A Cost Metric for Scene-Interior Ray Origins

Bartosz Fabianowski, Colin Fowler and John Dingliana

GV2 Group, Trinity College, Dublin, Ireland

Abstract

Acceleration structures reducing the number of intersection tests are crucial for high ray tracing performance. The best acceleration structures are currently obtained using the surface area heuristic (SAH) which greedily optimizes a cost metric based on a series of heuristic assumptions about the rays. Noting that rays in many cases originate within the scene's bounding box, we assume a uniform distribution of their origins throughout the scene and derive alternative cost metrics. Our experimental results show that at a slight increase in acceleration structure build time, an improvement in ray tracing speed over the SAH is achieved for most scenes. A more accurate metric based on our assumptions yields even more reliable speed-ups at the cost of higher construction time. We conclude that making more realistic assumptions than the SAH is a promising route for better ray tracing acceleration.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques —Graphics data structures and data types I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

Fast ray tracing requires the number of ray-primitive intersection tests to be reduced by an acceleration structure, most commonly a kd-tree or bounding volume hierarchy. The best currently known method for their construction is the surface area heuristic (SAH) [MB90]. Each candidate node is assessed with the product of the time required to process it during ray tracing and the probability of it being visited by a ray as the cost metric. A top-down construction then greedily minimizes this cost. To estimate the probability, three fundamental assumptions are made: One, ray origins are uniformly distributed outside the scene's bounding box S. Two, their directions are uniformly distributed. Three, the rays miss all primitives. Under these assumptions, the probability of a node N being visited is given by the ratio of the surface areas of N and S [KM63].

We propose that the SAH's first assumption be revised: The ray origin in many cases is actually known to be located *inside* the scene, not outside it. This is trivially the case for reflected and refracted rays as they originate from scene surfaces. For indoor scenes, camera and light sources are also commonly located within the scene's bounding box. We investigate the probability of visiting a node under this premise and derive a cost metric that improves ray tracing speed for many scenes at a slight increase in construction time.

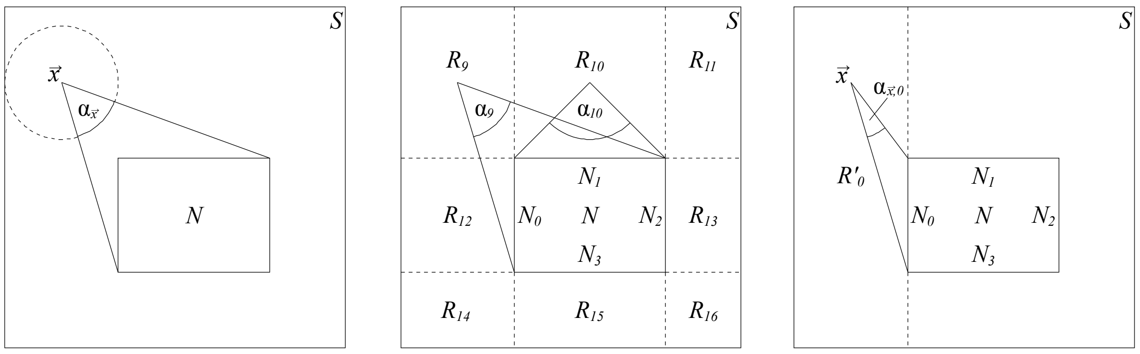
2. Related work

As the established technique for building acceleration structures [Hav00], the SAH has seen much interest in further improving its cost metric. Empty space cut-off [HKRS02] and the explicit consideration of mailboxing for intersection tests [Hun08] are examples of such work. The question of when to cease further subdivision is addressed in [HB02]. A need to also keep the construction time in mind is highlighted by the recent interest in dynamic scenes where acceleration structures need to repeatedly be rebuilt. Approximations to the SAH are being considered by some [HMS06, PGSS06]. Another method for accelerating the construction is an efficient ordering of the computations, as described in [WH06].

