ISSN 2464-4617 (print)
ISSN 2464-4625 (online)
Computer Science Research Notes - CSRN 3401
http://www.wscg.eu
WSCG 2024
POSTERS

Dynamic Many-Light Importance Sampling for Real-Time Ray Tracing

Pedro da Silva Freire[1][a]
pedro.silva.freire@tecnico.ulisboa.pt

João Madeiras Pereira[1][b]
jap@inesc-id.pt

[1] INESC-ID, Instituto Superior Técnico, University of Lisbon, Rua Alves Redol, 9, 1000-029 Lisbon, Portugal

November 2023

Abstract

One of the main performance-heavy areas in raytracing is light sampling. Light sampling is solved in a process called next-event estimation(NEE), where light samples are taken at each ray intersection. Since real-time rendering is an objective, instead of sampling all the luminaries, just a set of lights deemed more important are sampled by using a technique called Monte Carlo Importance Sampling. Aiming further acceleration, some importance sampling-based approaches build hierarchical data structures over all light sources, which results in high maintenance costs for dynamic scenes. This paper describes a two-level light bounding volume hierarchy (BVH) of mesh-based lights to accelerate light sampling while minimising the quality loss in dynamic scenes common in similar algorithms. Its main advantage is the ability to have dynamic lights without losing excessive accuracy or performance to maintenance operations. Our approach was developed as an extension to the Mogwai's PathTracer application from NVIDIA. The algorithm rebuilds the top-level structure in the CPU allowing it to retain its accuracy and refits only the bottom-level structures in need of updating on the GPU. The CPU rebuild is a relatively costly operation, to avoid excessive performance loss it is done asynchronously only being used in the next frame. We tested several quality metrics as well as frame times, our implementation achieved an up to 36% increase in MSE and 6% in SSIM with an average of 7% slowdown when compared with the single-level BVH.

Keywords: Ray Tracing, Bounding Volume Hierarchies, Vulkan Ray Tracing, Illumination, Acceleration Structures, Importance Sampling

1. Introduction

[a] [b] Until recently raytracing has proven overly demanding for real-time rendering. Now, with the hardware acceleration available in modern RTX GPUs and highly efficient algorithms real-time raytracing can be achieved.

One of the main performance-heavy areas in raytracing is light sampling. Light sampling refers to the problem of calculating lighting for a surface from all light sources in the scene without considering reflections(indirect lighting). This is done by tracing rays to lights to see if the light is occluded by any object along the way, for complex scenes this requires many ray intersections that are impossible to calculate in real-time as the number of rays to be traced each lit patch scales linearly with the number of lights.

Modern algorithms use importance sampling to reduce the amount of rays being traced. One popular approach to importance sampling is through the use of tree data structures[8]. These usually have problematic upkeep costs but have proven to be an adequate solution for real-time rendering.

1.1. Objectives

The goal of project was to implement a two-level light bounding volume hierarchy (BVH) to improve mesh-based light sampling and reduce quality loss in dynamic scenes. Based on Pierre Moreau's work, this BVH extends NVIDIA's Mogwai PathTracer in the Falcor framework, using the Vulkan ray tracing API for real-time renders.

We compared our two-level BVH with Mogwai's single-level BVH and the ReSTIR algorithm, using new metrics for better picture quality evaluation.

The submission includes a video comparing the single-level and two-level BVHs.

2. Background

2.1. Light Equation

The rendering equation introduced by Kajiya[3] is a mathematical formulation, to predict the intensity of light passing from a point x' in a surface to a point x, it takes into account intricate phenomena such as reflection, refraction, and shading, this equation enables us to calculate the appearance of objects with high accuracy and realism.

The equation is as follows:

$$I(x,x') = g(x,x')\left[\epsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')\,dx\right] \tag{1}$$

Where $I(x,x')$, is related to the light intensity going from point x' to point x in $joule/m^4sec$, $g(x,x')$, is related to geometry, it is zero if the surface of x' is occluded. $\epsilon(x,x')$, is related to the light intensity emitted from emissive surfaces from point x' to x. $\rho(x,x',x'')$, is related to the light intensity scattered from an emissive point x" to x by a patch of surface at x'. And S, is the union of surfaces.

