Utrecht University
Graduate School of Natural Sciences

Game and Media Technology Master Thesis

Assessing Alternatives to the Surface Area Heuristic for Bounding Volume Hierarchy Construction

---

Author
Athos L. van Kralingen
ICA6037062

In Cooperation With
Traverse Research

Traverse Research Supervisors
Dr. ing. J. Bikker
Ing. M. Oomen

Supervisor
Dr. P. Vangorp

Second Supervisor
Prof. dr. A.C. Telea

December 22, 2023

Abstract

Ray tracing has long been a widespread technique for high-quality offline renderings. Research into acceleration structures for real-time ray-tracing has paved the way for the millions to billions of rays involved to be traced, but the heuristic cost model used for the construction of some of the highest-quality acceleration structures to date, the Surface Area Heuristic (SAH), has long remained the same despite being known to rely on assumptions that rarely all hold.

This thesis explores alternatives to the greedily applied SAH cost model for constructing Bounding Volume Hierarchies. First, a fully non-greedy evaluation method is introduced to compare against the standard greedy construction algorithm. While methods for constructing Bounding Volume Hierarchies exist that achieve a far lower cost than the original greedy method, they are not guaranteed to achieve the global optimum. This method is then used to measure the impact of a non-greedy versus a greedy evaluation on the quality of a Bounding Volume Hierarchy in terms of the Surface Area Heuristic cost in low primitive count scenes.

Secondly, this research extensively compares previously proposed alternatives and improvements to Surface Area Heuristic. Their ray-tracing performance is specifically tested on the differences in assumptions related to the rays and their distributions and compared against SAH. To our knowledge, these alternative heuristics were previously only compared to the Surface Area Heuristic or previous versions of the heuristic that they propose an improvement for and not against each other. The ray-tracing performance of the hierarchies constructed using these alternative heuristics is compared against SAH and each other. The constructed hierarchies are evaluated with the cost model suggested by Aila et al. [1] that can be used to estimate their expected ray-tracing performance.

Observations show that the impact of greedy evaluation is limited to about 5% for the tested maximum of 15 triangles, but the found trend predicts that this difference may be upwards of a factor of a 2.5 lower cost for non-greedy evaluation at only a thousand triangles, in line with other construction algorithms discussed by e.g., Meister et al. in 2022 [2]. There is almost no measured difference when the same is evaluated for 12 triangles as three uniformly sized tetrahedra of aspect ratio 1.

Heuristics that provide an alternative to each of the assumptions of the Surface Area Heuristic are observed to deliver at least comparable performance for given viewpoints. Outliers of superior performance are primarily based on the scene in which the measurements are taken. Compared to the Surface Area Heuristic, the time per ray can decrease by over 30% for given viewpoints in and up to 5% for animated paths through scenes. Reductions of even over 50% are observed for specific ray distributions for these static viewpoints. Additionally, it is observed that some of the heuristics that require viewpoint-dependent input information to construct the hierarchy show a limited performance impact of under 5% in most scenes.

Alternative heuristics are found to produce similar and sometimes superior quality BVHs by the quality metrics described by Aila et al., but the embedding of occlusion information causes the metrics to seemingly underestimate the ray-tracing performance.

Our research concludes that the impact of greedy versus non-greedy evaluation is small in terms of measured tree cost for small scenes, but the linearity of growth in relation to the number of primitives shows that the impact of greediness may be significant. Though the greedy SAH-based top-down construction performs the best on average, the alternative heuristics applied greedily show superior ray tracing performance for the majority of tested scenes individually, and it is shown that selecting the construction heuristic to use should be done per scene. Finally, we find that the adjusted cost metrics by Aila et al.[1] may underestimate the ray-tracing performance for BVH construction techniques that use occlusion-based information, indicating that their metric for scalar ray-tracing may not describe all factors that relate to ray-tracing performance.

Contents

1 Introduction and Background

1.1 Ray Tracing

Ray tracing is a technique in computer graphics that can render images of objects with high fidelity to mimic the real-world traversal of light by following the path light takes along a ray. This technique extends to fields outside of rendering, such as for audio [3] and physics, but is mostly known for its use in computer-generated images. In 2018, NVidia started releasing GPUs under the RTX name [4], which contain dedicated hardware for ray tracing intended for use in real-time rendering applications such as video games; other hardware vendors followed suit in recent years [5]. This introduction improved the popularity of ray tracing, but well before that, it was already prevalent for ground truth reference and the movie industry [6], for example.

With ray tracing, a ray is cast into the scene, where it is tested against the geometry to determine a potential intersection and processes the result for further evaluation of shading effects. The approach that is used today was formalized by Whitted [7]. Later work by Kajiya introduced *the Rendering Equation* [8], which defined the integral that would allow for evaluating the shading of surfaces. They also introduced the concept of *path tracers*, which approximate the integration of this integral through a Monte Carlo process. *Primary rays* are cast from the camera into a scene of to-be-rendered objects, which recursively reflect off these objects' surfaces, continuously bouncing until they reach a light source. Randomness in the directions of these rays is used to approximate the integral and converge towards ground truth once sufficient samples are taken. Examples of reflections that can occur within the scene are specular and diffuse reflections, depending on the surface material properties. As the ray is reflected, its path will ultimately end at a light source that, together with the surface attributes, determines an output color for belonging to the cast primary ray.

Path tracing implicitly models effects such as shadows, reflections, and ambient occlusion, but ray tracing can be more generalized to model these effects individually [9, 10]. Reflections in the form of *diffuse* and *specular* rays were suggested in the initial discovery by Whitted [7], with approaches for real-time use-cases studied, for example, by McGuire & Mara in 2014 [10]. *Shadow rays* are used to find obstructing geometry between a to-be-shaded point and a light source, showing a cast shadow instead of a lit surface point if such obstructions are found; an application already explored by Appel in 1968 [11]. Landis in 2002 [12] was one of the first found sources to formalize the use of *ambient occlusion rays*, which are used to mimic local shadowing effects of global illumination by measuring geometric obscurance locally per the model by Zhukov et al. [13].

In order for these effects to be rendered, ray tracers require many rays to be cast, where each ray has to individually check for intersection against the scene geometry. In rendering, geometry is commonly formed out of smaller *primitives*, usually triangles, where each ray traced needs to be tested for intersection against each individual triangle. With modern-day scenes composed of millions of triangles and using at least one primary ray per pixel, a naive approach would require trillions of intersection tests for just these primary rays. To prevent this, implementations of ray tracing use *acceleration structures* to group the primitives in a scene such that unnecessary intersection tests can be filtered and significantly reduced.

1.2 Acceleration Structures

Acceleration structures divide the scene primitives such that redundant intersection tests can efficiently be eliminated for large subsets of those primitives. The most commonly used techniques in real-time rendering employ a hierarchical tree-like recursive partitioning scheme that recursively partitions either the space around the primitives or the sets of primitives themselves.

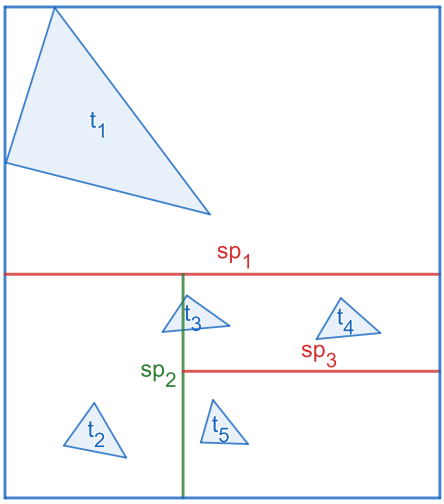


Figure 1: A visualization of how a 2-dimensional k-d tree represents a set of primitives in a scene. The red segments, sp_1 and sp_2, represent splits parallel to the x-axis, while the green segment represents one parallel to the y-axis

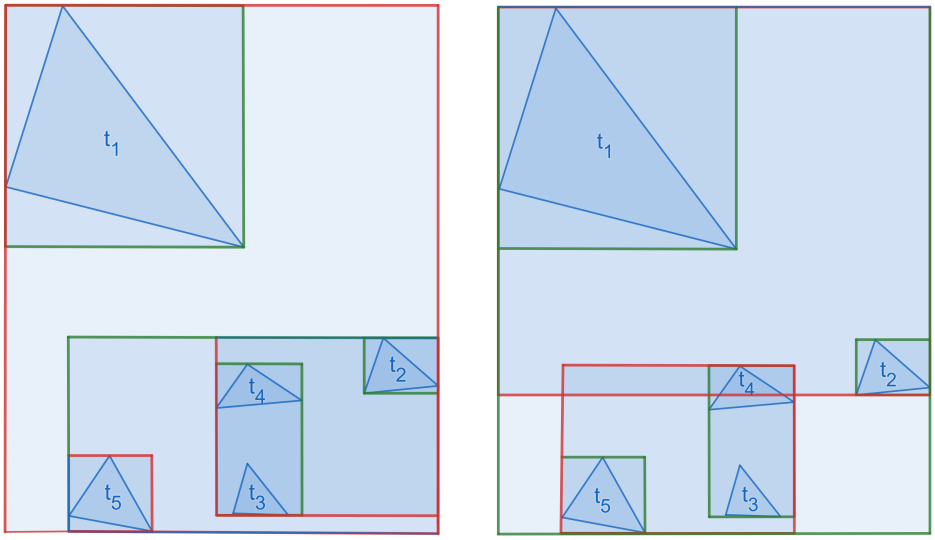
The two most commonly used in ray-tracing for real-time rendering are the *k-d tree* [14] and *Bounding Volume Hierarchy* (BVH) [15].

The k-d tree recursively splits the space of the primitives into two partitions along the x-, y- or z-axis, represented by a *split plane*. The built structure contains nodes of two types: *interior* and *leaf* nodes. Interior nodes represent locations at which the space is split in half and store the axis and position of the split plane, as well as references to its direct child nodes that represent the subspaces induced by this split plane. A leaf node only contains references to the primitives contained within its subspace and has no child nodes. An example is given in Figure 1, where $sp1..3$ respectively represent the split planes of the first three subdivisions created for the k-d tree. The interior node representing split plane sp_1 has two child nodes: a leaf node that only references the primitive t_1 and an interior node that represents the split plane sp_2, which with the subdivision sp_3 ultimately contains leaf nodes for $t_{2..4}$. As shown with t_3, primitives may straddle the split plane and are therefore referenced by multiple leaf nodes. In this example, t_3 is referenced in the leaf node left of sp_2 and above sp_3.

As a result of these recursive spatial subdivisions, a k-d tree can reduce the intersection queries in the best case to a complexity of $O(\log n)$ by recursively eliminating half of the primitive space with each traversal step.

Different from the k-d tree, the BVH recursively splits the scene primitives into n sets, where n is the chosen *degree* of the BVH. Nodes store tightly enclosing *bounding volume* for an approximate, overestimated representation of the union of the primitives that lie inside of it. Interior nodes of a BVH only store the bounding volume rather than the split axis and position of the split. Leaf nodes also store these bounding volumes along with the references to contained primitives. The bounding volumes can be of any primitive type, such as a sphere or tetrahedron, but this is commonly an *axis-aligned bounding box* (AABB) due to them fitting the space sufficiently tightly and the low cost for ray-AABB intersection tests that are needed during traversal. Examples of valid BVHs using AABBs are shown in Figure 2.

Construction algorithms often assume a degree of two, but other multiples of two are regularly used. However, BVHs with degree two are often modified after initial construction [16, 17, 18].



(a) (b)

Figure 2: Two valid BVH registrations for the same set of primitives; the left shows no overlap between the bounding volumes of sibling nodes, while the right shows an overlap due to t_4 straddling a split plane. The AABB edges are colored alternatively based on their hierarchy depth

BVHs mentioned throughout this text are therefore assumed to be of a degree of two and use AABB as volumes, without loss of generality.

There are some other differences to a k-d tree. BVHs do not use the surrounding space to determine the subdivision of primitives; any grouping of the primitives and nodes makes a valid BVH, and the representative bounding volumes may overlap, see Figure 2b. This potential overlap of bounding volumes and arbitrarity in the configuration of nodes implies that, unlike traversing down a k-d tree, there is no guarantee that the search space will be reduced with each step. Therefore most construction algorithms for BVHs do split the set of objects based on their sorted position along one of the axes like k-d trees, but this split plane is not stored explicitly and does not necessarily separate the primitives spatially in their entirety.

Another implication of the allowed overlap of the bounding volumes is that primitives straddling such an implicit split plane do not have to be referenced by both nodes, as the BVH is valid either way. In the example in Figure 2b, the primitive t_4 lies partially in the bounding volume for the parent node of t_1 and t_2 and partially in the one of t_4 and t_3, but is here only stored in the latter. This can lead to a lower memory usage than an equivalent k-d tree.

Both data structures have multiple high-performance and highly parallel construction algorithms available [19, 20, 21], but k-d tree construction suffers from a lower bound of $O(n \log n)$ time [22]. The BVH being trivially valid allows for construction algorithms such as the LBVH by Lauterbach et al. [23] that can be constructed in $O(n)$. Extensive performance tests by Vinkler et al.[24] also show the additional primitive references k-d trees resulting in increased memory usage in, with worst cases of up to ten times as many nodes and 14 times as many triangle references compared to their respective BVH. They additionally demonstrate that ray-tracing performance can decrease by over 40% in selected scenes for the k-d tree as a result of this, despite reducing the number of ray-primitive intersection tests and traversal steps compared to a BVH. These aspects have made the BVH more commonly used for ray tracing in recent years.

```
FindFirstIntersection(rootNode, ray):

if isLeaf(rootNode) then
    closestIntersection = {}
    for primitive in primitives(rootNode) do
        if intersects(ray, primitive) and distance(intersection) < distance(
            closestIntersection)
            closestIntersection = intersection
    return closestIntersection
else if intersects(ray, boundingBox(rootNode)) then
    intersection = ∅
    if intersectionLeft = FindFirstIntersection(leftChild(rootNode), ray) then
        intersection = intersectionLeft
    if intersectionRight = FindFirstIntersection(rightChild(rootNode), ray) and
        distance(intersectionRight) < distance(intersection) then
        intersection = intersectionRight
    return intersection
else
    return ∅
```

Listing 1: First-hit traversal for a BVH

1.3 BVH Traversal

Traversal of a BVH to find a primitive intersection is considered trivial. Starting at the root node and given the ray, an intersection is first tested against the bounding box of the BVH's root node, i.e., the one enclosing the entire space of primitives. If the ray is found to intersect the bounding box, there is a possibility of intersection with a primitive in one of the root's child subtrees, and the bounding volumes of the root's child nodes should therefore be tested in the same manner. For each child node volume intersected, the child node will be set as the traversal's new root node, and the process is repeated until neither child node's volume is intersected. If the current root node is a leaf node, one of its primitives may be intersected by the ray, requiring a ray-primitive intersection for all of its primitives.

After an intersection is found, the result returned by the algorithm depends on the *traversal scheme*. Meister et al., in their survey on BVHs [21], identify the following three traversal schemes:

1. First-hit traversal
2. Any-hit traversal
3. Multi-hit traversal

First-hit traversal returns the primitive intersected by the ray if it is the one closest to the ray origin out of all primitives intersecting it. Most shading algorithms use this as they require surface information about the first intersected primitive for computing the emitted color of the surface. Any-hit traversal returns any primitive lying between the ray origin and the directional extent or some possible endpoint, regardless of which primitive would be hit first. This gives sufficient information for applications such as shadow rays, which only require knowing whether any occluding object lies between the ray origin and a particular light source. Multi-hit traversal is for applications that depend on the surface information of all intersected primitives, such as light traversing through translucent objects. First-hit and any-hit traversal are the most common and, therefore, will be primarily discussed. Pseudocode for a recursive first-hit traversal is given in Listing 1.

1.4 BVH Construction

A BVH construction algorithm splits the original set of primitives recursively until a termination criterion is met. The goal is to determine at build-time which configuration results in the minimum number of intersection tests and traversal steps at runtime. This requires making two primary decisions: where to split the set of primitives and which termination criterion or criteria to use.

Splitting the set of primitives is, in practice, done using a split plane, as explained in Section 1.2. The next decision is where this split plane should be placed, which Rubin & Whitted already found to be a non-trivial decision [15]. Initial approaches used a straightforward approach of placing the split plane at the spatial median of the primitive set along a given axis. This, however, has no guarantee of being the split plane with the best ray tracing performance.

MacDonald & Booth suggested the use of a cost function to determine the quality of a split plane location [25].

They find that using a cost function with the probability of intersection of the surrounding volume leads to a high correlation between cost and ray-tracing performance. They express the cost of an acceleration structure trough as a summation of costs per node, with the full expansion in Equation 1. A heuristic is used to estimate the probability of intersection for the nodes induced by a potential split plane, where a higher probability of intersection results in a higher cost.

$$C_{root} = \frac{1}{SA(root)} \cdot (C_t \cdot \sum_{N \in I} p_N + C_i \cdot \sum_{N \in L} (T_N \cdot p_N)) \tag{1}$$

Here L and I are the set of leaf nodes and interior nodes of the BVH, respectively; C_i is the cost of performing a ray-primitive intersection; C_t the cost of adding a new node, adding two ray-bounding volume intersection tests; T_N the number of primitives in the respective leaf node; and p_N the probability a ray will intersect the node. Note that C_t and C_i are user-defined estimates of their respective costs. This cost function is evaluated for all or a set of candidate split planes, and the lowest cost candidate split plane is used for subdivision.

This cost metric is then used to define the cost of a particular split plane as in Equation 2 and, subsequently, find the local minima when subdividing a primitive set for bounding volume hierarchy construction.

$$C = C_t + C_i(p_l \cdot T_l + p_r \cdot T_r) \tag{2}$$

Here, $p_{l|r}$ and $T_{l|r}$ are the respective intersection probabilities and number of triangles in the left and child nodes induced by the split plane.

As will be covered in Section 3, the chosen heuristic to estimate this probability can have significant performance implications. The performance differences can vary based on the scene and other variables. An extreme case in an example comparison of two heuristics can be found in the research by Vinkler et al. [26] in which a performance difference can be seen up to a factor of four for their rotated Power Plant scene.

1.5 The Surface Area Heuristic

MacDonald & Booth presented their cost function alongside a heuristic that can be used in the cost function: the Surface Area Heuristic (SAH) [25]. They found that, under certain conditions, the probability of intersection between a ray and a node in an acceleration structure is proportional to the ratio of the node surface area over the subtree's root node surface area, see Equation 3.

```
Subdivide(rootNode):

foreach primitive in rootNode
    split = splitAtBoundary(primitive)
    if cost(split) < cost(bestSplit)
        bestSplit = split

if terminationCriteriaMet(rootNode, cost(bestSplit)) then
    return makeLeaf(rootNode)
else
    Subdivide(leftNode(bestSplit))
    Subdivide(rightNode(bestSplit))
```

Listing 2: Top-down BVH Construction

$$p^{sah} = \frac{SA_n}{SA_p} \tag{3}$$

Inserting these probabilities into the cost function from Equation 2 leads to the formula in Equation 4, which has SA_p, SA_l, and SA_r as the respective surface areas of the parent node, left child, and right child node AABBs. Since the division by SA_p is the same for each split plane candidate, it is usually omitted.

$$C_{sah} = C_t + C_i(\frac{SA_l}{SA_p} \cdot T_l + \frac{SA_r}{SA_p} \cdot T_r) \tag{4}$$

Evaluating Equation 4 for all possible split planes of a BVH node, the split plane minimizing C for the theoretical best ray trace performance can be found. Havran shows that the minimum of this cost function always lies at the boundary of a primitive along a given axis [27], meaning only evaluation of split planes positioned at the boundaries of primitives is needed. Once an optimal split plane for a hierarchy level is found, this process is repeated for each created child node downwards until a termination criterion is met. This process assumes that the created nodes will remain leaf nodes. It will only select the split plane that immediately lowers the local cost, despite the possibly lowered cost if it were subdivided further. This identifies the evaluation as a greedy algorithm.

The cost function of Equation 4 is also commonly used for defining a termination criterion. Subdivision is stopped once the lowest cost candidate split plane exceeds the cost of the parent node if that parent node were to remain a leaf node. Retaining the previous notation where the division by SA_p is factored out, a potential termination criterion for SAH becomes as shown in Equation 5. An example of this *top-down BVH construction* is shown in Listing 2.

$$C \geq SA_p \cdot C_i \cdot (T_l + T_r) \tag{5}$$

Calculating the SAH for all potential split planes is computationally expensive: for n scene primitives, there are $n-1$ candidate positions to place the split plane along each axis with a total of $(n-1)*3$ candidates. Several optimizations have been proposed to limit the number of split plane evaluations. A now common approach proposed by Wald et al. [28] is *binning*. The space within the parent bounding box is split into $k+1$ uniform bins along the chosen split axis, where $n = min(T-1, \lambda)$ and λ a user-configured integer. Rather than evaluating a split plane at every primitive's extents, potential split plane candidates are limited to the boundary between every two bins. This reduces the number of split plane candidates for each hierarchy level to k. A second potential optimization is to only consider split planes along the parent node's AABB's longest axis. Typical values for λ are powers of two up to 16. With binning and

using the optimization to only evaluate the longest axis, ray tracing performance is reduced by at most 9% and, on average, closer to 5%. At the same time, construction time decreases by approximately a factor of ten for evaluated scenes [28].

