BVH Trees of Many Dynamic Lights for Real-Time Ray Tracing

Hubert Sosnowski[1] and Dominik Szajerman[1][0000−0002−4316−5310]

Institute of Information Technology, Lodz University of Technology, Łódź, Poland
dominik.szajerman@p.lodz.pl

Abstract. Real-time rendering is limited in the number of lights that can be used to shade a singular point, therefore it is crucial to select the most appropriate lights. To efficiently search through the large collection of lights a BVH tree is used, which contains positions and emissive characteristics of lights. Instead of rebuilding, an update process is used for animated objects. The update is faster, but it degrades the quality of the tree over time, resulting in increased rendering error. Therefore, the tree should be occasionally rebuilt. This paper presents four key attributes of the light tree, which can be quickly and easily calculated during the tree update process in every frame. They can be used to estimate the degree of the tree quality degradation. Two algorithms are proposed that can be used to decide on the necessity of tree rebuilding, based on the previously mentioned attributes. Applying those algorithms results in lower rendering error compared to a tree that has not been rebuilt, with a minimal number of rebuilds. At the same time, this solution is more versatile compared to rebuilding at fixed intervals.

Keywords: Bounding Volume Hierachy (BVH) · dynamic lights · real-time ray tracing.

1 Introduction

Lighting is crucial in virtual scenes. However, managing large sets of lights negatively impacts real-time rendering performance. Current hardware limitations restrict the number of lights that can be used to calculate lighting for a single pixel in a frame, making the selection of appropriate lights a complex task. An effective approach is to use a binary BVH (Bounding Volume Hierarchy) tree to store the positions and attributes of lights. While this solution is efficient, initializing the structure is costly, and building a tree for every frame in animated scenes is impractical. Although algorithms exist to update tree data [3], the update process reduces the performance of the tree search algorithm (quality of the tree).

The developed heuristic clearly indicates when to rebuild the tree while allowing adjustments to the frequency of rebuilds and acceptable rendering error increases. It also accommodates two-level trees [6].

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_13

2 Related Work

In [2], the authors create a BVH, considering not only the position of each light but also the direction of light emission and the scattering angle. The method handles a large number of lights effectively, as well as the local influence of some lights. Initializing the tree structure is still expensive, but this is done only once for the scene. The tree search itself is not very costly, and the binary tree structure provides opportunities for data updates. Methods for updating data in a BVH tree after elements have been moved were described in [1] and further developed in [3] and [8]. The light tree tolerates updates well but loses accuracy over time with subsequent updates. These phenomena were studied in [6], where a two-level tree was proposed, analogous to those used in ray tracing.

The existing algorithms to solve the following tasks were adapted in this work: tree building[8], tree search[2], and tree update [6].

3 Method

This work identifies **four** tree attributes that affect the representation of lights. The first is the **distance of subtrees** from each other). If child subtrees are too far apart (Fig. 1), the accuracy of approximating a node's influence on a scene point may decrease due to including overly distant areas in calculations.

Similarly, the opposite situation occurs when the **subtrees overlap** significantly (Fig. 2). This results in a situation where the approximated influence of both subtrees on a given point is similar, which makes it uncertain which path can lead to a better solution.

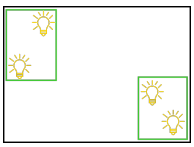

Fig. 1. Empty areas in the volume – too large distance of subtrees

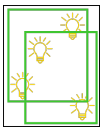

Fig. 2. Subtrees overlap

Another attribute is a measure describing the **variance of lights** in a given cluster, as noted in [2]. The changes in the tree structure can also be tracked using the **splitting cost** formula.

The build algorithm minimizes the values of the four attributes presented, while the update algorithm ignores them. These values can be computed as the tree is rebuilt and their changes monitored in subsequent frames.

This paper proposes two **rebuild strategy** algorithms for deciding when to rebuild the tree based on one of the four attributes.

During subsequent frames and updates, the values of them can be calculated. In order to correctly determine the parameter value for which the tree should

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_13

be rebuilt, it is possible to analyze the changes in these parameters. Several fixed points can be chosen during the animation to calculate the values of the selected parameter after the tree rebuilding. From the set of values calculated in this way, the standard deviation (SD) can be computed. Updating the tree will yield quite variable parameter values, while rebuilding will usually lead to their minimization. The standard deviation in this case is an estimate of what changes in values can be expected after the tree reconstruction. In cases where the calculated parameter value (the "calculateParameter" function) increases by a specified multiple (S) of the standard deviation above the previous minimum parameter value (Pmin) since the last rebuild, a decision can be made to rebuild the tree as presented in the **algorithm 1**.

