1.88)	**298 (1.88)**	222 (1.40)	281 (1.77)
# triangles	Shadow rays	401 (1.00)	455 (1.13)	553 (1.38)	603 (1.50)	**656 (1.63)**	656 (1.63)	583 (1.45)	584 (1.46)	481 (1.20)	531 (1.32)
262k	Second. rays reord.	205 (1.29)	238 (1.50)	271 (1.71)	302 (1.91)	313 (1.97)	336 (2.11)	**368 (2.32)**	366 (2.31)	279 (1.76)	349 (2.20)
	Shadow rays reord.	597 (1.49)	678 (1.69)	758 (1.89)	822 (2.05)	880 (2.19)	**885 (2.21)**	808 (2.02)	807 (2.01)	708 (1.77)	758 (1.89)
	BVH cost	154 (1.00)	112 (0.72)	85 (0.55)	83 (0.54)	80 (0.52)	79 (0.51)	79 (0.51)	**79 (0.51)**	92 (0.59)	92 (0.60)
Conference	Secondary rays	278 (1.00)	327 (1.18)	384 (1.38)	402 (1.45)	401 (1.44)	403 (1.45)	**487 (1.75)**	477 (1.72)	407 (1.47)	439 (1.58)
# triangles	Shadow rays	525 (1.00)	648 (1.24)	783 (1.49)	799 (1.52)	821 (1.57)	833 (1.59)	**898 (1.71)**	892 (1.70)	821 (1.57)	797 (1.52)
331k	Second. rays reord.	331 (1.19)	385 (1.39)	449 (1.62)	468 (1.68)	468 (1.69)	469 (1.69)	**562 (2.03)**	549 (1.98)	468 (1.69)	525 (1.89)
	Shadow rays reord.	836 (1.59)	1047 (1.99)	1153 (2.20)	1178 (2.25)	1191 (2.27)	1208 (2.30)	**1234 (2.35)**	1215 (2.32)	1214 (2.31)	1160 (2.21)
	BVH cost	197 (1.00)	195 (0.99)	188 (0.95)	167 (0.85)	161 (0.82)	157 (0.80)	146 (0.74)	**146 (0.74)**	159 (0.81)	149 (0.76)
Gallery	Secondary rays	148 (1.00)	149 (1.00)	155 (1.05)	168 (1.13)	176 (1.19)	178 (1.20)	187 (1.26)	186 (1.25)	179 (1.20)	**188 (1.27)**
# triangles	Shadow rays	355 (1.00)	338 (0.95)	382 (1.08)	392 (1.10)	416 (1.17)	416 (1.17)	**427 (1.20)**	424 (1.20)	403 (1.14)	426 (1.20)
998k	Second. rays reord.	165 (1.11)	165 (1.11)	172 (1.16)	186 (1.25)	196 (1.32)	197 (1.33)	202 (1.37)	202 (1.36)	199 (1.34)	**205 (1.38)**
	Shadow rays reord.	479 (1.35)	458 (1.29)	516 (1.45)	524 (1.48)	553 (1.56)	553 (1.56)	563 (1.58)	561 (1.58)	539 (1.52)	**564 (1.59)**
	BVH cost	204 (1.00)	184 (0.91)	178 (0.87)	168 (0.83)	159 (0.78)	156 (0.76)	148 (0.73)	**147 (0.72)**	156 (0.77)	150 (0.74)
Happy Buddha	Secondary rays	105 (1.00)	105 (1.01)	108 (1.03)	115 (1.10)	122 (1.16)	122 (1.16)	125 (1.20)	124 (1.18)	124 (1.19)	**127 (1.22)**
# triangles	Shadow rays	159 (1.00)	161 (1.01)	173 (1.09)	179 (1.12)	184 (1.16)	186 (1.17)	189 (1.19)	188 (1.18)	189 (1.19)	**193 (1.21)**
1087k	Second. rays reord.	98 (0.93)	98 (0.94)	102 (0.97)	107 (1.02)	112 (1.07)	112 (1.07)	116 (1.11)	115 (1.09)	115 (1.10)	**117 (1.12)**
	Shadow rays reord.	135 (0.85)	138 (0.87)	150 (0.94)	154 (0.97)	157 (0.98)	158 (0.99)	162 (1.02)	160 (1.01)	161 (1.01)	**165 (1.04)**
	BVH cost	254 (1.00)	217 (0.86)	174 (0.69)	163 (0.64)	143 (0.56)	141 (0.56)	140 (0.55)	**140 (0.55)**	186 (0.73)	163 (0.64)
Sodahall	Secondary rays	220 (1.00)	255 (1.15)	248 (1.13)	284 (1.29)	319 (1.45)	319 (1.45)	**375 (1.70)**	370 (1.68)	310 (1.41)	363 (1.65)
# triangles	Shadow rays	603 (1.00)	615 (1.02)	721 (1.20)	758 (1.26)	**808 (1.34)**	806 (1.34)	786 (1.30)	777 (1.29)	727 (1.21)	773 (1.28)
2169k	Second. rays reord.	240 (1.09)	272 (1.23)	266 (1.21)	300 (1.36)	331 (1.50)	332 (1.50)	**378 (1.72)**	373 (1.69)	323 (1.46)	375 (1.70)
	Shadow rays reord.	667 (1.11)	660 (1.09)	750 (1.24)	772 (1.28)	**824 (1.37)**	809 (1.34)	792 (1.31)	781 (1.29)	759 (1.26)	790 (1.31)
	BVH cost	1233 (1.00)	1225 (0.99)	1082 (0.88)	1051 (0.85)	981 (0.80)	979 (0.79)	835 (0.68)	**834 (0.68)**	1054 (0.85)	925 (0.75)
Hairball	Secondary rays	74 (1.00)	74 (1.00)	71 (0.96)	86 (1.16)	91 (1.24)	89 (1.20)	**104 (1.41)**	101 (1.37)	91 (1.24)	101 (1.37)
# triangles	Shadow rays	170 (1.00)	169 (1.00)	175 (1.03)	193 (1.14)	202 (1.19)	200 (1.18)	**221 (1.30)**	218 (1.29)	202 (1.19)	217 (1.28)
2880k	Second. rays reord.	79 (1.07)	80 (1.08)	77 (1.04)	92 (1.25)	98 (1.32)	96 (1.30)	**112 (1.52)**	109 (1.48)	98 (1.33)	110 (1.49)
	Shadow rays reord.	192 (1.13)	191 (1.12)	202 (1.19)	219 (1.29)	229 (1.35)	228 (1.34)	**247 (1.46)**	243 (1.43)	229 (1.35)	243 (1.43)
	BVH cost	76 (1.00)	70 (0.92)	68 (0.89)	63 (0.83)	61 (0.80)	60 (0.79)	58 (0.77)	**58 (0.77)**	62 (0.82)	60 (0.79)
Crown	Secondary rays	94 (1.00)	92 (0.98)	103 (1.09)	109 (1.16)	118 (1.24)	115 (1.22)	131 (1.39)	128 (1.35)	120 (1.27)	**135 (1.43)**
# triangles	Shadow rays	221 (1.00)	221 (1.00)	254 (1.15)	257 (1.16)	267 (1.21)	267 (1.21)	290 (1.31)	289 (1.31)	275 (1.25)	**301 (1.37)**
4868k	Second. rays reord.	96 (1.02)	94 (1.00)	105 (1.11)	111 (1.18)	119 (1.26)	118 (1.25)	134 (1.42)	130 (1.38)	123 (1.30)	**139 (1.47)**
	Shadow rays reord.	227 (1.03)	226 (1.02)	264 (1.20)	269 (1.22)	278 (1.26)	276 (1.25)	303 (1.37)	300 (1.36)	282 (1.28)	**310 (1.40)**
	BVH cost	428 (1.00)	314 (0.73)	201 (0.47)	173 (0.41)	159 (0.37)	159 (0.37)	109 (0.26)	**109 (0.26)**	190 (0.44)	120 (0.28)
Pompeii	Secondary rays	86 (1.00)	101 (1.17)	123 (1.43)	135 (1.57)	145 (1.68)	143 (1.66)	229 (2.66)	225 (2.61)	145 (1.68)	**234 (2.71)**
# triangles	Shadow rays	190 (1.00)	225 (1.19)	290 (1.53)	308 (1.62)	326 (1.72)	324 (1.71)	472 (2.49)	464 (2.44)	317 (1.67)	**479 (2.52)**
5632k	Second. rays reord.	89 (1.03)	103 (1.19)	126 (1.46)	139 (1.61)	149 (1.72)	145 (1.69)	227 (2.63)	222 (2.58)	147 (1.70)	**231 (2.68)**
	Shadow rays reord.	215 (1.13)	251 (1.32)	324 (1.70)	344 (1.81)	366 (1.93)	356 (1.88)	487 (2.56)	480 (2.53)	355 (1.87)	**500 (2.63)**
	BVH cost	251 (1.00)	183 (0.73)	142 (0.56)	136 (0.54)	128 (0.51)	125 (0.50)	115 (0.46)	**115 (0.46)**	157 (0.63)	133 (0.53)
San Miguel	Secondary rays	72 (1.00)	87 (1.21)	103 (1.44)	111 (1.55)	128 (1.79)	134 (1.87)	**150 (2.09)**	148 (2.06)	120 (1.67)	141 (1.97)
# triangles	Shadow rays	194 (1.00)	232 (1.20)	309 (1.59)	329 (1.70)	340 (1.75)	347 (1.79)	**378 (1.95)**	372 (1.92)	330 (1.70)	358 (1.85)
7880k	Second. rays reord.	84 (1.17)	100 (1.39)	121 (1.69)	129 (1.80)	148 (2.06)	153 (2.13)	**171 (2.39)**	168 (2.35)	141 (1.96)	167 (2.33)
	Shadow rays reord.	262 (1.35)	313 (1.61)	396 (2.04)	426 (2.19)	445 (2.30)	448 (2.31)	**477 (2.46)**	469 (2.42)	434 (2.24)	467 (2.41)
	BVH cost	299 (1.00)	215 (0.72)	135 (0.45)	110 (0.37)	101 (0.34)	98 (0.33)	80 (0.27)	**80 (0.27)**	115 (0.38)	87 (0.29)
Vienna	Secondary rays	101 (1.00)	103 (1.02)	175 (1.74)	192 (1.91)	204 (2.03)	212 (2.11)	**240 (2.38)**	237 (2.36)	208 (2.07)	236 (2.34)
# triangles	Shadow rays	179 (1.00)	194 (1.08)	334 (1.86)	357 (1.99)	380 (2.12)	382 (2.13)	434 (2.42)	430 (2.40)	384 (2.14)	**437 (2.43)**
8637k	Second. rays reord.	103 (1.02)	106 (1.05)	180 (1.79)	194 (1.93)	208 (2.07)	213 (2.12)	**236 (2.35)**	234 (2.33)	209 (2.08)	233 (2.32)
	Shadow rays reord.	195 (1.08)	208 (1.16)	367 (2.05)	377 (2.10)	402 (2.24)	402 (2.24)	438 (2.44)	435 (2.42)	400 (2.23)	**439 (2.45)**
	BVH cost	124 (1.00)	111 (0.90)	78 (0.63)	75 (0.61)	69 (0.56)	69 (0.55)	52 (0.42)	**52 (0.42)**	85 (0.69)	58 (0.46)
Powerplant	Secondary rays	53 (1.00)	60 (1.14)	68 (1.28)	81 (1.53)	67 (1.27)	65 (1.22)	162 (3.05)	165 (3.11)	67 (1.26)	**191 (3.60)**
# triangles	Shadow rays	190 (1.00)	207 (1.09)	249 (1.31)	250 (1.31)	295 (1.55)	264 (1.39)	429 (2.26)	421 (2.21)	250 (1.31)	**433 (2.28)**
12759k	Second. rays reord.	52 (0.98)	61 (1.16)	67 (1.26)	80 (1.52)	62 (1.17)	60 (1.13)	172 (3.25)	173 (3.27)	65 (1.23)	**201 (3.79)**
	Shadow rays reord.	224 (1.18)	239 (1.25)	269 (1.41)	258 (1.36)	344 (1.81)	265 (1.39)	444 (2.33)	437 (2.30)	273 (1.43)	**454 (2.39)**