As shown in the survey of BVH construction algorithms by Meister et al. [21], alternative construction methods to the top-down construction in Listing 2 are used, but SAH is still frequently used even for other construction approaches. Two primary causes of SAH being the most used construction heuristic. First, evaluating the intersection probability of a single BVH node is cheap, as calculating the surface area of an axis-aligned bounding box can be done using only few operations. This is essential for situations where construction speed dominates performance. Secondly, the work by Aila et al. [1] shows that a desired Pearson correlation coefficient of 0.99 r between nanoseconds per ray and SAH is often achieved, indicative of a good correlation between SAH cost and ray-tracing performance.

1.5.1 SAH Assumptions

If it is assumed that, based on the formulations by MacDonald and Booth [25], the surface area directly models the probability of intersection and assuming exact values are known for C_i and C_t, the cost function and termination criterion should theoretically give us the optimal BVH. However, for the probability of intersection to be proportional to the surface area, the following assumptions need to hold [25, 29, 1]:

1. Ray origins and directions are uniformly distributed.
2. Ray origins have to lie outside the scene's bounding box.
3. Rays do not terminate inside the scene.

The first assumption means that all rays fired into the scene will have a uniform distribution of origins and directions. A different distribution could skew the resulting probabilities. This is similarly true for the second assumption, where rays starting from inside the scene are not correctly estimated. The third assumption implies that ray intersections are tracked beyond the first-encountered hit, as is required for multi-hit traversal.

1.6 Problem Definition

The Surface Area Heuristic has long been one of the main heuristics chosen to guide the construction of k-d trees and BVHs. Several suggested optimizations have made it reasonably inexpensive to compute a SAH-based BVH [28]. Though a SAH-based cost function generally yields a BVH with significantly better ray tracing performance than using median splits, it has been shown that a lower SAH cost does not always lead to better performance. As addressed previously, the research by Aila et al. [1] shows that despite a correlation coefficient r of 0.99 between SAH and the measured nanoseconds per ray for several scenes, some scenes also show values as low as 0.652. This while values of 0.90 and under were already observed for insufficient correlation of metrics in practice.

Causes for this deviation are, in part, the assumptions listed in Section 1.5.1, which for ray tracing will rarely all hold simultaneously. First, ray origins and directions rarely form a uniform distribution. An example is tracing primary rays, which will originate from a single location and directions limited to within the camera field of view. Countering the second assumption, ray origins are often close to the objects to be tested for intersection and lie within the scene. While primary rays commonly originate from outside the scene, diffuse, specular, and shadow rays are cast from surfaces inside. This generally causes the total number of rays originating

from inside the scene to outnumber the ones starting outside it. The final assumption holds only for multi-hit traversal but rarely for first-and any-hit traversal. Rays that miss all geometry will satisfy this, but other rays should not pass through after the first primitive intersection. As an example for these latter two traversal types, a primitive p fully occluding another primitive p_o would make it impossible for a ray to intersect p_o, while SAH would indicate that its probability is still proportional to p_o's surface area.

In addition to these assumptions, SAH is evaluated for every level in the hierarchy separately in a greedy fashion. The decision where to place the split plane assumes that the result of the split is two leaf nodes rather than potential interior nodes that have to evaluate SAH for their own split decision subsequently. The minimal SAH value on one particular hierarchy level may not lead to a split plane that yields the minimization of SAH over the entire hierarchy. This greedy approach may therefore yield a hierarchy inferior to a non-greedy approach. The exact impact of this greedy approach is unknown.

To summarize, the minimization of the SAH cost can be used to produce BVHs that are considered of high quality, but the implications of these assumptions make it so that SAH cannot accurately model the ray tracing performance for all applications. Greediness means that a full minimization of this cost function is rarely achieved and can further impact the quality of a BVH.

The following chapters will further explore how our research relates to these assumptions, which alternatives have already been created, and discuss the testing of and a testing framework for assessing these alternative solutions.

2 Research Questions

The assumptions needed for SAH to be accurate and the non-greedy evaluation means there is potential for higher quality BVHs, possibly depending on factors such as the ray distribution or scene. As such, it seems necessary to accumulate the research of recent years into improvements to SAH and alternative heuristics that can overcome the limitations mentioned in Section 1.5.1. As SAH is still a frequently used metric for performance correlation, this research should also show whether any of these alternative heuristics leads to a significant difference in correlation to performance.

Finally, the greedy evaluation of SAH should be researched, as the greedy algorithm of Listing 2 only uses local minima. Mapping the impact compared to a non-greedy evaluation may lead to finding potentially better ray-tracing performance.

This leads us to our main research question: how can we construct the optimal BVH for ray tracing using existing heuristics and evaluation methods, regardless of construction time?

This research question is broken down into the following four research questions.

1. With which BVH construction heuristic can we obtain the optimal BVH in terms of ray tracing performance for a specific scene and ray distribution, regardless of construction time?

2. With which BVH construction heuristic can we obtain the optimal BVH in terms of ray tracing performance for the average scene and with varying ray distributions, regardless of construction time?

3. Which existing alternative heuristic results in the best correlation between the estimated tree quality and actual ray tracing performance?

4. What is the impact of the greedy nature of top-down SAH-guided BVH construction on the final tree quality in terms of the resulting SAH?

In Section 3, the previous work that covers all these research questions will be discussed. Section 4 will detail an implementation for non-greedy evaluation and the impact of greediness, and Section 5 will compare the alternative heuristics discussed in Section 3.

3 Previous Work

Over the years, multiple methods have been introduced to improve or completely replace SAH as a heuristic. Construction algorithms that also modify the depth-first traversal algorithm of Listing 1 are not discussed, as the primary interest is in what heuristic has the best correlation to intersection probability and with that, BVH quality. Some discussed heuristics deviate slightly from this, but in subsequent research, these have been adapted to standard depth-first traversal.

3.1 Universal Improvement

The first category of existing improvements to SAH that we define are heuristics or changes to the BVH that can improve ray tracing performance for any ray type but do not necessarily for every scene. To build the BVH, no information or assumptions are required about which rays or with which distributions rays will be traced.

3.1.1 Split Bounding Volume Hierarchy

The Split BVH (SBVH) by Stich et al., [30] is a well-known extension to the BVH, although it still uses the unmodified SAH as a heuristic to guide the partitioning at each hierarchy level. Instead, it has the following changes compared to a regular SAH-based BVH:

1. Primitives are not restricted to be referenced by a single leaf node but can rather be referenced by multiple.

2. Split planes can also split spatially rather than as sets of objects.

The removal of the restriction of the first change above does not functionally affect the traversal of a BVH. Still, it is theoretically not a necessary restriction for a regular BVH, as it will always split the object set such that duplication cannot occur. This is different for the SBVH, which may split the object set spatially and have primitives that lie in both partitions, having to reference that primitive in both resulting nodes.

The motivation for introducing this possibility of spatial splits is inspired by earlier research by Ernst & Greiner [31]. Essentially, SAH for BVHs is found to rely on another assumption next to the three listed in Section 1.5.1: BVH nodes that do not share their subtree are assumed not to overlap.

When two such nodes do overlap, a traversing ray may intersect both child nodes at the same intersection point. As a result, ray traversal must evaluate both child nodes for intersection. For first-hit traversal, at most one child node will contain the desired intersection. This means that the overlap of nodes can increase the number of traversal steps and primitive intersection tests, decreasing ray-tracing performance. This fourth assumption is similar to the assumption that rays do not terminate on intersection, but this adds to it as overlapping nodes lead to unnecessary evaluations regardless of whether a primitive in either node was intersected or not. The assumption is also specific to a BVH, as the nodes of a k-d tree would not be able to overlap.

The solution to such overlap by Ernst & Greiner uses a pre-processing step to deal with large primitives that cause overlap. The SBVH is a continuation that takes a slightly different approach. SBVH construction starts the evaluation of an object split; the same procedure as the top-down construction of Listing 2. Using SAH, evaluate each candidate split plane and select the best one to partition the objects into two sets. The first added step is to calculate the overlapping area of the split plane induced nodes' bounding boxes and measure its surface area. If the overlap surface area over the surface area of the BVH root node exceeds the user-specified

```
FindBestSplitCandidate(Root, Node):

partition1, partition2 = FindBestSAHSplit(Node)
if SA(Aabb(partition1) ∩ Aabb(partition2)) / SA(Root) ≥ α then
    bins = SelectSpatialBins(Node)
    clippedPrimitives = ClipPrimitivesPerBin(Primitives(Node), bins)
    bestBin = null
    foreach bin in bins do
        leftPartition, rightPartition = clippedPrimitives[bin]
        if SAH(leftPartition, rightPartition) < SAH(bestBin) then
            bestBin = bin
    end
    partition1, partition2 = leftPartition, rightPartition
return partition1, partition2
```

Listing 3: SBVH Construction

α, a spatial split is evaluated as well. The spatial split first selects N spatial bins to divide the primitives into. Next, any primitive spanning multiple bins is clipped to the bounding box of the bin, and the clipped bounding box is used to evaluate SAH again per candidate split plane at the boundaries of each spatial bin. The best split plane candidate amongst these is used as the final split. Pseudocode for this algorithm can be found in Listing 3. The authors also propose an optimization on top of this known as *reference unsplitting*, which further reduces the total SAH cost.

The traversal of the BVH remains identical, as there are no changes to the intersection tests and traversal decision-making. However, the number of references to a primitive doubles every time it is clipped via a spatial split, increasing memory usage to some extent. Stich et al. show that an SBVH improves performance by up to 39% for primary rays for standard scenes and more for SAH worst-case scenarios. Vinkler et al. [26], while evaluating another heuristic in 2012, show even more such worst-case scenarios in which the SBVH outperforms a regular SAH-based BVH by up to a factor of four in ray tracing performance. The authors' suggested α of 10^{-5} limits the average memory overhead to under 30%, though they report outliers of as much as 103%. Another downside is that this approach has been found harder to optimize than normal BVH construction techniques. For traditional BVHs, the primary example is the LBVH [23] that can construct a BVH in $O(n)$ and is considered relatively straightforward to parallelize.

3.1.2 End-point Overlap

In 2013, Aila et al. [1] explored the idea that SAH may not be the only correlation factor for ray tracing performance of BVHs. Similar to the intuition behind the SBVH, they also note that overlap is a major factor but formulate a slightly more specific metric: the *end-point overlap* (EPO).

EPO measures the overlap between triangles and sub-tree nodes that they are not a child of. This means that similar to the SBVH algorithm, we perform clipping of the primitives against the node volumes, but for EPO the metric involves clipping each primitive in the entire BVH against each interior node that is not an ancestor. EPO is the total surface area of these clipped primitives. A visualization of what EPO represents is shown in Figure 3.

Currently, EPO is purely a BVH metric and not implemented as a heuristic for BVH construction. It only intends to find a better correlation between BVH cost and ray tracing performance. Aila et al. tested nine different construction algorithms, with and without explicit optimization over SAH and both top-down and bottom-up algorithms. They found that incorporating EPO into SAH through a linear function increases the Pearson correlation coefficients between the

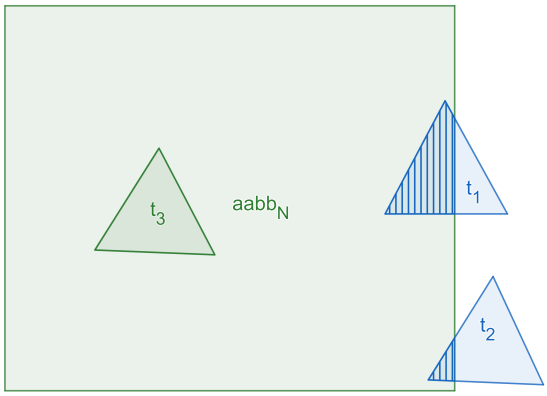


Figure 3: A visualization of EPO. EPO is the marked area of t_1 and t_2 within $aabb_N$

heuristic and scalar ray tracing performance from an average of r of 0.915 to 0.994 over their tested scenes. Especially interesting is that the found worst-case correlation coefficient over all scenes increased from an r of 0.652 to 0.988.

One caveat is that the combined metric requires a scene-dependent weighing factor α, as seen in Equation 6,

$$(1 - \alpha) \cdot SAH + \alpha \cdot EPO \tag{6}$$

With $0 \leq \alpha \leq 1$. For the mentioned increase in correlation, α was hand-configured for every scene. Using a single α of 0.71 for all scenes decreases the average Pearson coefficient to r 0.98, which still significantly improves over SAH.

Along with the observed performance correlations, they also note that top-down BVH builders are generally better at implicitly optimizing EPO. For these, the separation of primitives into minimal surface area bounding volumes does minimize overlap to some extent. Bottom-up builders, on the other hand, start at the individual primitives and build upwards, which makes them unable to minimize this overlap in the same manner as it recursively merges nodes upwards.

Currently, an EPO-based heuristic for greedy evaluation has not been construed. However, this metric can still be used to gain insight into the implicit optimization of EPO for other heuristics and evaluate their expected hierarchy costs compared to their ray-tracing performance.

3.1.3 Leaf Count Variability

Not only did Aila et al. [1] introduce end-point overlap, but they also introduced a quality metric named *leaf count variability* (LCV). Modern raytracing is often parallelized, using *single instruction multiple data* (SIMD) or through other massive parallelization devices like the GPU. Even with EPO, SAH does not correlate very well with ray tracing performance once executed using instruction-level parallelism on the CPU Or GPU. This is mostly caused by how these methods need to trace multiple rays through the BVH simultaneously: a performance hit will occur if the rays simultaneously traced will differ in the number of leaf nodes that they intersect. The instructions for SIMD-like architectures require operation in lockstep. Therefore, a ray-primitive intersection can not be performed at the same time as a traversal step, and a performance hit occurs when the number of leaf nodes is not uniform for all simultaneously traced rays.

Aila et al. summarize such a characteristic to the LCV metric in Equation 7,

$$LCV = \sqrt{E[N_l^2] - E[N_l]^2} \tag{7}$$

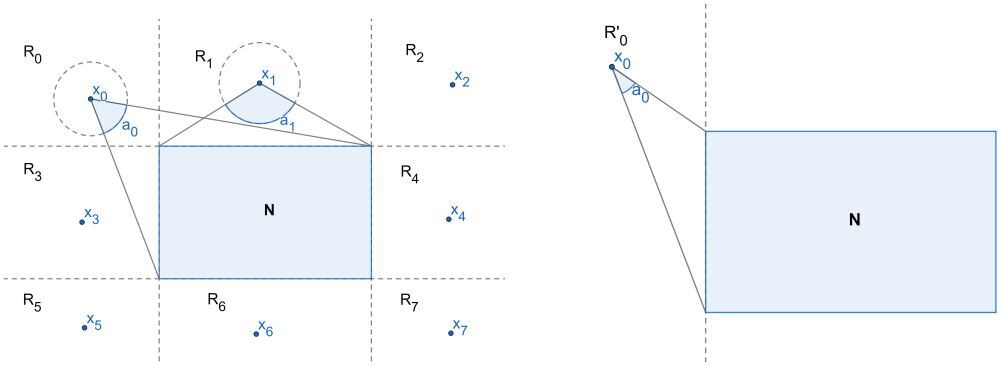
Where E is the expected value for the number of rays intersecting leaf nodes and N_l is the number of leaf nodes intersected. In other words, LCV is the standard deviation of the number of leaf nodes intersected per ray. To accurately predict the performance for SIMD-based raytracing, LCV is then combined with EPO, SAH, and another scene-dependent weighting factor β into the convex function shown in Equation 8.

$$(1 - \alpha - \beta) \cdot SAH + \alpha \cdot EPO + \beta \cdot LCV \tag{8}$$

Measuring this for their tested set of BVH builders improves correlation coefficients even further than EPO when raytracing is done using their SIMD raytracer. Compared to the combined metric from Equation 6, correlation coefficients improve from a minimum of an r of 0.907 up to 0.988; and, on average, from an r of 0.979 up to 0.993.

The main downside of LCV is that it requires a form of sampling for the expected value of the number of leaves intersected by rays, but the only way of evaluating this is by tracing rays through the BVH. This makes LCV generally only suitable as a quality evaluation metric of a constructed BVH and not something to optimize during construction.

3.1.4 Scene-Interior Ray Origin Metric



(a) The middle slice of regions for the high-quality approximation

(b) The region and difference in solid angle evaluation for the fast approximation

Figure 4: A 2-dimensional slice of the subdivided regions of the parent bounding box used for approximating the scene-interior ray origin metric

Fabianowski et al. in 2009 [29] changed SAH's assumption of rays starting outside of the scene and proposed an approximation specifically for rays starting inside the scene. This change is, in particular, beneficial for the previously mentioned use case of *diffuse rays* that will originate from surfaces within the scene.

They first derive that the intersection probability between a ray origin x and a BVH node N is equal to the fractional solid angle between the ray origin and N, assuming ray directions are still uniformly distributed. This can be computed by projecting N's visible faces onto a unit sphere centered around x and summing the covered surface area of each projected face. Doing this for every possible ray origin x_i that lies outside the bounding box of node N but inside

the scene bounds S. Averaging this and normalizing for the scene bounds leads to equation Equation 9.

$$p_N = \frac{V(N)}{V(S)} + \frac{1}{V(S)} \int_{S \setminus N} \frac{a_x}{4\pi} dx \tag{9}$$

S are the scene bounds, N is the node in evaluation, V is the volume of the respective bounding box, and a_x is the solid angle of all faces of N projected onto the ray origin x. Specifically, a_x is the surface area of N visible from x. The integral is used to average this for all points inside the volume of S but outside of N, measuring the average visible surface area of the entire node N within the scene bounds. This causes the heuristic, in contrast to SAH, to assume ray origins always lie within the scene bounds.

The integral in Equation 9 is one that cannot be expressed through elementary functions such as summations, meaning it has no closed-form solution. Instead, two approximations are defined: one that favors BVH quality and another that favors BVH construction speed.

Their first approximation is considered the higher quality approximation of Equation 9. The faces or N are extended to planes, which split the 3D space around node N into 26 regions of possible ray origins within S, as displayed in Figure 5 as 3D visualization of the regions and Figure 4a as a 2D slice of a subsection of these regions. Each region R_i contains the origins $x0..26$ from which a single face or unique tuple of faces N_i is visible. Rather than evaluating all possible ray origins x within a region R_i, only the center of each region R_i is used to find a solid angle a_i per region. This results in the formula in Equation 10.

$$p_N \approx \frac{V(N)}{V(S)} + \frac{1}{V(S)} \sum_{i=1}^{26} V(R_i) \frac{a_i}{4\pi} \tag{10}$$

Here, V is again the volume of the bounding box or respective region formed by the subdivision of S.

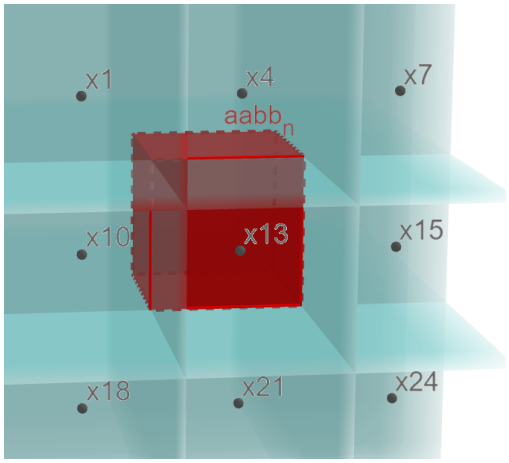


Figure 5: A 3D view of a slice of the regions at one side of the bounding box with each region center x_i

The second proposed approximation is less accurate but can offer lower construction times. Rather than also considering the crossing of the planes of N as separate regions, only the six overlapping regions R'_i are considered that each contains the origins x from which the respective

face N_i is visible. In other words, there are six regions, with each region R'_i corresponding to the volume of points from which at least the face N_i is visible; one such region is displayed in Figure 4b where x_0 represents one of the possible ray origins in the region R'_0. These regions can also be described as the union of regions from the high-quality approximation from which the face N_i is visible. Given Figure 4a, the slice of the four regions displayed would change to $R'_0 = R_0 \cup R_1 \cup R_2$, $R'_1 = R_0 \cup R_3 \cup R_5$, $R'_2 = R_5 \cup R_6 \cup R_7$ and $R'_3 = R_2 \cup R_4 \cup R_7$, meaning that unlike the previous definition, the regions overlap. Note that the respective 3D regions are a union of more regions, but this description is limited for illustration purposes to only the regions that are shown in Figure 4a. Additionally, the solid angle is approximated using the relative surface area of the face N_i to its corresponding region R'_i; see equation Equation 11. This approximation relies on the property that the center of a larger region will lie further away from the AABB of N, amounting to a smaller solid angle. This means that both the surface area ratio and solid angle have a positive relation to the size of the region R'_i relative to the AABB of N.

$$p_N \approx \frac{V(N)}{V(S)} + \frac{1}{V(S)} \sum_{i=1}^{6} V(R'_i) \frac{SA(N_i)}{SA(R'_i)} \tag{11}$$

The authors integrated this modified heuristic into a k-d tree builder. For the majority of their tested scenes, they achieved a reduced number of traversal steps and intersections performed. Outliers show decreases in traversal steps by as much as 25%, intersections by as much as 21%, and an FPS increase of up to 7.5%. However, the performance increase can vary significantly, as seen from the average results and standard deviations in Table 1. The average performance difference between the two approximations is rather small, and one test case even shows situations in which the fast approximation of Equation 11 does not yield any performance increase, while their higher quality approximation of Equation 10 does. It should be noted that their reported results in terms of FPS may give a false impression compared to measurements such as rays per second [32].

