A parallel construction method based on linear hierarchical bounding box without stack

Mingquan Wang¹, Huiyan Qu¹*, Yi Liu¹
¹School of Information Technology, Jilin Agricultural University, Changchun, Jilin Province, 130118, China

*Corresponding author e-mail: quhuiyan68@163.com

Abstract. The intersect operation of geometry is a common method of collision detection. To achieve a fast intersect operation, a hierarchical bounding box must be quickly constructed. This paper proposes an effective space division method for constructing high-quality linear hierarchical bounding boxes through ray tracing. This method can generate more regular axial bounding boxes into a complete binary tree. This structure can be completely parallel on the GPU, and effective parallelization can ensure the fastest tree-building time. The algorithm in this paper uses a non-stack ray tracing method to increase the frame rate of the 3D model rendering in the graphics processing unit and reduce the construction time of the bounding box. The experimental analysis of construction time, frame rate, average number of intersections and other performance indicators, the experimental results show that the use of stack-free midpoint splitting method and surface heuristic splitting method to ensure rapid construction time and efficient ray traversal performance, improve the object's light traversal efficiency.

1. Instruction

In this paper, a new method for constructing linear hierarchical enveloping boxes is proposed, which uses ray tracing technology to generate regular data structures in a complete binary tree, which speeds up the construction time of high-quality hierarchical enveloping boxes and provides an effective ray traversal method. The parallel model of OpenMp is used to construct the accelerated structure on GPU in a fast parallel way. In the optimization stage, high-performance data parallel computing is adopted, and the traversal process is realized through the internal light algorithm without stack traversal.

2. Theoretical Basis

Wald [1] designed a fast algorithm for building hierarchical bounding box in 2010. Hierarchical bounding box realizes parallel computing on GPU and CPU, which provides a good method for accelerating the construction of hierarchical bounding box. In 2010, Bauszat et al. designed a hierarchical bounding box construction method that uses minimum storage space for each node. This method reduces the memory consumption during the construction of standard hierarchical bounding box, and also reduces the traversal path and time.

Garanzha [2] et al. extended and improved the hierarchical bounding box algorithm using queues in 2011, which accelerated the construction process. The main contribution is to implement target
partitioning instead of breadth first traversal primitives on a single pipeline based on effective work queues.

In 2011, Sulaiman[3] et al. proposed a method to construct a balanced bounding box by using the middle point in collision detection. This method is to find the middle point of AABB envelopment box by using the longest axis, and adopt a top-down subdivision process. During the subdivision task execution, the modeling part is removed, which improves the ray tracking efficiency.

In 2013, Hapala[4] et al. proposed a hierarchical bounding box traversal algorithm without the use of stack. This algorithm does not need to build a stack in the construction of hierarchical bounding box, but only needs to store the pointer of parent node of each node to traverse all nodes in the post-sequence and complete the construction of hierarchical bounding box.

In 2016, ZheFu et al. [5] proposed a CPU-based partitioning algorithm for hierarchical enveloping boxes, which could delete unnecessary rays during subdivision, quickly complete the ray tracing process, and reduce the construction time of hierarchical enveloping boxes.

In 2017, Murguia et al. [6] proposed a construction method of a linear hierarchical bounding box without stack. This method USES the method of dividing intermediate points based on Morton codes and graph elements, which gives fast construction time, but increases the difficulty of traversal.

3. Parallel construction method

3.1. OpenM machine technology

OpenMP directives to expand the C language is a parallel language, but OpenMP itself is not an independent parallel language, but the writing on the multiple processor designed parallel programs, guiding the Shared memory, multi-thread parallel compilation instructions and application programming interface (API), available in C/C++ and Fortran, (77, 90 and 95) in the application, and in the serial code compiler can identify the comment form.OpenMP standard is jointly defined and proposed by some internationally influential software and hardware manufacturers. It is a portable programming model of Shared storage architecture, widely used on Unix, Linux, Windows and other platforms.

3.2. Complete binary tree

A binary tree up to only two layers on the bottom of the nodes can be less than 2 degree, and on the bottom layer nodes are concentrated in the most on the left side of the number of position, then the binary tree to become a complete binary tree, and the bottom were focused on the layer of nodes on the some position, the left and on the last layer, the right of the number of missing binary tree node, then the binary tree to become fully binary tree. Figure 1 shows a complete binary tree with a node number.