A thorough analysis of the cost metric is provided by [Hav00]. The author here also proposes changing the ray distribution assumptions. However, only ray distributions favoring a particular traversal direction or a single origin are considered, yielding view-dependent acceleration structures.

3. Cost metric

Top-down construction of a binary space or object hierarchy for ray tracing acceleration involves the repeated subdivision of a parent node P into two children, L and R. Simple strategies such as splitting at the object or spatial median result in relatively poor acceleration [Hav00]. The method currently



(a) For a ray originating at $\vec{x}$, the probability of visiting N is the fractional solid angle $\alpha_{\vec{x}}/4\pi$ that the node subtends.

(b) Cut through a scene showing eight of the twenty-six regions R_i surrounding N and the solid angles at two of their centers.

(c) Cut through a scene showing region R'_0 of all points outside face N_0 and the solid angle $\alpha_{\vec{x},0}$ subtended by it at one of them.

Figure 1: *Cuts through a scene bounding box S and node N illustrating the calculation and approximation of probability p_N.*

yielding the highest acceleration is the SAH, minimizing the following cost metric on each subdivision:

$$C = p_P \cdot C_T + p_L \cdot N_L \cdot C_I + p_R \cdot N_R \cdot C_I \tag{1}$$

Here, p_P, p_L, p_R are the probabilities of the nodes being visited by a ray, C_T is the traversal cost for an inner node, C_I the intersection cost for a single primitive and N_L, N_R are the numbers of primitives associated with the children. By treating the children as leaves, the effects of further recursive subdivisions may appear to be ignored. However, as shown by [Hav00], linear cost estimates are in fact valid for subhierarchies as well, confirming this as a suitable metric.

4. Geometrical probability

Use of equation 1 requires visitation probabilities to be estimated for all candidate nodes. With S the scene's bounding box, the SAH makes the following assumptions [MB90]:

- Ray origins are uniformly distributed in space *outside S*.
- Ray directions are uniformly distributed.
- No ray intersects any primitives.

Under these conditions, the probability that a ray passing through S encounters a node N is given by the ratio of their surface areas (SA) [KM63]:

$$p_N = \frac{SA(N)}{SA(S)} \tag{2}$$

This definition of p_N leads to high ray tracing acceleration. At the same time, it raises the question whether more realistic assumptions could produce even better results.

Although we do not know the actual distribution of their origins, reflected and refracted rays always begin at surfaces within the scene. For indoor scenes, camera and light sources are also commonly located inside the scene's bounding box. We thus propose that the first assumption be replaced with:

- Ray origins are uniformly distributed in space *inside S*.

A ray can then encounter a node N by either originating directly inside it or by originating anywhere else in the scene and subsequently entering N. Due to the assumption of uniformly distributed ray directions, the probability of entering N for each potential origin $\vec{x} \in S \setminus N$ is equal to the fractional solid angle $\alpha_{\vec{x}}/4\pi$ subtended by N at $\vec{x}$ (fig. 1(a)):

$$p_N = \frac{V(N)}{V(S)} + \frac{1}{V(S)} \int_{S \setminus N} \frac{\alpha_{\vec{x}}}{4\pi} d\vec{x} \tag{3}$$

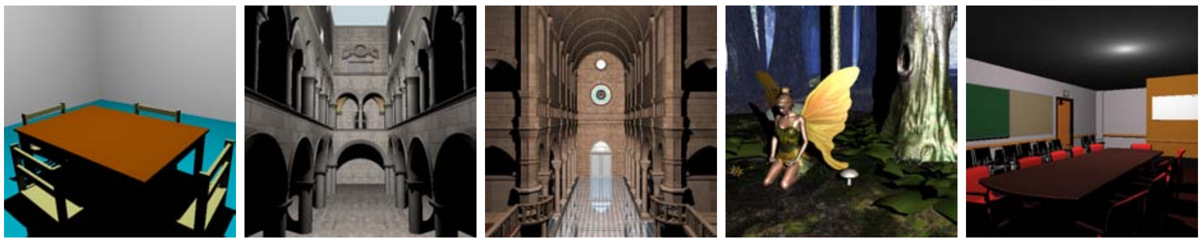
The solid angle $\alpha_{\vec{x}}$ is obtained by projecting all faces of N visible from $\vec{x}$ onto the unit sphere around it and computing the area covered. This leads to an elliptic integral for which we have no closed-form solution. We therefore present two approximations, one focusing on estimation quality, the other on evaluation speed.