When the light set and animation itself change significantly, there is a need to recalculate the standard deviation, which results in limited applicability of the method. **Algorithm 2** presents the solution for cases when the standard deviation cannot be calculated. In such situations, it is possible to estimate the parameter value at which the tree should be rebuilt based on changes in the parameter itself over time. Specifically, the smallest parameter value after the last rebuild (Pmin) can be stored. When the calculated parameter value exceeds the smallest value Pmin multiplied by a chosen scale factor (S) (e.g., 10%), the decision to rebuild the tree can be made.

Algorithm 1

```
if calculateParameter(D) < Pmin then
    Pmin ← calculateParameter(D)
end if
if calculateParameter(D) > Pmin + SD * S then
    rebuildTree(D)
    Pmin ← calculateParameter(D)
else
    updateTree(D)
end if
```

Algorithm 2

```
if calculateParameter(D) < Pmin then
    Pmin ← calculateParameter(D)
end if
if calculateParameter(D) > Pmin * S then
    rebuildTree(D)
    Pmin ← calculateParameter(D)
else
    updateTree(D)
end if
```

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_13

4 Results and Discussion

The testing program renders HDR images for result comparison, using the Monte Carlo method with importance sampling and the BRDF function for shading. After each frame is rendered, the scene is updated according to defined animations, and the light tree is adjusted based on new object positions. The decision to update or rebuild the tree depends on specified conditions. The resulting rendered frames were compared to a reference frame (ground truth) rendered using 4000 rays per pixel. When compared to the reference frame, the error was calculated according to MAPE (Mean Absolute Percentage Error) (1) [4]:

$$\text{MAPE} = \frac{1}{N} \sum^{N} \frac{|I - I_{ref}|}{0.01 * \bar{I}_{ref} + I_{ref}} \tag{1}$$

where N is the number of pixels, I is the light intensity value for a given pixel, I_{ref} is the same as I but for the reference frame, $\bar{I}_{ref}$ the average light intensity of all pixels for the reference frame. The tests used two scenes: Amazon Lumberyard Bistro[5] – 2,848,615 triangles and NVIDIA Emerald Square City Scene[7] – 2,722,804 triangles. Experiments were conducted on a computer with an NVIDIA GeForce RTX 2070 (8 GB VRAM), an Intel i5-9600K 3.7 GHz processor, and 16 GB RAM. The rendering resolution was 1920×1080 pixels.

Fig. 3 illustrates the use of the distance of subtrees parameter to determine when to rebuild the tree in Scene 2, as per algorithm (2). A similar trend is observed for $S = 1.05$ in Fig. 4, where a smaller S results in more frequent rebuilds. However, the SD was too large relative to parameter changes for the strategy using algorithm (1) to be effective, causing the first rebuild to occur too late.

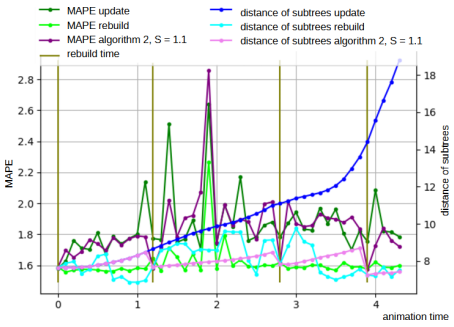

Fig. 3. MAPE and distance of subtrees over time at $S = 1.1$ for Scene 2

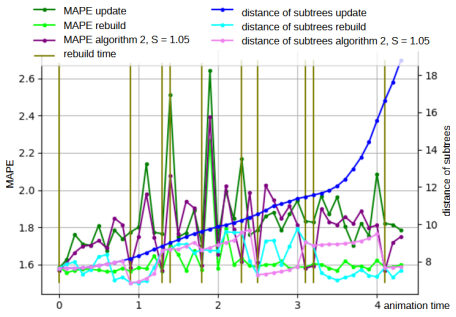

Fig. 4. MAPE and distance of subtrees over time at $S = 1.05$ for Scene 2

A similar trend is observed for the subtree overlap parameter. Results are shown in Fig. 5 and Fig. 6 for Scene 1, and in Fig. 7 and Fig. 8 for Scene 2, using algorithm (2). In Scene 2, the same scale parameter S results in more frequent

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_13

rebuild decisions due to more rapid changes in parameter values. Results from algorithm (1) are presented in Fig. 9 and Fig. 10 for Scene 1 ($SD = 0.00296$), and in Fig. 11 and Fig. 12 for Scene 2 ($SD = 0.00673$).

