

Received July 10, 2019, accepted July 20, 2019, date of publication July 31, 2019, date of current version August 15, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932151

Parallel BVH Construction Using Locally Density Clustering

YINGSONG HU, WEIJIAN WANG, DAN LI^{ID}, QINGZHI ZENG^{ID}, AND YUNFEI HU

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Dan Li (lidanhust@hust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61502185 and in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFYXJJ071.

ABSTRACT A novel bounding volume hierarchy (BVH) construction method based on locally dense clustering (LDC) was proposed for the low quality of BVH constructed in a complex scene with uneven distribution of primitives. The quality of the BVH was effectively improved by defining primitive density, analyzing the relation between density and traversal efficiency, selecting primitives with high density as the clustering center at the early stages of construction, and combining with top-down iterative clustering and bottom-up agglomerative clustering (AC). In order to effectively calculate primitive density, a local search strategy based on Morton coding was adopted. This strategy can quickly get approximate primitive density information through GPU. We evaluate the method and show that, in the complex building scene with uneven distribution of primitives, our method is 13% higher in traversal speed and 11% lower in surface area heuristic (SAH) cost. That means our method can improve the rendering speed of ray tracing effectively.

INDEX TERMS Primitive density, Morton code, local search, parallel Construction.

I. INTRODUCTION

Ray tracing is a global illumination rendering algorithm. Its physical principle based on ray propagation can better render the light phenomena of the scene such as blurring, shading, and highlighting, thus making the image more realistic and closer to the goal of photo-quality image pursued by humans. Although ray tracing has many advantages, it also has limitations. It is a computationally expensive recursive operation and its key is to find the nearest intersection between a given ray and the scene. In order to get high-quality renderings, a lot of rays will certainly be traced. In the actual calculation process, billions of rays may be required for the intersection test with millions of primitives in the scene. Therefore, we must organize and manage the 3D scenes with specific acceleration structures to improve computational efficiency. The most common acceleration structure is BVH. Compared with other acceleration structures based on space division, such as uniform lattice and K-D tree, the BVH has the following advantages: (1) Each triangular facet is stored only one reference in the BVH, thus the BVH has a predictable memory usage. (2) The BVH can better process

dynamic scenes by adjusting the topology. In recent years, many scholars have improved and optimized the construction algorithm of the BVH. Currently, the construction methods of the BVH can be roughly divided into three categories: top-down, bottom-up and incremental construction methods. Among these methods, in terms of SAH cost, the bottom-up construction method can generate high-quality BVH [1]. Recently, some novel construction ideas have been proposed. Weller *et al.* [2] proposed to construct BVH on the GPU by the clustering algorithm Batch Neural Gas (BNG) of machine learning. However, their method is not for the 3D scenes represented by the mesh model, but for those represented by the voxel. The construction of BVH is to divide the graphics primitives in 3D scenes and actually is a hierarchical clustering problem. Therefore, Meister and Bittner [3] first applied K-Means clustering method to the construction methods of BVH and completed the construction on the GPU. Meanwhile, tests have shown that their results are comparable to the most advanced methods available today.

In this paper, we proposed, by analyzing the distribution of primitive density, to construct BVH on the GPU by the method based on LDC. Our method selects the clustering center by calculating the local primitive density to construct a BVH by top-down iteration and then construct the final BVH

The associate editor coordinating the review of this manuscript and approving it for publication was Noor Zaman.

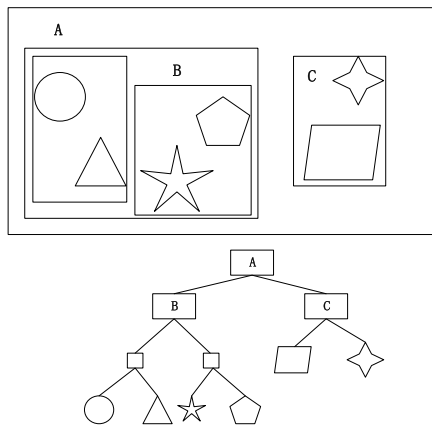


FIGURE 1. An example of the bounding volume hierarchy in 2D. The root nodes in this figure represent the bounding boxes in the whole scene; A, B and C represent the bounding boxes of geometric primitives, forming the internal nodes of the BVH.

by simple AC on this basis. The experimental comparison showed that our method is equivalent to that of Meister and Bittner [3] in terms of construction speed. Meanwhile, in most scenes with uneven distribution of primitives, our SAH cost, traversal speed, end-point overlap (EPO) [4] and other quantitative criteria are superior to their method.

II. RELATED WORK

As an object-based scene management technique, the BVH has been widely applied in collision detection and ray tracing. It can generate ordered structured data, i.e. a binary tree, from the polygon soup with no structure or topological information by following certain strategies. The BVH can organize scenes in a hierarchical tree structure. Leaf nodes contain geometric primitives; the internal nodes represent the bounding box containing all descendants; root nodes are expressed as the bounding box containing the whole scene. The schematic diagram of BVH is shown in Figure 1.

Based on BVH, a tree structure, the average ray intersection complexity of ray tracing decreases from $O(n^2)$ to $O(\log n)$. Currently, the construction methods of the BVH can be roughly divided into three categories: top-down, bottom-up and incremental construction algorithms.