4.1. High-quality approximation

For our first approximation, we extend N's faces into planes, dividing the surrounding space $S \setminus N$ into twenty-six regions (fig. 1(b)). Each region R_i contains the potential ray origins $\vec{x}$ for which $\alpha_{\vec{x}}$ is the projection of a particular subset of N's faces. For example, R_{10} contains all points from which only N_1 can be seen and R_9 those seeing both N_0 and N_1. We then replace $\alpha_{\vec{x}}$ by its value at the center of each region. Accounting for every subset of N's faces separately, we obtain a high-quality approximation:

$$p_N \approx \frac{V(N)}{V(S)} + \frac{1}{V(S)} \sum_i V(R_i) \frac{\alpha_i}{4\pi} \tag{4}$$

We compute α_i by decomposing N's faces into triangles and using the method of [OS83] to calculate the solid angles.



(a) Scene 6 (b) Sponza Atrium (c) Sibenik Cathedral (d) Fairy Forest, frame 0 (e) Conference Room

Figure 2: *Screenshots from our real-time ray tracer showing the benchmark scenes used for evaluation.*

Scene	SAH	High-quality p_N approximation					Fast p_N approximation						
	FPS	Traversals	Intersections		FPS		Traversals	Intersections		FPS		Construction time (s)	
			Plane	Triangle				Plane	Triangle				
Scene 6	382.2	-24.9%	-15.3%	-7.3%	410.8	**+7.5%**	-25.2%	-14.3%	-3.0%	409.1	**+7.0%**	0.01	+7.4%
Sponza	99.6	-0.5%	+1.1%	-12.3%	101.5	**+1.9%**	-4.3%	+25.5%	-13.7%	99.6	**0.0%**	0.96	+5.8%
Sibenik	101.0	-5.8%	-11.0%	-7.2%	103.1	**+2.0%**	-5.6%	-10.5%	-4.0%	103.1	**+2.1%**	0.97	+2.3%
Fairy	75.9	+5.5%	-5.7%	-2.8%	76.9	**+1.3%**	-1.1%	-3.9%	-6.5%	78.0	**+2.7%**	3.85	+13.0%
Conference	71.9	-13.0%	-21.4%	-6.1%	75.6	**+5.1%**	-18.6%	-17.4%	+1.4%	74.9	**+4.2%**	3.50	+7.6%
Average		-7.7%	-10.4%	-7.1%		**+3.5%**	-11.0%	-4.1%	-5.2%		**+3.2%**		+7.2%
Std. Dev.		11.8%	8.7%	3.4%		2.7%	10.4%	17.3%	5.6%		2.6%		3.9%

Table 1: *Performance characteristics of our ray tracer at* 512×512 *for the proposed* p_N *approximations, relative to the SAH. Inner node traversals and primitive intersections are counted across all rays (primary, shadow and reflection).*

4.2. Fast approximation

Equation 4 can be used to prebuild acceleration structures but is too computationally expensive for on-the-fly construction. Our second approximation aims to significantly reduce evaluation cost while maintaining better ray tracing acceleration than the SAH. We divide $S \setminus N$ into six overlapping regions R_i', each containing all potential origins $\vec{x}$ from which a face N_i can be seen (fig. 1(c)). Throughout every R_i', the fractional solid angle $\alpha_{\vec{x},i}/4\pi$ subtended by N_i should be integrated. As all solid angle calculations involve expensive trigonometric functions, we approximate the fraction as a ratio of surface areas instead:

$$p_N \approx \frac{V(N)}{V(S)} + \frac{1}{V(S)} \sum_i V\left(R_i'\right) \frac{SA(N_i)}{SA\left(R_i'\right)} \qquad (5)$$

This is equivalent to treating the ray origins as uniformly distributed in space outside R_i', in analogy to the SAH. Our choice of this estimation is based on its low cost and experimental results indicating ray tracing performance similar to that achieved by using equation 4 for many scenes.