![Complete Binary Tree Diagram](image)

Figure 1. A complete binary tree of node numbering.

4. Ray tracing for complete binary trees

4.1 Construct a complete binary tree suitable for ray tracing

Using a complete binary tree, the number of nodes requires 2N, where N represents the number of primitives. In order to be a complete tree, how many primitives are stored in each branch must be defined to ensure that the deepest layer is complete. It can be observed in Figure 1 that the subtree of each node is also a complete tree, and the number of left subleaf nodes is not less than the number of
right subleaf nodes. Murguia et al. proposed the stackless LBVH method, as shown in Figure 1. One of the main problems is that the result tree is unbalanced. In order to be efficient, it is necessary to guarantee the useless memory area. This results in a lot of wasted memory areas, with only 15% of the nodes actually used. To avoid this, every floor is completely filled except the last one, which is as far to the left as possible.

4.2 Node structure based on no stack layer bounding box

Each node size is 32 bytes. Six 32-bit floating-point stores each node, representing one AABB and two 32-bit integers storing all AABB primitives and one offset of a primitive. In addition, a split coordinate axis is stored to determine whether a node is a leaf node or an internal node, as shown in Figure 2. The coordinates are stored on the two least significant digits, and the node flag (Leaf = 1, internal = 0) is stored on the third significant digit.

![Figure 2. Structure of nodes, each node consists of 32 bytes](image)

Figure (a) represents the minimum point of each AABB bounding box coordinate, figure (b) represents the maximum point, Figure (c) represents the initial position of the tree, and Figure (d) represents the initial data. If the node is a leaf, the coordinate axis will be split.

After the nodes are determined, the next step is to create a complete binary tree. Since all 3D model elements are rendered, this process is accomplished by creating the complete tree described by regular equations 3.1--3.4, which calculate the size of each AABB’s component on the X, Y, and Z axes after splitting. Initialize each layer of the tree with MPI+OpenMP parallel processing from top to bottom, and this processing loop executes $N$ The time complexity is $O(N)$. 

4.3 Space allocation algorithm based on non-stack hierarchical bounding boxes

The space division is completed by the following space allocation algorithm:

**Step1: find out the maximum and minimum values of graph tuples on the coordinate axis;**

Take $\text{Maxaxis}[x][x] = \text{Max} - \text{Min}[x]$;

$\text{Maxaxis}[y] = \text{Max}[y] - \text{Min}[y]$;

$\text{Maxaxis}[z] = \text{Max}[z] - \text{Min}[z]$;

**Step2: Store the longest coordinate axis;**

Step3: Sort existing primitives by using the half-sort algorithm;

In the complete binary tree, the half-sort algorithm is performed, and the node points of the graph are qualified and parallel sorted by MPI+OpenMP.

**Step4: Divide the graph element for the reconstructed tree;**

Before partition, the number of graph primitives is calculated first. After sorting the calculation results, the graph primitives are allocated to the left and right subtrees respectively.

**Step5: Call parallel MPI+Open MP parallel algorithm, assign graph elements to complete binary tree, return Step1, and find the longest axis.**
5. Experimental results and analysis

The following is the acceleration process of the linear hierarchical bounding box 3D model proposed in this paper based on OpenMp without stack and experimental data of the rendering model. Compared with the references [7] and [8], the construction time of the accelerated structure, the frame frequency of rendering, the memory occupancy and the ray tracing efficiency are also compared. The experiment was conducted under the environment of 3.19ghz Intel Core i7 CPU, 6GB DDR RAM and NVIDIA Geforce GTX780 GPU.

Figure 3. It is based on a full linear hierarchical bounding box 3D accelerated rendering model with no stack

The comparative experiments were carried out by using 73,632 drawing elements of warrior model, 338,157 drawing elements of architectural model, 378,658 drawing elements of robot model, 1,357,892 drawing elements of concept car model and 2,379,236 drawing elements of Peony model. Figure 3 shows the ray-tracing algorithm [8] to perform a 3D model rendering process, and real-time application in this paper, based on no stack completely linear bounding box of the acceleration of process, through the contrast experiment shows the superiority of the proposed accelerating structure as shown in figure 4, table 1 analysis of the structure performance of the proposed optimization.