A. TOP-DOWN CONSTRUCTION ALGORITHM

This method is the earliest method and starts with the root nodes that initially contain all primitives and recursively divides the set of primitives into two subsets according to a certain strategy until all leaf nodes are processed. In 1980, Rubin and Whitted [5] first proposed the concept of BVH and manually constructed the BVH acceleration structure of 3D scene objects. Afterwards, Weghorst *et al.* [6] proposed to construct the BVH with modeling hierarchy. The selection of each internal node in the BVH has $O(n)$ selection modes. The key problem is how to select the division mode. Kay and Kajiyi [7] designed and used the BVH construction algorithm based on median division. In this algorithm, by selecting the

bounding boxes of the geometric primitives closest to the two sides of the axis, a segment is obtained by connecting the centers of the two bounding boxes; the midpoint of the segment is selected as the division point, with the left part divided into the left treelet and the right part into the right treelet. However, this method has poor performance for the circumstance where objects are unevenly distributed. Later, Goldsmith and Salmon [8] proposed the SAH cost model, the main idea of which is that on the selected divisional plane, it should be possible to bound more geometric primitives with fewer domains. The division is carried out according to this strategy. The smallest divisional plane of the SAH cost is selected at each level of BVH construction, with the final results constructed in a top-down and greed manner. This method can obtain a superior balance between the traversal times and intersection times of rays, thus the acceleration structure with higher quality can be obtained. However, since the computational complexity of the SAH cost is $O(n^2)$ and almost every node needs to be calculated, the construction time of this cost model is much longer. Within a period afterwards, many scholars have improved the SAH-based top-down construction algorithm. For example, Wald [9], Wald *et al.* [10], and Ize *et al.* [11] used the approximate SAH cost function of binning thought for BVH construction; and Hunt [12] recommended using the structure of the scene graph to accelerate the BVH construction process and reduce the computational complexity. Dammertz *et al.* [13] proposed to use the BVH with a high branching coefficient to make better use of the SIMD units of modern CPU.

With the rapid development of modern CPU and GPU, many top-down parallel BVH construction methods based on multi-core CPU and GPU have been proposed accordingly. Lauterbach *et al.* [14] held that the objects that are adjacent in space should be divided into the same level of the bounding box, thus they put forward a GPU method based on Morton code. This method is known as LBVH and is deemed as the fastest BVH construction algorithm. However, the BVH constructed by this method is of low quality. Pantaleoni and Luebke [15] expanded the LBVH and proposed the HLBVH method. This method divides the construction process into the upper and lower levels. The construction method based on the SAH cost model is adopted in the higher level and the parallel construction method based on Morton coding in the lower level, thus improving the quality of the BVH to some extent. Apetrei [16] optimized the LBVH by reducing the number of iterations. Afterwards, the studies on the BVH construction based on Morton code did not stop. Vinkler *et al.* [17] extended on the basis that the original Morton code only encodes the spatial coordinates, increased a coding bit of geometric primitive size. The objects with more primitives can be conveniently split out via the primitive size bit at a higher level during the construction as soon as possible. The top-down method is advantageous in the construction speed. However, since the divisional plane selected is optimal in the current step, it is difficult to obtain the global optimal results.

B. BOTTOM-UP CONSTRUCTION ALGORITHM

Different from the top-down method, the bottom-up method focuses on the selection of the global optimal divisional plane. This method starts with leaf nodes and combines the primitives in pairs according to different combination methods to form an internal node of BVH until the final root node is completed.

Walter *et al.* [18] first used Agglomerative Clustering (AC) algorithm to construct a BVH. Compared with the top-down construction algorithm, AC can obtain the BVH with minimal SAH cost, but it also takes a longer time. The AC time complexity is $O(n^3)$. Later, Gu *et al.* [19] improved this algorithm and proposed approximate agglomerative clustering (AAC) algorithm. Their method first conducts space division of the 3D space. The AC algorithm is separately performed in each space after the division, thus the calculation amount of AC method can be reduced; the treelet with the lowest local cost in each part is approximately obtained, thus improving the construction speed. Inspired by this idea, Meister and Bittner [20] proposed the bottom-up ordered clustering construction algorithm PLOC. The clustering step was completed by bottom-up iteration and by assigning the two primitives that are close to each other into the same cluster. Although this method is slower than the top-down method in terms of construction speed, the BVH constructed is of higher quality and is more suitable for complex scenes.

C. INCREMENTAL CONSTRUCTION ALGORITHM

The incremental construction method is quite different from the former two methods and is based on the existing BVH structure. On this basis, certain strategies are optimized to obtain a BVH with higher quality. Its main research direction also focuses on optimizing strategies. Bittner *et al.* [21] significantly optimized the SAH cost by the deletion, search and insertion operations of nodes. Karras and Aila [22] successively proposed to optimize the quality of the BVH by means of the topological modification of the existing BVH. Meister and Bittner [23] completed the incremental construction on the GPU on this basis, improving the construction speed. Hendrich *et al.* [24] proposed a progressively refined cut strategy, by which the refined cutting is performed on the BVH constructed by LBVH method. This method can reduce the workload involved in each step in the construction process. Currently, the incremental construction method is more suitable for dynamic scenes with object updates. Its disadvantage is that it takes a longer time to construct and relies on the existing BVH construction methods.

