High Performance Graphics (2012)
C. Dachsbacher, J. Munkberg, and J. Pantaleoni (Editors)

SRDH: Specializing BVH Construction and Traversal Order Using Representative Shadow Ray Sets

Nicolas Feltman Minjae Lee Kayvon Fatahalian

Carnegie Mellon University

Abstract

We derive the Shadow Ray Distribution Heuristic (SRDH), an accurate cost estimator for shadow ray traversal through a bounding volume hierarchy (BVH). The SRDH leverages up-front knowledge of the distribution and intersection results of previously traced shadow rays to construct a shadow-ray-specialized BVH and choose an associated traversal order policy which together promote early termination by quickly finding occlusions. In scenes containing large amounts of occlusion, SRDH reduces the number of BVH node traversal steps needed for shadow computations between 22% and 56% compared to average-case traversal through SAH-constructed trees. Evaluating the SRDH using a sparse shadow ray set recorded from a 16×16 pixel rendering of the scene consistently produces BVHs whose traversal cost is within 6% of those built when all shadow rays are available to the metric at the time of construction. The benefits of the SRDH come at the cost of storing an additional BVH in memory and a 2.4× increase (on average) in BVH construction time.

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

1. Introduction

In this paper, we use a priori knowledge of a scene's shadow ray queries and shadow-casting geometry to reduce the cost of tracing subsequent shadow rays. Our work is built on assumptions that are similar to those of Bittner and Havran [BH09]; we expect that a small representative set of shadow rays, along with a list of intersections with scene geometry, is available at the time of acceleration structure construction. The main contributions of our work include a cost metric for traversing shadow rays through a bounding volume hierarchy (BVH) that is based on the available ray information and a BVH construction algorithm that uses this metric to simultaneously specialize BVH structure and also shadow ray traversal order to minimize this cost. That is, we do not build a high-quality acceleration structure and then determine an efficient ray traversal order; instead, our algorithm makes both sets of decisions *at the same time*. The result is a BVH that contains frequent occluders at shallow tree depths and a traversal order policy that directs shadow rays toward these occluders to achieve early termination.

We find that by co-optimizing BVH structure and traversal order using representative ray data we can reduce by up to 56% shadow ray traversal steps in scenes containing significant amounts of light occlusion. The approach is effective even when representative ray sets are sparse. For example, shadow ray data recorded from a small 16×16 pre-rendering of the scene is sufficient to achieve traversal cost within 6% (sometimes nearly identical) of that obtained when all the shadow rays to be traced are known in advance. Thus the benefits of ray tracing using the SRDH come at the expense of a modest increase in BVH construction time (on average preprocessing takes 2.4 times longer than BVH construction using the SAH) and increased memory footprint for storing the additional shadow-ray specialized BVH.

2. Background

High-performance ray tracing systems construct adaptive acceleration structures such as BVHs and K-D trees using the surface area heuristic (SAH) [GS87]. (We refer the reader to [WBS07] for a description of employing the SAH during BVH construction). SAH-based construction attempts to minimize expected traversal cost as determined by a cost metric that assumes rays are distributed uniformly in a scene and that traversal through the hierarchy does not terminate when an occlusion is found. Previous work [Hav01, HM08, FFD09] has sought to improve the accuracy of the SAH by modifying it to account for common non-uniform ray distributions. Rather than model ray distribution analytically, Bittner and Havran [BH09] use ray data logged from previous frames (or sparsely sampled from the current frame) in an attempt to obtain an accurate cost metric for the scene. Although the resulting trees yielded only modest improvements in traversal performance for radiance rays, we take inspiration from Bittner and Havran here.

We hypothesize that access to full ray information during BVH construction can yield more substantial benefits for shadow rays, rather than radiance rays (Ize and Hansen make similar comments [IH11]). Shadow rays, unlike radiance rays, only require determination if *any* intersection exists along a ray (rather than the first intersection). As a result, traversal may terminate when the first intersection is found. Thus, it is advantageous in situations of high occlusion for rays to traverse through the scene hierarchy in a manner that quickly encounters the occluders.

However, since the location of occluders is determined by both the geometric configuration of the scene and the shadow rays cast, it is not obvious how to traverse through a BVH in the direction of the most likely hit [Smi98, BH10]. Ize and Hansen [IH11] measure the performance of a number of traversal schemes (e.g., front-to-back, back-to-front, random) and report that the preferred traversal order varies depending on scene configuration.