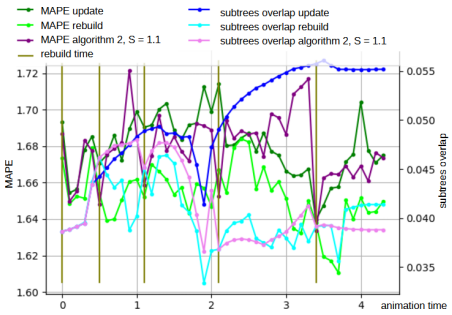

Fig. 5. MAPE and subtrees overlap over time at $S = 1.1$ for Scene 1

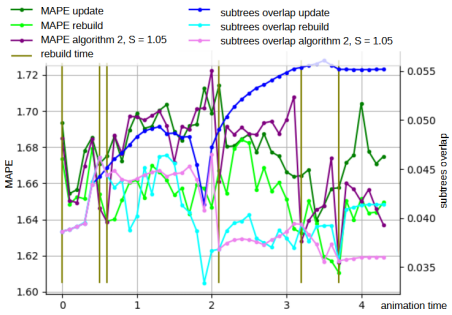

Fig. 6. MAPE and subtrees overlap over time at $S = 1.05$ for Scene 1

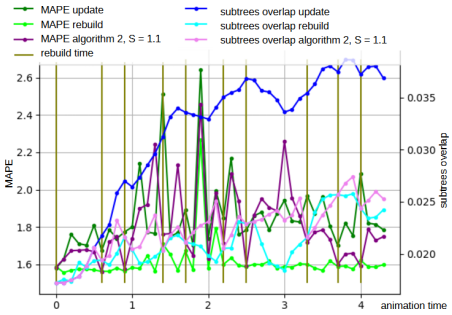

Fig. 7. MAPE and subtrees overlap over time at $S = 1.1$ for Scene 2

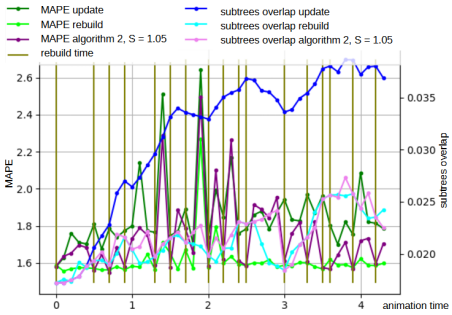

Fig. 8. MAPE and subtrees overlap over time at $S = 1.05$ for Scene 2

The variance of lights attribute increases steadily in both scenes, but the changes are too small for the chosen strategies to prompt tree reconstruction. Only tests for Scene 2 using algorithm (1) with $SD = 0.0013$ and $S = 1$ yielded positive results.

For the splitting cost attribute, $S = 1.1$ in algorithm (2) for Scene 1 was too high, resulting in too few rebuilds; $S = 1.05$ proved more suitable (Fig. 13). The first rebuild decision was delayed, causing most of the MAPE curve to overlap with that of the updated tree, with noticeable error reduction only after the second rebuild. A test using algorithm (1) with $SD = 89.409$ yielded similar results, with rebuild decisions occurring at the same times. In Scene 2 (Fig. 14), $S = 1.1$ was more appropriate, but the strategy with $SD = 96.257$ resulted in