Figure 4. Mean construction time (MS) of three different accelerated structures

Table 1. Comparison of construction time of different accelerated structures (MS)

| Model of accelerated structure | Samurai | Architecture | Robot | Concept car | Peonies |
|-------------------------------|---------|---------------|-------|-------------|---------|
| Hierarchical bounding box algorithm | 141     | 536           | 766   | 1570        | 1835    |
| Linear hierarchical bounding box | 163     | 289           | 310   | 538         | 733     |
| Linear hierarchical bounding box based on non-stack | 38      | 137           | 187   | 368         | 576     |
5.1 Structural Performance

Figure 4 and Table 1 show the time required for the three algorithms to accelerate the process. When the three algorithms are run on the GPU at the same time, the rendering frame frequency of the three accelerated models can be seen. Based on the linear bounding box construction method without stack proposed in this chapter, the average cost of building the hierarchical bounding box tree of the 1M graph element 3D model is 0.33s. This construction method can be used in the dynamic reconstruction process to implement the ray tracing process of dynamic scene.

5.2 Ray tracking efficiency of the three structures

In this paper, the linear hierarchical bounding box method based on stackless is on average 30% higher than the intersecting detection bounding box of hierarchical bounding box and linear hierarchical bounding box, and 50% less than the graph element. The algorithm proposed in this paper requires more bounding boxes, because the traversal algorithm needs to check the bounding boxes of left and right subtrees, which also provides an advantage of reducing the intersection test of leaf nodes and providing better performance for the algorithm.

6. Conclusion

This paper presents a partition method for building high performance hierarchical bounding box. An efficient parallel construction process is used to quickly allocate the graph elements by the projection sequence of the longest bounding box plane, and a fast parallel segmented broken half sort algorithm is applied to the graph elements. The advantages of this partition method are optimized data parallel scanning, fast determination of the coordinates of the longest intersection of AABB envelop boxes, and good parallel construction time. The proposed method uses a position tracking traversal process to speed up ray tracing and improve frame rate, and gives a comparison process with two reference algorithms to compare the advantages and disadvantages of structure performance, rendering frame frequency, memory occupancy and average intersection test. The proposed partition and traversal method provides fast build time and high quality construction process, efficient memory storage and fast traversal implementation process. The future work will use this acceleration method combined with prediction and adaptation strategies to speed up the real-time rendering process of dynamic scenes.

7. Acknowledgments

This work was financially supported by Key Project of "The thirteenth-Five" science and technology research of Jilin Provincial Department of Education on Optical Motion Capture Technology "Rapid image generation and detection technology based on virtual reality", JJKH20190907KJ fund.

References

[1] WALD, I. 2010. Fast construction of sah bvhs on the intel many integrated core (mic) architecture. IEEE Transaction on Visualization and Computer Graphics.

[2] Garanzha, K., Pantaleoni, J., and McAllister, D. 2011. Simpler and faster hlbvh with work queues. In Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics, ACM, New York, NY, USA, HPG '11, 59-64.

[3] SULAIMAN, H. A., AND Bade, A. 2011. The construction of balanced bounding-volume hierarchies using spatial object median splitting method for collision detection. In International Journal of New Computer Architecture and their Application, 396-403.
[4] Hapala, M., Davidović, T., Wald, I., Havran, V., and Slusallek, P. 2013. Efficient stack-less bvh traversal for ray tracing. In Proceedings of the 27th Spring Conference on Computer Graphics, ACM, New York, NY, USA, SCCG ’11, 7-12.
[5] Zhefu, W., Hong, Y., and Bin, C. 2016. Divide and conquer ray tracing algorithm based on bvh partition. In Virtual Reality and Visualization (ICVRV), 2016 International Conference on, 49-55.
[6] Murguia, S., Avila, F., Reyes, L., and Garcia, A. 2017. Bit-trail Traversal for Stack-less LBVH on Direct Compute, 1 ed. CRC Press, 319-335.
[7] Pharr, M., and Humphreys, G. 2018. Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
[8] Garcia, A., Avila, F., Murguia, S., and Reyes, L. 2012. Interactive Ray Tracing Using DirectX11 on the Compute Shader, 1 ed. A K Peters/CRC Press, 353-376