Observing that dense regions of the scene are more likely to contain occluders, Ize and Hansen [IH11] propose a cost metric (RTSAH) that models the cost of shadow ray traversal through a existing SAH-built BVH. RTSAH uses geometric density within the volume of a subtree as an estimate of likelihood of ray hit. During traversal, rays always proceed in the direction of the minimum cost subtree, often (but not always, as we show in Section 4.1) leading to improved traversal performance.

Our work exploits actual ray data, rather than assumptions about the scene, to guide traversal decisions. Like Ize and Hansen, we derive a cost model specific to shadow ray traversal. However, our metric uses actual intersection data from a representative shadow ray set in place of geometric density estimates and also allows traversal policy to change at each node in a BVH. We then use this metric to specialize both BVH structure and traversal order for shadow rays. (We produce a BVH optimized for shadow ray traversal.)

Alternative acceleration techniques, such as caching frequent occluders [HG86] or employing low-resolution proxy geometry [DKH09, LBBS08] are also valid optimizations for shadow computations in ray tracing systems. We view these techniques as complementary to accelerating the depth-first traversal cost of shadow rays.

3. Shadow Ray Cost Metric and Cost Heuristic

In this section we describe a general algorithm for shadow ray traversal that encompasses the depth-first schemes described in Section 2. We derive a cost model for traversal that is evaluated using data from a representative shadow ray set, and then use the cost model as a basis for a cost heuristic used during greedy, top-down BVH construction.

3.1. Kernel-Guided Traversal

In depth-first traversal methods, the goal of a traversal order scheme is to guide each shadow ray to its shallowest occlusion in order to terminate the trace routine as quickly as possible. Thinking about traversal as a series of left/right decisions at each branch, we encapsulate this decision process as a *traversal-order kernel function*. Given a BVH node $\mathcal{N}$ with associated kernel $\kappa_{\mathcal{N}}$, $\kappa_{\mathcal{N}}(r)$ represents the probability that the ray r will traverse to the left child first after passing the bounding box test of $\mathcal{N}$. (The motivation for adopting a probabilistic definition will become clear in Section 3.2.) As it may be advantageous to adopt traversal-order decision logic specialized for different regions of a scene, we permit assignment of a unique kernel to each node in the BVH. Example traversal-order kernels include:

- Constant: Returns the constant p. For example, left-child-first traversal associates each node with the constant kernel $\kappa(r) = 1$. RTSAH [IH11] uses a bottom-up preprocess on a SAH tree to assign one of two constant kernels (p=0 or 1) to each node based on estimated child traversal costs.
- Front-to-back: Returns 1 if the left child center is closer to the origin of r, or 0 otherwise
- Back-to-front: Returns 1 if the left child center is farther from the origin of r, or 0 otherwise

The method `anyhit_traverse` given below performs ordered depth-first shadow ray traversal of BVH $\mathcal{N}$ in accordance with traversal kernels defined at each tree node. All traversal schemes described in Section 2 are special cases of `anyhit_traverse` for various kernel choices.

```
define anyhit_traverse($\mathcal{N}$,$r$):
  if $r$ misses bbox($\mathcal{N}$):
    return MISS
  if $\mathcal{N}$ is a leaf:
    return intersect prims in $\mathcal{N}$
  p = $\kappa_{\mathcal{N}}(r)$
  if rand() < p:
    first = left_child($\mathcal{N}$)
    second = right_child($\mathcal{N}$)
  else:
    first = right_child($\mathcal{N}$)
    second = left_child($\mathcal{N}$)
  if anyhit_traverse(first, $r$) == HIT:
    return HIT
  return anyhit_traverse(second, $r$)
```

Note that in `anyhit_traverse` ray-node bounding box tests are performed only upon traversing to a child node (rather than as part of visiting its parent branch, as is common when tracing radiance rays). A shadow ray occluded by geometry in the first sub-tree need not be tested with the bounds of the second. This key optimization of shadow ray traversal is not possible for radiance rays, which require identification of the closest ray-scene intersection.