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_13

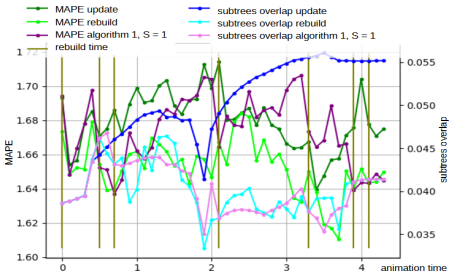

Fig. 9. MAPE and subtrees overlap for $SD = 0.00296, S = 1$, Scene 1

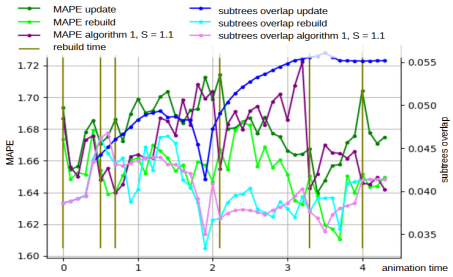

Fig. 10. MAPE and subtrees overlap for $SD = 0.00296, S = 1.1$, Scene 1

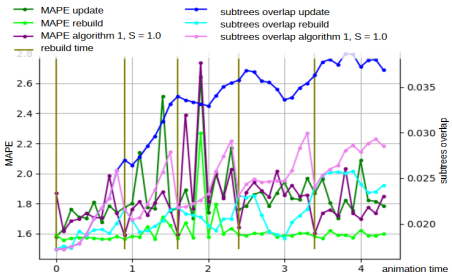

Fig. 11. MAPE and subtrees overlap for $SD = 0.00673, S = 1$ for Scene 2

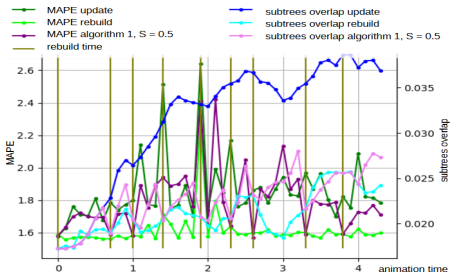

Fig. 12. MAPE and subtrees overlap for $SD = 0.00673, S = 0.5$, Scene 2

excessive rebuilds. Only increasing S to 1.5 reduced rebuild frequency, yet tests showed no significant improvement in rendering error.

Tables 1 and 2 summarize test results. The average MAPE is lowest for the always-rebuilding tree and highest for the update-only tree. Rebuilding strategies offer intermediate performance, better than updating alone but not as good as always rebuilding.

It can be seen that the largest MAPE is selected for the tree that is not rebuilt. Unfortunately, the tree rebuild operation is too expensive to do in every frame. For Scene 1, the tree update operation requires 6.26ms, while the tree rebuild takes 15.92ms, which is more than twice as long. Similarly, for Scene 2: 17.71ms and 89.81ms, which is more than a factor of 5.

Considering an optimization algorithm that will determine the optimal combination of the algorithm, its parameters, and the impact of individual tree attributes on performance and quality of the tree requires taking many factors into account. The best results were achieved for subtrees overlap and splitting cost in both scenes, with distance of subtrees also performing well in Scene 2. In Scene 1, the most effective decision-making algorithm was (1) using subtree overlap with a scale factor of $S = 1.1$. In Scene 2, the largest error reduction occurred with the (2) algorithm and $S = 1.05$, resulting in 17 rebuilds over 44 frames. A better rebuild-to-error ratio was obtained with (1) using $SD = 0.00673$

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_13

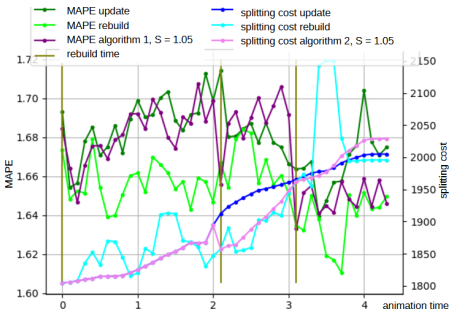

Fig. 13. MAPE and splitting cost over time at $S = 1.05$ for Scene 1

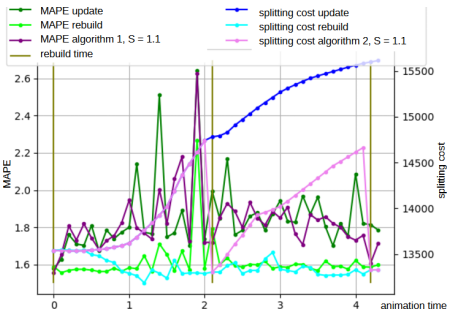

Fig. 14. MAPE and splitting cost over time at $S = 1.1$ for Scene 2

Table 1. Number of rebuilds and average MAPE for different rebuild strategies for Scene 1. Total frame count is 44. The given times are expressed in milliseconds.