Table 3. Performance comparison for all tested methods and scenes showing the BVH cost and trace speed in MRays/s for secondary and shadow rays with and without ray sorting using binary BVHs.

ing times). One reason might be that wide BVHs use a different traversal algorithm [Lier et al. 2018] than binary BVHs [Aila and Laine 2009]. For quaternary and octal BVHs, PRBVH$_S$ and PRBVH$_S^A$ outperform SBVH for both secondary and shadow rays. GPU ray tracing is memory bandwidth limited, and thus to achieve higher trace performance for higher branching factors, it is necessary to employ some kind of node compression [Ylitie et al. 2017], which was not used in our comparison.

Scene	Statistic	LBVH	HLBVH	PLOC	ATRBVH	PRBVH	PRBVH^{A}	PRBVH$_{S}$	PRBVH$_{S}^{A}$	SAH	SBVH
	BVH cost	108 (1.00)	92 (0.85)	90 (0.83)	84 (0.78)	80 (0.74)	79 (0.74)	76 (0.71)	**76 (0.70)**	84 (0.78)	77 (0.71)
Sibenik	Secondary rays	229 (1.00)	266 (1.16)	248 (1.08)	280 (1.22)	296 (1.29)	300 (1.31)	**308 (1.35)**	307 (1.34)	287 (1.25)	307 (1.34)
# triangles	Shadow rays	496 (1.00)	541 (1.09)	554 (1.12)	583 (1.18)	613 (1.24)	**630 (1.27)**	623 (1.26)	620 (1.25)	580 (1.17)	597 (1.21)
75k	Second. rays reord.	267 (1.16)	304 (1.33)	282 (1.23)	317 (1.38)	333 (1.45)	337 (1.47)	**349 (1.52)**	348 (1.52)	327 (1.42)	349 (1.52)
	Shadow rays reord.	626 (1.26)	672 (1.36)	688 (1.39)	710 (1.43)	742 (1.50)	**760 (1.53)**	743 (1.50)	744 (1.50)	712 (1.44)	721 (1.46)
	BVH cost	163 (1.00)	132 (0.81)	118 (0.72)	115 (0.71)	115 (0.71)	113 (0.70)	**107 (0.66)**	107 (0.66)	131 (0.81)	113 (0.70)
Crytek Sponza	Secondary rays	158 (1.00)	180 (1.14)	185 (1.17)	197 (1.25)	226 (1.43)	226 (1.43)	**241 (1.53)**	238 (1.51)	186 (1.18)	230 (1.46)
# triangles	Shadow rays	320 (1.00)	376 (1.17)	415 (1.30)	445 (1.39)	**482 (1.50)**	480 (1.50)	452 (1.41)	455 (1.42)	376 (1.18)	420 (1.31)
262k	Second. rays reord.	196 (1.24)	226 (1.43)	232 (1.47)	242 (1.53)	274 (1.74)	276 (1.75)	**297 (1.88)**	293 (1.85)	228 (1.45)	284 (1.80)
	Shadow rays reord.	419 (1.31)	485 (1.52)	519 (1.62)	549 (1.71)	591 (1.85)	**594 (1.86)**	562 (1.76)	562 (1.76)	482 (1.51)	530 (1.65)
	BVH cost	90 (1.00)	74 (0.83)	61 (0.68)	61 (0.68)	61 (0.68)	**60 (0.66)**	61 (0.68)	61 (0.68)	61 (0.68)	63 (0.70)
Conference	Secondary rays	256 (1.00)	299 (1.17)	359 (1.40)	357 (1.39)	368 (1.44)	375 (1.46)	**384 (1.50)**	380 (1.48)	356 (1.39)	342 (1.34)
# triangles	Shadow rays	484 (1.00)	599 (1.24)	698 (1.44)	710 (1.47)	**758 (1.57)**	736 (1.52)	738 (1.53)	736 (1.52)	658 (1.36)	602 (1.24)
331k	Second. rays reord.	317 (1.24)	364 (1.42)	423 (1.65)	422 (1.65)	434 (1.69)	439 (1.71)	**455 (1.77)**	453 (1.77)	425 (1.66)	425 (1.66)
	Shadow rays reord.	649 (1.34)	778 (1.61)	893 (1.85)	906 (1.87)	**938 (1.94)**	907 (1.87)	909 (1.88)	905 (1.87)	851 (1.76)	790 (1.63)
	BVH cost	119 (1.00)	117 (0.98)	114 (0.96)	104 (0.88)	101 (0.85)	99 (0.83)	92 (0.78)	**92 (0.78)**	99 (0.84)	93 (0.78)
Gallery	Secondary rays	134 (1.00)	135 (1.01)	139 (1.04)	147 (1.10)	153 (1.14)	154 (1.15)	161 (1.20)	160 (1.20)	153 (1.14)	**161 (1.20)**
# triangles	Shadow rays	290 (1.00)	284 (0.98)	305 (1.05)	314 (1.08)	330 (1.14)	331 (1.14)	**338 (1.17)**	335 (1.15)	320 (1.10)	331 (1.14)
998k	Second. rays reord.	146 (1.09)	147 (1.10)	153 (1.14)	161 (1.20)	167 (1.25)	168 (1.26)	173 (1.30)	172 (1.29)	168 (1.26)	**174 (1.30)**
	Shadow rays reord.	349 (1.20)	340 (1.17)	367 (1.26)	374 (1.29)	391 (1.35)	393 (1.35)	**402 (1.38)**	399 (1.37)	382 (1.32)	392 (1.35)
	BVH cost	118 (1.00)	110 (0.94)	107 (0.91)	103 (0.88)	98 (0.83)	96 (0.82)	92 (0.79)	**92 (0.78)**	96 (0.82)	93 (0.79)
Happy Buddha	Secondary rays	118 (1.00)	120 (1.01)	119 (1.01)	126 (1.06)	131 (1.10)	129 (1.09)	**135 (1.14)**	134 (1.13)	132 (1.11)	134 (1.13)
# triangles	Shadow rays	169 (1.00)	170 (1.01)	177 (1.05)	181 (1.07)	186 (1.10)	186 (1.10)	**190 (1.12)**	188 (1.12)	187 (1.11)	189 (1.12)
1087k	Second. rays reord.	114 (0.96)	115 (0.97)	115 (0.98)	120 (1.01)	124 (1.05)	123 (1.04)	127 (1.07)	126 (1.07)	125 (1.05)	**128 (1.08)**
	Shadow rays reord.	154 (0.91)	155 (0.92)	159 (0.94)	163 (0.96)	167 (0.99)	166 (0.99)	169 (1.00)	169 (1.00)	168 (1.00)	**170 (1.01)**
	BVH cost	149 (1.00)	134 (0.90)	116 (0.77)	110 (0.74)	100 (0.67)	100 (0.67)	**98 (0.66)**	98 (0.66)	116 (0.78)	104 (0.69)
Sodahall	Secondary rays	212 (1.00)	250 (1.18)	245 (1.16)	274 (1.29)	314 (1.48)	307 (1.45)	**334 (1.58)**	327 (1.54)	283 (1.34)	319 (1.50)