In conclusion, the construction methods of BVH are still continuously studied. How to ensure real-time construction and the quality of BVH is still the major problem at present. On the one hand, although the top-down method has high construction speed and is convenient and parallel, the quality of the final BVH is not as high as that constructed by the bottom-up method. Therefore, this method is not suitable for complex scenes. On the other hand, the construction speed

of bottom-up and incremental methods is much lower than that of the top-down method. Considering the advantages and disadvantages of the three methods, our method started in two dimensions. In the horizontal dimension, the fast iterative clustering is performed on the nodes at each level of the BVH; in the vertical dimension,

AC is conducted on the treelet to optimize the quality of the BVH. Our method maintained high-quality BVH structure while ensuring the construction speed by the separate parallel processing in two dimensions, and showed excellent performance in complex scenes with uneven distribution.

III. PRIMITIVE DENSITY

A BVH with the higher quality should have the following characteristics: (1) higher construction speed: the timeliness of the construction speed is the basis to ensure the timeliness of ray tracing; (2) lower SAH cost; and (3) lower spatial overlap. The excessive spatial overlap is a major cause for the low quality of BVH. The higher the “overlap” of the bounding boxes corresponding to the two treelets, the more likely it is that a ray will hit both treelets as it passes through the region. This means that both treelets have to undergo intersection test and that the costs of traversal and intersection tests will be increased accordingly, thus lowering the final rendering efficiency. On this basis, we established a primitive density model via the experimental contrastive analysis of the relation between primitive density and the traversal speed of BVH. This paper first introduces the definition of primitive density and then the relation between primitive density and traversal efficiency.

A. DEFINITION OF PRIMITIVE DENSITY

This paper defines primitive density by analyzing the distribution of geometric primitives in 3D scenes to determine the closeness between primitives. In a given 3D scene model with a size of n , n represents the number of primitives, i.e. $C = \{C_1, C_2, \dots, C_n\}$. The search radius r is defined. For any primitive $C_i, i \in n$. A sphere is created in the 3D space with the primitive center as the center of sphere and the search radius r as the radius Q_i , then the density ρ_i of the primitive C_i is defined as follows:

$$\rho_i = \sum_{j \in C/C_i} \chi(\text{dist}(C_i, C_j) - r)$$

$$\chi(d) = \begin{cases} 1, & x \leq 0 \\ 0, & x > 0 \end{cases} \quad (1)$$

where, dist represents the Euclidean distance between the two primitive centers. For the sake of convenient calculation, the distance between the centers of primitive bounding boxes is simply used to replace the Euclidean distance. The specific definition of the primitive density ρ_i expressed by this equation is as follows: the number of other primitives contained in the sphere with the primitive center as the center of sphere and with a radius of r . Figure 2 is the two-dimensional (2D) diagram of primitive density.

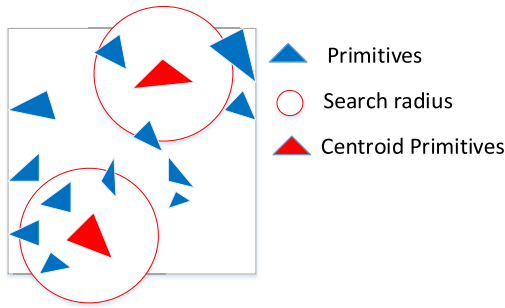


FIGURE 2. 2D diagram of primitive density. In this figure, the triangles represent primitives, red circles the circle with a radius of r and with the center of the primitive as its center. It can be seen from the above definition of primitive density that the density of the upper red primitives is 3 and that of the lower red primitives 4.

According to the definition of primitive density, the greater the primitive density, the greater the number of adjacent primitives in space, the higher the closeness between primitives, and the more likely it is to be the center of the objects in the scene. The density map of the scene can be obtained by calculating the density of each primitive by the primitives in the traversal scene. Figure 3 provides the density and heat maps of the three scenes under three different fields of view.

B. RELATION BETWEEN PRIMITIVE DENSITY AND TRAVERSAL EFFICIENCY

When the left and right nodes of BVH have high spatial overlap, in the stack-based traversal process, the left and right nodes will be pushed into a stack for traversal to determine whether a ray intersects with the nodes. Therefore, when this happens in the higher level of the BVH, the traversal of the corresponding several levels may be needed to determine the nearest intersection, which increases the computation overhead of traversal steps and primitives intersections. The following conclusion is drawn by calculating the Pearson correlation coefficient between primitive density and traversal times: there is a certain degree of correlation between primitive density and traversal times. For primitives with greater primitive density, their computation overhead in the ray traversal process is also greater. Figure 4 provides the correlation between primitive density maps and traversal heat maps.

During the experiment, we selected 6 scenes with different complexity; for each experimental scene, we selected 5

different views for ray traversal. The construction method of the BVH was LBVH, the fastest method at present. The traversal times of each ray in the BVH, the density of the nearest primitive that intersects with the ray were recorded respectively to generate the traversal heat map T and the density heat map G , respectively. The correlation coefficient between T and G was obtained by calculating the color distribution between the two heat maps:

$$Correlation := \frac{N \sum x_i y_i - \sum x_i \sum y_i}{\sigma_x \sigma_y} \quad (2)$$

where, x_i and y_i represent the color values of the traversal heat map T and the density heat map G at pixel I , $\sigma_x = \sqrt{N \sum x_i^2 - (\sum x_i)^2}$, $\sigma_y = \sqrt{N \sum y_i^2 - (\sum y_i)^2}$. We calculated the correlation coefficient between ray traversal times and primitive density by experiment. The data is shown in Table 1.

TABLE 1. Correlation coefficient between density and traversal efficiency.