Strategy		Average MAPE	Rebuild count /44	Update time	Rebuild time
Always updated		1.680	1	269	16
Always rebuild		1.653	44	0	700
subtrees overlap	alg. 1, $SD = 0.00296$, $S = 1$	1.675	7	232	111
	alg. 1, $SD = 0.00296$, $S = 1.1$	1.674	6	238	96
	algorithm 2, $S = 1.1$	1.679	5	244	80
	algorithm 2, $S = 1.05$	1.676	6	238	96
splitting cost	algorithm 2, $S = 1.05$	1.676	3	257	48

Table 2. Number of rebuilds and average MAPE for different rebuild strategies for Scene 2. Total frame count is 44. The given times are expressed in milliseconds.

Strategy		Average MAPE	Rebuild count /44	Update time	Rebuild time
Always updated		1.856	1	762	90
Always rebuild		1.614	44	0	3952
distance of subtrees	algorithm 2, $S = 1.1$	1.842	4	708	359
	algorithm 2, $S = 1.05$	1.792	10	602	898
subtrees overlap	alg. 1, $SD = 0.00673$, $S = 1$	1.832	5	691	449
	alg. 1, $SD = 0.00673$, $S = 0.5$	1.799	9	620	808
	algorithm 2, $S = 1.1$	1.802	10	602	898
	algorithm 2, $S = 1.05$	1.745	17	478	1527
variance of lights	alg. 1, $SD = 0.0013$, $S = 1$	1.774	11	584	988
splitting cost	algorithm 2, $S = 1.1$	1.833	3	726	269

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_13

and $S = 0.5$ for subtree overlap (9 rebuilds), and with $SD = 0.0013$ and $S = 1$ for variance of lights (11 rebuilds).

Studies on BVH also report performance measurements. In [3] it ranges from 813 ms to 1659 ms per million triangles (Mt) for the rebuild operation, depending on the processed scene, and from 220 ms to 375 ms per Mt for the update operation. These performance results are the only available data suitable for comparison. The proposed method achieves 6 ms per Mt for Scene 1 and 33 ms per Mt for Scene 2 during the rebuild phase. The corresponding update times are 2 ms per Mt and 7 ms per Mt, respectively.

5 Conclusions

In this paper, four attributes of the BVH tree are considered: distance of subtrees, subtrees overlap, variance of light, and splitting cost. Two algorithms are proposed to determine tree reconstruction based on the selected attributes and an additional scale factor or calculated standard deviation. Using these attributes for decision-making resulted in fewer tree rebuilds during testing, but led to a noticeable decrease in rendering error compared to the non-rebuilt tree. While rebuilding the tree every frame was the most effective solution for reducing error, it was highly inefficient in terms of the time required compared to updating.

Different attributes and coefficients in the algorithms can produce varying effects depending on the number and types of lights in the scene and their animations. Therefore, the parameters of the proposed algorithms should be tailored to individual needs.

References

1. van den Bergen, G.: Efficient collision detection of complex deformable models using aabb trees. Journal of Graphics Tools **2** (1997). https://doi.org/10.1080/10867651.1997.10487480
2. Estevez, A.C., Kulla, C.: Importance sampling of many lights with adaptive tree splitting. Proceedings of the ACM on Computer Graphics and Interactive Techniques **1** (2018). https://doi.org/10.1145/3233305
3. Lauterbach, C., Yoon, S.E., Manocha, D., Tuft, D.: Rt-deform: Interactive ray tracing of dynamic scenes using bvhs (2006). https://doi.org/10.1109/RT.2006.280213
4. Lin, D., Kettunen, M., Bitterli, B., Pantaleoni, J., Yuksel, C., Wyman, C.: Generalized resampled importance sampling: Foundations of restir. ACM Transactions on Graphics **41** (2022). https://doi.org/10.1145/3528223.3530158
5. Lumberyard, A.: Amazon lumberyard bistro, open research content archive (orca) (July 2017), http://developer.nvidia.com/orca/amazon-lumberyard-bistro
6. Moreau, P., Pharr, M., Clarberg, P.: Dynamic many-light sampling for real-time ray tracing. vol. 2019-July (2019). https://doi.org/10.2312/hpg.20191191
7. Nicholas Hull, K.A., Benty, N.: Nvidia emerald square, open research content archive (orca) (July 2017), http://developer.nvidia.com/orca/nvidia-emerald-square
8. Wald, I., Boulos, S., Shirley, P.: Ray tracing deformable scenes using dynamic bounding volume hierarchies. ACM Transactions on Graphics **26** (2007). https://doi.org/10.1145/1189762.1206075

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_13