The authors' tests were limited to primary, shadow, and reflection rays. The type of reflection rays tested for is unspecified, but it is assumed from the images that these are specular. Similarly, these images show that the camera seems to be placed inside the scene bounding box, which would also cause all ray origins to lie inside it, favoring their described metric. As not all real situations have the viewpoint inside the scene, there may be a measurable impact on primary rays with this metric. Additionally, assuming the reflection rays they tested for are considered specular reflection rather than diffuse, there may be ray distributions such as diffuse reflections that are more uniformly distributed throughout the scene, more closely matching the intended use.

Finally, it seems possible for this new metric to be merged into the metric with EPO and LCV in the form of Equation 8, possibly finding an even better correlation.

	Traversals	Intersections		FPS
		Plane	Primitive	
HQ Approximation Average	-7.7%	-10.4%	-7.1%	+3.5%
HQ Approximation Stdev.	11.8%	8.7%	3.4%	2.7%
Fast Approximation Average	-11.0%	-4.1%	-5.2%	+3.2%
Fast Approximation Stdev.	10.4%	17.3%	5.6%	2.6%

Table 1: Performance characteristics of using the interior ray metric compared to SAH and standard deviation for the metric measured. Measurements by Fabianowski et al. [29].

3.2 Fixed Ray Distribution

SAH is used because it approximates the intersection probability given that the distribution of rays is uniform, which as discussed in Section 1.5.1, is rare in practice. Instead of changing the cost metric to another generalized model, this assumption of SAH can also be alleviated by building a BVH for a specific ray type or set of rays. This means a BVH may only be optimal for a given set of rays and decrease in performance when the scene or camera changes, though the BVH may be reusable at least partially if there exists temporal coherence between frames. Alternatively, a BVH built for one type of ray can still be preferred if that ray type dominates the other distributions the BHV is used for or by having multiple BVHs, one per ray type.

3.2.1 Preferred Ray Sets

Bittner & Havran first demonstrated specialization for ray distributions in 1999 by adjusting SAH to a preferred distribution or ray set [33]. They introduce metrics for three distributions of rays, of which two are relevant to primary rays: correcting for parallel projection and perspective projection. Their third metric is similar to what was done later in the work by Fabionowski et al, see Section 3.1.4, where they specialize the acceleration structure for a uniform distribution from inside the scene. The spherical distribution is covered more extensively by the metric by Fabionowski et al. for the ray distributions of interest, so we will focus specifically on the modifications by Bittner & Havran for primary rays.

For parallel and perspective rays, they observe that the probability of intersection can be found through the projection of a node N AABB onto the parallel or perspective projection plane and then clipping it to the viewing frustum. The probability of the ray intersecting a node N can then be expressed using Equation (12), where p_N represents the new probability of intersecting node N, $SA^{projected}$ is the projected surface area of N onto the projection plane and S the bounding box of the root node.

$$p_N = \frac{SA^{projected}(N)}{SA^{projected}(S)} \tag{12}$$

Though they only perform tests in four different scenes, the primitive intersection tests per ray are observed to decrease by 29% on average and traversal steps by 16%.

This work is interesting because it shows the possibility of improving ray tracing performance through simple heuristic changes but is limited to primary rays. Despite that, using this as a separate BVH that only traces primary rays or a weighted combination, with the change proposed by Fabianowski et al. mentioned in Section 3.1.4 seems like an option worth considering if the performance for primary rays significantly improves.

3.2.2 Ray Distribution Heuristic

The aspect of specializing an acceleration structure to a specific ray distribution was conceptualized through the *Ray Distribution Heuristic* (RDH) by Havran & Bittner [34], which they suggested for the construction of k-d trees. They present the notion of a *representative ray set* (RRS); a subset of the to-be-cast rays that accurately represents the total ray distribution. The RRS has several possible definitions for an animation frame i:

- The RRS for frame i is a subset of the rays cast in frame i.
- The RRS for frame i is a subset of the rays cast in frame $i-1$.
- The RRS for frame i is a subset of the rays cast in all animation frames.

Given an RRS with one of the above definitions, a split plane candidate is evaluated by determining which rays intersect the bounding box of the left partition and which intersect the right partition. The probability of intersection is then estimated per Equation 13.

$$p^{RDH}_{\{L|R\}} = \frac{|R_{\{L|R\}}|}{|R|} \tag{13}$$

R_L and R_R are the set of rays intersecting the left and right child node, respectively, and R is the set of rays intersecting the parent node. A visualization of the RRS per child node can be seen in Figure 6.

The resulting probability itself is not found to be a fully suitable heuristic, so an interpolation between the probability found through RDH and SAH is used; see Equation 14. The weight, w, is defined by a separate function listed in Equation 15, with α and β being pre-configured constants. With the author suggested values of $\alpha = 0.9$ and $\beta = 0.1$, this results in a function that strongly decreases the influence of RDH on the estimated probability once the RRS shrinks to below one hundred rays. The cost function remains the same as in Equation 2, where $p = p^{RDH}$.

$$p^{RDH}_{\{L|R\}} = w \cdot p^{RDH}_{\{L|R\}} + (1 - w) \cdot p^{SAH}_{\{L|R\}} \tag{14}$$

$$w = \alpha \cdot (1 - \frac{1}{1 + \beta} \cdot |R| \tag{15}$$

Their experiments compare their method to only a BVH build using SAH to guide split plane decision-making. The average rendering time for only primary rays was observed to go down by approximately 15%, though the results include scenes with increases of over 10% and one with a decrease of over 44%, showing that variance can be high. It is also observed that RDH performs better relative to SAH when there is relatively high occlusion in the scene. The authors also show that the RRS can be subsampled with low impact on performance. Reducing the RRS by sampling only one ray per 4x4 pattern still gave results that outperformed SAH by 6% on average.

When shadow rays and three diffuse bounces are included to simulate path tracing, the performance compared to SAH is actually worse by 5%. If this performance were equal, it would be explained by the combination of ray distributions being more uniform compared to only primary rays, closer to the SAH assumption about ray uniformity. However, the worse performance than SAH may be caused by the $4x4$ sampling pattern of the RRS that is used; results for higher densities other than for casting only primary rays are not presented. Although the primary rays may follow a coherent pattern, the diffuse rays, for example, will be much more uniformly scattered, possibly requiring more dense sampling for better results.

Another possible influence is how the subsampling is performed. When subsampling the rayset by subsampling the set of primary rays, the number of diffuse bounces varies more greatly per primary ray than the in the original RRS. Especially for second and third bounces, the intersections of rays with nodes may be sparse and a reduction by one ray may significantly influence the approximated intersection probability. The path tracing results for highly occluded scenes are not included, which are scenes that may result in a higher ratio of primary rays leading to first, second, and third diffuse bounces.

The reported measurements are not separated by ray distribution. If the RRS consists of significantly more primary rays than those of other distributions, the primary rays may improve more than, for example, shadow rays. Since RDH specifically aims to capture the ray distribution and the observation of the authors regarding the uniformity of rays for additional diffuse bounces, it may be worthwhile building and tracing per ray distribution separately.

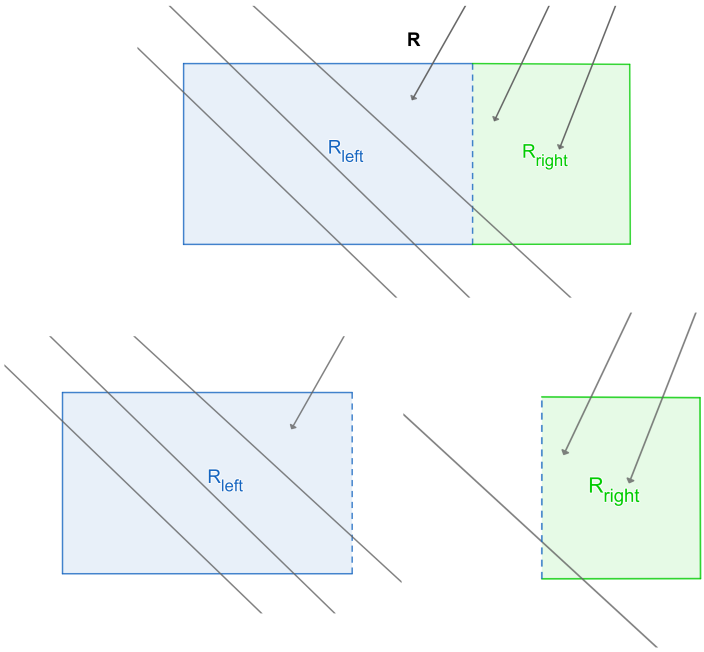


Figure 6: A visualization of the representative ray set for each child node

One important note regarding the weight function is that this might not necessarily translate directly to BVHs. For k-d trees, the union of the sets of rays intersecting the left and right child nodes is the set of rays intersecting the parent node. This union may be a proper subset for a BVH, as the union of child node bounding box volumes does not necessarily form the parent bounding box. As a result, weighing by the number of rays intersecting the parent node may be incorrect.

3.2.3 Occlusion Surface Area Heuristic

As covered in the previous sections, occlusion is a significant factor for rays traversing a BVH. With a different approach, Vinkler et al. [26] introduced the *Occlusion Surface Area Heuristic* (OSAH). The general idea is to add a visibility term to the building process by using visibility information from the previous frame to guide the building of the BVH instead. This is similar to the RDH process of using a representative ray set.

As noted by Vinkler et al., a BVH of only the visible primitives is shallower and implicitly requires fewer traversal steps during traversal. Their approach is a cost function that favors placing the invisible triangles in a subtree separate from the visible ones. The cost function is as per Equation 16.

$$p_{\{left|right\}} = w \cdot \frac{N^V_{\{left|right\}}}{N^V_{left} + N^V_{right}} + (1.0 - w) \cdot \frac{SA_{\{left|right\}}}{SA_N} \tag{16}$$

N^V_{left} and N^V_{right} are the number of visible primitives in the respective partition for the candidate split, SA the respective surface areas, N the current node we are evaluating the cost for and w a user-defined weight to blend between the SAH factor and the visibility factor. The authors use a weight of w to 0.9 but also mention that this is scene dependent. In any situation, a value

of 1.0 is not advised, as the SAH term should be included for tie breaks. This formula is and cannot be used if all triangles in the parent node are either considered visible or invisible. In these situations, SAH is used.

Similar to how the surface area is the probability of intersection relative to the parent node, the number of visible triangles can be used in the same manner. The higher the estimated number of triangles that will be intersected, the higher the probability that a node will be intersected. This is also similar to RDH, except that the construction does not rely on the size of the ray set and that the assigned visibility is binary. This binary visibility has the property that fully separating visible from invisible triangles leads to a low cost, as the fraction for the node with no visible triangles is zero at $w = 1$. A downside is that nodes that are intersected very infrequently are considered equal to those intersected for nearly every ray and by the notion of the OSAH factor in Equation 16, would be optimized the same.

To counter this, the algorithm applies a depth limit and a viability condition. The viability condition first determines whether an OSAH split is worth considering and uses SAH otherwise. Both the best splits using SAH and OSAH are computed and the OSAH split is only performed if OSAH is considered to hide more primitives. For OSAH, the number of hidden primitives is considered to be the number of primitives in the found child node with the lower number of visible primitives, as it aims to hide these from the traversal path of visible triangles. For SAH, this is considered the maximum over the number of primitives in each split-plane induced child node, as these triangles are hidden and excluded from the hierarchy for the sibling node assuming that the node with more triangles contains the primitives that are not desired to be encountered during traversal. This condition is summarized in Equation 17.

$$N_h = \begin{cases} T_{left} & \text{if } N^V_{left} < N^V_{right}, \\ T_{right} & \text{else} \end{cases} \tag{17}$$
$$N_h > max(T^{SAH}_{left}, T^{SAH}_{right})$$

$T^{SAH}_{left|right}$ are the respective numbers of primitives in the left and right partitions for the best SAH split and N_h is the number of primitives hidden from the traversal path for visible triangles by the best OSAH split. In an ideal situation, N_h only considers the invisible triangles due to the OSAH cost favoring a full split between invisible and visible if possible.

The authors implement OSAH specifically for an SBVH, though there is no notion that it does not transfer to a regular BVH. Compared to the SBVH by Stich et al., render times decrease by about 17% in high-occlusion scenes. They also conduct their tests to measure the results for specific ray distributions: primary, shadow, ambient occlusion, diffuse, and path tracing, with each showing improvements between 13% and 18% and high consistency between different ray distributions.

There is no mention of experiments with OSAH in a normal BVH, which may indicate that the performance increase for this is suboptimal. Experiments will have to show whether the difference in impact is substantial.

3.3 Shadow Rays

Some heuristics that have been researched are specialized to shadow rays. Shadow rays are specifically used to test for any obstruction between a ray origin and the endpoint, the light source. Unlike primary and diffuse rays, it is irrelevant which primitive is intersected first but rather whether any object is hit along this path at all. For shadow rays, the traversal can be terminated once any intersected objects have been found, which is used as a property for the

following heuristics to optimize ray traversal. Other optimizations are used explicitly during the construction of the BVH.

Though the interest of this research lies explicitly in the construction of the BVH and not its traversal, it is briefly discussed how these modifications of traversal can be used during construction only instead.

3.3.1 RTSAH

The ray termination surface area heuristic (RTSAH) by Ize & Hansen [35] is a heuristic that exploits the aspect of being able to terminate rays early for shadow rays. This modifies the traversal by determining at BVH traversal which node has a lower cost for testing an intersection, favoring nodes with high probabilities of intersection as these allow the traversal to terminate. SAH is used to determine these probabilities of intersection.

For interior nodes, several probabilities are first defined for a ray to pierce its child nodes: p_{lr} is the probability of hitting both left and right child nodes, p_{jl} that of hitting only the left child and p_{jr} that of only hitting only the right child node. The latter two can be expressed in terms of p_{lr}: $p_{\{jl|jr\}} = p_{\{l|r\}} - p_{lr}$, with $p_{l|r}$ the probabilities of intersecting the left or right, regardless of whether the other child node was intersected. They also define the probability of intersecting empty space as $p_e = 1 - (p_{jl} + p_{jr} + p_{lr})$. These definitions are then used to define separate costs of entering the left and right child nodes, which can be used to determine the cost of entering either child node first, as described in Equation 18.

$$\{left|right\}First = C_t + p_{\{l|r\}} \cdot C_{\{l|r\}} + p_{\{jl|jr\}} \cdot (C_t + C_{\{r|l\}}) + P_e \cdot C_t \tag{18}$$

C_t is again the cost of traversal and $p_{\{l|r\}} \cdot C_{\{l|r\}}$ the cost of having to traverse the respective left or right child node. $leftFirst$ and $rightFirst$ are the respective costs of traversing the left or right child node individually. At the time of traversal, only the minimum cost amongst the child nodes matters. The result is that child nodes with a lower cost are explicitly traversed before those of lower intersection probability, leading to potentially fewer traversal steps and intersection tests. The traversal of paths through the BVH that have significant empty space are penalized, as these are unlikely to lead to an intersection and, therefore, an early termination. To evaluate the probabilities p_{jl} and p_{jr} of the cost function in Equation 18, the probabilities p_l and p_r need to be determined, which is done through SAH.

They also propose an approximation where they set $p_e = 0$, meaning the probability of traversing empty space when going to a child node is zero and removes the need for computing the previously mentioned form factors, improving construction times.

Their experiments compare the front-to-back traversal of a SAH-constructed BVH to their modified traversal. In their four test scenes for BVHs, speed-ups between 44% and 50% were observed compared to standard traversal for a SAH-constructed BVH. The approximation has a performance impact compared to the full evaluation between 10% and 16%. It should be noted that the speed-ups compared to SAH are almost entirely from shadow rays that do intersect with geometry, as this leads to earlier termination than normal SAH. For shadow rays that do not intersect geometry, the speed-up is negligible, as there is no early termination of traversal.

Changing the BVH's traversal order falls outside the scope of our research. Still, a promising aspect of the proposed cost function is that it represents a probability of intersection that can be used as a heuristic. With as in Listing 1, it can be assumed that traverse the left child node is always fully traversed before the right child node. This allows for using RTSAH as a BVH construction heuristic that also considers the ordering of child nodes or does this as a post-process step. In other words, during construction, the lower cost child node should be set to be the left

child node and the higher cost as the right, based on the assumption that the left child node's path will always be traversed first.

3.3.2 Surface Area Traversal Order

Surface Area Traversal Order (SATO) by Nah & Manocha [36] is a modification to RTSAH that primarily aims to reduce the pre-processing performance of RTSAH. Despite the aim of reduced times, the cost function is significantly changed from RTSAH and shows different performance characteristics. Three different traversal order schemes are presented, but this will focus on the traversal order that leads to improved ray-tracing performance over RTSAH.

SATO is heavily based on the two properties that can be observed with SAH-based construction:

1. Higher-level nodes of the hierarchy are more likely to contain large primitives.
2. The probability of a node containing larger primitives in its subtree is directly related to the surface area of the node's bounding box.

They use this to define their PrimSATO submetric, which assigns a lower cost for the traversal of the node with the higher average surface area of primitives within that subtree. In other words, if the average surface area of all primitives of all leaf nodes in the left subtree is higher than that of the right, we traverse to the left node first. Due to the listed assumptions, this should give the quickest path to finding a potential occluder for shadow rays. They also propose using the maximum surface areas rather than the average.

The result of PrimSATO is directly used as the traversal order function, similar to the one listed for RTSAH in Equation 18. The results for PrimSATO show that it slightly outperforms RTSAH and outperforms front-to-back traversal of a SAH-built tree by a factor of 1.52 on average. The results of using the maximum surface area are slightly less consistent and show lower performance gains.

Like our comment for RTSAH, the authors note that a BVH can be built such that the left node is the best option to traverse first without needing any runtime information. This again leaves us options for experimentation to compare against other methods.

3.3.3 SRDH

The Shadow Ray Distribution Heuristic (SRDH) by Feltman et al. [37] can be seen as a combination of RTSAH and RDH. Their approach uses a second BVH specifically for shadow rays, which the representative rayset is used on.

Similar to RTSAH, costs are determined for traversing a particular child node. They generalize the general decision-making of traversal as a *kernel function*, which will determine whether the left or right child node is to be evaluated first by returning a value between zero and one for a ray r. similar to the RTSAH cost function in Equation 18. For example, if we desire to traverse front-to-back with ray r with node N, the kernel function will return one if the left child node of N is closer to the origin of r and zero if the right child of N is closer. This kernel function can be defined on a per-node basis.

This definition can then be used for a cost model for estimating split plane costs like for RDH and SAH. Given a representative ray set R, the total split plane cost for a ray $r \in R$ is a combination of the kernel function as defined above, the ray hitting the sibling node and the ray hitting a primitive within the subtree of the child node. This leads to the total cost function in Equation 19

```
SRDHBuild(P, R):

if length(P) < splittingThreshold then
    return CreateLeaf(P)

minCost = ∞
bestSplit = null
foreach (leftPartition, rightPartition) in ObjectPartitions(P) do
    foreach kernel in kernels do
        if R is empty then
            cost = SAH(leftPartition, rightPartition)
        else
            cost = SRDH(leftPartition, rightPartition, kernel, R)
        if cost < minCost
            minCost = cost
            bestSplit = (leftPartition, rightPartition, kernel)

rrsLeft = RRSExcludingIntersectWith(R, (1 − bestSplit.kernel), bestSplit.kernel)
rrsRight = RRSExcludingIntersectWith(R, bestSplit.kernel, bestSplit.right)
return CreateInterior(SRDHBuild(bestSplit.left, rrsLeft), SRDHBuild(bestSplit.right
    , rrsRight), bestSplit.kernel)
```

Listing 4: Building a BVH with SRDH as heuristic

$$C_{SRDH}(P_{left}, P_{right}, k, R) = \sum_{r \in R} (1 - k(r) \cdot H(P_{right}, r)) \cdot I(P_{left}, r) \cdot |P_{right}| \\ +(1 - (1 - k(r)) \cdot H(P_{left}, r)) \cdot I(P_{right}, r) \cdot |P_{right}| \tag{19}$$

P_{left} and P_{right} are the respective sets of primitives of the two partitions for a candidate split plane, and k is the kernel function for N. Furthermore, H and I are binary functions, where H returns one if there is at least one intersection between r and the set of primitives and I whether r intersects the bounding box of N; both return zero otherwise.