Scene	Correlation
Sponza	0.896
Sibenik	0.831
Crytek Sponza	0.767
Conference	0.859
Soda Hall	0.739

According to the calculation of the correlation coefficient, the primitive density is deemed to be significantly correlated to the speed at which the ray traverses the primitive. The reason for this is that since the primitive density is great and there is a lot of overlap with surrounding primitives when constructing the BVH, the overhead of traversal times will be increased accordingly in the traversal process. On this basis, this paper proposed a BVH construction method based on local primitive density clustering. This method can position the clustering center to the model center in combination with the density distribution information of geometric primitives, separate the primitives with greater density from other primitives at the higher level of hierarchical clustering, and can effectively improve the quality of the BVH, make the

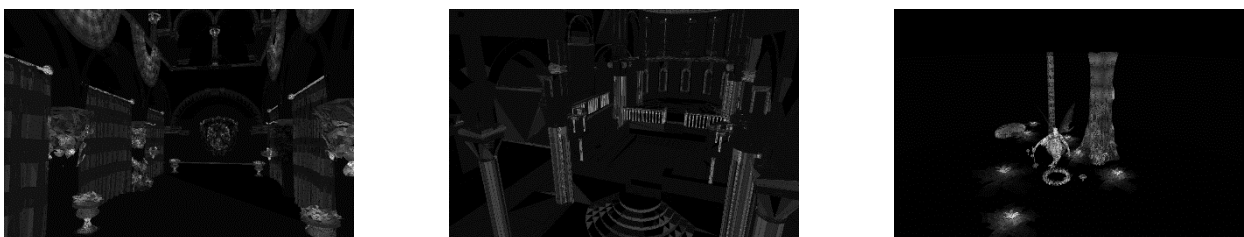


FIGURE 3. Sponza (262K) (Left), Sibenik (752K) (Middle) and FairyForest (174K) (Right) and their density maps in the corresponding three views the white part represents that the density of the region is greater.

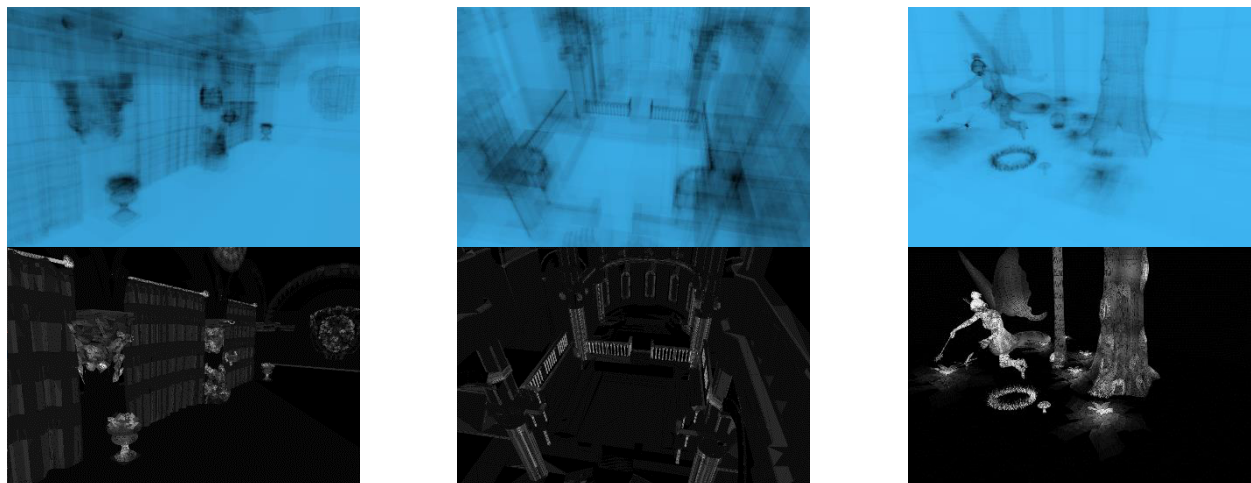


FIGURE 4. Traversal heat map (Upper). The darker the color, the more the traversal times at the primitive. The lower one is the primitive density map obtained by calculating the primitive density. The brighter the color, the greater the primitive density. The left, middle and right are models Sponza (262K), Sibenik (752K) and FairyForest (174K), respectively.

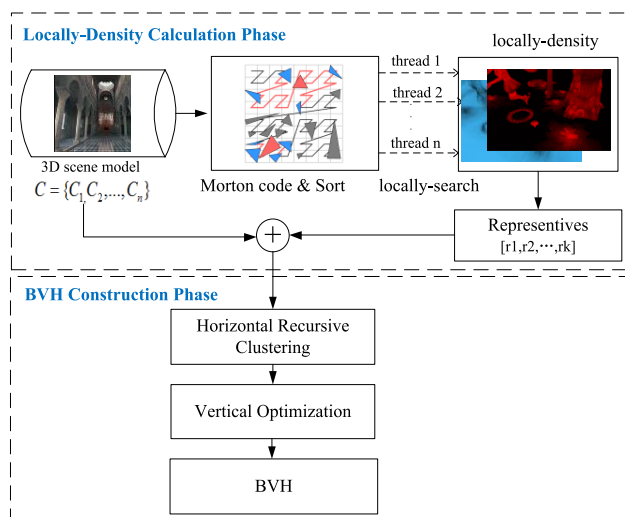


FIGURE 5. Frame diagram of the algorithm. In this figure, our algorithm is mainly divided into two phases: Local density calculation and BVH construction phases.

bounding box more compact and reduce redundant space. The steps of the algorithm proposed in this paper will be introduced in detail.