# triangles	Shadow rays	484 (1.00)	524 (1.08)	557 (1.15)	593 (1.23)	**604 (1.25)**	603 (1.25)	599 (1.24)	580 (1.20)	575 (1.19)	579 (1.20)
2169k	Second. rays reord.	234 (1.10)	268 (1.26)	264 (1.24)	292 (1.38)	326 (1.54)	318 (1.50)	**342 (1.61)**	335 (1.58)	299 (1.41)	328 (1.55)
	Shadow rays reord.	528 (1.09)	556 (1.15)	589 (1.22)	620 (1.28)	**629 (1.30)**	624 (1.29)	616 (1.27)	601 (1.24)	605 (1.25)	602 (1.24)
	BVH cost	922 (1.00)	919 (1.00)	797 (0.86)	788 (0.85)	742 (0.80)	742 (0.80)	608 (0.66)	**606 (0.66)**	802 (0.87)	658 (0.71)
Hairball	Secondary rays	71 (1.00)	72 (1.02)	71 (1.00)	83 (1.18)	90 (1.28)	90 (1.27)	**93 (1.32)**	92 (1.31)	87 (1.23)	90 (1.28)
# triangles	Shadow rays	151 (1.00)	154 (1.02)	163 (1.07)	177 (1.17)	**190 (1.25)**	188 (1.24)	189 (1.25)	186 (1.23)	183 (1.21)	184 (1.22)
2880k	Second. rays reord.	80 (1.13)	82 (1.16)	80 (1.14)	94 (1.33)	101 (1.43)	100 (1.42)	**105 (1.49)**	103 (1.47)	97 (1.37)	102 (1.45)
	Shadow rays reord.	173 (1.14)	175 (1.16)	191 (1.26)	203 (1.34)	**216 (1.43)**	214 (1.42)	210 (1.39)	208 (1.38)	210 (1.39)	207 (1.37)
	BVH cost	46 (1.00)	44 (0.96)	42 (0.91)	40 (0.87)	39 (0.85)	39 (0.84)	38 (0.81)	**38 (0.81)**	40 (0.86)	38 (0.83)
Crown	Secondary rays	104 (1.00)	104 (1.01)	114 (1.11)	118 (1.14)	126 (1.21)	125 (1.20)	135 (1.30)	131 (1.27)	125 (1.21)	**135 (1.31)**
# triangles	Shadow rays	211 (1.00)	215 (1.02)	240 (1.14)	242 (1.15)	251 (1.19)	252 (1.19)	260 (1.23)	256 (1.22)	249 (1.18)	**261 (1.24)**
4868k	Second. rays reord.	114 (1.10)	115 (1.12)	126 (1.22)	130 (1.26)	138 (1.33)	136 (1.31)	147 (1.42)	145 (1.40)	136 (1.32)	**148 (1.43)**
	Shadow rays reord.	220 (1.04)	223 (1.06)	249 (1.18)	250 (1.19)	259 (1.23)	258 (1.22)	267 (1.27)	265 (1.26)	255 (1.21)	**268 (1.27)**
	BVH cost	255 (1.00)	202 (0.79)	151 (0.59)	137 (0.54)	130 (0.51)	129 (0.51)	**81 (0.32)**	81 (0.32)	136 (0.53)	82 (0.32)
Pompeii	Secondary rays	95 (1.00)	111 (1.17)	136 (1.44)	153 (1.61)	162 (1.70)	160 (1.69)	228 (2.40)	224 (2.36)	157 (1.65)	**230 (2.42)**
# triangles	Shadow rays	173 (1.00)	213 (1.23)	276 (1.60)	292 (1.69)	314 (1.82)	310 (1.79)	406 (2.35)	400 (2.31)	305 (1.76)	**414 (2.39)**
5632k	Second. rays reord.	99 (1.05)	116 (1.22)	143 (1.50)	156 (1.64)	165 (1.74)	164 (1.73)	228 (2.40)	223 (2.35)	161 (1.70)	**230 (2.42)**
	Shadow rays reord.	192 (1.11)	230 (1.33)	292 (1.69)	307 (1.78)	326 (1.88)	322 (1.86)	409 (2.37)	401 (2.32)	320 (1.85)	**415 (2.40)**
	BVH cost	146 (1.00)	117 (0.80)	98 (0.67)	96 (0.66)	92 (0.63)	90 (0.62)	**82 (0.56)**	82 (0.56)	102 (0.69)	86 (0.59)
San Miguel	Secondary rays	69 (1.00)	86 (1.25)	103 (1.49)	107 (1.55)	116 (1.68)	121 (1.75)	**131 (1.90)**	131 (1.90)	106 (1.53)	128 (1.85)
# triangles	Shadow rays	174 (1.00)	217 (1.25)	255 (1.46)	264 (1.51)	276 (1.59)	276 (1.59)	**296 (1.70)**	293 (1.68)	267 (1.53)	288 (1.65)
7880k	Second. rays reord.	82 (1.19)	102 (1.47)	119 (1.73)	124 (1.80)	135 (1.95)	140 (2.02)	**151 (2.18)**	150 (2.17)	125 (1.80)	147 (2.13)
	Shadow rays reord.	224 (1.29)	269 (1.54)	310 (1.78)	322 (1.85)	333 (1.91)	332 (1.91)	**350 (2.01)**	347 (1.99)	324 (1.86)	345 (1.98)
	BVH cost	163 (1.00)	124 (0.76)	89 (0.54)	77 (0.47)	71 (0.44)	70 (0.43)	52 (0.32)	**52 (0.32)**	77 (0.47)	54 (0.33)
Vienna	Secondary rays	106 (1.00)	118 (1.11)	170 (1.60)	184 (1.74)	201 (1.89)	204 (1.92)	**227 (2.14)**	225 (2.11)	194 (1.83)	219 (2.06)
# triangles	Shadow rays	166 (1.00)	195 (1.18)	291 (1.75)	318 (1.92)	339 (2.05)	346 (2.08)	**375 (2.26)**	370 (2.23)	331 (2.00)	367 (2.21)
8637k	Second. rays reord.	114 (1.07)	127 (1.19)	178 (1.67)	193 (1.81)	208 (1.96)	210 (1.97)	**232 (2.19)**	231 (2.17)	199 (1.88)	224 (2.11)
	Shadow rays reord.	182 (1.10)	210 (1.26)	303 (1.83)	327 (1.97)	347 (2.09)	351 (2.11)	**376 (2.27)**	373 (2.25)	338 (2.03)	368 (2.22)
	BVH cost	75 (1.00)	70 (0.93)	55 (0.73)	54 (0.72)	51 (0.68)	50 (0.67)	36 (0.49)	**36 (0.48)**	57 (0.76)	37 (0.49)
Powerplant	Secondary rays	81 (1.00)	92 (1.14)	116 (1.44)	115 (1.43)	119 (1.48)	120 (1.49)	196 (2.43)	195 (2.42)	108 (1.34)	**203 (2.52)**
# triangles	Shadow rays	231 (1.00)	248 (1.07)	310 (1.34)	312 (1.35)	340 (1.47)	340 (1.47)	**392 (1.69)**	382 (1.65)	295 (1.28)	388 (1.68)
12759k	Second. rays reord.	94 (1.17)	106 (1.31)	128 (1.59)	128 (1.59)	128 (1.59)	129 (1.60)	211 (2.62)	207 (2.57)	120 (1.49)	**215 (2.67)**
	Shadow rays reord.	259 (1.12)	275 (1.19)	335 (1.45)	333 (1.44)	361 (1.56)	361 (1.56)	**397 (1.71)**	387 (1.67)	326 (1.41)	395 (1.71)