The build process iterates over all possible combinations of split plane positions and kernel functions. Though the set of possible kernels is not fixed for the build process, the authors note the examples of front-to-back, back-to-front, and a constant function (i.e., always select right or left, independent of the ray). This is one of the main differences between RTSAH and SATO, as SRDH dynamically changes the traversal scheme locally.

Once the lowest cost combination of split plane and kernel is found, we can recurse into the created child nodes N_{left} and N_{right}, with $R_{left} = \{r \in R | k(r)H(p_2, r) \neq 1\}$ and $R_{right} = \{r \in R | (1-k)(r)H(p_1, r) \neq 1\}$ respectively. If the input RRS R is empty, SAH is used instead, similar to RDH using SAH as a fallback. Pseudocode for this algorithm can be found in Listing 4. It should be noted that the only stopping criterion for splitting into partitions is the number of primitives that remain.

on traversal, we evaluate the kernel function k stored in each interior node to continue. The authors note that a separate BVH is needed next to this one because the traversal function is specific to any-hit traversal.

Their experiments show a speed-up of 41% compared to back-to-front and 44% compared to RTSAH traversal. These speed-ups are significant, but curiously, RTSAH performed worse than back-to-front traversal, with the latter being constructed through a SAH-built BVH.

Finally, we have also found it to be shared among these traversal order techniques that there is the potential to re-order child nodes instead of storing which node is best to traverse. This is no less true for SRDH, and an extension had been proposed by Ogaki & Derouet-Jourdan [38]. The changes reflect almost identically to those we stated for RTSAH and SATO.

3.4 Non-greedy SAH Computation

As noted previously, SAH as a cost function is applied is done greedily. The exact impact of this greediness has been unknown. Only brief notes, such as in the work by Wald in 2009 [28], state that in certain situations, the greediness performance impact can be observed when fewer split plane candidates are evaluated than the total number of possibilities.

The computation of a non-greedy SAH acceleration structure was briefly explored in Havran's Ph.D. thesis [27]. The nature of the cost function in Equation 1 is subsequently shown to require solving an $\mathcal{NP}$-hard problem, with no polynomial-time algorithm known. This means that finding a solution for a sizeable n is infeasible. In practice, Equation 1 has to be evaluated for all possible hierarchies to find the lowest cost BVH. This constitutes a combinatorial problem in which the combination of split plane positions at every possibly created level in the hierarchy makes evaluation infeasible for any regular test scene.

Ng & Trigonov further show that the number of combinations is at least exponential [39].

Other than the research by Havran and proof by Ng & Trifonov, there is relatively little exploration of this topic, though there has been research on producing lower SAH cost hierarchies than top-down greedy construction.

Another approach for BVH construction with SAH cost values lower than top-down greedy construction was proposed by Ng & Trifonov [39]. They use stochastic search and evolutionary methods that find the best split plane with lower costs than the greedy approach. It is also shown for these evolutionary algorithms, the convergence speed to a local optimum depends on the scene. Yet, there is no method to escape these local minima.

In 2017, Wodniok & Goesele [40] suggested the possible SAH BVH by using temporary subtrees to evaluate the SAH for each split plane position candidate. A temporary BVH for the left and right partition at each split plane position candidate is constructed greedily. The split plane position with the minimum cost for its left and right sub-hierarchies is chosen as the split plane. This allows results in a partially, non-greedily evaluated BVH in $O(n \log^2 n)$, but does not yet pose the optimal. The sub-trees at every split-plane location are greedy, imposing the same greediness implications only one level down. Additionally, binning is used for feasible construction times, which does impact the total SAH cost negatively.

Finally, Kensler [41] uses simulated annealing with tree rotations for optimizing local SAH beyond local minima and Meister et al. [2] use this further to create a baseline SAH cost for BVH hierarchies. This improves the global SAH cost beyond top-down construction, but similar to the previously discussed methods, this is not guaranteed to find the optimum.

3.5 Final Notes

The discussed alternative heuristics each address individual weaknesses of SAH. While these all lead to improved performance, some are limited with regard to the extensiveness of performance tests, and the heuristics are not evaluated against each other.

The nature of the greedy evaluation method has not evaluated extensively either. While the other discussed methods can achieve lower than the construction algorithm of Listing 2, they still do not guarantee the escape of local minima, and the exact distance from the global optimum remains unknown.

4 Non-greedy SAH

In this section, the potential improvement of using a non-greedy evaluation method for calculating the lowest SAH-cost BVH is experimented with and analyzed quantitatively.

As discussed previously, a cause for non-optimal BVH performance can be the greedy nature of SAH-based construction methods. The evaluation of SAH in a non-greedy manner has been briefly discussed in the thesis by Havran [27], who concludes that there is no known polynomial-time algorithm to find a global optimum. The only known method is a combinatorial evaluation of all split-plane candidates for every axis. Additionally, with a SAH-based termination condition, all potential termination criteria must be similarly evaluated. The work by Ng and Trifonov shows that the lower bound for the number of split-plane candidates to evaluate lies is exponential for only one axis. Consequently, even evaluating scenes with few primitives can require significant computation time. To explore the impact of greediness, our research focuses on scenes with very few primitives with high numbers of different configurations to measure the impact of the greedy approach on a small scale. This can give insight into the potential gains a non-greedy algorithm may yield in practice. Additionally, a measurable difference for a low number of primitives and the near-linear relation between the number of primitives and SAH in a scene, then such differences should only increase.

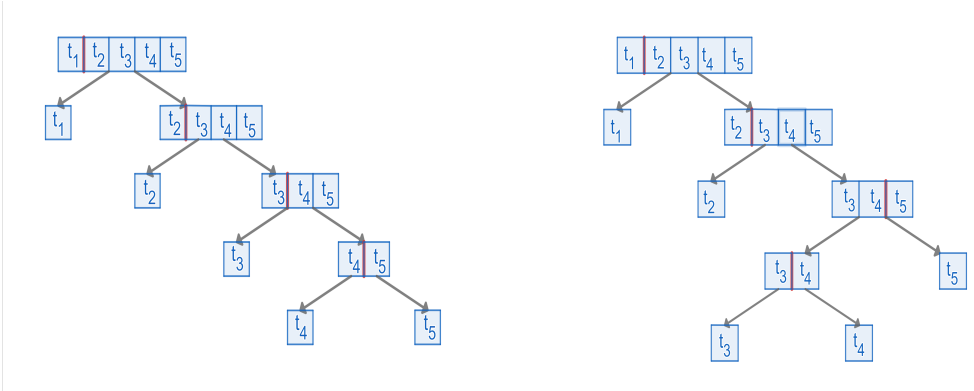
4.1 Implementation

Our implementation for iterating all potential split plane combinations starts with an initial configuration: a BVH in which every split plane is placed between the first and second primitive of the sorted primitive subset. See Figure 7a for an example with five primitives. To find a new configuration of split planes, post-order tree traversal is used to find the first node whose' split plane position can be incremented to a new valid position. The split plane is moved, and the traversal is terminated. Valid split plane positions lie between 0 and $n - 1$ for a subset of n primitives. If the traversal encounters a node that does not have a next valid split plane position, the split plane position for this node resets to its initial configuration, 0, and the post-order traversal continues to the next node. Figure 7b shows the first new configuration of split planes found after the initial configuration in Figure 7a. Once the algorithm is back at the initial configuration of the BVH, the traversal has finished, and all options have been evaluated.

The primitives in each node are initially sorted along the x-axis, but this does not yet guarantee the discovery of all split plane combinations, as the result can be different along all canonical axes. Therefore, once a node has no next valid split position left, the axis is iterated next rather than moving on to the next post-order traversal node. Once all split plane candidates over all axes have been evaluated for a node, the traversal continues as previously described. Pseudocode for the algorithm can be found in Listing 5.

```
FindNextSplitPlaneCombination(initialConfiguration, primitives):
while nextNode = PostOrderTraversal(splitConfiguration)
    if SplitPlanePosition(firstNode) < NumberOfPrimitives(firstNode) - 1
        IncrementSplitPosition(firstNode)
        break
    else if SplitPlaneAxis(firstNode) < 2
        ResetSplitPlanePosition(firstNode)
        IncrementAxis(firstNode)
        break
    else
        ResetSplitPlanePosition(firstNode)
        ResetAxis(firstNode)
```

Listing 5: Iterate over all split plane candidates for a BVH



(a) The initial configuration **(b)** The state after the first iteration

Figure 7: Two steps of the algorithm of the split-plane exploration algorithm. Note that the object set at the root is assumed to be sorted by the evaluated axis, and for illustration purposes, only one axis is evaluated. The red marker indicates the current position of the split plane in the primitive set or subset.

At each discovered configuration, the BVH's SAH cost is calculated as per Equation 1. Only the minimum SAH cost BVH so far is kept.

4.2 Termination Criteria

Finding the non-greedy SAH build has little relation to building it greedily and no longer involves a heuristic or cost function to evaluate; the total SAH is only calculated after construction. As such, the SAH build termination criterion of Equation 5 can not be applied for non-greedy evaluation. The other discussed termination criterion of stopping the subdivision when a node shrinks to below a predefined number of primitives can still be applied. However, the greedy algorithm using the cost-based termination criterion may be capable of constructing a BVH with a lower SAH-cost value than the one from the non-greedy approach.

The non-greedy algorithm should, therefore, also iterate every possible threshold for the number of primitives in a leaf in the same manner as is done for the evaluation of axes. The number of combinations increases similarly, but instead with a $n - 1$ recurrence, where n is the number of primitives in this subtree. This also leads to evaluating configurations unlikely to yield a decrease in the total SAH cost for many scenes. As an example, for a scene of a thousand primitives in which greedy SAH would have at most three primitives in a leaf node,

```
PruneSplitPlaneCombinations(currentMinSAH):
    newBVH, leftNode, rightNode = nextSplitPlane()
    if SAHCost(newBVH) > currentMinSAH
        skipSplitPlanesFor(leftNode)
        skipSplitPlanesFor(rightNode)
    else
        currentMinSAH = min(SAHCost(newBVH, currentMinSAH))
```

Listing 6: Pruning subtrees with split plane candidates

not subdividing the root node will be an evaluated configuration that is severely unlikely to yield a better BVH.

To partially combat this at the cost of losing the full guarantee at finding the optimal SAH cost, we estimate the termination criterion using greedy SAH. From the greedy evaluation, the maximum number of primitives of all leaf nodes is obtained. During iteration, the termination criterion is then dynamically using the following:

$$i = min(i + 1, \underset{N \in N_L^{greedy}}{max}(T_n)) \tag{20}$$

Where i is the number of primitives per leaf, N_L^{greedy} is the set of leaf nodes in the greedily constructed BVH, and T is the number of primitives in leaf node N. This definition ensures that, at worst, the non-greedily constructed BVH's cost equals that of the greedily constructed one.

4.3 Split Plane Configuration Pruning

An immediate observation from Equation 4 is that the SAH cost of any subset of nodes is always equal to or lower than the containing tree. When a new split plane option is evaluated via the algorithm in Listing 5, two new subtree roots will be below that node with different SAH costs. Suppose the newly created subtree root nodes combined with the existing nodes outside these subtrees have a SAH cost higher than the minimal total SAH cost found so far. In that case, a better SAH cost will never be found by evaluating the candidate split planes in the child nodes of these newly created subtrees. Therefore, configurations of the current BVH containing these newly induced nodes in it do not require further evaluation and are pruned by advancing the iteration of Listing 5. Pseudocode for this pruning algorithm is in Listing 6.

Another observation is that a BVH constructed using greedy evaluation will be amongst the evaluated BVHs for non-greedy evaluation. As the greedily evaluated BVH is likely of low-cost compared to arbitrary configurations encountered during non-greedy evaluation, the initial found lowest cost can be set to the SAH cost found for the BVH constructed using greedy evaluation. As a result, candidate split planes that cannot improve to or below the SAH cost of the greedily evaluated are pruned and, therefore, never evaluated. This is illustrated in Figure 8. In this example, if the SAH cost of nodes $N_{1..6}$ summed, as per Equation (1), exceeds the cost of any previously found BVH, the nodes containing t_1 and t_2 are pruned from further evaluation and the next split plane is found using Listing 5 starting from node N_5.

A downside of this method of pruning of candidate split planes is the inconsistency, as the number of branches of split plane combinations pruned depends on the calculated SAH costs during the traversal. This makes it heavily dependent on the scene that it is used for and it is currently unknown how many of the potential split plane candidates may be pruned. However, the end result remains the same and the number of combinations of split planes never increases.

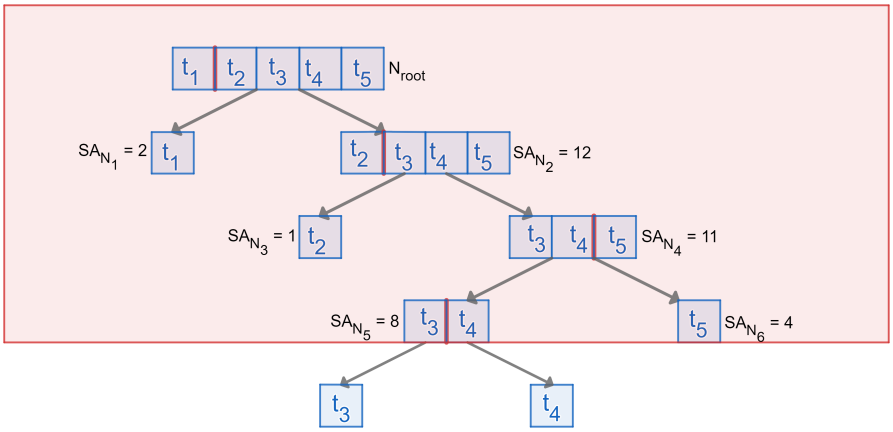


Figure 8: An overview of how the branches are pruned. The SAH of the nodes in the red marked area is summed, as per Equation (1). If these nodes' combined SAH cost exceeds the best-known SAH cost, the nodes outside of the marked area are not evaluated.

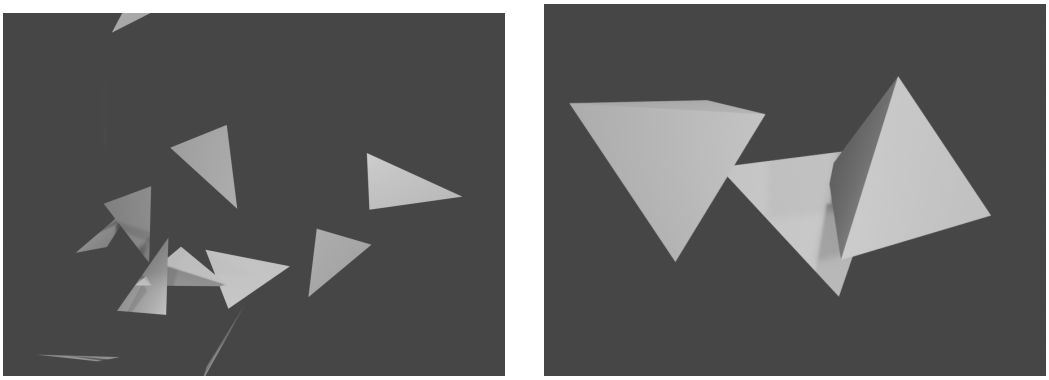
4.4 Testing Methodology

Our tests aim to find the overhead of calculating the SAH cost of a scene greedily. Despite the introduced optimizations of 4.3, the construction of the non-greedy optimal BVH for a scene with only 15 triangles was observed to require the evaluation of trillions of built BVHs and may still grow exponentially with the number of primitives, a far cry from the modern-day scenes with millions of triangles in them. However, some of the impact of the standard SAH's greediness can still be measured on this smaller scale. To do so, the construction algorithm we applied the construction algorithm to three types of scenes:

- Fully randomized triangles.
- Uniformly sized triangles at random displacement.
- Triangulated tetrahedra at random displacement.

This allows us to measure the average impact and differences in SAH cost between the greedily and non-greedily constructed BVH, resulting from regularities and irregularities in scenes.

For the fully randomized triangles, sets of 15 triangles are generated in 50 different. For each scene, the greedy and non-greedy SAH is calculated over subsets of size 3..15 of these triangle sets. This estimates the impact of greediness as the number of triangles increases. The other two types of scenes are intended to show more similarity to regular graphics test scenes, where a grid of primitives is displaced through rotation and translation, with minimal spatial overlap between these primitives. The tests consist of 25 input scenes of 12 triangles and, separately, 25 input scenes of 4 tetrahedra with an aspect ratio of one. Example configurations for these two types of scenes can be seen in Figure 9.



(a) An example configuration of randomly displaced triangles

(b) An example configuration of randomly displaced tetrahedra

Figure 9: The test scenes for the randomly displaced triangles and tetrahedra, each of 12 triangles.

For all three test types, the resulting non-greedily built BVHs are compared to a greedy, full-sweep evaluation counterpart using the algorithm described by McDonald & Booth [25].

4.5 Results and Analysis

We first want to explicitly state that this test data is limited: the number of primitives that can be processed is unlikely to lead to definitive conclusions about the impact of greediness that hold for scenes of more than the 15 triangles that were tested for. This analysis will cover such extrapolated numbers. However, until algorithmic improvements are made for non-greedy algorithms, the likelihood of this extrapolation being correct can only be speculated about.

The resulting SAH differences for the randomized triangles are displayed in Figure 10. Our first observation is that, on average, the impact of greediness is minimal for the smallest scenes: a percentual difference of approximately 0.5% for three triangles. This is as expected since the BVH will automatically be shallow in depth, which is the optimal case for the greedy algorithm that assumes the following subdivision will be the final one in this subtree. Although this difference is higher for scenes of 15 triangles at approximately 5%, this difference is still minimal. For reference, the comparison by Aila et al. [1], for example, shows that SAH differences of a factor two are not uncommon for different types of greedy-SAH builders in scenes with hundreds of thousands and up to millions of triangles. A second and more interesting observation is that the percentual difference between greedy and non-greedy evaluation appears to increase over the number of triangles. This is also expected due to the gradual increase in depth of the BVH that needs to occur, but the linear growth of this ratio shows that the difference may become significant. Following this trend, the impact on a thousand triangles would be upwards of a factor of 2.5. This resembles the measured differences in SAH cost for different builders for even larger scenes. However, a third observation from the data in this figure is that this trend may be limited. The top 95^{th} percentile of the percentual difference remains within the same bandwidth over the number of triangles. For these scenes, the greedy algorithm is limited to a less than 5% lower cost value for these particular scenes. As this measurement does not grow over the number of triangles, it may indicate that the previously described upwards trend is more limited as this upper bound and the mean would eventually cross.

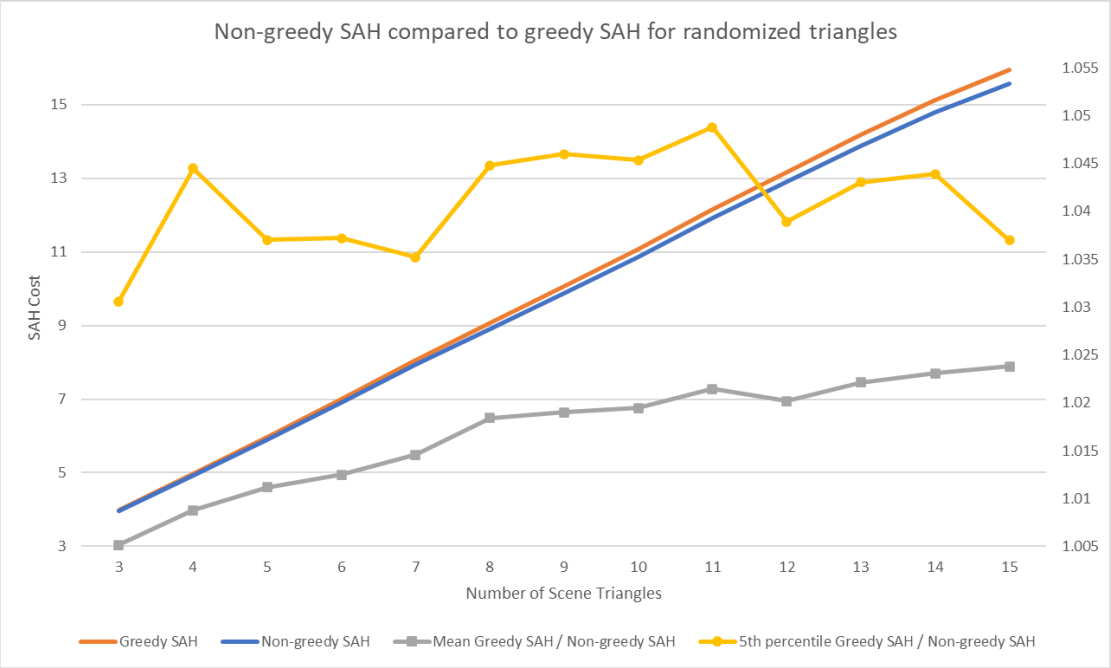
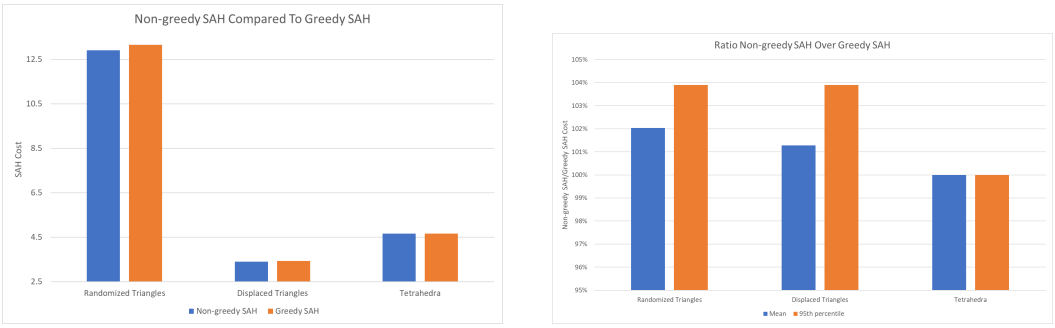


Figure 10: A comparison of SAH costs for scenes of randomized triangles between a non-greedily and greedily evaluated SAH cost for the BVH. The average and upper 95^{th} percentile of ratios of the greedy evaluation over the non-greedy evaluation cost are shown alongside it.