The light equation is a useful tool for realistic rendering, however, solving the integral over S would require an in-

[a] https://orcid.org/0009-0008-6672-5940
[b] https://orcid.org/0000-0002-8120-7649

finite amount of samples. Monte Carlo Methods are used to solve quantitative problems in science through the use of statistical sampling[7]. Sampling can be improved by using importance sampling, meaning we try to take more relevant samples to increase quality at the same sample count. In our case, we want to estimate $I(x, x')$ using a limited amount of samples acquired from traced rays.

2.2. Mesh Light Sampling

When using mesh-based lights it is too computationally demanding to sample every emissive triangle. Furthermore, in more complex ray tracing algorithms, direct lighting calculations are performed many times per pixel, making it also excessively demanding to try to sample every light.

This research focuses on importance sampling algorithms for light sampling, mainly algorithms based on a BVH data structure. By sampling only a select few lights deemed more important good quality and performance can be obtained.

2.3. BVH Data Structure

The bounding volume hierarchy is a tree data structure where each node is associated with a volume of space and the lights contained within. The root node is associated with the entire scene, subsequent child nodes result from splitting the parent node's set of elements into smaller sets, one for each child node.

When building BVHs for light sampling, it is important to properly estimate each node's contributions. Alejandro Estevez and Christopher Kulla [2] proposed an algorithm to ensure that each node's importance is calculated with enough accuracy.

Additionally, Estevez and Kulla proposed a new heuristic for evaluating the quality of a node split. The new heuristic is called the surface area orientation heuristic(SAOH) it expands on the surface area heuristic by adding a light orientation component.

In this work we implement a novel approach using a two-level BVH introduced by P. Moreau et al. [5]. With a two-level BVH we can use both refits and rebuilds to achieve real-time rendering while preserving quality.

3. Implementation

3.1. Development Environment

NVIDIA's Mogwai is an open-source platform that is modular allowing fast prototyping of algorithms. Mogwai's modularity is achieved through the use of renderGraphs. The renderGraph in which we implemented our algorithm is the PathTracer renderGraph.

The PathTracer renderGraph is capable of running a simple unbiased path tracer great for creating ground truth images. It also already includes some NEE algorithms such as the single-level BVH in which our implementation is based, and ReStir [1] the current best algorithm for direct lighting.

3.2. Two-Level Light BVH

The two levels of the BVH divide the scene differently, the top-level acceleration structure(TLAS) divides the scene into groups of emissive meshes, each BLAS divides an emissive mesh into groups of emissive triangles. Each leaf node of the TLAS contains only one emissive mesh and a link to the corresponding bottom-level acceleration structure(BLAS) that will divide said emissive mesh, this structure is as seen in Figure 1.

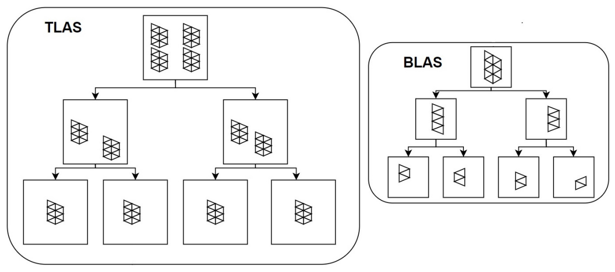

Figure 1: Two-level BVH structure. The TLAS keeps the meshes while the BLASes subdivide meshes.

By building the tree this way the refit operation only has to be applied to BLASes of changed mesh lights and the TLAS. When lights change TLAS and BLASes that need updating get refitted. The refitted TLAS is an asynchronously rebuilt TLAS from the previous frame.

Next, we describe each of the operations needed to use and maintain a two-level light BVH

3.3. BVH Building

Building the BVH is done in a top-down approach, in the case of a two-level BVH we start at the root of the TLAS, this node has an AABB that encompasses the entire scene and a list of all lights.

Then we split the current node being processed into two child nodes, the parent AABB and light list are divided between the two child nodes. Then we repeat this slitting for created child nodes until we only have one light, creating a TLAS leaf node.