IV. BVH CONSTRUCTION VIA LDC

This paper focuses on the construction of BVH by the LDC method. The overall framework is shown in Figure 5.

This method can be roughly divided into two phases:

1) Calculation of local density. We found in the study that the computation overhead of the density is very high. The time complexity is $O(n^2)$. In the test, we adopted a local density calculation method based on Morton coding. We can get the approximate local density map of the whole scene by the local search strategy. Later, k primitives with the greatest local density were selected as the initial clustering center to be entered into the next phase.

2) BVH construction. This phase can be specifically divided into two subphases: horizontal iterative clustering and vertical optimization subphases. (1) Horizontal iterative clustering Each iterative process clusters the primitives currently processed and assigns them to the corresponding clusters represented by the clustering center according to the distance formula. All primitives in the scene can be divided into K clusters by an iteration. The K clusters form a one-level structure in the BVH. Later, each cluster is iteratively re-clustered to obtain a k -ary tree. (2) Vertical optimization For the k -ary tree obtained in the previous phase, a simple AC method is used to optimize it into a complete binary tree in a bottom-up manner, thus the BVH construction of the whole scene is finally completed.

In the next part, we will specifically describe the detailed process of these two steps and later provide the process outline of the whole algorithm.

A. CALCULATION OF LOCAL PRIMITIVE DENSITY

In the study, we found that the calculation of the density of 3D scene is time-consuming for the calculation of the density of a primitive needs to traverse the distance between this primitive and all other primitive in the scene. The time complexity is $O(n^2)$. It obviously cannot meet the speed of BVH construction. In fact, we do not need to traverse all primitives, when the center distance of a certain primitive is greater than the density radius r , we can sort out all the primitives after that primitive for the center distance of such primitives must be greater than the density radius r . For this purpose, we adopted a local density calculation method based on Morton code. This method can significantly improve the calculation efficiency by performing parallel computing on the GPU. The calculation process is roughly divided into two steps: (1) sorting all primitives in the space with Morton curve; and (2) performing a local search in sorted primitives and calculating the approximate primitive density, i.e. the local density. Next, we will specifically describe the local

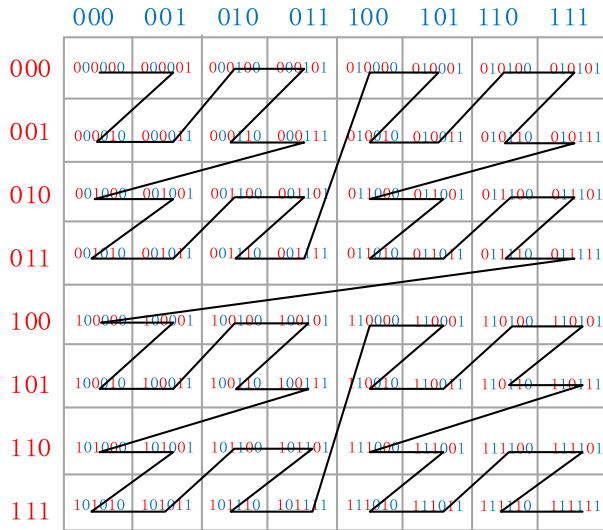


FIGURE 6. 2D diagram of Morton code. The continuous Morton curve on a 2D plane is obtained by the sequential connection of Morton code.

density calculation process and provide the error between the approximate density and the actual density by error analysis.

1) SPATIAL SORTING BASED ON MORTON CODE

Morton code is a coding method that maps quantized multi-dimensional vectors into one-dimensional (1D) vectors and can maintain the local characteristics of data points. The Z-shaped space-filling curve can be obtained by encoding and sorting the spatial uniform grid by means of Morton code. The Z-shaped curve has a certain spatial locality, that is, the spatially adjacent grids are adjacent on the mapped Z-curve. Compared with other space-filling curves with a better spatial locality, such as the Hilbert curve, the calculation method of Z-shaped curved is simpler and more efficient. This paper used Morton code for the spatial sorting of the primitives in the 3D scene. The Z-shaped curve in 2D spaces is shown in Figure 6.

We calculated the Morton code corresponding to the midpoint of all primitives on the GPU in a parallel manner. Next, we used the RadixSort on the GPU to sort such data, thus a spatially coherent collection of primitives as shown in Figure 7 can be obtained.

2) LOCAL SEARCH

For each primitive, the local density is needed to approximately replace the density, so a local search radius r is defined in this paper. We calculated the local density of the primitive by local search radius r on the 1D Z-order. For example, for the primitive C_i with an index of i , we searched the primitive within the range $(1 - r, i + r)$ along the sorted Z-order, and calculated the density of the primitive C_i by the defined density calculation formula. See Figure 8 for an example.