In Figure 11 the same differences in SAH cost are shown for the test scenes of the randomly displaced tetrahedra and triangles alongside the data for scenes of 12 fully randomized triangles as previously described. It can be observed that the impact varies. The tetrahedra scenes show no measurable impact and are significantly more representative of real scenes than the fully randomized triangles. This is an important aspect as if there is also limited scaling to the number of tetrahedra, the impact of greediness for real use cases would be much more limited than depicted by the randomized triangles. On average, the non-greedy evaluation of the randomly displaced triangle scenes shows similar improvements to that of the fully randomized triangle scenes. There may be an explanation for this difference with tetrahedra: tetrahedra are more grouped together and thus relatively easier to determine optimal partitioning for. The triangles of one tetrahedron should likely be grouped for the lowest SAH cost, which makes the decision tree of which triangles to place in a single node more shallow by default; greedy evaluation excels at this. However, given that measurable impact was observed for as few as three randomized triangles as shown in Figure 10, the lack of improvement for tetrahedra is somewhat surprising. This may indicate that more grouped and structured data may benefit significantly less from a non-greedy evaluation, as the greedy evaluation can already perform at a near-optimal level.



(a) **(b)**

Figure 11: The difference in SAH between greedy and non-greedy SAH for the randomly transformed tetrahedra and triangles

4.5.1 Pruning Results

While our research did not aim to optimize the calculation of non-greedily evaluated SAH-based BVHs to make it more feasible, the discussed pruning step allows significant reductions in the number of iterations needed to calculate the lowest SAH-cost BVH. As such, here we use this opportunity to discuss some of the observed effects briefly.

With the previously captured data for the non-greedy SAH evaluation, the number of BVHs constructed is also kept. The results for the number of BVHs that need to be evaluated for a given number of triangles in the scene are shown in Figure 12.

As observed in Figure 12, the required number of constructed BVHs shows exponential growth relative to the number of primitives without pruning, with a curve fit at $0.6875e^{2.038 \cdot n}$ where n is the number of primitives and an r^2 of 0.999 for the prediction. This matches the expectations set and deduced by Ng & Trifonov [39]. Split plane pruning, on average, shows some reduction of the number of constructed BVHs, but only minimally on average. Some outliers show a much more substantial reduction, such as the minimum for 14 triangles, which reduces an estimated 1.4 trillion evaluation to only 2.1 million. Such a reduction is not shown for subsets of the triangle sets at $m < 14$, but the random nature may cause an added triangle to significantly increase the SAH cost and lead to large subtrees of split plane configurations being pruned early. These are considered coincidences for random scenes, but it still shows the potential of the reduction.

The data for tetrahedra and displaced triangles in Figure 13 shows noticeably more impact on average. Especially for displaced triangles, this average reduction is over two orders of magnitude.

This observation is difficult to analyze, as the pruning solely relies on the found cost values during an iteration of split plane configurations, for which we do not know an optimum. It is also unknown beforehand whether the worst split plane configurations are attempted first, which significantly impacts the final number of BVHs that need to be built for evaluation. However, It can be concluded that, depending on the scene and how the scene primitives are structured, split plane pruning is an effective optimization for non-greedy SAH evaluation.

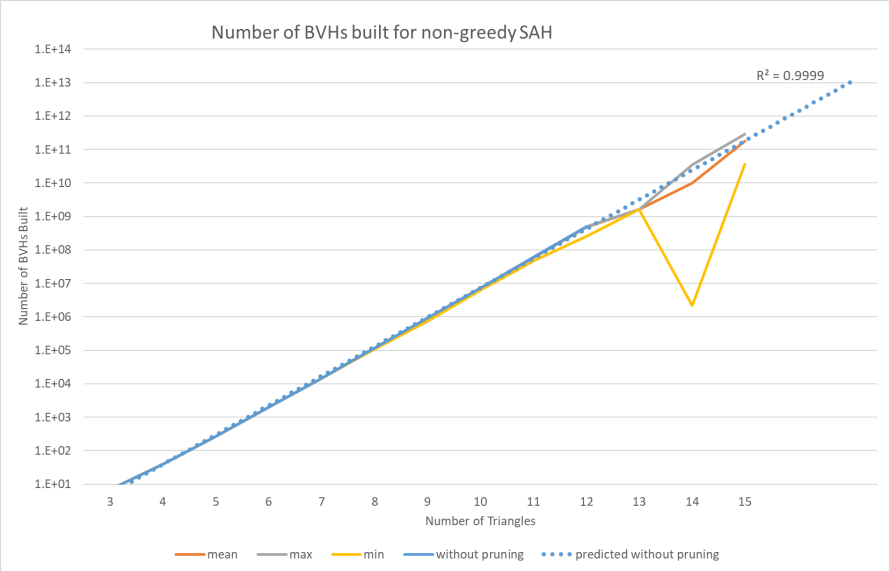


Figure 12: Effect of split plane pruning on the number of BVHs needed for evaluation over the number of triangles in the scene. Note the vertical logarithmic scale

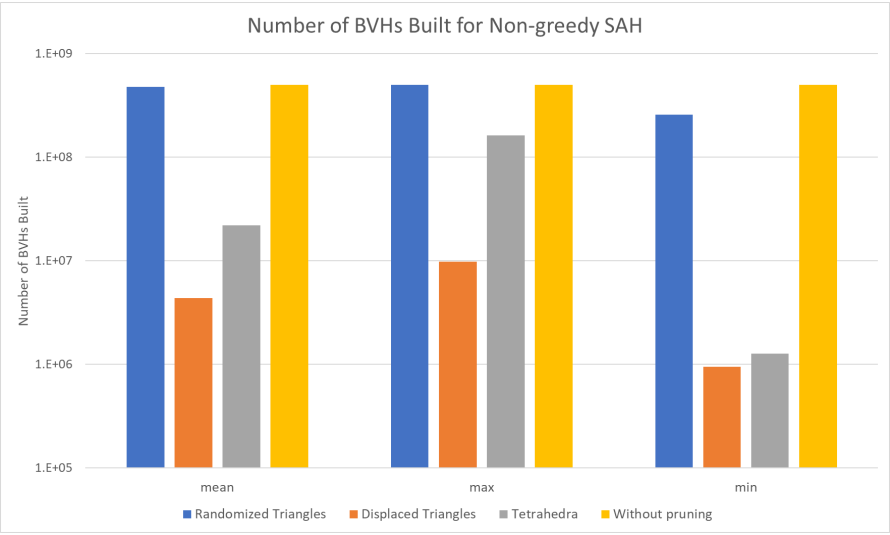


Figure 13: Effect of split plane pruning on the number of BVHs needed for evaluation for 12 triangles randomized, randomly displaced, and as tetrahedra, compared to evaluating these without pruning. Note the vertical logarithmic scale

5 Comparison of Alternative Heuristics

The previous section focused on the possible performance improvements that can be obtained from a non-greedy evaluation of the heuristic-based cost function. As discussed in Section 3, these heuristics are not limited to SAH, and a better BVH can also be obtained by using different heuristics in a greedy manner.

Each of these previously discussed heuristics has its distinct characteristics for ray-tracing performance. Though in their respective research, each presented heuristic already analyzes their performance through their own methodology, they vary in how extensively they are tested, which measurements are taken for their test scenarios, and are not tested against each other in their respective works. In short, they are tested in different frameworks. Except for shadow-ray-based heuristics, there is little to no follow-up work that extensively explores their characteristics, proposes improvements, or compares a selection of them. Especially the differences in measurements used are important; Vinkler et al. [26], for example, cover an extensive performance analysis that includes per-ray-distribution measurements and performance over animation sequences, but these are not present for the other heuristics. This section discusses a framework for comparing different heuristics that breaks down the performance measurements similar to the framework by Vinkler et al. [26].

5.1 Implementation and Testing Methodology

An existing BVH top-down construction implementation on the CPU, provided by Traverse, is used to implement the following five heuristics:

- SAH
- Scene-Interior Ray Origin Metric
- Perspective Adjustment using Preferred Ray Sets
- RDH
- OSAH

The BVH construction algorithm uses the binning approach by Wald [28] with ten bins for all measured heuristics to speed up construction. Still, evaluation over all axes is preserved as this was deemed feasible. Each heuristic uses the termination criterion in Equation 5.

Both suggested by Fabianowski et al. [29] approximations are implemented for the Scene-Interior Ray Origin Metric as they might exhibit different performance characteristics for varying ray distributions. The perspective-adjusted version of SAH and RDH using preferred ray sets both originate as heuristics for k-d trees, but both their probability functions can be trivially extended to BVHs. Similarly, the work of Vinkler et al. only implements OSAH for an SBVH, but the probability functions seem equally applicable for a normal BVH.

The suggested subsampling pattern of $4x4$ that was found by Bittner & Havran [34] to trade-off construction and run-time performance well is used. For scenes with camera animation, the RRS is either the set of rays from the starting viewpoint or, when animated, the set of rays from the current animation frame. The impact of using varying ray distributions is also evaluated by testing with only primary, shadow, AO, or both diffuse bounce rays as RRS. The used weights are set to $\alpha = 0.9$ and $\beta = 0.1$, as suggested by the authors.

The visibility information for OSAH is computed by ray tracing the scene at the same resolution as the proceeding test. No subsampling like for RDH was used, as the briefly experimented

subsampling at $4x4$ was found to impact performance noticeably, and there has been no research like that for RDH that would suggest this subsampling leads to visibility queries of sufficient quality for the BVH construction. The weight of OSAH is set to 0.9 for all scenes, as is also done by the authors.

These heuristics are tested over the following nine following test scenes:

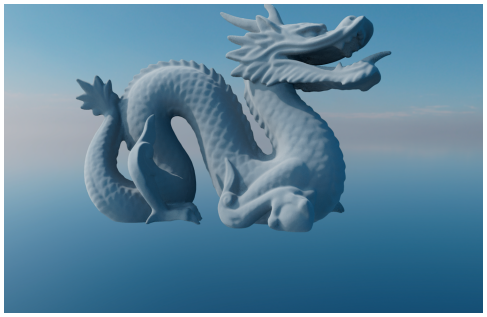
- Stanford Bunny
- Dragon
- Sibenik Cathedral
- Sponza
- Conference Room
- Soda Hall
- Bistro
- San Miguel

These scenes from the tested viewpoints, along with their triangle counts, can be seen in Figure 14.

Except for the *Stanford Bunny* and *Dragon*, the occlusion in these scenes is relatively high. This is intentional, as it is generally more challenging for SAH due to its assumption that rays do not terminate within the scene. Additionally, the measurements by Aila et al. [1] show a relatively low correlation performance, for example, for the *Soda Hall*, *San Miguel* and *Dragon* scenes. This allows for evaluating the correlation between the measured performance and the found hierarchy costs in terms of SAH and EPO, as discussed in Section 3.1.2. Our testing framework does not utilize SIMD during ray traversal, which would lead to increased performance, but introduces additional complexities. As such, the LCV metric as per Section 3.1.3 is not included, as it is not shown to improve the correlation to ray-tracing performance in Aila et al.'s tests.

Each test scene is tested from a static viewpoint and with an animated camera track that moves the camera through the scene to record data from a more extensive set of viewpoints. Except for *Conference Room* and the *Sibenik Catedral*, these viewpoints lie outside the building interiors. The camera track aims to move the camera from an outside static viewpoint into the scene and move throughout it. In theory, this would benefit heuristics that use visibility information or assume ray origins lie inside of the scene, but we argue that the recorded track for the camera represents practical use cases quite well. Each test with a moving camera is repeated six times: the first as a warm-up run in which slightly higher render times were observed for the first few frames, with the other five being the actual test runs.

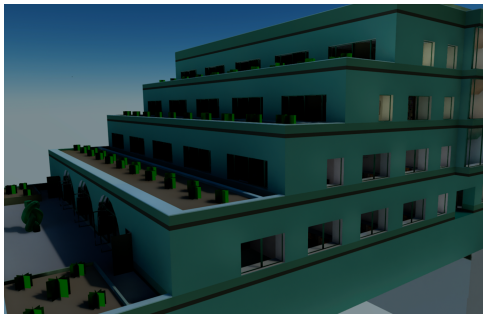
(a) Bunny, 4,968 triangles



(b) Dragon, 871,306 triangles

(c) Sponza, 262,267 triangles

(d) Conference Room, 124,631 triangles



(e) Soda Hall, 2,167,238 triangles

(f) San Miguel, 5,600,390 triangles

(g) Bistro, 3,872,281 triangles

(h) Sibenik Cathedral, 75,283 triangles

Figure 14: The scenes rendered in Traverse Research's proprietary ray-tracer from their respective static viewpoints, along with the number of triangles in each scene.

As we are also interested in the extent of temporal coherence for RRS and OSAH, these are separately evaluated for the animated scenes by testing them without rebuilding the BVH, rebuilding the BVH every three frames, and rebuilding it every frame on a reference camera animation of 10 FPS. Not rebuilding can show if the BVH can still be used reasonably well if the visibility information or RRS is stale for long periods. The other two test cases show whether there is a need to rebuild the BVH as frequently as every frame or if it suffices only to rebuild infrequently. The constructed BVH is valid in all scenarios, but rebuilding every frame may be sub-optimal for the frame in which the performance was measured.

A CPU path tracer evaluates the time per ray, traversal steps, and intersection tests for primary, shadow, diffuse, and ambient occlusion rays. Diffuse rays sample uniformly, whereas ambient occlusion is sampled according to a cosine-weighted distribution. It casts two bounces of diffuse, with per bounce one shadow ray per light and three samples of ambient occlusion rays. Diffuse and shadow rays use one sample per pixel and three samples of ambient occlusion per pixel at a radius of 5.0. As three to five lights are located in every scene, three to five shadow rays per bounce are cast. These relatively low sampling densities are compensated through longer-duration tests and allow the usage of the camera animations. Diffuse bounces are limited to two for testing purposes since the second diffuse bounces approach a random uniform distribution, as briefly illustrated by Bikker [42]. It should be noted that the randomness is not deterministic due to the multi-threaded nature of the application.

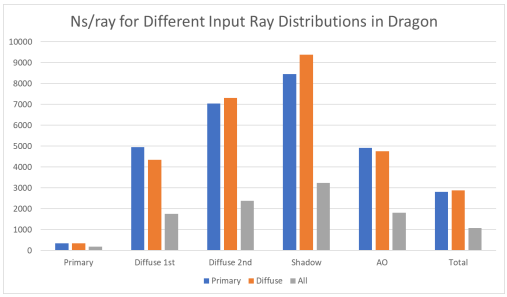
All tests were conducted on an AMD Ryzen 9 7900X 12-core processor with 64Gb of physical memory, with the path tracer utilizing all cores and never exceeding the physical memory limit. The scenes were traced at a resolution of 1280 by 720 pixels. The timings only include the time needed to traverse the BVH measured using the 'QueryPerformanceCounter' API.

5.2 Results and Analysis

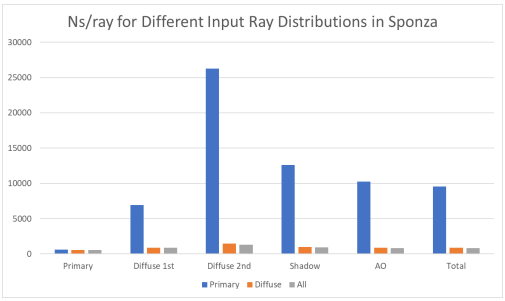
The results of the tests are individually discussed for both the static viewpoints and animated sequences. The discussion below focuses on the measurements in nanoseconds per ray, but intersection tests and traversal steps have also been measured. These measurements can be found in Appendix A.

5.2.1 Varying Ray Distributions as RRS

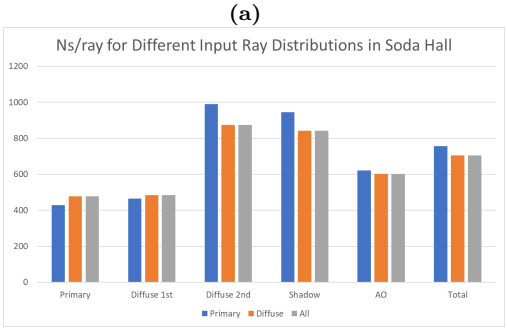
The results for evaluating RDH with varying distributions of rays in static scenes as inputs are shown in the figures in Figure 15. For most scenes, the primary rays as RRS does not build a suitable hierarchy and leads to much worse performance over all the ray distributions than when all ray distributions are used for the RRS. Using only the diffuse rays as RRS performs significantly better in all ray distributions but is still slower than using the entire set of rays. There is generally no observed benefit of using a subset of the RRS specific to the ray distribution for which the optimal ray tracing performance is desired. The only exception is *Soda Hall*, in which the performance increases slightly for primary and diffuse rays with those respectively as input RRS. As these are static captures from an outside perspective of the building, this may be due to the limited number of diffuse rays in the scene. For the full RRS, the primary rays would more likely dominate the number of diffuse rays, decreasing the relative accuracy of intersection probabilities for diffuse rays. When diffuse rays are the only rays as input, this is balanced differently. Still, the improvement is minimal compared to the overhead incurred for the other test scenes. Therefore, the remainder of our tests uses all rays from the current frame for the RRS of RDH.



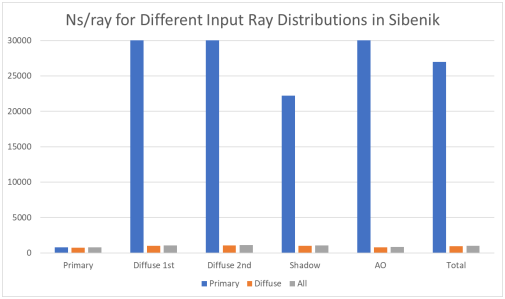
(a)



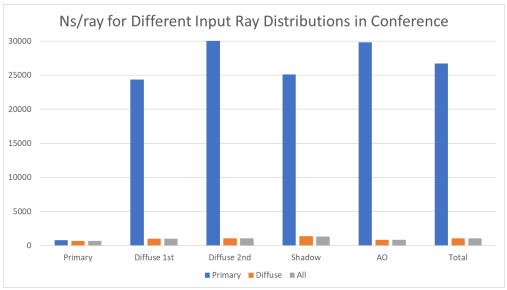
(b)



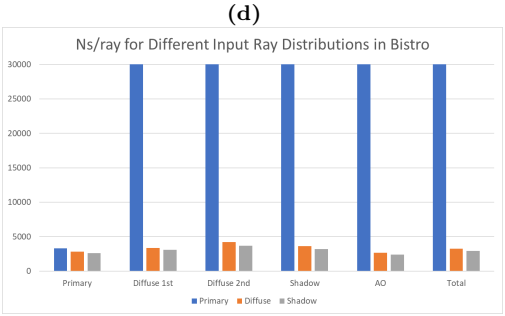
(c)



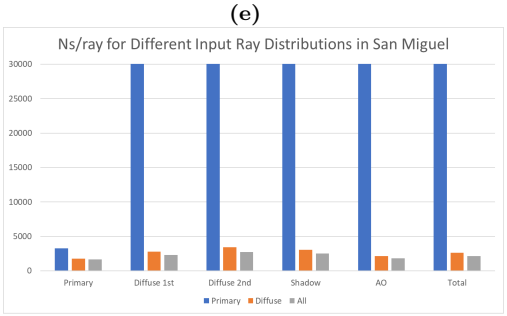
(d)



(e)



(f)



(g)

Figure 15: The differences in ray tracing performance from a single viewpoint when different input sets are provided as RRS. Each group represents the performance per ray distribution where either only primary or diffuse from the RRS or the entire RRS was used.

5.2.2 Tests for Static Viewpoints

In Table 2, the results can be seen for evaluating the time measured per ray traced through the BVH. This shows that the best heuristic depends quite heavily on the scene but shows comparable performance between most despite their different definitions for intersection probabilities. Even the fast approximation of the Scene-Interior Ray Origin Metric, denoted Int-Scene Fast, is often not far off from the highest performance.