3.2. Shadow Ray Traversal Cost Metric

We now develop a cost model for `anyhit_traverse`. To facilitate derivation, we introduce four basic {0,1}-valued functions, defined for BVH node $\mathcal{N}$ and ray r:

$M(\mathcal{N},r)$	"Miss"	r misses the bbox of $\mathcal{N}$
$I(\mathcal{N},r)$	"Pierce"	r intersects the bbox of $\mathcal{N}$
$H(\mathcal{N},r)$	"Hit"	r intersects a primitive in the subtree of $\mathcal{N}$
$F(\mathcal{N},r)$	"Faux-Hit"	r intersects the bbox of $\mathcal{N}$, but misses all primitives

Note the identity:

$$H(\mathcal{N},r)+F(\mathcal{N},r)+M(\mathcal{N},r)=I(\mathcal{N},r)+M(\mathcal{N},r)=1$$

Letting c_b and c_p be the costs of bounding box and primitive intersection tests, the cost of performing `anyhit_traverse`$(\mathcal{N},r)$, for leaf node $\mathcal{N}$ is:

$$C(\mathcal{N},r)=c_b+I(\mathcal{N},r)c_p$$

That is, we always test r with the bounding box of the leaf node. If the ray intersects the box we also test the primitive. Now let c_k be the cost of a kernel execution, and $\mathcal{N}$ be a branch node with left child $\mathcal{A}$ and right child $\mathcal{B}$. The cost when descending to $\mathcal{A}$ first is:

$$\mathcal{A}_{first}=c_b+I(\mathcal{N},r)(c_k+C(\mathcal{A},r)+(1-H(\mathcal{A},r))C(\mathcal{B},r))$$

That is, if r pierces the bounding box of $\mathcal{N}$, traversal proceeds to $\mathcal{A}$, and then to $\mathcal{B}$, provided no primitive intersection was found in $\mathcal{A}$. Symmetrically, the cost when traversing $\mathcal{B}$ first is:

$$\mathcal{B}_{first}=c_b+I(\mathcal{N},r)(c_k+C(\mathcal{B},r)+(1-H(\mathcal{B},r))C(\mathcal{A},r))$$

The expected cost of `anyhit_traverse`$(\mathcal{N},r)$ is given by combining $\mathcal{A}_{first}$ and $\mathcal{B}_{first}$ according to the probability $\kappa_{\mathcal{N}}(r)$. (We write $\overline{\kappa_{\mathcal{N}}}(r)\equiv 1-\kappa_{\mathcal{N}}(r)$ for convenience):

$$\begin{aligned}C(\mathcal{N},r)&=(\kappa_{\mathcal{N}}(r))\mathcal{A}_{first}+(\overline{\kappa_{\mathcal{N}}}(r))\mathcal{B}_{first}\\&=c_b+I(\mathcal{N},r)(c_k+C'(\mathcal{N},r))\end{aligned}$$

where,

$$\begin{aligned}C'(\mathcal{N},r)=&(1-\overline{\kappa_{\mathcal{N}}}(r)H(\mathcal{B},r))C(\mathcal{A},r)\\&+(1-\kappa_{\mathcal{N}}(r)H(\mathcal{A},r))C(\mathcal{B},r)\end{aligned}$$

Observe that $C(\mathcal{N},r)$ gives the expected cost over the left-first/right-first probability distribution defined by $\kappa_{\mathcal{N}}(r)$ for the specific ray r. The expected cost of a set of rays R is simply the sum of the cost for each ray:

$$C(\mathcal{N},R)=\sum_{r\in R}C(\mathcal{N},r)$$

And for the set $R'\subseteq R$ that pierces the bounds of $\mathcal{N}$:

$$C'(\mathcal{N},R')=\sum_{r'\in R'}C'(\mathcal{N},r')$$

3.3. Shadow Ray Distribution Heuristic (SRDH)

Consider the construction of BVH branch $\mathcal{N}$ from a collection of primitives P and a set of rays R which pierce the bounding box of P. Our goal is to estimate $C'(R,\mathcal{N})$ for the various possible constructions of $\mathcal{N}$. For instance, if we partition P into P_1 and P_2, and use the kernel κ, we seek

$$SRDH(P_1,P_2,\kappa,R)\approx C'([\mathcal{N}_{P0},\mathcal{N}_{P1},\kappa],R)$$

where $\mathcal{N}_{P0}$ and $\mathcal{N}_{P1}$ are the subtrees that will be built out of P_0 and P_1. Of course, in a top-down build process these subtrees have yet to be built, so we approximate the sub-costs as being linear in the number of contained primitives, analogously to [WH06]:

$$\begin{aligned}SRDH(P_1,P_2,\kappa,R)=\sum_{r\in R}&(1-\overline{\kappa}(r)H(P_2,r))I(P_1,r)|L_1|\\&+(1-\kappa(r)H(P_1,r))I(P_2,r)|P_2|\end{aligned}$$

We refer to the cost estimator above as the *Shadow Ray Distribution Heuristic* (SRDH). Note that the H function above is an extension of the H function from section 3.2. In this context, it indicates whether there is any intersections between a ray and set of primitives.

