ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/333205600

Comparison of BVH and KD-Tree for the GPGPU Acceleration on Real Mobile Devices

Chapter · May 2019

DOI: 10.1007/978-981-13-3648-5_62

CITATIONS
0

READS
193

4 authors, including:

SeongKi Kim
Keimyung University
23 PUBLICATIONS **13** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

GPU DVFS for the low power and high performance of mobile GPGPU applications View project

Realistic Mobile-based Immersive Virtual Reality with Low-power/ High-performance View project

All content following this page was uploaded by SeongKi Kim on 27 May 2019.

The user has requested enhancement of the downloaded file.

Comparison of BVH and KD-tree for the GPGPU acceleration on real mobile devices

SeungWoo Chung[1], MinKyoung Choi[1], DaeGeun Youn[1] and SeongKi Kim[1]*

[1]Keimyung University, Korea
*skkim@kmu.ac.kr

Abstract. Portability and thermal issues significantly limit the performance of mobile devices, hindering the use of the global illumination that enables high-quality rendering. For a long time, the BVH and KD-tree algorithms through GPU acceleration have been widely used to accelerate global illumination in the desktop environment. However, these algorithms have not been sufficiently studied in the mobile environment, and therefore, the use of either of them presents many challenges. This paper describes the implementation of BVH and KD-tree algorithms that utilize GPU acceleration on mobile devices. The experimental results show that the path tracing performance can be increased 2.3 times on average when KD-tree is used and that KD-tree can improve the performance more than BVH. To the best of our knowledge, this study is the first to examine BVH and KD-tree in the real mobile environment.

Keywords: Path tracing, Mobile device, BVH, KD-tree, GPGPU

1 Introduction

On mobile devices, the performance of which is limited, it is difficult to run global illumination, which requires vast resources. However, as the number of users who want to run high-quality games and virtual/augmented realities on mobile devices increases, many types of research have been conducted with the purpose of realizing global illumination. In [1], a study on creating realistic images for augmented reality in a mobile environment was described. In [2], a new method for real-time global illumination in a mobile device was proposed. Despite these efforts, global illumination is not widely used because of the performance issue related to mobile devices.

The BVH and KD-tree algorithms on the GPU are a typical means of accelerating global illumination, and many researchers have studied them in the desktop environment. However, research on practical algorithms for mobile devices is still lacking. We implemented the acceleration algorithms through the GPGPU on mobile devices and evaluated their performance. The rest of this paper is organized as follows. Section 2 explains the related works. Section 3 describes the implementation of the acceleration algorithms for mobile devices and compares their performance on real mobile devices. Section 4 concludes the paper.

2 Related Works

BVH (Bounding Volume Hierarchy). BVH is a data structure of geometric objects with a tree and was proposed by Herman Johannes Haverkort [3] in 2004. It has the overall architecture in the below Fig. 1.

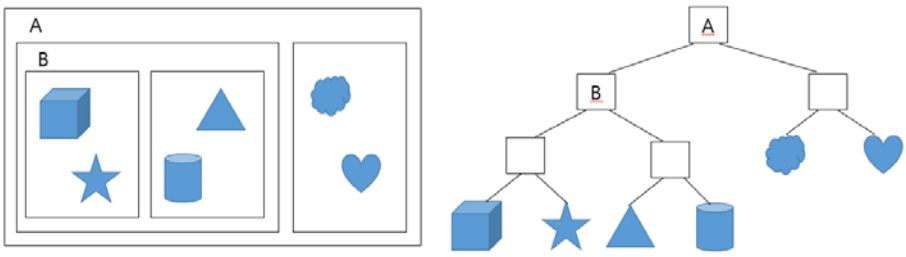

Fig. 1. Architecture of BVH.

As shown in Figure 1, all of the geometric objects are included in the leaf of a tree. The leaves are recursively included in the larger bounding volume, and all objects are finally included into a single root node.