The main outlier is SAH using the perspective adjustment using Preferred Ray Sets as discussed in Section 3.2.1, denoted as P-SAH, which can only build high-quality hierarchies for a few scenes such as *Soda Hall*. In those scenes, the performance is close to the other heuristics, specifically to SAH. Although there was no indication this heuristic would perform well for other ray distributions, for most of the scenes, even primary rays were slower by a significant margin compared to other heuristics. It generally seems to perform the worst in closed scenes with high occlusion. For *San Miguel*, no measurement could be obtained within a reasonable time as it did not manage to render a single frame. Analysis of the constructed BVH quickly showed why: it did not perform a single split. Two potential causes for the lack of performance can be identified. First, as it performs well in static scenes where the viewpoint lies outside the scene's bounding box, one problem is likely related to primary rays starting inside this bounding box. Though Bittner & Havran [33] specifically describe the edge-case where the viewpoint lies inside the bounding box, no notable differences were observed between including or excluding this, and it does not account for situations of projections of negative depth. Second, there is no description of the termination criteria used for Havran & Bittner's experiments, so it is possible that the termination criteria of Equation 5 does not extend well for the characteristics of this heuristic. This is illustrated by the lack of splitting performed for *San Miguel*.

Another outlier can be seen in Sponza for the Interior-Scene Ray Origin Metric High-Quality approximation, denoted as Int-Scene HQ. This test performs worse than the fast approximation, with primary rays being almost twice as slow. This would indicate that the fast approximation has different performance characteristics than the High-Quality version.

Finally, it can be observed that the RDH metric seems to show some of the most promising performance compared to the others, especially for shadow and ambient occlusion rays. Shadow rays are much less uniform than diffuse rays, meaning an RRS can more accurately capture the exact distribution.

Scene	Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
Bunny	primary	**186**	195	193	239	198	189
	diffuse 1st	577	583	592	706	**565**	572
	diffuse 2nd	656	**645**	662	885	676	673
	shadow	774	**752**	797	963	781	784
	ao	522	516	526	609	**511**	524
	total	446	**444**	456	546	448	450
Dragon	primary	161	**158**	164	165	181	159
	diffuse 1st	1553	1549	1674	1773	1758	**1535**
	diffuse 2nd	2053	2085	2153	2356	2389	**2012**
	shadow	2574	2606	2764	3066	3232	**2540**
	ao	1590	1582	1689	1775	1805	**1556**
	total	911	913	969	1039	1078	**896**
Sponza	primary	1281	1290	657	10036	**559**	629
	diffuse 1st	1419	1444	997	20716	**870**	991
	diffuse 2nd	1479	1497	1444	47120	**1316**	1368
	shadow	1214	1241	1085	25758	**922**	1031
	ao	1146	1164	958	23374	**842**	928
	total	1213	1234	966	22395	**838**	931
Sibenik	primary	837	816	903	2199	**776**	836
	diffuse 1st	1126	**1057**	1228	5651	1082	1060
	diffuse 2nd	1102	**1084**	1157	6319	1108	1128
	shadow	1119	1107	1153	25015	**1085**	1087
	ao	856	847	906	4613	857	**828**
	total	1021	1007	1064	15959	998	**992**
Conference	primary	716	730	745	1543	**685**	795
	diffuse 1st	**989**	1055	1092	4242	1013	1052
	diffuse 2nd	**1022**	1107	1129	5603	1050	1106
	shadow	1355	1360	1434	4628	**1347**	1427
	ao	**806**	859	885	3866	826	875
	total	**1056**	1088	1135	4182	1062	1127
Soda Hall	primary	429	**410**	500	438	478	519
	diffuse 1st	518	486	594	532	**485**	**485**
	diffuse 2nd	840	**806**	977	879	875	975
	shadow	853	**766**	878	861	842	1054
	ao	579	**544**	653	586	601	678
	total	699	**640**	750	708	704	839
San Miguel	primary	**1615**	1634	1625	N/A	1653	1693
	diffuse 1st	2467	2386	2432		**2313**	2387
	diffuse 2nd	2856	2782	2810		**2727**	2750
	shadow	2576	**2502**	2608		2505	2548
	ao	1926	1857	1906		**1807**	1882
	total	2255	2188	2256		**2162**	2219
Bistro	primary	2716	2804	2949	23946	**2614**	2791
	diffuse 1st	3192	3125	3236	129434	**3109**	3321
	diffuse 2nd	3677	**3621**	3710	183161	3660	3861
	shadow	3276	3287	3429	65859	**3223**	3510
	ao	2454	2379	2457	117251	**2376**	2563
	total	2989	2970	3088	87381	**2925**	3167

Table 2: The measured nanoseconds per ray for each scene at a single viewpoint, separated by ray distribution. The total is the total ray tracing performance per ray. For each row, the best result is in boldface. Note that San Miguel for the P-SAH could not compute a single frame

For each of these scenes, the quality of the constructed BVH is displayed in Table 3. This lists the hierarchy cost regarding SAH, EPO, and the weighted combination according to Equation 8. In the scenes for which the value is known, the α values from Aila et al. [1] are used, though it should be noted that the triangle counts are not precisely the same. The expected discrepancy of the weights is minimal, as the layout of the scenes would still match closely.

As would be expected, SAH-based construction generally achieves the lowest SAH cost. OSAH is similar to SAH for low occlusion scenes such as *Dragon* and *Bunny*, due to the low occlusion likely causing it to defer to SAH almost entirely. In the other scenes, where there is significant occlusion, OSAH leads to some of the highest SAH, if not the highest SAH cost. Also, despite not explicitly minimizing the surface area, both approximations of the scene-interior ray origin metric achieve relatively low SAH costs. SAH alone is known not to be a consistently good indicator of ray-tracing performance, but similar and even worse differences can be observed for EPO too. It is evident that these two metrics do not describe the ray-tracing performance of OSAH well. The assumption of SAH that rays do not terminate on intersection is not fully accounted for by EPO either, whereas OSAH does attempt to minimize this; perhaps another metric needs to be factored in for defining how the constructed BVH handles occlusion.

Another observation is that the fast approximation of the Scene-Interior Ray Origin Metric optimizes EPO well and, for two of the scenes, better than all other heuristics, but it is not immediately obvious why this is. It does give some explanation for the relatively good ray-tracing performance. While the α values obtained from Aila et al. may be different than the ground truth for the version of Sponza that was tested here, both EPO and SAH are separately also lower than SAH-based construction and the High-Quality approximation of the Scene-Interior Ray Origin Metric, meaning this relative outcome would remain the same regardless of the weight. Both approximations optimize SAH well, which is as expected: the solid angle is a notion of surface area adapted for a nearby origin, and as the origin moves infinitely far away, it approaches the surface area as used by SAH.

The P-SAH construction only produces builds of comparable quality for several scenes: precisely those in which the time per ray was not a multiple of the other heuristics. As shown for *San Miguel*, the termination criterion estimates that no split should be performed. It is possible that this cost-based termination criterion does not work for Preferred Ray Sets directly, as the metric also explicitly covers an edge case in which the intersection probability to 1.0 if the viewpoint lies inside the node of which the surface area is projected.

It can be concluded that heuristics such as RDH and the Interior-Scene still allow for optimization similar to SAH of these metrics that have proven to correlate well to performance but that the performance of OSAH is consistently underestimated. Additionally, the fast approximation for the Scene-Interior Ray Origin Metric clearly shows different characteristics in terms of the produced tree quality compared to the High-Quality approximation. From this perspective, the fast approximation may be well worthwhile to be evaluated for scenes that wish to optimize more for EPO, even though the SAH cost is often worse.

Scene	Cost Metric	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
Bunny	SAH	**32.2**	32.4	32.9	37.8	32.9	**32.2**
	EPO	1.2	**1.1**	1.2	1.6	1.2	1.2
Dragon	SAH	**56.4**	57.1	58.4	70.5	98.2	58.7
	EPO	**1.3**	1.3	1.5	1.8	2.4	1.4
	$SAH+EPO_{a=0.61}$	**22.8**	23.1	23.7	28.6	39.8	23.7
Sponza	SAH	**94.8**	95.6	109.4	203.4	99.5	113.2
	EPO	23.4	23.7	25.2	51.7	**22.3**	27.1
	$SAH+EPO_{a=0.84}$	34.8	35.2	38.7	76	**34.7**	40.9
Sibenik	SAH	**56.8**	56.8	64.5	131.5	59.1	94.3
	EPO	4.6	4.7	**4.4**	23.3	5	7.3
	$SAH+EPO_{a=0.76}$	**17.1**	17.2	18.8	49.2	18	28.2
Conference	SAH	**57.3**	63.7	62.1	164.7	60.8	75.2
	EPO	**7.6**	10.6	10.7	27.4	8.6	16.2
	$SAH+EPO_{a=0.42}$	**36.5**	41.4	40.5	107.1	38.9	50.4
Soda Hall	SAH	**126.2**	128.8	173.9	142.9	154.9	151.2
	EPO	40.4	38.7	**34.5**	46.8	49.1	64.8
	$SAH+EPO_{a=0.83}$	55.0	**54.1**	58.2	63.1	67.1	79.5
San Miguel	SAH	**79.4**	80.5	88.2	5600391.5	97.9	85.1
	EPO	17.1	16.9	16.2	**0.0**	20.6	22.1
	$SAH+EPO_{a=0.72}$	**34.5**	34.7	36.3	1568109.6	42.2	39.8
Bistro	SAH	148	**147.3**	149.2	500.4	157.2	175.2
	EPO	38.5	**36.3**	36.7	217.6	39.7	55.1

Table 3: The hierarchy costs in terms of SAH, EPO and the weighted combination used by Aila et al. using the indicated α weight. The weights are obtained from the work by Aila et al.

Scene	Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
Bunny	primary	**89**	95	90	94	98	**89**
	diffuse 1st	**556**	570	562	614	571	557
	diffuse 2nd	**655**	685	662	712	670	717
	shadow	805	809	823	985	814	**795**
	ao	521	517	527	558	522	**506**
	total	285	289	290	320	293	**282**
Dragon	primary	133	135	**128**	133	248	131
	diffuse 1st	**1496**	1543	1512	1667	3642	1522
	diffuse 2nd	2023	**2005**	2014	2290	4933	2059
	shadow	**2570**	2573	2609	2968	6286	2597
	ao	1550	**1530**	1552	1704	3624	1557
	total	**684**	**684**	685	759	1591	688
Sponza	primary	**1052**	1071	1161	1111	1114	1108
	diffuse 1st	**1404**	1427	1567	1975	1570	1525
	diffuse 2nd	**1523**	1533	1668	2822	1701	1658
	shadow	**1306**	1342	1532	3080	1423	1529
	ao	**1189**	1204	1299	2035	1317	1276
	total	**1248**	1270	1401	2340	1372	1381
Sibenik	primary	626	**615**	660	699	671	792
	diffuse 1st	1123	**1072**	1093	1195	1096	1170
	diffuse 2nd	1143	**1140**	1189	1363	1253	1328
	shadow	1182	1172	1177	1300	**1149**	1290
	ao	**884**	886	895	967	914	963
	total	1052	**1045**	1055	1158	1049	1154
Conference	primary	861	849	884	64927	919	**843**
	diffuse 1st	**981**	1072	1085	82508	1065	992
	diffuse 2nd	**1026**	1084	1105	75844	1096	**1026**
	shadow	**1303**	1314	1380	80025	1346	1319
	ao	**819**	867	880	66304	889	820
	total	**1046**	1078	1115	73345	1104	1053
Soda Hall	primary	884	**846**	899	38685	1155	976
	diffuse 1st	1043	**1020**	1054	52406	1264	1055
	diffuse 2nd	1100	**1090**	1113	58627	1436	1193
	shadow	1004	**941**	985	47794	1141	1085
	ao	973	**939**	949	53303	1193	1013
	total	994	**945**	977	49815	1174	1060
San Miguel	primary	1398	**1336**	1403	N/A	1914	1386
	diffuse 1st	1911	**1857**	1894		2350	1897
	diffuse 2nd	1996	**1963**	1999		2438	1971
	shadow	1961	**1895**	1965		2267	1953
	ao	1468	**1423**	1443		1719	1442
	total	1722	**1667**	1712		2028	1705
Bistro	primary	1861	1847	**1836**	313822	1986	1935
	diffuse 1st	2904	**2855**	2910	220796	2992	2926
	diffuse 2nd	3219	**3135**	3219	209207	3323	3252
	shadow	2986	**2984**	3025	168312	3065	3093
	ao	2192	**2110**	2148	157152	2273	2221
	total	2669	**2635**	2674	175099	2752	2742

Table 4: The measured nanoseconds per ray for each scene averaged over the entire camera animation, separated by ray distribution. The total is the total ray tracing performance per ray. For each row, the best result is in boldface. Note that San Miguel for the P-SAH is not included for these tests

5.2.3 Tests for Animated Camera Splines

For our animated tests, each scene is run with its distinct camera spline in the same manner as the static viewpoints, for which the results are shown in Table 4. Due to the inability to build a usable hierarchy in the previous test and the already discussed performance characteristics, the *San Miguel* test is excluded for P-SAH.

For P-SAH and the other heuristics that are view-dependent, OSAH and RDH, the BVH is only built at the initial position of the camera spline; these purely rely on the existence of temporal coherence, which is very dependent on the scene. Without such coherence, these BVHs are expected to perform worse due to the set of intersected nodes and visibility changing over time. This logically leads to SAH and the Scene-Interior Ray Origin Metric outperforming OSAH and RDH for the majority. However, OSAH is still found to be the fastest for two of the scenes. This shows that the visibility information can be reusable to a greater extent than just using that of the previous frame, as suggested by [26]. Additionally, RDH is shown to still perform well for shadow rays, despite not rebuilding the BVH for subsequent frames with new ray sets. The other differences are relatively small, and it depends from scene to scene which heuristic performs better. No silver bullet heuristic can be identified for primary or diffuse rays.

In the final two test sequences, OSAH and RDH's input visibility information and ray sets, respectively, are updated every three frames and every frame. *San Miguel* and *Bistro* are excluded due to their high build times. The results for these tests with rebuilds respectively every three frames and every frame are shown in Table 5 and Table 6.

Especially RDH benefits from rebuilding the BVH regularly, showing speed-ups for most scenes, and showing the highest performance in *Soda Hall*, at about 10% higher than the next fastest. The *Dragon* scene does produce a much worse BVH than the average when only rebuilt every three frames. The animation possibly caused specific frames to have BVHs of insufficient quality in which it had a lower number of rays in the RRS. This is to some extent in line with the observations by Havran & Bittner for ray-tracing of low-occlusion scenes, though our observation is much more extreme than the maximum decrease of 12% observed in their work. One possible explanation is the noted difference between k-d trees and BVHs; possibly, the weight function needs to be modified to account for the child nodes not being an exact union of the parent ray set.

The other results for RDH show the best ray-tracing performance for most scenes when aimed, specifically when occlusion is high. Though Bittner & Havran did not experiment with RDH path tracing for high-occlusion scenes, only tracing primary rays showed similar results to ours with significant occlusion. Additionally, the animated sequences, excluding the **Bunny** and *Dragon* scene, are either entirely or primarily rendered from the interior of the scene bounding box. The Arena v1 and v2 scenes in Havran & Bittner's original research were the only interior scenes and showed the most significant speed-up over SAH for tracing primary rays.

OSAH shows improvements as well, but these are much more minor. The fundamental difference in these two heuristics is that RDH measures the number of rays intersecting, whereas OSAH only analyzes whether any intersection for a primitive has occurred. The latter only assigns a binary visibility, compared to RDH directly assigning a probability. As the rays and viewpoints change with every frame during the animation, the probabilities may change significantly for RDH, but if the intersections for primitives and nodes only decrease while staying above zero, the evaluation will remain the same.

One scene that does show a significant improvement when regularly rebuilding is *Sibenik*. As the *Sibenik* camera animation starts at the exterior and moves inside to stay there for the remainder of the animation, the visibility initially is likely poorly estimated. For *Soda Hall*, the same type of animation sequence is implemented, but as the scene contains many gaps that allow

rays to bounce inside the building when viewed from the outside, the visibility information is likely to be much more accurate. The performance likely decreases since once inside the building, the available visibility information starts to decrease, and the camera may move somewhat quickly to places behind closed walls that may have been traced from the starting viewpoint.

Additionally, a more subtle performance increase can be observed for shadow rays. RDH has the highest shadow ray performance for nearly every scene tested, while OSAH is close to it and faster for the Dragon scene. A potential cause may be the fixed endpoints for shadow rays. This leads to specific ray distributions and accurate occlusion information, which RDH and OSAH do well at.

Scene	Ray Distribution	RDH	OSAH
Bunny	primary	93	**89**
	diffuse 1st	**553**	577
	diffuse 2nd	**644**	653
	shadow	**806**	812
	ao	**506**	510
	total	**285**	**285**
Dragon	primary	6420	**126**
	diffuse 1st	53565	**1441**
	diffuse 2nd	75727	**1977**
	shadow	67491	**2490**
	ao	58991	**1498**
	total	23615	**659**
Sponza	primary	**976**	1085
	diffuse 1st	**1327**	1457
	diffuse 2nd	**1477**	1544
	shadow	**1216**	1399
	ao	**1133**	1226
	total	**1178**	1303
Sibenik	primary	**619**	632
	diffuse 1st	**998**	1007
	diffuse 2nd	1152	**1129**
	shadow	**1076**	1113
	ao	841	**837**
	total	**975**	994
Conference	primary	**833**	865
	diffuse 1st	**979**	1070
	diffuse 2nd	**1022**	1114
	shadow	**1274**	1392
	ao	**820**	883
	total	**1032**	1120

Table 5: The measured nanoseconds per ray for each scene averaged over the entire camera animation when the BVH is rebuilt every three frames, separated by ray distribution. The total is the total ray tracing performance per ray. For each row, the best result is in boldface.

Scene	Ray Distribution	RDH	OSAH
Bunny	primary	**88**	91
	diffuse 1st	**551**	556
	diffuse 2nd	**630**	653
	shadow	**770**	799
	ao	**489**	517
	total	**274**	284
Dragon	primary	145	**125**
	diffuse 1st	1733	**1467**
	diffuse 2nd	2504	**1968**
	shadow	3200	**2486**
	ao	1793	**1474**
	total	810	**654**
Sponza	primary	**926**	1063
	diffuse 1st	**1297**	1432
	diffuse 2nd	**1432**	1533
	shadow	**1185**	1379
	ao	**1106**	1207
	total	**1147**	1283
Sibenik	primary	**551**	626
	diffuse 1st	**937**	1002
	diffuse 2nd	**1113**	1129
	shadow	**1037**	1105
	ao	**812**	833
	total	**934**	988
Conference	primary	**832**	857
	diffuse 1st	**988**	1066
	diffuse 2nd	**1035**	1101
	shadow	**1279**	1388
	ao	**826**	874
	total	**1038**	1114

Table 6: The measured nanoseconds per ray for each scene averaged over the entire camera animation when the BVH is rebuilt every three frames, separated by ray distribution. The total is the total ray tracing performance per ray. For each row, the best result is in boldface.

6 Conclusion

In this section, we separately wrap up our findings for each of our research goals, regarding the non-greedy evaluation of SAH and the testing framework for the other construction heuristics.

6.1 Non-greedy SAH Evaluation

Our initial goal was to find the impact of the greedy nature of top-down SAH-guided BVH construction for the final tree quality in terms of the resulting SAH cost. This impact is shown to be minimal, given the tested scenes. Multiple comparisons such as those by Meister & Bittner and Aila et al. [2, 1] show that compared to differences between SAH-based builders, the impact of greediness is negligible. However, the upward trend of measured greediness and SAH cost's linearity relative to the number of primitives in the scene may indicate that the difference in performance can rise to that of a similar magnitude. Even the observed differences of averages up to 3.5% and outliers at over 8% show that the de-facto greedy evaluation has measurable room for improvement. As a result of the limitation regarding the number of primitives per leaf, an even higher impact may be observable, but the expected additional impact is considered minimal.

As stated previously, it is hard to draw any definitive conclusions from this analysis due to the limited testing capabilities caused by the number of BVHs that require evaluation. Non-greedy evaluation remains largely infeasible, as even with the proposed pruning solution, it requires the evaluation of trillions of BVHs for only tiny scenes. Despite this, the pruning algorithm still shows some promise as it, on average, reduces the number of BVHs required to be evaluated for the structured scene data by two orders of magnitude and shows exceptional outliers of reductions up to a factor of six orders of magnitude.

6.2 Comparison of Alternative Heuristics

To conclude our goal of finding the most optimal BVH in terms of ray-tracing performance for the average and specific scenes and ray distributions, Section 5 compared a set of alternative heuristics to SAH. The comparison of these heuristics shows that the performance of a cost heuristic such as RDH is highly dependent on the ray distribution of the input set of rays, seemingly more significant than the impact of subsampling as found by Havran & Bittner [34]. Except for the *Soda Hall* scene, the performance is at least two times worse when only the primary or both diffuse distributions are used as input RRS. For *Soda Hall* specifically, the approach of using only primary or only the diffuse rays may be worth it as it allows for a smaller set of rays that require evaluation during construction and can slightly boost performance for the given ray distribution. However, this was only measured for a single viewpoint.