Table 4. Performance comparison for all tested methods and scenes showing the BVH cost and trace speed in MRays/s for secondary and shadow rays with and without ray sorting using quaternary BVHs.

Table 1 shows that using spatial splitting pays off overall, improving trace performance by about 24% for secondary rays and 12% for shadow rays on average. However, the speedup provided by spatial splits depends heavily on a particular scene. There is a rather marginal improvement for finely tessellated scenes such as Conference, Happy Buddha, and Hairball. On the other hand, for a complex scene such as Powerplant, the speedup is almost threefold.

Scene	Statistic	LBVH	HLBVH	PLOC	ATRBVH	PRBVH	PRBVH^{A}	PRBVH$_{S}$	PRBVH$_{S}^{A}$	SAH	SBVH
	BVH cost	113 (1.00)	102 (0.90)	97 (0.86)	94 (0.83)	91 (0.81)	90 (0.80)	85 (0.76)	85 (0.76)	92 (0.82)	**84 (0.74)**
Sibenik	Secondary rays	197 (1.00)	222 (1.13)	215 (1.09)	234 (1.19)	244 (1.24)	247 (1.26)	**253 (1.29)**	253 (1.28)	233 (1.19)	250 (1.27)
# triangles	Shadow rays	439 (1.00)	449 (1.02)	475 (1.08)	472 (1.07)	500 (1.14)	508 (1.16)	515 (1.17)	**516 (1.18)**	486 (1.11)	482 (1.10)
75k	Second. rays reord.	225 (1.14)	250 (1.27)	240 (1.22)	261 (1.33)	273 (1.39)	275 (1.40)	**282 (1.44)**	282 (1.43)	262 (1.33)	280 (1.42)
	Shadow rays reord.	497 (1.13)	505 (1.15)	531 (1.21)	524 (1.19)	555 (1.26)	562 (1.28)	565 (1.29)	**568 (1.29)**	540 (1.23)	533 (1.21)
	BVH cost	176 (1.00)	142 (0.81)	137 (0.78)	136 (0.77)	135 (0.77)	133 (0.76)	129 (0.73)	**128 (0.73)**	136 (0.77)	130 (0.74)
Crytek Sponza	Secondary rays	130 (1.00)	151 (1.16)	159 (1.22)	172 (1.32)	194 (1.49)	198 (1.52)	**200 (1.54)**	199 (1.53)	156 (1.20)	190 (1.46)
# triangles	Shadow rays	279 (1.00)	326 (1.17)	345 (1.24)	376 (1.35)	425 (1.52)	**431 (1.54)**	374 (1.34)	375 (1.35)	313 (1.12)	348 (1.25)
262k	Second. rays reord.	158 (1.21)	183 (1.41)	189 (1.45)	203 (1.56)	228 (1.75)	232 (1.78)	**237 (1.82)**	235 (1.81)	188 (1.44)	227 (1.74)
	Shadow rays reord.	326 (1.17)	373 (1.34)	393 (1.41)	428 (1.54)	477 (1.71)	**484 (1.73)**	422 (1.51)	420 (1.51)	360 (1.29)	396 (1.42)
	BVH cost	91 (1.00)	83 (0.91)	75 (0.83)	75 (0.82)	75 (0.83)	75 (0.83)	78 (0.86)	78 (0.86)	**70 (0.77)**	75 (0.82)
Conference	Secondary rays	242 (1.00)	274 (1.13)	322 (1.33)	328 (1.36)	334 (1.38)	**335 (1.38)**	322 (1.33)	320 (1.32)	318 (1.31)	292 (1.21)
# triangles	Shadow rays	481 (1.00)	557 (1.16)	638 (1.33)	643 (1.34)	655 (1.36)	**659 (1.37)**	622 (1.29)	621 (1.29)	583 (1.21)	539 (1.12)
331k	Second. rays reord.	287 (1.19)	322 (1.33)	367 (1.52)	371 (1.54)	**381 (1.58)**	379 (1.57)	369 (1.52)	366 (1.51)	367 (1.52)	350 (1.45)
	Shadow rays reord.	544 (1.13)	634 (1.32)	720 (1.50)	717 (1.49)	738 (1.53)	**738 (1.53)**	685 (1.42)	682 (1.42)	658 (1.37)	615 (1.28)
	BVH cost	119 (1.00)	117 (0.99)	111 (0.94)	106 (0.89)	101 (0.85)	99 (0.83)	93 (0.78)	**92 (0.78)**	100 (0.84)	93 (0.78)
Gallery	Secondary rays	114 (1.00)	116 (1.02)	117 (1.03)	126 (1.11)	130 (1.14)	131 (1.15)	**140 (1.23)**	139 (1.22)	131 (1.15)	139 (1.22)
# triangles	Shadow rays	252 (1.00)	249 (0.99)	267 (1.06)	282 (1.12)	287 (1.14)	295 (1.17)	305 (1.21)	**305 (1.21)**	286 (1.14)	302 (1.20)
998k	Second. rays reord.	126 (1.10)	128 (1.12)	131 (1.15)	140 (1.23)	143 (1.26)	147 (1.29)	**154 (1.35)**	153 (1.34)	147 (1.29)	153 (1.34)
	Shadow rays reord.	283 (1.12)	278 (1.10)	301 (1.19)	315 (1.25)	321 (1.28)	330 (1.31)	341 (1.35)	**341 (1.35)**	320 (1.27)	337 (1.34)
	BVH cost	115 (1.00)	111 (0.97)	105 (0.91)	102 (0.89)	97 (0.85)	95 (0.83)	92 (0.80)	**92 (0.80)**	96 (0.84)	93 (0.81)
Happy Buddha	Secondary rays	115 (1.00)	116 (1.01)	116 (1.02)	122 (1.06)	128 (1.12)	127 (1.11)	**131 (1.14)**	131 (1.14)	129 (1.12)	131 (1.14)
# triangles	Shadow rays	158 (1.00)	159 (1.00)	163 (1.03)	170 (1.07)	174 (1.10)	174 (1.10)	**176 (1.12)**	175 (1.11)	174 (1.11)	175 (1.11)
1087k	Second. rays reord.	109 (0.95)	110 (0.96)	110 (0.96)	115 (1.01)	119 (1.03)	119 (1.04)	**122 (1.06)**	121 (1.06)	119 (1.04)	122 (1.06)
	Shadow rays reord.	141 (0.90)	143 (0.90)	146 (0.92)	151 (0.96)	154 (0.98)	154 (0.98)	**156 (0.99)**	155 (0.98)	154 (0.97)	156 (0.99)
	BVH cost	153 (1.00)	142 (0.93)	126 (0.82)	122 (0.79)	115 (0.75)	115 (0.75)	112 (0.73)	112 (0.73)	119 (0.78)	**111 (0.73)**
Sodahall	Secondary rays	185 (1.00)	224 (1.21)	214 (1.16)	239 (1.29)	274 (1.48)	272 (1.47)	**278 (1.50)**	276 (1.49)	244 (1.32)	269 (1.45)