Afterward, Y. Gu et al. [4] added Approximate Agglomerative Clustering (AAC) function and improved performance about 2.2 times. Lauterbach et al. [5] used GPU to speed up the generation/ the traversal of a BVH tree and increased the performance by 8.5 times when compared to the CPU-based algorithm.

KD tree (K-Dimensional tree). KD tree is one of the trees to accelerate the intersection test between the triangle and the ray, and recursively splits a plane in an axis-aligned manner like the below Fig. 2.

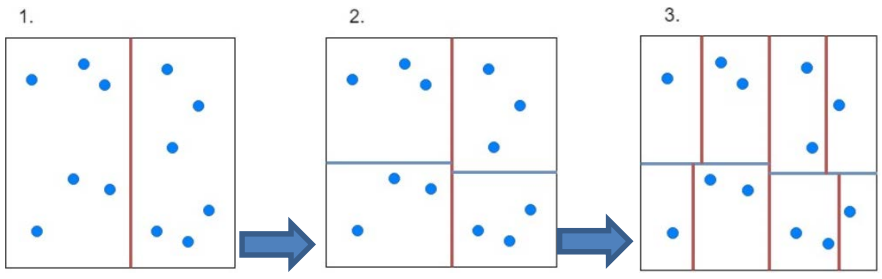

Fig. 2. Recursive division of KD-tree.

The KD tree divides a space into binary nodes aligned with the X, Y, and Z axes. Each node has an Axis Aligned Bounding Box (AABB) for the corresponding space.

Two child nodes in an arbitrary internal node are two spaces of the divided plane and the AABB of the child node cannot be larger than the AABB of the parent node.

The KD tree consists of three different types of nodes: a root node at the top of a tree, inner nodes with children at the middle of a tree, and leaf nodes without any children at the bottom of a tree. When the KD tree is traversed in the rendering step, the algorithm decides which children should be traversed first with the partition information of the parent node and the children. When reaching a leaf node, the algorithm checks the intersection between the ray and the included triangles.

Zhou et al. [6] proposed an algorithm for constructing the KD tree on the GPU and increased the performance by 6-15 times when compared to the CPU-based algorithm. Wu et al. [7] proposed a method of accelerating the construction of a KD tree with the SAH method on a GPU. The SAH method is a greedy algorithm that uses a cost function with the surface area of the node space.

3 Implementation and Results

Implementation. Using Android Studio and NDK, we implemented BVH and KD-tree through the GPGPU using the following procedure. First, we create a BVH tree for the path tracing processes. We divide the information of all the triangles and spheres and initialize the BVH structure [3]. Fig. 3 shows these processes.

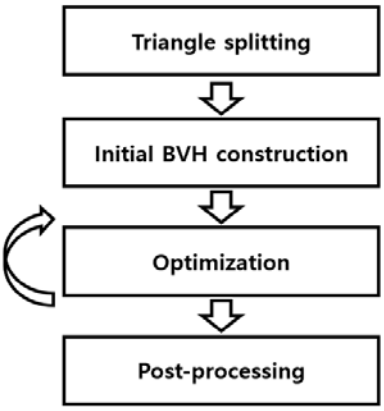

Fig. 3. Processes of building BVH tree.

Before building the BVH tree, our algorithm separates all geometric objects into triangles and spheres. It makes a radix tree to allow faster parallel processing and generates a BVH tree through bubble sorting. After the BVH tree has been generated, our algorithm reconstructs a small treelet in the optimization phase [3]. When comparing the BVH, our algorithm classifies the children into treelets. Each partitioned tree recur-

sively executes the algorithm, and our algorithm obtains the SAH (Surface Area Heuristics) cost. The cost of an entire tree is then calculated. Then, we reconstruct the small treelet with the calculated cost. The reason why the treelet is constructed and optimized is that it can reduce the cost and its structure can be easily processed using a parallel method when the node is sequentially expanded. For this structure, we need to keep a portion of the entire AABB (Axis-Aligned Bounding Box) in a separate array and specify the number of iterations for each thread.