The measurements for all construction heuristics show that the difference can be relatively small but also that the best heuristic through all scenes depends on the exact scene, as can be more thoroughly identified using the results in Table 2. RDH is the fastest for a significant number of scenes, though it can perform significantly worse than other heuristics when there is little occlusion and diffuse bounces. The heuristics used generally do not excel in one particular ray distribution, although RDH shows to be slightly faster at shadow ray traversal, explained by the consistent end-points.

When the BVH is not rebuilt for every frame, OSAH performs best for all scenes averaged over all ray distributions. Standard top-down SAH does outperform OSAH for the *Conference Room* and *Soda Hall* scenes, and the Scene-Interior Ray Origin Metric even more so.

Finally, the measurements in Table 3 lead to our occlusion about the best alternative heuristic yielding the best correlation between estimated tree quality and actual ray tracing performance.

Although the respective metrics were not measured as the total cost of the tree, which is not well-defined for metrics for OSAH, the performance was weighed against the metrics by Aila et al. Outside of SAH, identifying the best heuristic through its relation with EPO and SAH, the Scene-Interior Ray Origin Metric appears to be the best choice. Still, due to the SAH builder producing the lowest SAH cost hierarchies and, compared to the other heuristics, relatively low EPO values indicate SAH as a construction heuristic to remain the best for correlation between performance and rays. It is also shown that the combined metric of EPO and SAH significantly underestimates the performance of OSAH, meaning that other metrics may need to be added to obtain an even more accurate performance correlation.

We conclude that if construction time is no factor, the best BVH for a given scene is often a BVH using RDH. In the average case, the Surface Area Heuristic and Scene-Interior Ray Origin Metric high-quality approximation perform well, especially when there is little occlusion. Still, the overall best heuristic is more scene-dependent. If the desired performance lies in shadow ray traversal, RDH and OSAH can also produce slightly better performance for scenes of high occlusion. If only a single or a limited set of viewpoints is required, the optimal heuristic for BVH for ray tracing largely depends on the scene and viewpoint for which measurements are taken. Depending on the coherency of the scene primitives, using a greedy approach may have minimal impact based on small-scale estimations.

7 Future Work

Our research lays a foundation for finding the best heuristic and evaluation of the heuristic for a given scene and scenario through the heuristic comparisons and non-greedy evaluation algorithms in our testing framework. From this starting point, we see a vast set of options for future research.

7.1 Non-greedy SAH

The non-greedy evaluation of SAH is an experiment that opens pathways for further research. This research shows a starting point for evaluating and pruning a significant set of configurations to test. However, further research may find new pruning-like methods to reduce the space of possible configurations. Faster implementations may also help, such as GPU-based iteration, but it should be noted that the current pruning algorithm relies on sequential evaluation and is harder to translate to a parallel algorithm. Our pruning in the best-case scenario leads to six orders of magnitude decrease in configurations, which would likely outperform any attempt at parallelization. We estimate other pruning methods to be more effective than parallelization, though a combination of both may be possible.

Extending the number of primitives is necessary to derive further conclusions, especially for the structured data in the form of tetrahedra. For the current numbers, the virtually non-existent impact on tetrahedra may pose implications for real test scenes in which data is commonly authored to be grouped to some degree. It should also be researched whether the approximation of the termination criterion limits the measured impact.

Additionally, as the BVH is only tested on its final cost value, the non-greedy evaluation can be applied to any of the discussed heuristics from Section 3, including EPO. Further research could determine whether the impact of greediness is different for different types of heuristic-based cost functions used and, if differences are observed, analyze the cause for such differences. A downside is that based on their definitions given in Section 3 and observations during the experiments, though not outweighing the effect of the exponentially growing number of BVHs to evaluate, the alternative heuristics can be severely more expensive to compute than for SAH. Most of the tested heuristics have efficient methods suggested for greedy evaluation that re-uses the data and often approaches the performance of calculating SAH, such as discussed in Section 3.2.2. Still, it is unknown if these can be applied to modify the BVH in global cost calculations.

Several methods have been discussed that optimize SAH beyond the greedy algorithm, such as by use of simulated annealing [41], stochastic split-plane searches [39] or partially greedy construction [40]. Adding comparisons to these methods can be used to measure the impact of these approaches compared to non-greedy evaluation and then derive better conclusions about the impact on significantly larger scenes for non-greedy SAH.

Finally, future work could also research the difference between greedy and non-greedy SAH evaluation regarding the impact on ray-tracing performance directly. A measurable difference in run-time performance could not be observed for the tested scenes due to the shallow hierarchy or limitations of the used benchmark tool's precision. If more sophisticated pruning possibilities were found, the impact could be measured for the larger scenes it would allow for testing.

7.2 The Scene-Interior Ray Origin Metric

Our results show that both approximations of this metric as proposed by Fabianowski et al. [29] perform very well in practice, though neither outperforms SAH. More interestingly, however, is the discrepancies between the low-and-high quality approximations, where the fast approximation

performs better. Since this is observed to be significant in some scenarios, future work may be able to find the parameters that lead to better performance for the fast approximation.

7.3 Impact on Massively Parallel Devices

Our tests focused on finding the best-fitting heuristic for each of the most common ray distributions, but these tests were performed on a CPU without explicit optimizations for using SIMD.

As Aila et al. show, the Leaf Count Variance can noticeably impact BVH traversal performance when instruction-level parallelism is present. As most modern real-time ray and path-tracing solutions integrate this, primarily on the GPU, testing these same heuristics in a GPU or CPU with SIMD implementation may be worthwhile.

7.4 Shadow Ray Heuristics

As outlined in Section 3, several heuristics are optimized for any-hit traversal, particularly for shadow rays. Despite being intended for only that ray distribution, multiple reasons warrant inclusion in our testing framework of heuristics-based construction methods. First, such a heuristic may not negatively impact the performance of other ray distributions, such as the diffuse and primary rays, as seen with heuristics like the Scene-Interior Ray Origin metric. Sufficiently large speed-ups for shadow rays compared to the impact on the other ray distributions mean a BVH can be used for all ray distributions regardless. Moreso, these heuristics do not seem to have been evaluated against the heuristics of this research outside of SAH. Research may also show they perform better for other ray distributions than shadow rays, for example, due to the reordering of child nodes being beneficial to other types of traversal. However, such research would also require an analysis of why this would be true.

7.5 Combining of Heuristics

Heuristics such as OSAH and RDH use SAH to prevent degenerate hierarchies in situations lacking data to build a well-performing BVH. Replacing SAH with either Scene-Interior Origin Ray Metric's approximations may lead to better results. Further specialized mixtures be researched, such as using OSAH for all rays with a weighted blend between SAH and the Interior-Scene Ray Origin Metric as a fallback. The proposed testing framework could be used to see if this leads to particular speed-ups for rays whose origins lie inside the scene.

7.6 Stochastic Evaluation of BVH Quality

The recent work by Tessari et al. [43] shows the possibility of improving BVH construction speed through initial construction over a stochastically selected subset of the input primitives. Importance sampling is used to determine the primitives that contribute most to performance. This technique leads to a factor 2 improved build times with similar quality hierarchies for top-down construction. There is potential for applying this technique to other heuristics than SAH for the initial high-quality BVH that it constructs over the stochastic subset, but we see even more potential for two different use cases. First, using the stochastic subset may allow for a reasonable measurement of the impact of greediness for scenes with higher primitive counts, as only a subset needs to be evaluated. Second, our results show that the best heuristic can be highly dependent on the scene. To select the best heuristic for a scene, this method may be suitable for constructing BVHs with different heuristics over the importance sampled subset

and efficiently finding the best heuristic to use for the scene. However, for the estimated ray-tracing performance of heuristics, the BVH quality metrics must ensure this is not severely underestimated, as observed for OSAH.

Bibliography

[1] T. Aila, T. Karras, and S. Laine. "On quality metrics of bounding volume hierarchies". In: July 2013, pp. 101–107. DOI: 10.1145/2492045.2492056.

[2] D. Meister and J. Bittner. "Performance Comparison of Bounding Volume Hierarchies for GPU Ray Tracing". In: *Journal of Computer Graphics Techniques (JCGT)* 11.4 (Oct. 2022), pp. 1–19. ISSN: 2331-7418. URL: http://jcgt.org/published/0011/04/01/.

[3] C. Cao et al. "Interactive Sound Propagation with Bidirectional Path Tracing". In: 35.6 (Dec. 2016). ISSN: 0730-0301. DOI: 10.1145/2980179.2982431.

[4] *NVIDIA RTX™ platform*. July 2020. URL: https://developer.nvidia.com/rtx.

[5] *AMD RDNA™ 2*. July 2021. URL: https://gpuopen.com/rdna2/.

[6] P. Christensen and W. Jarosz. *The Path to Path-Traced Movies*. Jan. 2016. ISBN: 9781680832112. DOI: 10.1561/9781680832112.

[7] T. Whitted. "An Improved Illumination Model for Shaded Display". In: *Commun. ACM* 23.6 (June 1980), pp. 343–349. ISSN: 0001-0782. DOI: 10.1145/358876.358882.

[8] J. T. Kajiya. "The Rendering Equation". In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '86. New York, NY, USA: Association for Computing Machinery, 1986, pp. 143–150. ISBN: 0897911962. DOI: 10.1145/15922.15902.

[9] M. Agrawala et al. "Efficient Image-Based Methods for Rendering Soft Shadows". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 375–384. ISBN: 1581132085. DOI: 10.1145/344779.344954.

[10] M. McGuire and M. Mara. "Efficient GPU Screen-Space Ray Tracing". In: *Journal of Computer Graphics Techniques (JCGT)* (Nov. 2014). URL: https://jcgt.org/published/0003/04/04/paper.pdf.

[11] A. Appel. "Some Techniques for Shading Machine Renderings of Solids". In: AFIPS '68 (Spring). Atlantic City, New Jersey: Association for Computing Machinery, 1968, pp. 37–45. ISBN: 9781450378970. DOI: 10.1145/1468075.1468082.

[12] H. Landis. "Production-Ready Global Illumination". In: *L. Gritz (Ed.), RenderMan in production: SIGGRAPH 2002 course 16*. 2002.

[13] S. Zhukov, A. Iones, and G. Kronin. "An ambient light illumination model". In: *Rendering Techniques '98*. Ed. by G. Drettakis and N. Max. Vienna: Springer Vienna, 1998, pp. 45–55. ISBN: 978-3-7091-6453-2. DOI: 10.1007/978-3-7091-6453-2_5.

[14] J. L. Bentley. "Multidimensional Binary Search Trees Used for Associative Searching". In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007.

[15] S. M. Rubin and T. Whitted. "A 3-Dimensional Representation for Fast Rendering of Complex Scenes". In: vol. 14. 3. New York, NY, USA: Association for Computing Machinery, July 1980, pp. 110–116. DOI: 10.1145/965105.807479. URL: https://doi.org/10.1145/965105.807479.

[16] H. Dammertz, J. Hanika, and A. Keller. "Shallow Bounding Volume Hierarchies for Fast SIMD Ray Tracing of Incoherent Rays". In: *Computer Graphics Forum* 27.4 (2008), pp. 1225–1233. DOI: https://doi.org/10.1111/j.1467-8659.2008.01261.x.

[17] A. S. Pinto. “Adaptive Collapsing on Bounding Volume Hierarchies for Ray-Tracing”. In: *Eurographics 2010 - Short Papers*. Ed. by H. P. A. Lensch and S. Seipel. The Eurographics Association, 2010. DOI: 10.2312/egsh.20101051.

[18] I. Wald, C. Benthin, and S. Boulos. “Getting rid of packets - Efficient SIMD single-ray traversal using multi-branching BVHs”. In: *2008 IEEE Symposium on Interactive Ray Tracing*. 2008, pp. 49–57. DOI: 10.1109/RT.2008.4634620.

[19] M. Shevtsov, A. Soupikov, and A. Kapustin. “Highly Parallel Fast KD-tree Construction for Interactive Ray Tracing of Dynamic Scenes”. In: *Comput. Graph. Forum* 26 (Sept. 2007), pp. 395–404. DOI: 10.1111/j.1467-8659.2007.01062.x.

[20] S. Popov et al. “Experiences with Streaming Construction of SAH KD-Trees”. In: *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, IEEE, 89-94 (2006)* (Sept. 2006). DOI: 10.1109/RT.2006.280219.

[21] D. Meister et al. “A Survey on Bounding Volume Hierarchies for Ray Tracing”. In: *Computer Graphics Forum* 40 (May 2021), pp. 683–712. DOI: 10.1111/cgf.142662.

[22] I. Wald and V. Havran. “On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N)”. In: *2006 IEEE Symposium on Interactive Ray Tracing*. 2006, pp. 61–69. DOI: 10.1109/RT.2006.280216.

[23] C. Lauterbach et al. “Fast BVH Construction on GPUs”. In: *Computer Graphics Forum* (2009). ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01377.x.

[24] M. Vinkler, V. Havran, and J. Bittner. “Performance Comparison of Bounding Volume Hierarchies and Kd-Trees for GPU Ray Tracing”. In: *Computer Graphics Forum* 35.8 (2016), pp. 68–79. DOI: https://doi.org/10.1111/cgf.12776.

[25] D. J. MacDonald and K. S. Booth. “Heuristics for Ray Tracing Using Space Subdivision”. In: *Vis. Comput.* 6.3 (May 1990), pp. 153–166. ISSN: 0178-2789. DOI: 10.1007/BF01911006. URL: https://doi.org/10.1007/BF01911006.

[26] M. Vinkler, V. Havran, and J. Sochor. “Visibility Driven BVH Build Up Algorithm for Ray Tracing”. In: *Computers & Graphics* 36 (June 2012). DOI: 10.1016/j.cag.2012.02.013.

[27] V. Havran. “Heuristic Ray Shooting Algorithms”. Ph.D. Thesis. Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Nov. 2000. URL: http://www.cgg.cvut.cz/~havran/phdthesis.html.

[28] I. Wald. “On Fast Construction of SAH-based Bounding Volume Hierarchies”. In: *Proceedings of IEEE Symposium on Interactive Ray Tracing 2007* (Oct. 2007), pp. 33–40. DOI: 10.1109/RT.2007.4342588.

[29] B. Fabianowski, C. Fowler, and J. Dingliana. “A Cost Metric for Scene-Interior Ray Origins”. In: (2009). Ed. by P. Alliez and M. Magnor. DOI: 10.2312/egs.20091046.

[30] M. Stich, H. Friedrich, and A. Dietrich. “Spatial splits in bounding volume hierarchies”. In: *Proceedings of the Conference on High Performance Graphics 2009 (HPG’09)* (Aug. 2009), pp. 7–13. DOI: 10.1145/1572769.1572771.

[31] M. Ernst and G. Greiner. “Early Split Clipping for Bounding Volume Hierarchies”. In: *2007 IEEE Symposium on Interactive Ray Tracing*. 2007, pp. 73–78. DOI: 10.1109/RT.2007.4342593.

[32] T. Akenine-Möller and B. Johnsson. “Performance per What?” In: *Journal of Computer Graphics Techniques (JCGT)* 1.1 (Oct. 2012), pp. 37–41. URL: http://jcgt.org/published/0001/01/03/.

[33] V. Havran and J. Bittner. “Rectilinear BSP Trees for Preferred Ray Sets”. In: 2001.

[34] J. Bittner and V. Havran. “RDH: Ray Distribution Heuristics for Construction of Spatial Data Structures”. In: SCCG ’09. Budmerice, Slovakia: Association for Computing Machinery, 2009, pp. 51–58. ISBN: 9781450307697. DOI: 10.1145/1980462.1980475.

[35] T. Ize and C. Hansen. “RTSAH traversal order for occlusion rays”. In: *Comput. Graph. Forum* 30 (Apr. 2011), pp. 297–305. DOI: 10.1111/j.1467-8659.2011.01861.x.

[36] J.-H. Nah and D. Manocha. “SATO: Surface Area Traversal Order for Shadow Ray Tracing”. In: *Computer Graphics Forum* 33 (Mar. 2014). DOI: 10.1111/cgf.12341.

[37] N. Feltman, M. Lee, and K. Fatahalian. “SRDH: Specializing BVH Construction and Traversal Order Using Representative Shadow Ray Sets”. In: *Proceedings of Conference on High Performance Graphics 2012* (June 2012), pp. 49–55. DOI: 10.2312/EGGH/HPG12/049-055.

[38] S. Ogaki and A. Derouet-Jourdan. “An N-ary BVH Child Node Sorting Technique for Occlusion Tests”. In: *Journal of Computer Graphics Techniques (JCGT)* 5.2 (June 2016), pp. 22–37. ISSN: 2331-7418. URL: http://jcgt.org/published/0005/02/02/.

[39] K. Ng and B. Trifonov. “Automatic Bounding Volume Hierarchy Generation Using Stochastic Search Methods”. In: *CPSC532D Mini-Workshop ”Stochastic Search Algorithms”* (2003).

[40] D. Wodniok and M. Goesele. “Construction of bounding volume hierarchies with SAH cost approximation on temporary subtrees”. In: *Computers & Graphics* 62 (2017), pp. 41–52. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2016.12.003.

[41] A. Kensler. “Tree Rotations for Improving Bounding Volume Hierarchies”. In: *Proceedings of the 2008 IEEE Symposium on Interactive Ray Tracing* (Sept. 2008), pp. 73–76. DOI: 10.1109/RT.2008.4634624.

[42] J. Bikker. “Ray Tracing for Real-time Games”. PhD thesis. 2012. DOI: 10.4233/uuid:a5847568-c21e-4af1-b914-5d8139efc785.

[43] L. Tessari et al. “Stochastic Subsets for BVH Construction”. In: *Computer Graphics Forum* 42 (May 2023), pp. 255–267. DOI: 10.1111/cgf.14759.

Appendices

A Additional Heuristic Measurement Data

A.1 Intersection Tests Static Scenes

Scene	Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
Bunny	primary	0.73	**0.72**	**0.72**	2.68	0.73	0.73
	diffuse 1st	4.01	4.01	**4.0**	13.85	4.06	4.01
	diffuse 2nd	4.48	4.47	**4.46**	22.67	4.55	4.47
	shadow	5.37	**5.36**	**5.36**	18.39	5.48	5.37
	ao	3.42	**3.41**	**3.41**	9.47	3.47	3.42
	total	2.81	**2.8**	2.81	9.12	2.85	2.81
Dragon	primary	0.3	0.3	0.3	0.31	1.17	**0.29**
	diffuse 1st	6.22	6.24	**6.21**	6.23	20.09	6.23
	diffuse 2nd	7.17	**7.14**	7.19	7.18	39.89	7.17
	shadow	8.51	8.51	**8.5**	8.51	46.98	8.51
	ao	**6.33**	**6.33**	6.34	6.35	25.68	6.34
	total	**3.17**	**3.17**	**3.17**	3.18	14.5	**3.17**
Sponza	primary	6.06	6.48	5.12	1067.85	4.62	**4.48**
	diffuse 1st	**7.83**	7.95	11.05	2161.04	9.85	9.4
	diffuse 2nd	**5.85**	5.94	13.41	4851.6	11.21	8.32
	shadow	**4.91**	4.96	8.67	2680.85	9.08	7.76
	ao	**4.98**	5.14	9.42	2422.98	7.63	6.66
	total	**5.28**	5.42	8.83	2329.25	7.92	6.98
Sibenik	primary	2.77	**2.76**	4.04	44.96	2.91	2.77
	diffuse 1st	4.43	**4.41**	7.19	342.58	4.61	**4.41**
	diffuse 2nd	5.1	5.1	8.07	420.7	5.35	**5.09**
	shadow	6.14	**6.11**	7.24	2394.94	6.5	6.14
	ao	3.84	3.84	6.12	317.61	4.02	**3.83**
	total	5.15	**5.13**	6.78	1466.28	5.43	5.14
Conference	primary	7.14	6.19	**4.83**	69.81	6.8	7.17
	diffuse 1st	10.78	10.08	**7.77**	305.62	10.91	10.79
	diffuse 2nd	10.76	10.09	**7.88**	434.32	10.98	10.78
	shadow	12.57	11.12	**10.45**	301.31	12	12.63
	ao	8.44	7.66	**6.01**	290.43	8.56	8.45
	total	10.37	9.3	**8.03**	292.16	10.18	10.4
Soda Hall	primary	2.08	**2.02**	10.13	2.34	2.67	2.14
	diffuse 1st	**1.47**	1.48	6.33	3.01	1.83	1.53
	diffuse 2nd	4.36	**4.35**	16.03	6.22	6.77	4.48
	shadow	4.51	**4.26**	12.21	9.31	6.07	4.74
	ao	2.93	**2.89**	10.57	3.87	3.86	3.01
	total	3.55	**3.4**	11.15	6.39	4.74	3.7
San Miguel	primary	5.29	5.26	**5.08**	N/A	7.17	5.77
	diffuse 1st	7.81	7.81	8.22		10.62	**7.7**
	diffuse 2nd	9.22	9.21	9.55		13.92	**9.05**
	shadow	**8.16**	8.21	8.53		11.84	8.31
	ao	6.58	6.58	6.77		8.74	**6.47**
	total	**7.36**	7.38	7.62		10.3	7.39
Bistro	primary	35.02	35.91	37.01	1889.39	34.63	**34.03**
	diffuse 1st	**29.14**	29.24	29.32	12142.5	30.66	29.74
	diffuse 2nd	32.83	**32.56**	33.03	17333.6	36.33	33.15
	shadow	**36.82**	39.31	40.29	6090.48	39.94	39.56
	ao	20.07	**19.73**	20.13	11119.6	21.51	20.4
	total	**30.77**	32.05	32.78	8175.53	33.1	32.35