3.4. Building BVHs Using the SRDH

We have developed a greedy, top-down BVH build algorithm based on the SRDH. Pseudocode for the algorithm is given in the function `SRDHBuild` below. `SRDHBuild` accepts as input both a set of primitives P and a set of rays R. The set R only needs to be *representative* of the actual rays to be traced and can be quite sparse (see Section 4.2). In order to calculate the general H function, *all* intersections between rays in R and primitives in P need to be known at the time of construction, not just one per ray.

In each build step, the algorithm minimizes the SRDH by searching over both a candidate set of partitions (`partitions`, e.g., axis-aligned planes) and traversal-order kernels (`kernels`). It then determines which rays would reach the child nodes given the selected traversal scheme, and produces filtered ray sets for the subsequent child node builds. In pseudocode, hatted variables denote the best values found so far.

```
define SRDHBuild($P$,$R$):
  if $|P| <$ leaf_threshold:
    return LeafNode($P$)
/* select partition, kernel that minimizes SRDH */
  minCost = $\infty$
  foreach $(P_1,P_2) \in$ partitions($P$):
    foreach $\kappa \in$ kernels($P_1$,$P_2$):
/* revert to SAH when there are no rays */
      if $R$ is empty:
        cost = SAH($P_1$,$P_2$)
      else:
        cost = SRDH($P_1$,$P_2$,$\kappa$,$R$)
      if cost < minCost:
        minCost = cost
        $(\hat{P}_1,\hat{P}_2,\hat{\kappa})$ = $(P_1,P_2,\kappa)$
/* compute ray sets reaching child nodes */
  $R_1 = \{r \in R : \bar{\hat{\kappa}}(r)H(\hat{P}_2,r) \neq 1\}$
  $R_2 = \{r \in R : \hat{\kappa}(r)H(\hat{P}_1,r) \neq 1\}$
/* return BVH branch */
  return [SRDHBuild($\hat{P}_1$,$R_1$),
          SRDHBuild($\hat{P}_2$,$R_2$),$\hat{\kappa}$]
```

Although the kernel functions of our cost model can return any probability, the implementation above is restricted to only those kernels with the range $\{0,1\}$. Supporting unrestricted kernels would increase the complexity and runtime cost of SRDHBuild by requiring that it track the probabilities that each ray has traversed to the current subtree and account for these probabilities within the SRDH. When R contains no occluded rays, this restriction no longer applies (indeed the choice of κ becomes irrelevant to the cost), and so any kernel can be assigned. More strongly, when R is empty, we revert to standard SAH entirely. Although [BH09] employed blended heuristic under sparsity, we find that an abrupt transition is sufficient. In both of these cases, we place a uniform constant ($p = 0.5$) kernel at the branch being built.

Because the H function is sparse, it can be represented efficiently as a list per ray of all primitives intersected by that ray. Our implementation performs a low-resolution prerendering of the scene to quickly generate a representative shadow ray set R and also compute the associated intersection lists. A SAH-constructed BVH (already needed for radiance rays) expedites the precalculation.

4. Evaluation

We evaluate the SRDH using the five scenes shown in Figure 1. BEDROOM and FAIRY contain objects that occlude large regions of the scene, and thus present opportunities for significant optimization of BVH construction and traversal. MADSCI and ARCADE feature area light sources and also generate a high percentage of occluded rays. In contrast to the other four scenes, the visible geometry in SPONZA is primarily unshadowed, so SPONZA serves as a test case for low amounts of occlusion. In all experiments, scenes are rendered at 1024×1024 resolution. One level of diffuse interreflection is computed during rendering so shadow rays originate from both surfaces visible to the camera and also from surfaces reached after one diffuse bounce. The diffuse bounce results in a wider distribution of shadow ray origins in the scene and serves to create a more challenging workload for SRDH optimization.