# triangles	Shadow rays	394 (1.00)	436 (1.11)	462 (1.17)	468 (1.19)	496 (1.26)	**498 (1.27)**	480 (1.22)	478 (1.21)	467 (1.19)	469 (1.19)
2169k	Second. rays reord.	201 (1.09)	238 (1.29)	229 (1.24)	250 (1.35)	282 (1.53)	280 (1.52)	**286 (1.55)**	283 (1.53)	256 (1.39)	277 (1.50)
	Shadow rays reord.	408 (1.04)	443 (1.12)	468 (1.19)	470 (1.19)	493 (1.25)	**495 (1.26)**	480 (1.22)	476 (1.21)	470 (1.19)	470 (1.19)
	BVH cost	979 (1.00)	979 (1.00)	828 (0.85)	828 (0.85)	781 (0.80)	780 (0.80)	**621 (0.63)**	624 (0.64)	825 (0.84)	632 (0.65)
Hairball	Secondary rays	70 (1.00)	72 (1.02)	66 (0.94)	79 (1.12)	84 (1.19)	83 (1.18)	**91 (1.29)**	89 (1.27)	80 (1.14)	86 (1.22)
# triangles	Shadow rays	157 (1.00)	160 (1.02)	161 (1.03)	175 (1.11)	181 (1.15)	180 (1.14)	**188 (1.20)**	186 (1.18)	176 (1.12)	180 (1.15)
2880k	Second. rays reord.	79 (1.12)	80 (1.13)	75 (1.06)	88 (1.25)	93 (1.32)	92 (1.31)	**101 (1.43)**	99 (1.40)	89 (1.27)	96 (1.36)
	Shadow rays reord.	170 (1.08)	173 (1.10)	177 (1.12)	189 (1.20)	195 (1.24)	193 (1.23)	**200 (1.27)**	198 (1.26)	189 (1.20)	192 (1.22)
	BVH cost	47 (1.00)	46 (0.98)	43 (0.91)	42 (0.90)	41 (0.87)	41 (0.87)	39 (0.84)	**39 (0.84)**	41 (0.88)	40 (0.84)
Crown	Secondary rays	100 (1.00)	100 (1.00)	108 (1.08)	112 (1.12)	118 (1.18)	117 (1.17)	**125 (1.25)**	124 (1.24)	116 (1.16)	124 (1.24)
# triangles	Shadow rays	201 (1.00)	202 (1.01)	220 (1.10)	224 (1.12)	230 (1.14)	232 (1.15)	235 (1.17)	**236 (1.17)**	227 (1.13)	233 (1.16)
4868k	Second. rays reord.	108 (1.08)	108 (1.08)	116 (1.16)	121 (1.21)	126 (1.26)	125 (1.25)	**133 (1.33)**	132 (1.32)	125 (1.25)	133 (1.32)
	Shadow rays reord.	198 (0.99)	199 (0.99)	216 (1.07)	220 (1.09)	225 (1.12)	226 (1.12)	231 (1.15)	**231 (1.15)**	223 (1.11)	229 (1.14)
	BVH cost	250 (1.00)	222 (0.89)	169 (0.68)	157 (0.63)	151 (0.60)	151 (0.60)	92 (0.37)	92 (0.37)	147 (0.59)	**87 (0.35)**
Pompeii	Secondary rays	87 (1.00)	103 (1.18)	129 (1.48)	141 (1.62)	152 (1.75)	151 (1.73)	204 (2.34)	201 (2.31)	146 (1.67)	**205 (2.35)**
# triangles	Shadow rays	160 (1.00)	192 (1.20)	242 (1.51)	261 (1.63)	272 (1.70)	271 (1.69)	344 (2.15)	339 (2.12)	265 (1.65)	**345 (2.16)**
5632k	Second. rays reord.	90 (1.03)	105 (1.21)	132 (1.51)	143 (1.64)	152 (1.75)	151 (1.74)	202 (2.32)	199 (2.28)	147 (1.69)	**204 (2.34)**
	Shadow rays reord.	164 (1.03)	194 (1.21)	242 (1.51)	259 (1.62)	268 (1.67)	267 (1.67)	333 (2.08)	327 (2.04)	262 (1.64)	**335 (2.09)**
	BVH cost	149 (1.00)	128 (0.86)	110 (0.74)	109 (0.73)	105 (0.70)	104 (0.70)	93 (0.63)	93 (0.63)	110 (0.74)	**93 (0.62)**
San Miguel	Secondary rays	67 (1.00)	79 (1.18)	93 (1.39)	97 (1.45)	106 (1.58)	111 (1.65)	**120 (1.79)**	117 (1.75)	95 (1.42)	110 (1.64)
# triangles	Shadow rays	170 (1.00)	202 (1.19)	238 (1.40)	242 (1.42)	253 (1.49)	255 (1.50)	**271 (1.59)**	266 (1.57)	244 (1.44)	252 (1.48)
7880k	Second. rays reord.	77 (1.16)	91 (1.36)	106 (1.58)	111 (1.66)	121 (1.80)	126 (1.88)	**134 (2.01)**	131 (1.96)	110 (1.64)	127 (1.89)
	Shadow rays reord.	197 (1.16)	231 (1.36)	270 (1.59)	276 (1.63)	285 (1.68)	287 (1.69)	**301 (1.77)**	297 (1.75)	280 (1.65)	287 (1.69)
	BVH cost	145 (1.00)	123 (0.85)	87 (0.60)	76 (0.53)	71 (0.49)	70 (0.49)	54 (0.37)	**54 (0.37)**	74 (0.51)	54 (0.37)
Vienna	Secondary rays	102 (1.00)	113 (1.11)	156 (1.53)	174 (1.70)	189 (1.85)	193 (1.89)	**209 (2.04)**	207 (2.02)	180 (1.76)	202 (1.97)
# triangles	Shadow rays	164 (1.00)	192 (1.18)	260 (1.59)	287 (1.75)	307 (1.88)	310 (1.89)	**328 (2.00)**	325 (1.99)	295 (1.80)	323 (1.97)
8637k	Second. rays reord.	108 (1.05)	119 (1.16)	160 (1.57)	178 (1.74)	192 (1.87)	194 (1.89)	**210 (2.05)**	207 (2.02)	182 (1.78)	203 (1.98)
	Shadow rays reord.	167 (1.02)	193 (1.18)	256 (1.57)	281 (1.72)	299 (1.83)	300 (1.83)	**318 (1.94)**	314 (1.92)	287 (1.75)	313 (1.91)
	BVH cost	70 (1.00)	67 (0.96)	55 (0.79)	55 (0.79)	53 (0.76)	53 (0.75)	38 (0.55)	38 (0.55)	55 (0.78)	**37 (0.52)**
Powerplant	Secondary rays	84 (1.00)	96 (1.14)	120 (1.42)	129 (1.54)	133 (1.58)	141 (1.67)	183 (2.18)	180 (2.13)	119 (1.41)	**184 (2.19)**
# triangles	Shadow rays	223 (1.00)	241 (1.08)	286 (1.29)	293 (1.32)	307 (1.38)	316 (1.42)	**341 (1.53)**	336 (1.51)	277 (1.25)	339 (1.52)
12759k	Second. rays reord.	97 (1.15)	108 (1.28)	131 (1.55)	141 (1.68)	149 (1.77)	156 (1.86)	191 (2.27)	187 (2.22)	132 (1.57)	**192 (2.28)**
	Shadow rays reord.	233 (1.05)	249 (1.12)	293 (1.32)	297 (1.33)	313 (1.41)	317 (1.42)	**337 (1.52)**	332 (1.49)	284 (1.27)	335 (1.50)