A.2 Traversal Steps Static Scenes

Scene	Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
Bunny	primary	7.86	**7.72**	7.92	7.77	8.22	7.86
	diffuse 1st	29.29	29.2	29.53	29.32	**27.55**	29.28
	diffuse 2nd	31.72	31.61	31.89	31.81	**30.44**	31.66
	shadow	37.01	37.2	37.61	38.67	**36.02**	37.01
	ao	26.68	26.48	26.88	26.8	**25.36**	26.68
	total	21.6	21.51	21.83	21.94	**20.97**	21.6
Dragon	primary	4.03	4.11	4.04	**3.89**	4.06	4.33
	diffuse 1st	54.06	54.67	55.42	59.84	**48.25**	54.25
	diffuse 2nd	61.97	62.36	63.71	70.96	**52.8**	62.8
	shadow	72.47	73.85	74.86	89.2	**63.02**	73.15
	ao	53.42	53.86	54.91	60.41	**46.78**	53.55
	total	27.82	28.18	28.56	32.05	**24.63**	28.14
Sponza	primary	62.62	64.74	33.11	**26.97**	27.03	31.09
	diffuse 1st	60.86	62.34	48.4	**40.25**	41.94	45.93
	diffuse 2nd	57.92	58.9	61.68	**52.14**	53.96	60.24
	shadow	54.64	55.98	54.29	**42.91**	44.51	50.6
	ao	48.34	49.28	45.17	40.41	**39.1**	43.4
	total	52.78	53.97	46.76	**39.41**	39.54	44.34
Sibenik	primary	37.04	35.05	39.04	79.07	**33.73**	37.68
	diffuse 1st	39.85	40.14	40.86	80.11	**37.43**	40.7
	diffuse 2nd	39.85	39.75	40.21	73.07	**38.84**	40.56
	shadow	43.67	43.21	43.11	60.2	**40.47**	44.27
	ao	31.43	31.53	31.46	53.49	**30.4**	31.85
	total	39.06	38.78	38.89	59.88	**36.69**	39.62
Conference	primary	29.71	31.84	32.91	41.12	**28.98**	32.66
	diffuse 1st	**38.3**	43.2	44.76	48.72	39.53	39.86
	diffuse 2nd	**38.32**	43	44.06	49.45	39.75	39.89
	shadow	55.88	57.58	60.41	70.62	**55.73**	57.12
	ao	**31.22**	34.73	35.57	40.01	32.15	32.55
	total	**42.39**	45.18	46.93	54.05	42.82	43.78
Soda Hall	primary	19.09	18.41	**18.03**	18.62	20.81	23.12
	diffuse 1st	17.45	16.69	18.75	17.39	**15.1**	19.05
	diffuse 2nd	34.51	32.85	35.03	35.3	**32.62**	37.74
	shadow	43.44	**39.28**	39.43	41.85	41.25	51.23
	ao	25.12	**23.96**	24.7	24.97	24.47	28.27
	total	33.22	**30.63**	31.07	32.33	31.93	38.67
San Miguel	primary	**64.58**	65.78	64.68	N/A	65.28	68.62
	diffuse 1st	76.64	76.19	76.58		**70.34**	77.14
	diffuse 2nd	83.55	83.28	83.68		**78.25**	84.51
	shadow	82.08	80.87	83.11		**77.18**	83.75
	ao	63.09	62.57	62.52		**58.5**	64.22
	total	72.85	72.15	73.05		**68.29**	74.33
Bistro	primary	127.51	131.76	135.65	168.97	**120.85**	130.22
	diffuse 1st	119.91	118.4	120.01	130.46	**113.12**	127.06
	diffuse 2nd	131.96	129.22	130.91	**122.2**	126.47	139.18
	shadow	139.69	139.76	143.43	150.22	**132.27**	149.49
	ao	92.25	89.9	90.82	94.13	**86.35**	98.09
	total	122.4	121.73	124.35	131.05	**115.61**	130.34

A.3 Intersection Tests Animated Scenes Without Rebuilds

Scene	Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
Bunny	primary	**0.29**	**0.29**	**0.29**	0.3	0.33	**0.29**
	diffuse 1st	**4.01**	**4.01**	**4.01**	**4.01**	4.49	**4.01**
	diffuse 2nd	**4.56**	4.57	**4.56**	4.57	5.08	4.57
	shadow	5.65	5.64	5.64	**5.63**	6.29	5.64
	ao	**3.42**	**3.42**	**3.42**	3.43	3.86	**3.42**
	total	**1.75**	**1.75**	**1.75**	**1.75**	1.96	**1.75**
Dragon	primary	**0.24**	**0.24**	**0.24**	**0.24**	12.51	**0.24**
	diffuse 1st	6.1	6.1	**6.09**	**6.09**	225.54	6.1
	diffuse 2nd	7.1	7.1	7.1	7.09	278.25	**7.08**
	shadow	8.42	**8.41**	8.42	8.43	330.62	8.42
	ao	**6.24**	**6.24**	**6.24**	6.25	226.49	**6.24**
	total	**2.34**	**2.34**	**2.34**	**2.34**	90.52	**2.34**
Sponza	primary	**7.41**	7.65	13.06	7.91	18.02	7.44
	diffuse 1st	7.57	7.6	13.13	53.05	25.87	**7.49**
	diffuse 2nd	5.89	5.89	11.88	121.66	25.77	**5.77**
	shadow	**5.45**	5.46	8.66	186.85	20.53	5.46
	ao	5.18	5.21	10.23	78.75	19.33	**5.09**
	total	5.63	5.67	10.21	108.98	20.38	**5.57**
Sibenik	primary	**3.04**	3.09	7.1	3.11	5.36	3.11
	diffuse 1st	5.72	5.72	7.8	**5.6**	10.18	5.76
	diffuse 2nd	6.14	6.14	8.73	**6.01**	11.3	6.22
	shadow	7.51	7.47	8.94	**7.42**	10.92	7.52
	ao	4.61	4.62	6.98	**4.52**	8.49	4.66
	total	6.21	6.19	8.15	**6.13**	9.82	6.24
Conference	primary	10.08	7.52	**6.3**	6670.64	9.88	9.77
	diffuse 1st	9.9	9.6	**7.76**	8485.71	10.25	9.91
	diffuse 2nd	10.2	9.5	**7.71**	7816.01	10.5	10.17
	shadow	11.99	10.53	**9.84**	8215.3	11.51	12.01
	ao	8.24	7.49	**6.15**	6854.77	8.49	8.21
	total	10.11	8.99	**7.89**	7552.53	10.03	10.09
Soda Hall	primary	3.73	**3.64**	8.31	4108.76	15.05	3.81
	diffuse 1st	3.87	**3.67**	8.66	5586.32	18.66	3.92
	diffuse 2nd	4.23	**4.06**	9.96	6240.14	22.02	4.35
	shadow	4.24	**3.8**	9.21	5179.8	14.47	4.62
	ao	3.87	**3.63**	7.77	5693.73	16.12	3.91
	total	4.08	**3.74**	8.7	5360.12	15.48	4.31
San Miguel	primary	4.73	4.74	8.1	N/A	86.59	**4.7**
	diffuse 1st	6.92	**6.89**	8.83		56.36	6.95
	diffuse 2nd	7.21	**7.2**	8.95		51.34	7.26
	shadow	6.54	6.53	7.36		42.64	**6.42**
	ao	5.41	**5.37**	6.69		35.63	5.43
	total	6.01	6.0	7.26		42.92	**5.98**
Bistro	primary	19.51	19.61	**19.34**	32161.8	27.8	19.69
	diffuse 1st	28.89	**28.26**	29.21	22059	40.82	28.79
	diffuse 2nd	29.54	**28.84**	29.99	20875	43.45	29.42
	shadow	**32.14**	33.34	33.85	16897	43.67	32.7
	ao	19.52	**18.42**	19.17	15815	29.22	19.48
	total	**27.09**	27.34	27.92	17611.9	37.95	27.38

A.4 Traversal Steps Animated Scenes Without Rebuilds

Scene	Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
Bunny	primary	3.04	**3.01**	3.07	3.23	3.07	3.08
	diffuse 1st	28.65	28.82	29.1	31.02	**27.86**	28.6
	diffuse 2nd	31.97	32.13	32.41	34.87	**31.29**	31.62
	shadow	38.13	38.52	38.91	48.2	**37.19**	38.22
	ao	26.53	26.55	26.87	28.72	**25.84**	26.46
	total	13.22	13.26	13.43	15.12	**12.95**	13.24
Dragon	primary	3.26	3.3	3.32	3.61	**2.48**	3.4
	diffuse 1st	52.36	52.72	53.61	58.37	**32.61**	52.75
	diffuse 2nd	60.54	60.96	62.08	68.96	**37.4**	61.38
	shadow	71.91	73.12	74.2	88.41	**44.3**	73.04
	ao	52.43	52.72	53.7	59.16	**32.3**	52.74
	total	20.7	20.9	21.24	24.03	**13.09**	20.97
Sponza	primary	50.71	50.91	53	51.17	**47.91**	54.05
	diffuse 1st	60.35	60.48	64.21	62.29	**57.37**	66.56
	diffuse 2nd	59.41	59.46	63.07	62.18	**57.36**	67.08
	shadow	57.68	58.07	65.91	55.41	**53.76**	68.48
	ao	49.77	49.81	51.75	51.62	**47.75**	54.24
	total	53.58	53.75	57.82	54.05	**50.83**	60.3
Sibenik	primary	28.4	28.23	**28.16**	31.89	29.52	37.13
	diffuse 1st	**40.42**	40.44	40.54	48.31	41.65	47.24
	diffuse 2nd	**42.41**	42.43	42.75	51.31	44.34	50.08
	shadow	46.6	46.16	46.39	54.32	**46.02**	54.25
	ao	33.02	33.17	**32.75**	38.55	34.17	38.33
	total	40.82	**40.63**	**40.63**	47.66	41.06	47.73
Conference	primary	**36.46**	38.35	39.99	40.87	38.31	36.89
	diffuse 1st	**38.21**	44.14	45.5	40.98	40.57	39.86
	diffuse 2nd	**37.9**	42.08	43.21	39.02	39.65	38.89
	shadow	**54.12**	56.12	59.38	55.48	54.43	55.28
	ao	**31.82**	35.42	36.1	32.28	33.37	32.69
	total	**42.19**	45.16	47.05	43.37	43.27	43.21
Soda Hall	primary	42.16	40.2	**40.02**	46.71	51.05	47.25
	diffuse 1st	42.18	41.14	**39.85**	44.89	48.61	46.23
	diffuse 2nd	45.2	44.96	**42.41**	48.26	51.98	49.73
	shadow	44.71	41.56	**40.65**	46.19	49.24	51.53
	ao	40.02	38.8	**36.62**	42.37	45.95	43.8
	total	42.97	40.69	**39.32**	44.96	48.3	48.52
San Miguel	primary	60.91	**58.83**	60.49	N/A	61.38	61.97
	diffuse 1st	64.96	64.18	65		**62.53**	66.04
	diffuse 2nd	64.03	63.29	64.08		**61.44**	65.14
	shadow	67.57	65.55	68.23		**65.13**	68.89
	ao	51.37	50.64	50.88		**49.42**	51.9
	total	60.03	58.68	60.08		**57.93**	60.98
Bistro	primary	89.55	88.12	**86.72**	176.91	91.45	92.82
	diffuse 1st	115.61	**112.04**	113.66	177.84	115.06	118.6
	diffuse 2nd	120.05	**115.83**	118.09	168.78	118.56	123.5
	shadow	130.05	127.94	129.25	190.6	**127.27**	135.19
	ao	87.27	**82.83**	83.83	127.86	86.37	89.85
	total	112.88	**109.92**	111.01	168.07	111.13	116.92

A.5 Intersection Tests Animated Scenes with Rebuilds Every 3 Frames

Scene	Ray Distribution	RDH	OSAH
Bunny	primary	0.3	**0.29**
	diffuse 1st	4.1	**4.01**
	diffuse 2nd	4.66	**4.57**
	shadow	5.79	**5.64**
	ao	3.5	**3.43**
	total	1.79	**1.75**
Dragon	primary	420.74	**0.24**
	diffuse 1st	3489.6	**6.1**
	diffuse 2nd	4887.81	**7.09**
	shadow	4338.72	**8.42**
	ao	3874.39	**6.24**
	total	1538.63	**2.34**
Sponza	primary	7.7	**7.5**
	diffuse 1st	7.97	**7.62**
	diffuse 2nd	6.64	**5.93**
	shadow	6.06	**5.48**
	ao	5.67	**5.26**
	total	6.15	**5.69**
Sibenik	primary	3.15	**3.06**
	diffuse 1st	5.9	**5.71**
	diffuse 2nd	6.4	**6.13**
	shadow	7.67	**7.51**
	ao	4.79	**4.6**
	total	6.38	**6.21**
Conference	primary	9.64	**7.91**
	diffuse 1st	10.18	**9.77**
	diffuse 2nd	10.42	**9.75**
	shadow	11.82	**11.17**
	ao	8.43	**7.79**
	total	10.12	**9.43**

A.6 Traversal Steps Animated Scenes with Rebuilds Every 3 Frames

Scene	Ray Distribution	RDH	OSAH
Bunny	primary	**3.02**	3.05
	diffuse 1st	**27.79**	28.68
	diffuse 2nd	**30.65**	32.07
	shadow	**37.43**	38.25
	ao	**25.63**	26.57
	total	**12.9**	13.25
Dragon	primary	**2.77**	3.38
	diffuse 1st	**40.62**	52.19
	diffuse 2nd	**45.63**	60.9
	shadow	**54.77**	72.38
	ao	**40.08**	52.29
	total	**16.0**	20.79
Sponza	primary	**47.48**	53.01
	diffuse 1st	**57.19**	63.36
	diffuse 2nd	**57.58**	62.46
	shadow	**53.14**	62.63
	ao	**47.37**	52
	total	**50.41**	56.79
Sibenik	primary	**28.55**	28.97
	diffuse 1st	**40.46**	41.01
	diffuse 2nd	43.4	**42.93**
	shadow	**44.81**	47.12
	ao	**33.34**	33.4
	total	**40.0**	41.3
Conference	primary	**37.02**	38.98
	diffuse 1st	**39.33**	44.1
	diffuse 2nd	**39.15**	43.02
	shadow	**53.42**	59.57
	ao	**32.82**	35.85
	total	**42.46**	46.9

A.7 Intersection Tests Animated Scenes with Rebuilds Every Frame

Scene	Ray Distribution	RDH	OSAH
Bunny	primary	0.3	**0.29**
	diffuse 1st	4.08	**4.01**
	diffuse 2nd	4.64	**4.58**
	shadow	5.75	**5.64**
	ao	3.48	**3.42**
	total	1.78	**1.75**
Dragon	primary	2.67	**0.23**
	diffuse 1st	45.64	**6.08**
	diffuse 2nd	68.77	**7.09**
	shadow	76.88	**8.44**
	ao	50.48	**6.23**
	total	20.3	**2.33**
Sponza	primary	**7.47**	7.55
	diffuse 1st	7.83	**7.61**
	diffuse 2nd	6.47	**5.93**
	shadow	5.85	**5.48**
	ao	5.56	**5.27**
	total	5.99	**5.69**
Sibenik	primary	**2.91**	3.07
	diffuse 1st	**5.56**	5.71
	diffuse 2nd	6.3	**6.13**
	shadow	7.54	**7.51**
	ao	4.66	**4.6**
	total	**6.21**	**6.21**
Conference	primary	9.97	**7.89**
	diffuse 1st	10.59	**9.77**
	diffuse 2nd	10.57	**9.74**
	shadow	12.23	**11.17**
	ao	8.68	**7.79**
	total	10.45	**9.43**

A.8 Traversal Steps Animated Scenes with Rebuilds Every Frame

Scene	Ray Distribution	RDH	OSAH
Bunny	primary	**3.0**	3.05
	diffuse 1st	**27.61**	28.62
	diffuse 2nd	**30.54**	32
	shadow	**37.28**	38.17
	ao	**25.48**	26.51
	total	**12.83**	13.22
Dragon	primary	**2.83**	3.33
	diffuse 1st	**42.5**	51.99
	diffuse 2nd	**47.62**	60.73
	shadow	**57.21**	72.42
	ao	**41.89**	52.13
	total	**16.68**	20.66
Sponza	primary	**45.59**	52.54
	diffuse 1st	**56.04**	62.7
	diffuse 2nd	**57.02**	62.1
	shadow	**52.18**	62.26
	ao	**46.56**	51.67
	total	**49.46**	56.41
Sibenik	primary	**25.62**	28.82
	diffuse 1st	**38.14**	40.84
	diffuse 2nd	**42.81**	**42.81**
	shadow	**43.85**	47.0
	ao	**32.48**	33.3
	total	**38.83**	41.19
Conference	primary	**37.0**	39.05
	diffuse 1st	**39.86**	44.16
	diffuse 2nd	**39.34**	43.08
	shadow	**53.89**	59.75
	ao	**33.13**	35.89
	total	**42.83**	47

Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
primary	429	**410**	500	438	478	519
diffuse 1st	518	486	594	532	**485**	**485**
diffuse 2nd	840	**806**	977	879	875	975
shadow	853	**766**	878	861	842	1054
ao	579	**544**	653	586	601	678
total	699	**640**	750	708	704	839

Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
primary	1281	1290	657	10036	**559**	629
diffuse 1^{st}	1419	1444	997	20716	**870**	991
diffuse 2^{nd}	1479	1497	1444	47120	**1316**	1368
shadow	1214	1241	1085	25758	**922**	1031
ao	1146	1164	958	23374	**842**	928
total	1213	1234	966	22395	**838**	931

Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
primary	884	**846**	899	38685	1155	976
diffuse 1st	1043	**1020**	1054	52406	1264	1055
diffuse 2nd	1100	**1090**	1113	58627	1436	1193
shadow	1004	**941**	985	47794	1141	1085
ao	973	**939**	949	53303	1193	1013
total	994	**945**	977	49815	1174	1060

Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
primary	133	135	128	133	145	**125**
diffuse 1^{st}	1496	1543	1512	1667	1733	**1467**
diffuse 2^{nd}	2023	2005	2014	2290	2504	**1968**
shadow	2570	2573	2609	2968	3200	**2486**
ao	1550	1530	1552	1704	1793	**1474**
total	684	684	685	759	810	**654**

Cost Metric	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
SAH	**126.2**	128.8	173.9	142.9	154.9	151.2
EPO	40.4	38.7	**34.5**	46.8	49.1	64.8
SAH+EPO$_{a=0.83}$	55.0	**54.1**	58.2	63.1	67.1	79.5
ns/ray	699	**640**	750	708	704	839

Cost Metric	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
SAH	**94.8**	95.6	109.4	203.4	99.5	113.2
EPO	23.4	23.7	25.2	51.7	**22.3**	27.1
SAH+EPO$_{a=0.84}$	34.8	35.2	38.7	76	**34.7**	40.9
ns/ray	1213	1234	966	22395	**838**	931

Scene	Ray Distribution	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
Conference	primary	**716**	730	745	1543	832	857
	diffuse 1st	989	1055	1092	4242	**988**	1066
	diffuse 2nd	**1022**	1107	1129	5603	1035	1101
	shadow	1355	1360	1434	4628	**1279**	1388
	ao	**806**	859	885	3866	826	874
	total	1056	1088	1135	4182	**1038**	1114
Soda Hall	primary	429	**410**	500	438	769	1024
	diffuse 1st	518	**486**	594	532	848	1106
	diffuse 2nd	840	**806**	977	879	995	1253
	shadow	853	**766**	878	861	815	1083
	ao	579	**544**	653	586	822	1071
	total	699	**640**	750	708	823	1084

Scene	Cost Metric	SAH	Int-Scene HQ	Int-Scene Fast	P-SAH	RDH	OSAH
Soda Hall	SAH	**126.2**	128.8	173.9	142.9	154.9	151.2
	EPO	40.4	38.7	**34.5**	46.8	49.1	64.8
	$SAH+EPO_{a=0.83}$	55.0	**54.1**	58.2	63.1	67.1	79.5
San Miguel	SAH	**79.4**	80.5	88.2	5600391.5	97.9	85.1
	EPO	17.1	16.9	16.2	**0.0**	20.6	22.1
	$SAH+EPO_{a=0.72}$	**34.5**	34.7	36.3	1568109.6	42.2	39.8
Bistro	SAH	148	**147.3**	149.2	500.4	157.2	175.2
	EPO	38.5	**36.3**	36.7	217.6	39.7	55.1
Carpenter	SAH	**145.4**	146.9	154.6	269.6	152.6	160.9
	EPO	29.8	30	29.1	65.3	**28.9**	44.4