An important property of equation 5 is the ability to reuse intermediary results. When assessing consecutive candidate nodes, the only difference is the position of one face. This affects the sizes of the four adjacent faces N_i and one region R_i' only. Updating p_N is trivial, involving two multiplications, three additions and one division. Most values can furthermore be passed from a parent node to its children, eliminating redundant calculations and yielding construction times similar to the SAH.

5. Results

We compare kd-trees built for a number of popular benchmark scenes (fig. 2) using the cost metric of equation 1 with p_N as defined by the SAH (equation 2) and our approximations (equations 4, 5). The construction is highly optimized, employing the $O(n \log n)$ method of [WH06] and empty space cut-off [HKRS02] to achieve a further increase in frame rate. All constants (empty space cut-off threshold 30%, $C_T = 1$, $C_I = 3$) are tuned to maximize ray tracing speed with the SAH and used identically in all evaluations.

Ray tracing performance is assessed using a real-time ray tracer implemented in CUDA [NVI08], supporting texture mapping, shadows from a single light source and one level of reflection. The primitive intersection code follows [Wal04].

Table 1 shows our results. All figures are obtained using a 512×512 viewport on an NVidia GeForce 280GTX, averaging over a flight through each scene. Absolute frame rates are given for our two approximations along with the relative differences to the SAH. The high-quality approximation can be seen to improve rendering speed for all benchmarked scenes by an average of 3.5%. The fast approximation is able to achieve similar speed-ups for most scenes. Only in the Sponza Atrium is the SAH's frame rate not exceeded. For this scene, the fast approximation appears not to follow the high-quality one with sufficient accuracy.

To further analyze the results, we give the number of inner node traversals and primitive intersection tests performed. As the intersection technique of [Wal04] allows for an early-out after intersecting a primitive's plane, we separately count

Scene	FPS	
Scene 6	125.8	**+8.5%**
Sponza	62.0	**-3.9%**
Sibenik	65.8	**+4.2%**
Fairy	21.7	**+0.4%**
Conference	48.8	**+5.1%**

(a) RT^2 [FC07]

Setting	FPS	
Unshaded	44.5	**+2.4%**
Unshadowed	44.4	**+2.5%**
Light outside	14.5	**-3.8%**
Light inside	14.2	**+1.7%**

(b) Radius-CUDA [Seg08]

Table 2: *Performance characteristics of two other ray tracers using the fast p_N approximation, relative to the SAH.*

the plane intersections and barycentric coordinate calculations. Both approximations lead to decreases in the number of traversals and intersections required in most cases, closely following each other. Only in the Sponza Atrium scene does the number of plane intersections significantly differ, corresponding to the lack of overall speed-up observed.

Construction speed for the high-quality approximation is on the order of minutes. We provide no exact numbers as the implementation is naïve and serves to assess the improvement in ray tracing performance only. Even with aggressive optimizations, it would remain uncompetitive due to the many trigonometric functions used. At the same time, the fast approximation is almost on par with the SAH, increasing kd-tree build time by only 7.2% on average.

To ensure the speed-ups are not specific to our implementation, we have ported the fast p_N approximation to the real-time ray tracers RT^2 [FC07] and Radius-CUDA [Seg08], the latter closely following [BAGJ08]. Both ray tracers originally used kd-trees based on the SAH for acceleration and we only replaced the computation of p_N, leaving all other code unchanged. Results are shown in table 2.

RT^2 uses static views of our benchmark scenes and experiences similar speed-ups. Only for the Sponza Atrium is there a slow-down, in line with the difficulties this scene poses in our ray tracer. Radius-CUDA renders frame 160 of the Fairy Forest animation with one animated light. Tracing only primary rays, a speed-up of 2.5% is achieved. Shadow rays lead to a slow-down. This is because the light is located outside the scene, violating our assumptions. After moving the light source inside the scene, we again obtain a speed-up.