When we implemented the KD-tree, first, we defined the coordinates of the AABB of a triangle. Then, we set the minimum and maximum values of each axis, which are required to allow use of the space division [4]. The range is determined by using the coordinates of the generated AABB. After we created the tree, the coordinate of the AABB was returned if it was out of range. If it was internal, the start/end addresses of each KD-tree were cropped by using the variance calculation and then arranged. The two addresses indicate the location of the points and the variance determines how widely the points are scattered.

We used variance calculation to determine the distance of the vertices from the first and last addresses:

$$\text{Variance} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} \quad (1)$$

$$\bar{x}(average) = \frac{(x_1 + x_2 + \cdots + x_n)}{n}$$

We obtained sorted values by comparing the start and end addresses with the incised part, finding the median value, assigning them to the left and right-hand sides of each path, and removing the root node of the KD-tree when the calculation of the vertices was completed. We ran these procedures recursively when implementing the KD-tree.

Results. We performed all our experiments on a Galaxy Note 8 with Exynos 8895 (2.3 GHz Quad-core + 1.7 GHz Quad-core, Mali-G71 MP20 546 MHz and 6 GB LPDDR4X SDRAM) as the internal AP. The device uses Android 8.0 (Oreo) as the platform and supports OpenCL for the GPGPU acceleration. We implemented the BVH and KD-tree algorithms with OpenCL, as described in Section 2. For the experiment, an ant model (486 vertices and 912 triangles) and 9 circles (2 circles with the properties of specular/diffuse reflection, respectively, 1 circle serving as the light source, and 6 circles serving as the walls) were rendered using path tracing. The image size was 640 × 480. Fig. 4 shows the results.

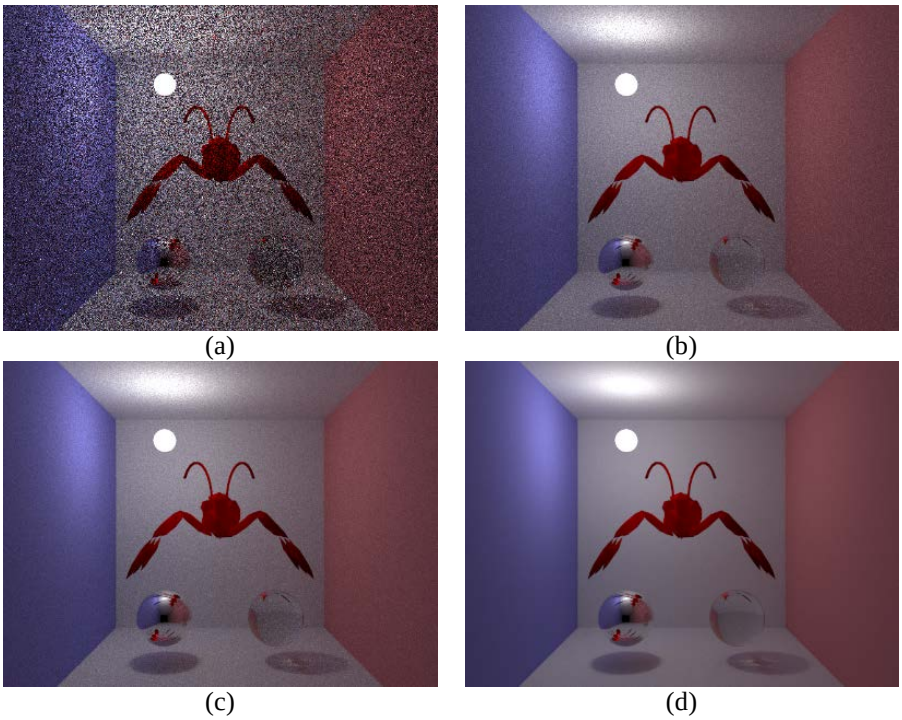

Fig. 4. Rendering results by the global illumination on mobile devices with the KD-tree. (a) 1 frame. (b) 50 frames. (c) 100 frames. (d) 1000 frames. The rendered results become more refined over time.