The density calculation formula is as follows:

$$\rho_i = \sum_{j \in (i-r, i+r)/i} d(C_j, C_i) = \sum_{j \in (i-r, i+r)/i} e^{-dist(C_j, C_i)} \quad (3)$$

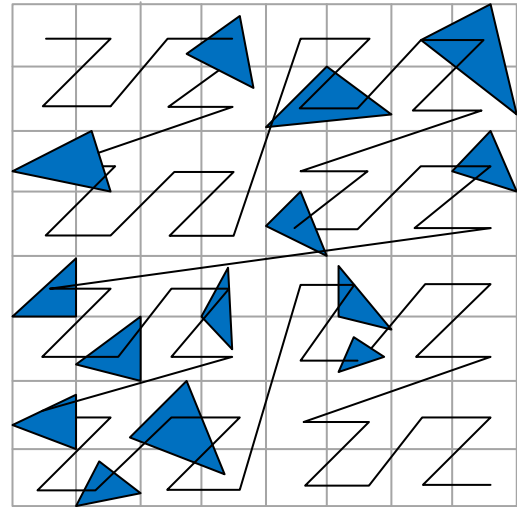


FIGURE 7. Collection of sorted primitives.

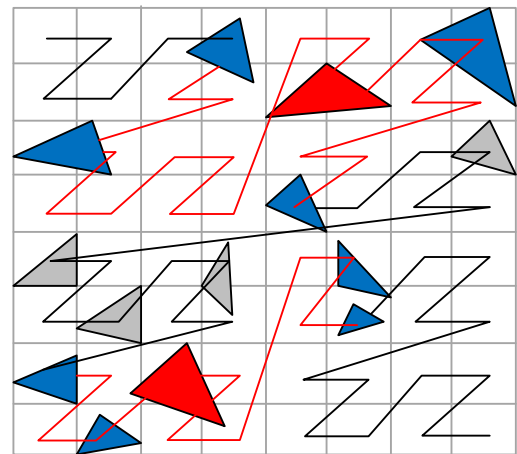


FIGURE 8. Schematic diagram of local search. In this figure, all primitives have been ordered in space by Morton code and connected by Z-order. In this figure, the search radius $r=2$; the red primitives are those needing density calculation; and the blue ones are those within the search radius.

where, ρ_i represents the local density of the primitive C_i . It should be noted that this formula is different from the formula mentioned in Section 3 in that the density value calculated by the latter is a discrete value. We found in the test that the probability of repetition is very high, thus we converted it into a continuous value by a gaussian kernel function to lower the continuous value. Besides, in terms of the function of the distance between two primitives $dist$, we referred to the study of Meister and Bittner [3]. For the axis-aligned bounding box (AABB), one bounding box is represented by two poles b_{min} and b_{max} , which represents the minimum and maximum coordinates among all coordinate points of the bounding box. The bounding boxes b_1 and b_2 of two geometric primitives are given to define the distance function:

$$d(b_1, b_2) = \|b_1^{min} - b_2^{min}\|^2 + \|b_1^{max} - b_2^{max}\|^2 \quad (4)$$

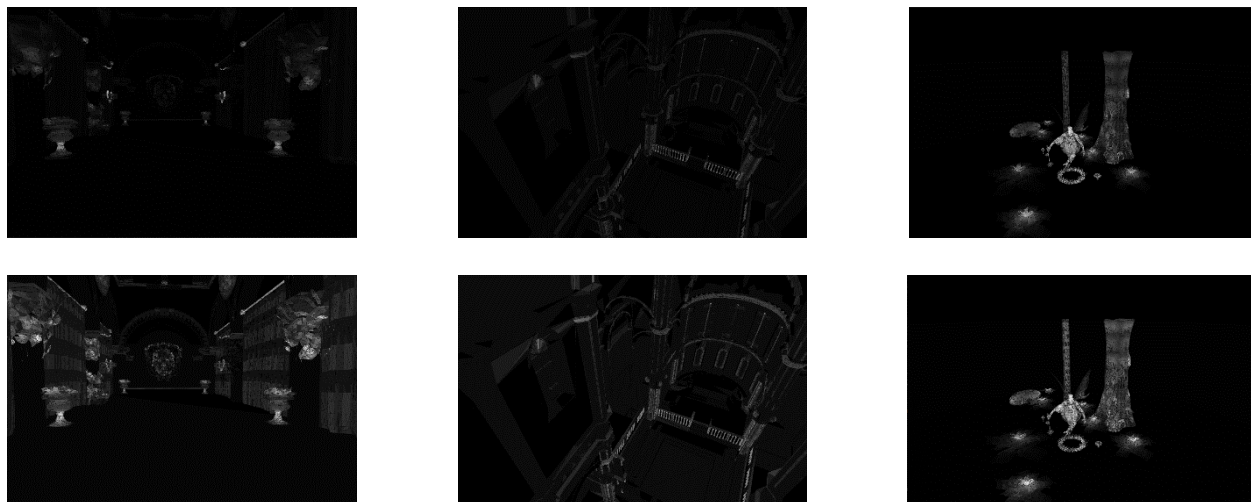


FIGURE 9. Approximate local density map with a search radius $r=100$ (Upper) and the density map (Lower). The brighter the color, the higher the primitive density there. The left, middle and right are models Sponza (262K), Sibenik (752K) and FairyForest (174K), respectively.

The experiment of Meister and Bittner [3] showed that the distance function corresponds to the most stable results of the final BVH constructed. In the actual calculation process, we assigned a thread for each C_i on the GPU and completed the parallel computing process of the local density of all primitives. See Algorithm 1 for the calculation pseudocode of this process.

Suppose that n represents the number of scene primitives entered, r is the search radius, and p is the number of thread cores assigned upon parallel computing. Next, we will specifically analyze the average time complexity of each method of calculating local density under serial computation and parallel computing.