6. Conclusions and future work

We have replaced the SAH's assumption of ray origins uniformly distributed outside the scene by a uniform distribution inside it. As the resulting probability p_N of a node being visited involves an intractable elliptic integral, we devised two approximations, one focusing on quality and the other on evaluation speed. Our results show an improvement in ray tracing performance with the high-quality approximation that the fast approximation is able to match for almost all scenes. The Sponza Atrium is the only benchmark scene for which the fast approximation yields no speed-up.

To better understand the difficulties in this scene, we plan to precisely investigate the differences between the acceleration structures generated by the SAH and our fast approximation. The large increase in plane intersections indicates that many primitives referenced by the kd-tree leaves may in fact lie outside or only skim them. Perfect splitting (clipping primitives to generated nodes) or split clipping [HB02] are interesting in this context. Another approach for approximating p_N we plan to look into is Monte Carlo evaluation of the exact solution for a large number of scenarios and fitting of a regression model. If a simple regression is found, high-quality approximations could efficiently be obtained.

7. Acknowledgements

The first two authors are funded by IRCSET under the Embark Initiative, the second additionally by Intel Corporation. Models used are courtesy of Peter Shirley (Scene 6), Marko Dabrovic (Sponza, Sibenik), Ingo Wald (Fairy Forest), Anat Grynberg and Greg Ward (Conference Room).

References

[BAGJ08] BUDGE B., ANDERSON J., GARTH C., JOY K.: *A hybrid CPU-GPU Implementation for Interactive Ray-Tracing of Dynamic Scenes*. Tech. Rep. CSE-2008-9, UC Davis, 2008. 4

[FC07] FOWLER C., COLLINS S.: Implementing the RT^2 real-time ray-tracing system. In *EG Ireland* (2007), pp. 1–8. 4

[Hav00] HAVRAN V.: *Heuristic Ray Shooting Algorithms*. PhD thesis, Czech Technical University, 2000. 1, 2

[HB02] HAVRAN V., BITTNER J.: On improving kd-trees for ray shooting. In *WSCG* (2002), pp. 209–217. 1, 4

[HKRS02] HURLEY J., KAPUSTIN A., RESHETOV A., SOUPIKOV A.: Fast ray tracing for modern general purpose CPU. In *Graphicon 2002* (2002). 1, 3

[HMS06] HUNT W., MARK W., STOLL G.: Fast kd-tree construction with an adaptive error-bounded heuristic. In *RT* (2006), pp. 81–88. 1

[Hun08] HUNT W.: Corrections to the surface area metric with respect to mail-boxing. In *RT* (2008), pp. 77–80. 1

[KM63] KENDALL M., MORAN P.: *Geometrical Probability*. Charles Griffin & Co., Ltd., London, 1963. 1, 2

[MB90] MACDONALD J., BOOTH K.: Heuristics for ray tracing using space subdivision. *The Visual Computer 6*, 3 (1990), 153–166. 1, 2

[NVI08] NVIDIA CORPORATION: NVIDIA CUDA programming guide version 2.0, 2008. 3

[OS83] OOSTEROM, VAN A., STRACKEE J.: The solid angle of a plane triangle. *IEEE Transactions on Biomedical Engineering BME-30*, 2 (1983), 125–126. 2

[PGSS06] POPOV S., GÜNTHER J., SEIDEL H.-P., SLUSALLEK P.: Experiences with streaming construction of SAH kd-trees. In *RT* (2006), pp. 89–94. 1

[Seg08] SEGOVIA B.: CUDA ray tracer source code, 2008. http://www710.univ-lyon1.fr/~bsegovia/demos/radius-cuda.zip. 4

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Saarland University, Saarbrücken, Germany, 2004. 3

[WH06] WALD I., HAVRAN V.: On building fast kd-trees for ray tracing, and on doing that in O(n log n). In *RT* (2006), pp. 61–69. 1, 3