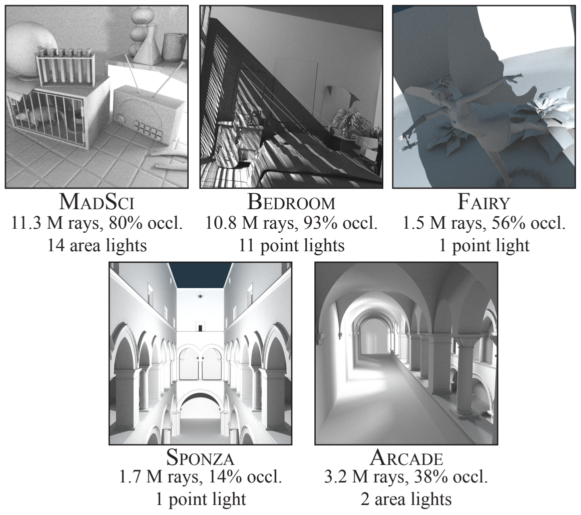


Figure 1: *Scenes used to evaluate the efficiency of BVHs built using the SRDH. Ray counts describe shadow rays only.*

We compare the cost of computing shadow ray intersections using SRDH determined BVH topology and traversal with that of four traversal policies through SAH-constructed BVHs: random ($\kappa(r) = .5$), front-to-back, back-to-front, and RTSAH. Both the SRDH and SAH trees contain exactly one primitive per leaf node, so throughout this evaluation, we use ray-bounding box intersection counts as a simple, machine-independent proxy for shadow ray tracing cost. (Counting ray-triangle intersection tests yields similar results.) All experiments utilize single-ray traversal (no packets).

4.1. SRDH Cost Comparison

Figure 2 compares the number of shadow ray-bounding box intersections performed by each of the five methods for each scene (lower values indicate lower cost). All counts are normalized to that of randomized traversal. In these experiments, the SRDH is evaluated using a ray set containing all shadow rays. Thus the red bars in the figure correspond to a "best case" scenario where complete information about the shadow rays (distribution and intersection results) is available at the time of construction.

SRDH yields the lowest total traversal cost for all scenes. In BEDROOM, where light enters through the window blinds on the left, SRDH decreases traversal cost by 56% compared to randomized traversal (41% and 44% compared to back-to-front and RTSAH traversal). Similar behavior occurs in

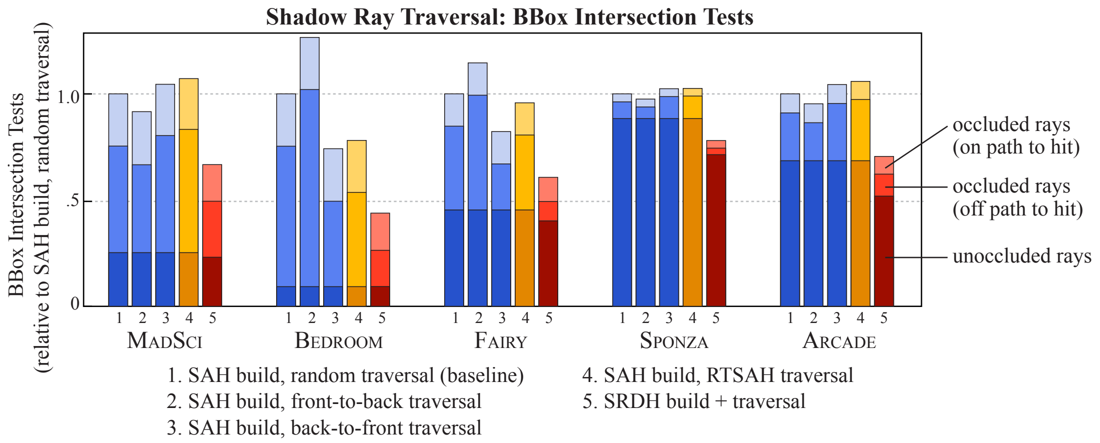


Figure 2: *Comparison of the number of ray-bounding box intersection tests performed during shadow ray traversal by five ray-tracing configurations (counts are normalized to random traversal order). SRDH generates BVHs and chooses a traversal order that results in the lowest total traversal cost for all scenes.*