Table 5. Performance comparison for all tested methods and scenes showing the BVH cost and trace speed in MRays/s for secondary and shadow rays with and without ray sorting using octal BVHs.

4.3. Cost and Trace Speed Correlation

It is a well-known fact that there is a discrepancy between the BVH cost and the actual trace performance [Aila et al. 2013]. In other words, cost reduction does not necessarily mean trace performance improvement. Top-down builders, such as SAH and SBVH, provide BVHs with higher costs than other approaches, but the top levels

Br. Factor	Statistic	LBVH	HLBVH	PLOC	ATRBVH	PRBVH	PRBVH^{A}	PRBVH$_{S}$	PRBVH$_{S}^{A}$	SAH	SBVH
2	BVH cost	1.00	0.83	0.70	0.65	0.61	0.60	0.55	0.55	0.68	0.59
	Secondary rays	1.00	1.11	1.21	1.18	1.21	1.21	1.11	1.12	1.07	1.02
	Shadow rays	1.00	1.13	1.14	1.18	1.19	1.21	1.22	1.24	1.10	1.14
	Secondary rays reord.	0.92	1.03	1.12	1.10	1.14	1.16	1.05	1.07	1.01	0.96
	Shadow rays reord.	0.84	0.95	0.98	1.03	1.03	1.07	1.10	1.11	0.95	1.01
4	BVH cost	1.00	0.88	0.77	0.73	0.70	0.69	0.62	0.62	0.74	0.64
	Secondary rays	1.00	1.03	1.08	1.06	1.04	1.04	1.07	1.08	1.03	1.06
	Shadow rays	1.00	1.03	1.04	1.05	1.04	1.05	1.16	1.17	1.05	1.16
	Second. rays reord.	0.89	0.93	0.98	0.97	0.96	0.97	1.00	1.01	0.94	0.98
	Shadow rays reord.	0.87	0.92	0.94	0.95	0.95	0.97	1.08	1.09	0.94	1.07
8	BVH cost	1.00	0.92	0.81	0.79	0.76	0.75	0.67	0.67	0.76	0.66
	Secondary rays	1.00	0.99	1.04	0.99	0.96	0.96	1.02	1.03	1.01	1.06
	Shadow rays	1.00	1.00	1.02	1.01	1.00	1.00	1.12	1.13	1.06	1.18
	Second. rays reord.	0.91	0.90	0.96	0.92	0.90	0.90	0.96	0.97	0.94	0.99
	Shadow rays reord.	0.94	0.95	0.98	0.97	0.97	0.97	1.10	1.11	1.02	1.14