In the steps of the calculating scene bounding box and Morton code, the serial version needs to traverse all primitives entered, with average time complexity of $O(n)$. The complexity can be lowered to $O(n/p)$ by the parallel computing on the GPU. In the step of sorting the Morton code, the time complexity of the use of quick sorting by the serial method is $O(n \log n)$. The complexity can be lowered to $O(\log n)$ by the radix sorting of the GPU. The thought of local density calculation is the core of our algorithm. This step needs to perform a local search for each primitive, thus the time complexity of this step under the serial version is $O(n)$ and the average time complexity under the GPU $O(n/p)$. In conclusion, the average time complexity under the serial version is $O(n \log n)$ and that under the parallel version $O(n/p)$.

3) ERROR ANALYSIS

By the above-mentioned two steps, we can obtain the local density of primitives. In fact, the Z-curves formed by Morton code are not completely ordered and adjacent. The existing singular points will cause errors. Therefore, the quality of the approximate density was evaluated by calculating the relative errors between the density and the local density in the test.

Algorithm 1 Pseudocode for Computing Locally-Density

Result: Locally Density $P_i=[P_1, P_2, \dots, P_n]$

initialization;

Input: $C_i=[C_1, C_2, \dots, C_n]$

compute scene box

BBOX sceneBox;

for $i=0$ to n

sceneBox = sceneBox.grow(C_i .getBBox());

end for

compute Morton-Code

$M_i=[M_1, M_2, \dots, M_n]$

for $i=0$ to n in parallel do

$M_i = \text{compute30MortonCode}(C_i.\text{centroid});$

end for

radix sort By Morion code

$C=CUB::\text{RadixSort}(C,M);$

compute locally density

for $i=0$ to $n-1$ in parallel do

for $j=\max(i-r,0)$ to $\min(i+r,n)$

if $i \doteq j$ then

$P_i += \text{CalculateDensity}(C_j,C_i);$

else

continue;

end for

end for

return $P;$

Figure 9 provides the density map G and the local density map G_m of some scenes.

The calculation formula for the relative error RE of the local density is defined as follows:

$$RE = \frac{\sum_{i \in n} |\rho_{r=n}(i) - \rho_{r=\lambda}(i)|}{\sum_{i \in n} |\rho_{r=n}(i)|} \quad (5)$$

TABLE 2. The relative error between local density and density.

Scene	RE		
	r=25	r=50	r=100
Sponza	0.37	0.29	0.25
Sibenik	0.41	0.32	0.29
Crytek Sponza	0.47	0.35	0.32
Conference	0.39	0.24	0.21
Soda Hall	0.36	0.19	0.17
San Miguel	0.42	0.33	0.30

where, n is the total number of primitives, $\rho_r = \lambda(i)$ is the local density of the primitive C_i when the search radius $r = \lambda$. When $r = n$, the value calculated is the density from the traversal of all primitives. Table 2 provides the relative error of local density of the nine experimental models when the search radius $r=25, 50$ and 100 . It can be seen from the relative error calculated by the experimental data that the relative error decreases with the increase of the search radius r and is already very small when $r=100$. The calculation amount increases with the increase of the search radius, so does the corresponding time consumption. Figure 10 provides the time consumption by the calculation of the local density under different search radii in the Conference model. With a progressive increase of 10 in the search radius, the time consumption is about 0.384 ms when $r=10$, about 1.825 ms when $r = 50$ and about 3.681 ms when $r = 100$. In general, the search radius is in a linear correlation with the time consumption, thus it can be concluded from the combination of the relative error and the time consumption that: the smaller the search radius, the greater the relative error, but the relatively shorter the time consumption; on the contrary, the greater the search radius, the more stable the relative error and the longer the time consumption. Therefore, the selection of an appropriate search radius, on the one hand, affects the final clustering effect, and on the other hand, affects the construction speed of BVH. In this paper, the search radius $r = 50$ was confirmed to be the appropriate search radius by experiments.

While calculating the local density property ρ_i of primitive, the max-heap and min-heap of size k is maintained by the shared memory within blocks; the k primitives with the highest local density are obtained by the merging between blocks; finally, the k primitives are entered into the next step as clustering centers.

B. CONSTRUCTION OF BVH

The density map of the scene and k initial clustering centers are obtained by approximate calculation and by the local density calculation method based on Morton coding. In this section, we will enter BVH construction phase which starts from the horizontal and vertical dimensions to obtain a K-ary

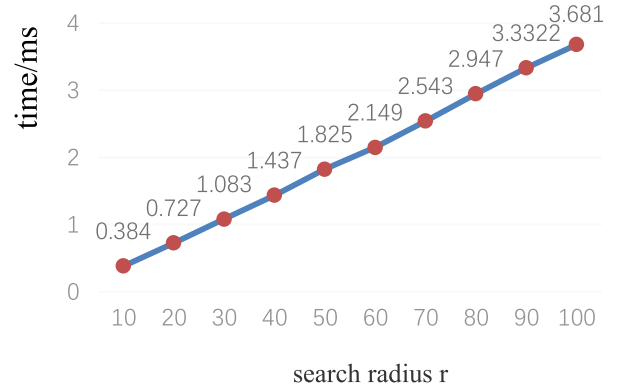


FIGURE 10. The time consumption of calculating density under different search radius in the conference model.

tree as an intermediate result by horizontal iterative clustering. Later, this K-ary tree is optimized to a binary BVH by vertical optimization.