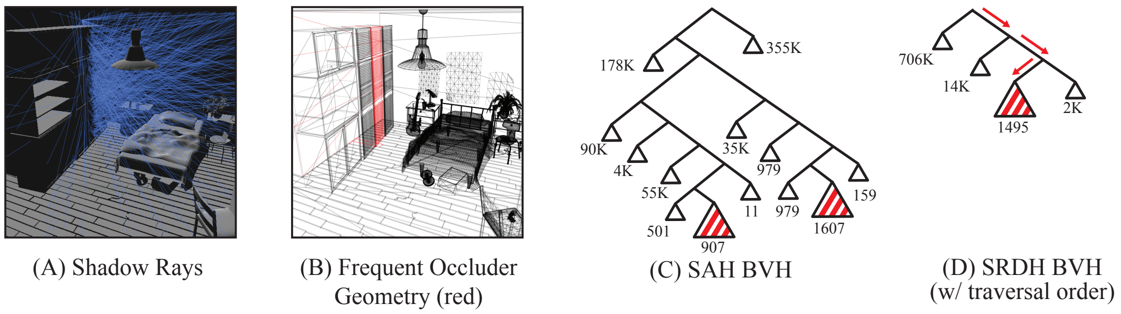


Figure 3: *Comparison of BVHs constructed for* BEDROOM *using the SAH and SRDH heuristics. (A) visualization of shadow rays in the scene. The view is chosen to aid visualization. It is not the camera position used in* BEDROOM*. (B) A large percentage of shadow rays intersect the middle set of blinds, highlighted in red. (C and D) BVHs built using the SAH and SRDH heuristics. Numbers indicate subtree node counts. Subtrees containing blinds geometry are marked in red. The SRDH tree pushes the highlighted blinds geometry towards the root and selects constant traversal kernels (red arrows) that guide rays to this subtree.*

FAIRY since the tree trunk lies between visible geometry and the light source. In BEDROOM, the SRDH identifies the blinds as the primary occluder in the scene. It produces a BVH, shown in Figure 3-D, that contains the blinds geometry in a subtree rooted at shallow depth (depth 3) and selects traversal order kernels that direct shadow rays to this region of the tree (red arrows). In contrast, SAH locates the blinds geometry deeper in the BVH and separates it into two subtrees (Figure 3-C). In the SAH-constructed BVH a majority of the blinds geometry is contained in subtrees rooted at depths 7 and 8.

To explain advantages of the SRDH tree quantitatively, we decompose the cost of ray traversal into three disjoint categories illustrated by the three segments within each vertical bar in Figure 2. The top two segments together comprise cost incurred during traversal of occluded shadow rays. The topmost segment of each bar corresponds to the cost of traversal along the path to the eventual hit, whereas the middle segment represents the cost due to traversing to nodes which do not lead to a hit (faux-hits). In comparison to other methods, SRDH shrinks the former component by placing frequent occluders near the root of the BVH. It shrinks the latter component by guiding traversal towards occluders and also keeping frequent occluders in close proximity in the tree.

The bottom section of each bar in Figure 2 gives the cost of traversing unoccluded rays. This cost is identical for all SAH-based schemes since early termination is not possible and all schemes visit the same tree nodes (traversal order does not impact the cost of unoccluded rays). The SRDH reduces unoccluded ray cost by 19% and 24% in the SPONZA and ARCADE scenes because it uses the actual distribution of rays to build a tree with higher culling efficacy (fewer faux-hits). The cost of tracing unoccluded rays dominates the overall cost of SPONZA (most rays are not

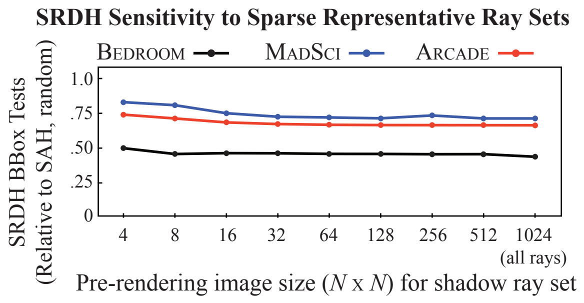


Figure 4: *SRDH output is robust to the size of the representative shadow ray set. Ray sets generated from a 16×16 pixel pre-rendering yield BVH quality comparable to that achieved when all the shadow rays are known in advance.*

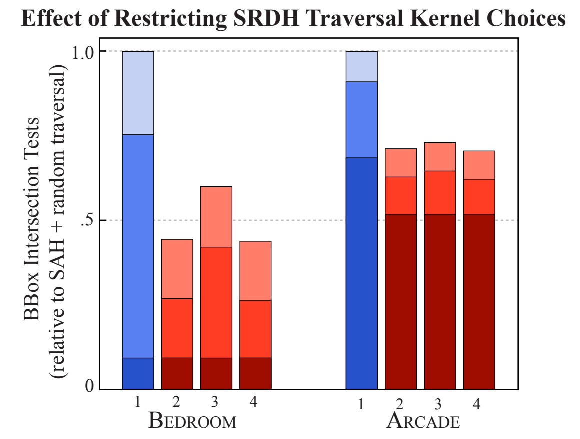
occluded) so SRDH's cost reduction for the small fraction of occluded rays does not substantially improve the overall cost of tracing shadows in the scene. As expected, the benefit realized by the SRDH on SPONZA echoes Bittner and Havran's [BH09] results for radiance rays (since in both cases early termination is not possible).