Table 6. Comparison of the BVH costs and the average trace times per a unit BVH cost (i.e., trace time divided by the BVH cost). The values are normalized prior to averaging by a value given by LBVH of the corresponding BVH branching factor.

are locally well optimized, which is important as most of the time during the traversal is spent in the top levels because they are visited by almost all rays. To quantize this fact, we compute the ratio between trace time and cost, which expresses the time needed for a unit cost (see Table 6). These values are averaged over all scenes. The values are normalized prior to averaging by a value given by LBVH of the corresponding branching factor. Top-down builders such as SAH and SBVH have lower values, whereas optimization algorithms such as PRBVH and bottom-up algorithms such as PLOC have higher values. These values can be considered as a correction to the actual cost values. By multiplying the cost by these values, we can obtain a corrected cost that actually corresponds to the trace times. We also plot normalized trace times and normalized BVH costs in Figure 1 to see the correlation between these two quantities.

4.4. Influence of Ray Reordering

Table 1 indicates that using ray reordering pays off on average, providing a 7% speedup for secondary rays and 8% for shadow rays. Particularly, ray reordering improves trace performance for Crytek Sponza (23% for secondary rays and 39% for shadow rays), Conference (15% for secondary rays and 37% for shadow rays), and San Miguel (14% for secondary rays and 26% for shadow rays) using binary BVHs. Ray reordering typically benefits from using larger ray batches as it is more likely to extract coherence from a large number of rays. Furthermore, the speed of the ray sorting algorithm, i.e., the number of rays sorted per second, increases with more rays. Thus, it pays off to use ray reordering for complex scenes (e.g., Hairball) or architectural scenes (e.g., San Miguel). Conversely, in object-like scenes (e.g., Happy Buddha), most of the rays escape the scene after the first bounce, and further bounces are generated only in concave regions, generating only a few secondary rays. Thus,

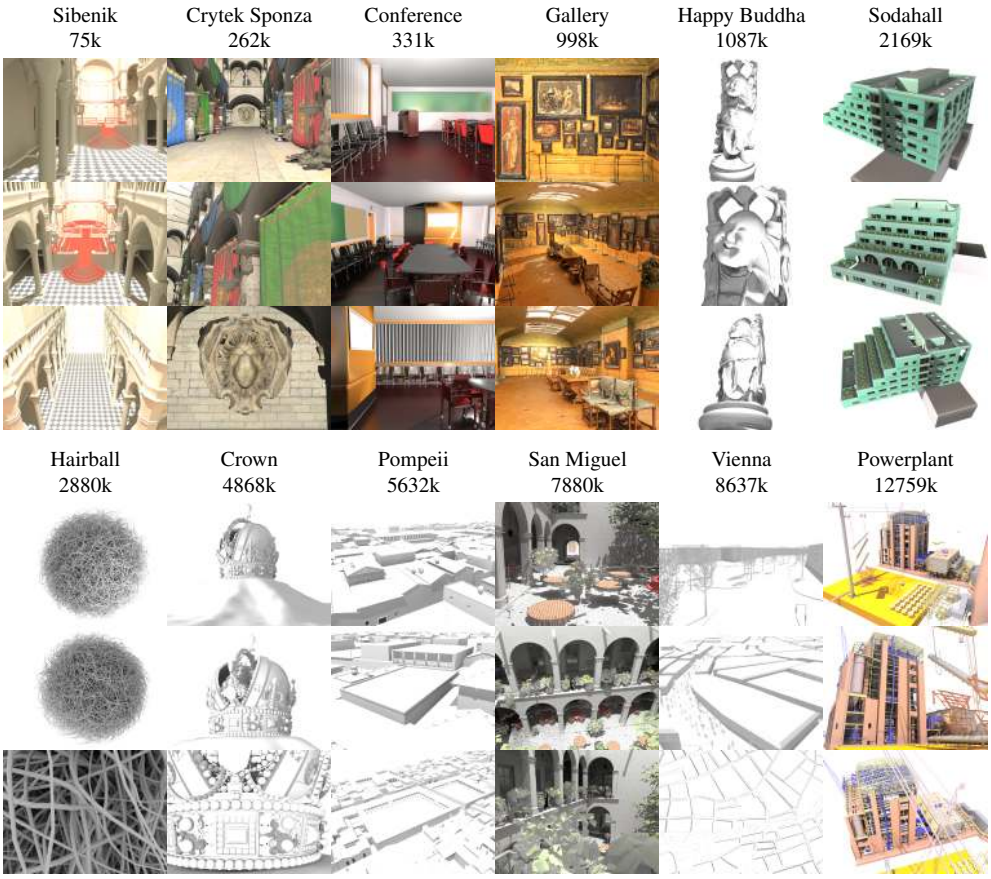

Figure 2. Rendered images of 12 tested scenes in resolution 1024×768. We use three representative views for each scene.

it is difficult to extract some coherence from a small number of rays and fully utilize parallel resources.

However, we can observe that the effect of ray reordering decreases with increasing branching factor, especially for shadow rays. For example, ray reordering improves trace performance for shadow rays in San Miguel by 26% for binary BVHs, 18% for quaternary BVHs, and 11% for octal BVHs using PRBVH_S. One reason might be again that for wide BVHs we use a completely different trace kernel [Lier et al. 2018] than for binary BVHs [Aila and Laine 2009]. Another reason might be that in wide BVHs we visit fewer nodes during the traversal, which results in fewer memory accesses, decreasing the importance of ray coherence.

5. Conclusion and Future Work

We provided an overview of the state-of-the-art BVH construction methods together with their extensive empirical performance comparison. In particular, we compared

the various state-of-the-art construction algorithms, spatial splitting, ray reordering, and wide BVHs. We also measured the discrepancy between trace times and BVH costs. We found that the best results are provided by binary BVHs constructed by SBVH, while ray reordering pays off in most cases. We also showed that parallel reinsertion in combination with simulated annealing can provide slightly better costs than parallel reinsertion itself.

The results indicate several promising directions for future work. One obvious direction would be a fast construction using spatial splits while keeping the quality of SBVH. Another interesting direction might be a hybrid construction using top-down construction for top levels and bottom-up construction for bottom levels. On a similar note, we could modify PRBVH to penalize or completely disable updates in top levels to preserve top splits provided by SBVH. Furthermore, during the conversion to wide BVH, some nodes are discarded. Thus, it makes sense to focus the optimization only on the nodes that remain and not on the discarded nodes. However, as was pointed by Meister et al. [2021], this is a difficult challenge as the cost function in wide BVHs is piecewise constant (i.e., some of the topological updates do not change the cost value), which makes the navigation in the search space more difficult.

Simulated annealing could also be used with other BVH construction methods such as treelet restructuring. Moreover, in simulated annealing, we clamp the probability used for pruning to a constant and prioritize nodes with negative values during the heating phase. In both cases, we could use more sophisticated approaches. We leave these problems for future work.