The BLAS build is done in the same way as the TLAS, subdividing the corresponding emissive mesh into subsets of triangles.

To split nodes we use binning to divide the element list into several sets of elements(bins). Then we calculate the cost of splitting between each of those bins with SAH or SAOH and finally choose the lowest cost split.

To evaluate the cost of a split that divides the list of elements into two groups, we use one of two heuristics, these are SAH Eq.(2) which takes into consideration the number of triangles and their total area, and SAOH Eq.(3) which also considers the light's orientation yielding better picture quality.

$$cost_S AH(L, R) = \frac{n(L)a(L) + n(R)a(R)}{n(L \cup R)a(L \cup R)} \tag{2}$$

Where n(G) and a(G) refer respectively to the number of triangles in group G and their area.

$$cost_S AOH(L, R, s) = k_r(s)\frac{\Phi(L)a(L)M_\Omega(L) + \Phi(R)a(R)M_\Omega(R)}{a(L \cup R)M_\Omega(L \cup R)} \tag{3}$$

Where $k_r(s)$ is a regularization factor given by $k_r(s) = \frac{length_max}{length_s}$ this factor is used to penalize the choosing thin boxes and $M_\Omega(L)$ is an orientation based scalar.

3.4. BVH Refitting

Refitting is a much simpler operation than building, it is done on the GPU, the refit is done bottom-up as seen in Figure 2, processing the lowest level of BLASes first, then we refit the TLAS also starting at the lowest level. This means a node is refitted based on its child nodes. This implies we need to first process leaf nodes and only then their parent nodes.

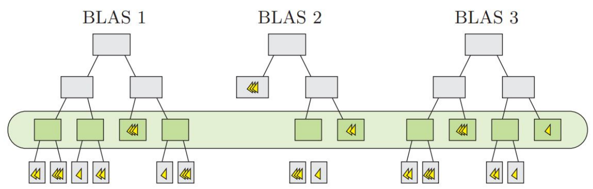

Figure 2: Figure form [5], showing all the BLASes being refit one dispatch per level with a bottom-up approach

To maximize refitting efficiency GPU batching is used, which means several GPU function calls are batched together to avoid delay overheads from multiple GPU calls.

To refit a BLAS leaf node we iterate over all of its triangles and calculate its new AABB and normal bounding cone. Internal nodes have their AABB and normal bounding cones adjusted to encompass their child nodes, and TLAS leaf nodes copy the corresponding BLAS root node properties.

3.5. BVH TLAS Rebuilding

The rebuilding of the TLAS happens in the same way as the building, but when a leaf node is created instead of building a BLAS it links to the correct already built BLAS.

Rebuilding happens asynchronously to minimize performance costs. With asynchronous processing comes a synchronisation problem.

The async rebuild is launched immediately after the TLAS refit, we allow syncing at the beginning of the next frame, and wait if needed for it to end before refitting in the next frame.

3.6. BVH Sampling

To generate a light sample from a two-level light BVH we traverse the BVH starting at the TLAS root until we reach a BLAS leaf.

To navigate the BVH (Bounding Volume Hierarchy), a random number is initially generated. This number determines which branch of the tree to explore. At each node, the random number is compared to the probability of selecting each branch.

The probability of choosing each node varies with their importance. This importance can be measured with the combination of the following metrics, distance, light flux, light orientation and light visibility.

Combining these metrics the following expression is used to calculate each node's importance:

$$importance(X, C) = \frac{\Phi(C)|\cos\theta_i'|}{||X - C||^2} x g(\theta') \tag{4}$$

Where X is the shading point and C is the center of the AABB, $g(\theta')$ is $cos\theta'$ if $\theta' < \theta_e$ and zero otherwise, $\theta_i' = max(0, \theta_i - \theta_u)$ and $\theta' = max(0, \theta - \theta_o - \theta_u)$where θ_i is the incident angle and θ_u is the uncertainty angle for the bounding cone to cover all emissive triangles, all these can be found in Figure 3.