4.2. Sparse Ray Sets

Bittner and Havran observe that the performance of their ray distribution heuristic is stable even when representative ray sets are sparse [BH09]. We find that the SRDH is similarly stable. Figure 4 plots the resulting shadow tracing cost for a 1024×1024 image as the size of the pre-rendered image used to generate the representative shadow ray set is changed. As in Section 4.1, we present traversal cost relative to random traversal through a SAH tree. In all of our experiments, evaluating the SRDH using shadow rays generated from a 16×16 rendering of the scene results in shadow tracing cost within 6% of that measured when all shadow rays are used by the SRDH during BVH construction.

Intuitively, when a large fraction of shadow rays is occluded, the occluders tend to be large or densely packed. As a result, even a sparse sampling of shadow rays reveals occlusions in that region of the scene. These occlusions allow SRDH to make intelligent traversal decisions, especially in the upper levels of the BVH where an incorrect traversal decision has the potential to have the biggest cost.

Resilience of the SRDH to representative ray set sparsity is important because precomputation costs are directly proportional to ray set size. Statistics of performing SRDH-based construction using ray sets obtained from 16×16- and 32×32-pixel pre-renderings are given in Table 1. (Measurements were conducted using a single-threaded implementation of SRDH written in C#, executing on a 3.1 GHz Intel Core i5-2400.) This implementation examines up to 31 axis-aligned partitions in each spatial dimension. Recall that an SRDH build leverages a SAH-constructed BVH to efficiently compute ray-triangle intersections for the representative ray set. (This BVH is also needed to support radiance ray



1. SAH build, random traversal (baseline)
2. SRDH build (constant kernels only)
3. SRDH build (front-to-back + back-to-front kernels only)
4. SRDH build (all kernels)

Figure 5: *The benefit of including the front-to-back and back-to-front kernels (in addition to constant kernels) in the traversal order search space is negligible for our test scenes (results for BEDROOM and SPONZA are shown). A minor benefit was noticeable in contrived setups.*

computations during ray tracing.) The time needed to perform SAH BVH construction (see SAH column), as well as the additional preprocessing time to construct a shadow-ray optimized BVH using the SRDH (SRDH column) is given in the table. On average, when evaluating the SRDH using ray sets from 16×16 pre-renderings, the additional SRDH build is 1.4× longer than a traditional SAH build. Therefore, overall construction time to generate both structures is 2.4× longer than SAH construction alone (SRDH Total). MADSCI and BEDROOM contain multiple light sources and thus feature the largest ray sets. Correspondingly these scenes require the longest preprocessing times.

4.3. Kernel Selection

For the results in Figures 2 and 4, we configured `SRDHBuild` to consider four possible traversal order kernels: left-first (constant $p = 0$), right-first (constant $p = 1$), front-to-back, and back-to-front. In general, optimization selects one of the two constant kernels except in the case where two distant light sources cause fields of rays going in opposite directions, which occurs in ARCADE. We have also run experiments in which we restrict available kernels choices, shown in Figure 5. In all but our most contrived scenes, the penalty from omitting front-to-back and back-to-front kernels from the search space is negligible. Still, the existence of a sizable center bar segment in Figures 2 and 5 suggests that unnecessary traversal still occurs for occluded rays. It is possible that other kernels that we have not considered might be able to avoid these poor traversal decisions.

			Ray Set from 16 × 16 Pre-render			Ray Set from 32 × 32 Pre-render		
Scene	# Tris	SAH (s)	# Rays	SRDH (s)	SRDH Total (s)	# Rays	SRDH (s)	SRDH Total (s)
MADSCI	80149	1.7	5505	3.0	4.7 (2.8×)	22477	6.2	7.9 (4.7×)
BEDROOM	361754	7.7	4864	10.1	17.8 (2.3×)	19198	12.2	19.9 (2.6×)
FAIRY	174117	3.6	366	4.6	8.2 (2.3×)	1451	4.7	8.3 (2.3×)
SPONZA	66444	1.2	416	1.6	2.8 (2.3×)	1648	1.8	3.0 (2.4×)
ARCADE	66444	1.2	774	1.7	2.9 (2.4×)	3110	2.1	3.3 (2.7×)

Table 1: *BVH construction time statistics using shadow ray sets generated from 16×16- and 32×32-pixel pre-renderings. Constructing a shadow-ray optimized BVH using the SRDH (SRDH, 16×16 pre-render) takes 1.3 to 1.8 times longer than SAH construction (SAH). Constructing both BVHs increases build cost 2.3 to 2.8 times (SRDH Total) over SAH construction alone.*

Shadow ray BVHs employing a mixture of only constant left-first and right-first kernels (as effective as any other evaluated scheme) can be implemented with no additional space or traversal time overhead by encoding traversal order into BVH pointer structure. Always storing the first child to traverse as the left child of a node allows simple "go left first" traversal logic to implement the SRDH-determined policy.