Acknowledgements

We would like to thank Jakub Černý for fruitful discussions and valuable comments. We thank Seth Teller for providing the Sodahall scene, Peter Wonka and Michael Wimmer for providing the Vienna and Pompeii scenes, and Morgan McGuire [2017] for providing the other scenes. This work was partly supported by the Czech Science Foundation (GA18-20374S) and Research Center for Informatics (CZ.02.1.01/0.0/0.0/16_019/0000765).

Copyright Notice and Trademarks

Index of Supplemental Materials

The source codes of the algorithm can be downloaded from GitHub:
`https://github.com/meistdan/hippie`.
Publication snapshot at
`https://jcgt.org/published/0011/03/06/hippie-main.zip`

References

AILA, T., AND LAINE, S. 2009. Understanding the efficiency of ray traversal on GPUs. In *Proceedings of the Conference on High-Performance Graphics*, Association for Computing Machinery, 145–149. URL: https://doi.org/10.1145/1572769.1572792. 7, 8, 11, 15

AILA, T., KARRAS, T., AND LAINE, S. 2013. On quality metrics of bounding volume hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference*, Association for Computing Machinery, 101–108. URL: https://doi.org/10.1145/2492045.2492056. 3, 13

DOMINGUES, L., AND PEDRINI, H. 2015. Bounding volume hierarchy optimization through agglomerative treelet restructuring. In *Proceedings of the 7th Conference on High-Performance Graphic*, Association for Computing Machinery, 13–20. URL: https://doi.org/10.1145/2790060.2790065. 5, 8

GARANZHA, K., PANTALEONI, J., AND MCALLISTER, D. 2011. Simpler and faster HLBVH with work queues. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, Association for Computing Machinery, 59–64. URL: https://doi.org/10.1145/2018323.2018333. 5, 8

GOLDSMITH, J., AND SALMON, J. 1987. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications 7*, 5, 14–20. URL: https://ieeexplore.ieee.org/document/4057175. 3

KARRAS, T., AND AILA, T. 2013. Fast parallel construction of high-quality bounding volume hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference*, Association for Computing Machinery, 89–99. URL: https://doi.org/10.1145/2492045.2492055. 5

KARRAS, T. 2012. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics*, Eurographics Association, 33–37. URL: https://research.nvidia.com/publication/maximizing-parallelism-construction-bvhs-octrees-and-k-d-trees. 4, 8

KENSLER, A. 2008. Tree rotations for improving bounding volume hierarchies. In *IEEE Symposium on Interactive Ray Tracing*, IEEE, 73–76. URL: https://ieeexplore.ieee.org/document/4634624. 6

LIER, A., STAMMINGER, M., AND SELGRAD, K. 2018. CPU-style SIMD ray traversal on GPUs. In *Proceedings of the Conference on High-Performance Graphics*, Association for Computing Machinery, 7:1–7:4. URL: https://doi.org/10.1145/3231578.3231583. 7, 8, 11, 15

MCGUIRE, M., 2017. Computer Graphics Archive, July. URL: https://casual-effects.com/data. 16

MEISTER, D., AND BITTNER, J. 2018. Parallel locally-ordered clustering for bounding volume hierarchy construction. *IEEE Transactions on Visualization and Computer Graphics 24*, 3, 1345–1353. URL: https://doi.org/10.1109/TVCG.2017.2669983. 5, 8

MEISTER, D., AND BITTNER, J. 2018. Parallel reinsertion for bounding volume hierarchy optimization. *Computer Graphics Forum (Proceedings of Eurographics) 37*, 2, 463–473. URL: https://diglib.eg.org:443/handle/10.1111/cgf13376. 5, 6, 8

MEISTER, D., BOKŠANSKÝ, J., GUTHE, M., AND BITTNER, J. 2020. On ray reordering techniques for faster GPU ray tracing. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, Association for Computing Machinery, 13:1–13:9. URL: https://doi.org/10.1145/3384382.3384534. 7, 8

MEISTER, D., OGAKI, S., BENTHIN, C., DOYLE, M. J., GUTHE, M., AND BITTNER, J. 2021. A survey on bounding volume hierarchies for ray tracing. *Computer Graphics Forum (Proceedings of Eurographics) 40*, 2, 683–712. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.142662. 3, 16

MERRILL, D., AND GRIMSHAW, A. 2011. High performance and scalable radix sorting: A case study of implementing dynamic parallelism for GPU computing. *Parallel Processing Letters 21*, 2, 245–272. URL: https://doi.org/10.1142/S0129626411000187. 8

MORTON, G. 1966. A computer oriented geodetic database and a new technique in file sequencing. Tech. rep., IBM Research. IBM Germany Scientific Symposium Series. URL: https://dominoweb.draco.res.ibm.com/0dabf9473b9c86d48525779800566a39.html. 4

PANTALEONI, J., AND LUEBKE, D. 2010. HLBVH: Hierarchical LBVH construction for real-time ray tracing of dynamic geometry. In *Proceedings of the Conference on High Performance Graphics*, Eurographics Association, 87–95. URL: https://research.nvidia.com/publication/hlbvh-hierarchical-lbvh-construction-real-time-ray-tracing. 5

STICH, M., FRIEDRICH, H., AND DIETRICH, A. 2009. Spatial splits in bounding volume hierarchies. In *Proceedings of the Conference on High Performance Graphics*, Association for Computing Machinery, 7–13. URL: https://doi.org/10.1145/1572769.1572771. 7, 8

VINKLER, M., HAVRAN, V., AND BITTNER, J. 2016. Performance comparison of bounding volume hierarchies and Kd-trees for GPU ray tracing. *Computer Graphics Forum 35*,

8, 68–79. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12776. 3

WALD, I., BENTHIN, C., AND BOULOS, S. 2008. Getting rid of packets—Efficient SIMD single-ray traversal using multi-branching BVHs. In *IEEE Symposium on Interactive Ray Tracing*, IEEE, 49–57. URL: https://doi.org/10.1109/RT.2008.4634620. 4

YLITIE, H., KARRAS, T., AND LAINE, S. 2017. Efficient incoherent ray traversal on GPUs through compressed wide BVHs. In *Proceedings of High-Performance Graphics*, Association for Computing Machinery, 4:1–4:13. URL: https://doi.org/10.1145/3105762.3105773. 8, 11

Author Contact Information

Daniel Meister
AMD Japan Co. Ltd.
Marunouchi Trust Tower
1-8-3 Marunouchi, Chiyoda-ku
Tokyo, Japan
daniel.meister@amd.com
https://meistdan.github.io

Jiří Bittner
Czech Technical University in Prague
Karlovo náměstí 13
121 35 Prague 2
Prague, Czech Republic
bittner@fel.cvut.cz
https://dcgi.fel.cvut.cz/home/bittner

Daniel Meister and Jiří Bittner, Performance Comparison of Bounding Volume Hierarchies for GPU Ray Tracing, *Journal of Computer Graphics Techniques (JCGT)*, vol. 11, no. 3, 1–19, 2022
http://jcgt.org/published/0011/04/01/

Received:	2022-02-20		
Recommended:	2022-04-14	Corresponding Editor:	Eric Haines
Published:	2022-10-18	Editor-in-Chief:	Marc Olano