When we used the BVH and the KD-trees, the rendering speed was recorded for comparison with the case where no acceleration algorithms were applied. Table 1 shows the rendering performance of BVH and KD-tree on the mobile device.

Table 1. Rendering performances of path tracing with each algorithm that adapts GPGPU acceleration.

Time (ms)	No acceleration	BVH	KD-tree
Rendering time	1762.7	905.8	753.0

Table 1 shows that the rendering performance is improved when we use the BVH or KD-tree algorithms with the GPU acceleration on the real mobile device, and the performance improvement is greater when the KD-tree algorithm is used.

When we used the BVH and KD-tree algorithms that utilize GPU acceleration, a performance improvement of 1.95 times and 2.3 times, respectively, was achieved as compared to the case where no acceleration algorithm was applied.

4 Conclusions

The hardware inside mobile devices has significantly improved, but performance limitations related to the portability and thermal problems remain. However, as user demand for high-quality content increases, global illumination in the mobile world is becoming essential. In this study, we implemented the BVH and the KD-tree algorithms with GPU acceleration on real mobile devices and compared their performances. Tests were performed using 3D models in a real mobile environment.

The results show that KD-tree algorithm achieved a better performance than the BVH algorithm. However, further diverse comparative studies are required. First, since the performance may vary according to the size of the models, the performance of models having various sizes should be examined. Second, the results may vary according to the mobile device. These limitations still exist, and we plan to resolve these problems. To the best of our knowledge, this paper is the first to compare the performance of the BVH and KD-tree algorithms in real mobile environments.

ACKNOWLEDGMENTS

David Bucciarelli shares the SmallptGPU codes, and we improved the codes. This research was supported by NRF in Korea (NRF-2017R1A1A1A05069806). SeongKi Kim is the corresponding author.

References

1. M. Csongei, L. Hoang, C. Sandor and Y. B. Le, Global illumination for Augmented Reality on mobile phones, Proceedings of THE IEEE Virtual Reality (VR), Minneapolis, MN, 2014, pp. 69-70.
2. Minsu Ahn, Inwoo Ha, Hyong-Euk Lee and James D. K. Kim, Real-time global illumination on mobile device, Proceedings of the SPIE 9030, mobile devices and multimedia: enabling technologies, algorithms, and applications 2014, 903005, 2014.
3. Haverkort, Herman Johannes. Results on geometric networks and data structures. ISBN 1-55860-732-3. 2004
4. Y. Gu, Y. He, K. Fatahalian and G Blelloch. Efficient BVH Construction via Approximate Agglomerative Clustering. Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, HPG 2013.
5. C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, D. Manocha. Fast BVH Construction on GPUs. Computer Graphics Forum. 2009
6. K. Zhou, Q. Hou, R. Wang, and B Guo, Real-time KD-tree construction on graphics hardware, ACM Trans. Graph., vol. 27, no.5, pp. 126:1-126:11, 2008.
7. Z. Wu, F. Zhao, and X. Liu, SAH KD-tree construction on GPU, Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, HPG 11, pp. 71-78, ACM, 2011.
8. Johannes Gunther, Stefan Popov, Hans-Peter Seidel and Philipp Slusallek, Realtime Ray Tracing on GPU with BVH-based Packet Traversal, IEEE symposium on Interactive Ray Tracing, RT, 2007.
9. K. Zhou, Q. Hou, R. Wang and B. Guo, Real-time KD-tree construction on graphics hardware, ACM TRANS. GRAPH., vol. 27, no. 5, pp. 126:1-126:11, 2008.

View publication stats