5. Discussion

In this paper we simultaneously optimize BVH structure and traversal order to decrease the cost of tracing shadow rays. Co-optimization is possible because information about a scene's shadow rays and their occlusions is available during build. By identifying frequent occluders during the build process, we are able to produce a BVH where the path to these occluders is short and specify a traversal order that directs rays along this path. This approach reduces the number of traversal steps needed to compute shadows in our test scenes by 22% to 56%.

The cost of our approach is additional preprocessing time to construct a shadow-ray optimized BVH using the SRDH as well as the memory overhead of this additional structure. The increase in overall acceleration-structure size is at most a factor of two, but may be less if primitive data is shared among both BVHs. Our preliminary experiments show that representative ray sets can be small, and that BVH construction costs can be kept within a factor of two to three of those of a normal SAH-based build. When rendering is dominated by the cost of tracing occluded shadow rays (a frequent case in high-quality rendering [BH10]), BVH specialization using the SRDH may yield a notable performance increase.

With traversal improvements due to the SRDH, a greater fraction of total shadow ray cost is due to unoccluded rays which, like radiance rays, are more resistant to optimization [BH09]. Decreasing this component of cost will likely require breaking fundamental assumptions of the BVH, perhaps by using more varied bounding volumes or redundant decompositions. We believe our co-optimization approach can be extended to include a wider variety of existing (and future) ray-tracing acceleration techniques and that many of these strategies stand to benefit from automatic optimization over representative rays.

6. Acknowledgments

Thiago Ize, Warren Hunt, and Solomon Boulos provided valuable advice during this project. This work was supported by a gift from the Intel Corporation.

References

[BH09] BITTNER J., HAVRAN V.: RDH: ray distribution heuristics for construction of spatial data structures. In *Proceedings of the 2009 Spring Conference on Computer Graphics* (2009), SCCG '09, ACM, pp. 51–58. 1, 2, 4, 5, 6

[BH10] BOULOS S., HAINES E.: Sorted BVHs. *Ray Tracing News 23* (2010). 2, 6

[DKH09] DJEU P., KEELY S., HUNT W.: Accelerating shadow rays using volumetric occluders and modified kd-tree traversal. In *Proceedings of the Conference on High Performance Graphics 2009* (2009), HPG '09, ACM, pp. 69–76. 2

[FFD09] FABIANOWSKI B., FLOWER C., DINGLIANA J.: A cost metric for scene-interior ray origins. In *Eurographics Short Papers* (2009), pp. 49–52. 2

[GS87] GOLDSMITH J., SALMON J.: Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications 7*, 5 (May 1987), 14–20. 1

[Hav01] HAVRAN V.: *Heuristic Ray Shooting Algorithms*. PhD thesis, Czech Technical University in Prague, 2001. 2

[HG86] HAINES E., GREENBERG D.: The light buffer: A shadow-testing accelerator. *IEEE Computer Graphics and Applications 6*, 9 (Sept. 1986), 6–16. 2

[HM08] HUNT W., MARK W. R.: Ray-specialized acceleration structures for ray tracing. In *IEEE Symposium on Interactive Ray Tracing* (2008), pp. 3 –10. 2

[IH11] IZE T., HANSEN C.: RTSAH traversal order for occlusion rays. In *Computer Graphics Forum (Proceedings of Eurographics 2011)* (2011), vol. 30. 2

[LBBS08] LACEWELL D., BURLEY B., BOULOS S., SHIRLEY P.: Raytracing prefiltered occlusion for aggregate geometry. In *IEEE Symposium on Interactive Ray Tracing* (Aug. 2008), pp. 19–26. 2

[Smi98] SMITS B.: Efficiency issues for ray tracing. *Journal of Graphics Tools 3*, 2 (Feb 1998), 1–14. 2

[WBS07] WALD I., BOULOS S., SHIRLEY P.: Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Transactions on Graphics 26*, 1 (Jan. 2007). 1

[WH06] WALD I., HAVRAN V.: On building fast kd-trees for ray tracing, and on doing that in O(NlogN). In *IEEE Symposium on Interactive Ray Tracing* (2006), pp. 61–70. 3