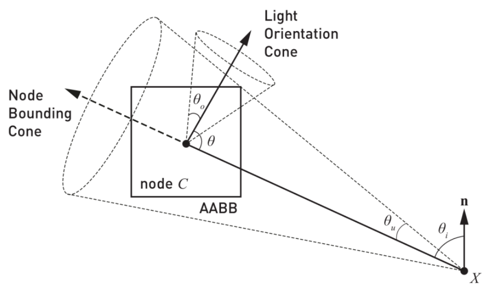

Figure 3: Image taken from[5].Geometry used for computing the importance of each node.

4. Experiments

To evaluate the performance of the developed algorithm we tested it against its single-level variant and ReStir. This way we can study the advantages of rebuilding the top structure as well as the consequences of restricting splitting to entire meshes in the top structure. We also compared the quality of the SAH and SAOH splitting heuristics with the two-level BVH.

The rendered images were tested against ground truth images we generated with Mogwais PathTracer.

We used two scenes to test the algorithms. The first, NVIDIA´s Bistro [4], features a long street allowing us to showcase quality decay over long stretches of movement. The other test scene was Emerald Square[6], it offers a more expansive environment filled with numerous lights, creating a complex lighting scenario for analysis.

The metrics taken in testing are MSE, SSIM, and PSNR. For performance we measured total frame time, as well as duration of rebuild and refit.

4.1. Results

In the Table 1 we have the averages over the several samples tested for each frame. Then in Figure 4 we show the graphs that map those averages shown in Table 1. We can see the better updating from the two-level BVH mitigates some of the quality loss.

When comparing the BVH´s we see the two-level BVH outperforming the single-level in MSE by 17% on average with an up to 36% improvement and in SSIM by 6% on average.

Bistro Exterior					
SingleLevel Average Frames:	0	50	100	150	Average
MSE	460,76	516,73	515,09	846,48	**584,76**
SSIM	0,525	0,500	0,509	0,462	**0,499**
PSNR	23,253	22,628	22,800	21,263	**22,486**
TwoLevel SAOH Average Frames:					
MSE	426,26	440,77	438,56	688,05	**498,41**
SSIM	0,542	0,538	0,538	0,493	**0,53**
PSNR	23,583	23,462	23,591	22,407	**23,26**

Table 1: Table with the average results for all metrics in tested frames of our version of Bistro.

The average SSIM results obtained in Emerald Square (Figure 5) show minimal difference between both BVH´s, the SAOH outperforms SAH heuristic. The ReStir results were considerably better than the other algorithms with

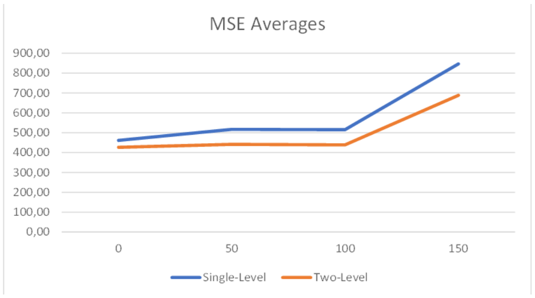

Figure 4: Graphs mapping the averages of MSE from Table 1.

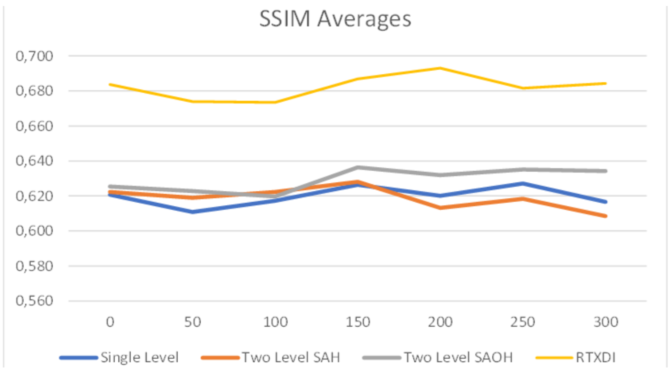

Figure 5: Graphs mapping the averages of SSIM in Emerald Square[6].

a 60% improvement in MSE and 8% in the SSIM metric. When looking at the close-up images of Emerald Square provided in Figure 6, we see much better results for the two-level SAH and SAOH pictures than for the single-level image.

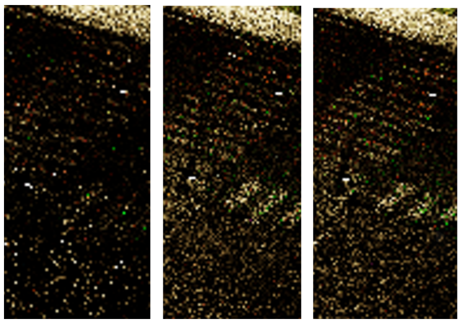

Figure 6: Zoomed in pictures of frame 150 with 4 samples. Left-Light BVH SAOH, center -two-level with SAH, right - two-level BVH with SAOH.

In Table 2 we have the frame time results for the single-level BVH and two-level BVH both using SAOH. These results were obtained in a PC with an Intel i5-4690k at stock frequency and an NVIDIA RTX 2080. The results are averages over all frames of animation with a simulated 25fps.

Overall the studied metrics showed the expected improvements and were close to what was reported by the work described in [5]. We can see from the close picture that the two-level BVH is working as expected improving quality in areas affected by moving lights.

Performance was as expected we saw an average of 7% slow down which is consistent with what was reported in

	Measure One		Emerald Square		Bistro Exterior	
Algorithm:	Single Level BVH	Two Level BVH	Single Level BVH	Two Level BVH	Single Level BVH	Two Level BVH
BVH Refit (ms)	4	7	7	9,2	4,2	6,3
BVH Rebuild (ms)	- -	7,2	- -	12,5	- -	3,2
Frame Time (ms)	61(16,4FPS)	66(15,2FPS)	60(16,7FPS)	63(15.9FPS)	36(27,7FPS)	38(26,3FPS)

Table 2: Performance Results

[5]

5. Conclusions

Not all scenes showed improved metrics metrics, however, when examining picture quality from zoomed-in pictures we saw more detail in certain areas with the two-level BVH compared to the single-level BVH.

Performance was as expected the added processing from the TLAS rebuild had little to no effect on performance due to its asynchronous processing but the refit was slightly slower.

When testing ReSTIR we saw an overall improvement in quality as was expected from the current best solution.

ReSTIR is the best solution of the three tested for all scenarios.

The two-level BVH showed some improvements over the single-level version, especially evident in scenarios where lights had undergone substantial updates.

Acknowledgements

The work reported in this article was partially supported under the auspices of the UNESCO Chair on AI & VR by national funds through Fundação para a Ciência e a Tecnologia with references DOI:10.54499/ UIDB/-50021/2020, DOI:10.54499/DL57/2016/CP1368/CT0002 and 2022.09212.PTDC (XAVIER project).

The work was also carried on under DL 60/2018, 3-08 – I&D activity at INESC -ID, project PEST 2020 -GI (graphics & interaction (UIDB/50021/2020).

References

[1] B. Bitterli, C. Wyman, M. Pharr, P. Shirley, A. Lefohn, and W. Jarosz. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (TOG)*, 39(4):148–1, 2020.

[2] A. Conty Estevez and C. Kulla. Importance sampling of many lights with adaptive tree splitting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2):1–17, 2018.

[3] J. T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[4] A. Lumberyard. Amazon lumberyard bistro, open research content archive (orca), July 2017. `http://developer.nvidia.com/orca/amazon-lumberyard-bistro`.

[5] P. Moreau, M. Pharr, and P. Clarberg. Dynamic many-light sampling for real-time ray tracing. In *High Performance Graphics (Short Papers)*, pages 21–26, 2019.

[6] K. A. Nicholas Hull and N. Benty. Nvidia emerald square, open research content archive (orca), July 2017. `http://developer.nvidia.com/orca/nvidia-emerald-square`.

[7] M. Pharr, W. Jakob, and G. Humphreys. *Physically based rendering: From theory to implementation.* MIT Press, 2023.

[8] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.