1) HORIZONTAL ITERATIVE CLUSTERING

This phase is divided into three subphases according to different iteration phases: (1) initial clustering; (2) intermediate iterative clustering; (3) termination iteration.

a: INITIAL CLUSTERING

The k initial clustering centers calculated by local density are characterized by high density. We can know, by analyzing the correlation between primitive density and traversal efficiency, that, compared with other primitives, these primitives have high overlap with surrounding bounding boxes and that, in the BVH constructed, such primitive will not only affect the traversal results of the bounding boxes at the same level, but also affect all nodes from the root node to that primitive node. Therefore, it is better to separate these primitives from other primitives at the higher level of the BVH tree. For this purpose, the K primitives were selected as the clustering centers upon initial clustering, in order to separate these primitives at a higher level. The specific process is roughly as follows: all other primitives are traversed; the method used here is the same as that used to calculate the local density; on the GPU, a thread is assigned to each primitive on warp; the distance between the primitive and K clustering centers is calculated; the Formula 1 proposed by Meister and Bittner [3] is also used as the distance function. All primitive points are divided into the nearest centers to complete a clustering process and obtain the K clusters after clustering. The K clusters can obtain the first level of K-ary tree, i.e. K intermediate nodes of the BVH, while the parent node of these nodes is the bounding box node in the scene. Figure 11 provides the effects of initial clustering.

b: INTERMEDIATE ITERATIVE CLUSTERING

A level of the BVH can be constructed by dividing K clusters into the K internal nodes in the BVH, thus completing an iterative process. In this phase, K centers are randomly

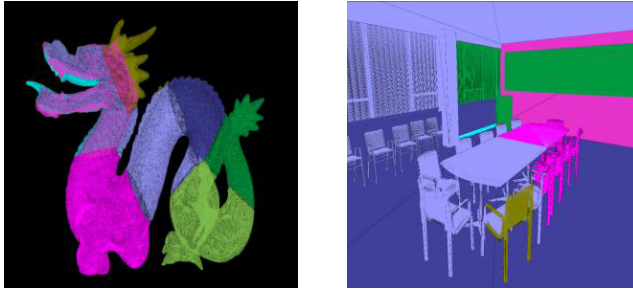


FIGURE 11. Schematic diagram of primary clustering segmentation. Parameter $K=8$; and the primitives belongs to different clusters are displayed in different colors.

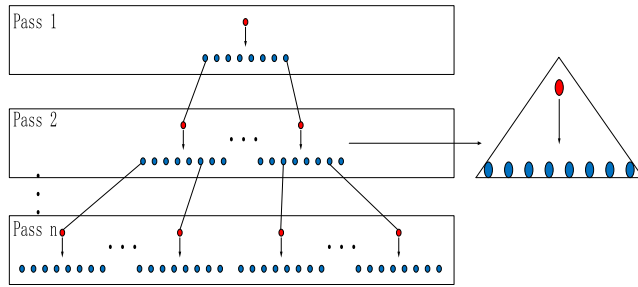


FIGURE 12. Schematic diagram of K-BVH construction. Pass n represents the n th horizontal clustering process. A level of the K-ary tree is constructed by each horizontal clustering process.

selected as the initial clustering centers of the intermediate iterative clustering process; the iteration parameter i is set; new cluster centers will be recalculated upon each iteration. The calculation formula for new cluster centers is as follows:

$$r_i = \bar{C}_i^i = \frac{1}{|C_i^i|} \left(\sum_{b_j \in C_i^i} b_j^{\min}, \sum_{b_j \in C_i^i} b_j^{\max} \right) \quad (6)$$

where $|C_i^i|$ is the number of primitives within the cluster; b_j is the bounding box of the primitive j ; the value calculated by the formula is the mean of the bounding boxes of the original clusters. When the number of iterations is greater than i , the iteration stops; when the number of iterations is set to 0, i.e. no cluster centers are calculated, this process is the same as the initial clustering. The clustering process at each level can be completed on the GPU by parallel computing by assigning thread. Two queues were used throughout the process: one as input queue and the other as output queue. The input queue is designed to process new unstarted tasks and the output queue to calculate and assign new tasks. At the beginning of the algorithm, the output queue has only one task called the initial clustering aimed to construct the upper level node of the BVH. When this task is fulfilled, it is dequeued to the output queue. Another corresponding relation between the index array storage primitive index and the queue was additionally used. Each task is entered into the input queue by calculating and assigning K new tasks. The complete process is shown in Figure 12.

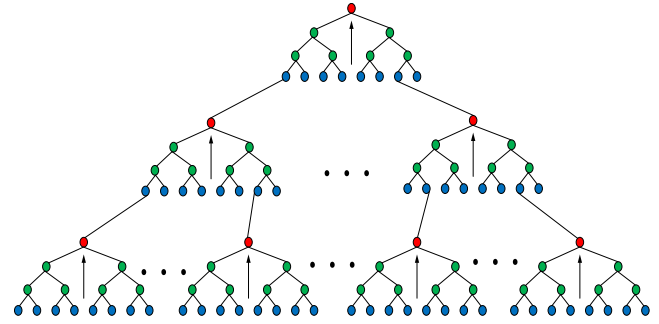


FIGURE 13. Schematic diagram of bottom-up AC. In this figure, the blue nodes are the leaf nodes before the processing with the AC method; the green ones are the intermediate nodes constructed by the AC method; the red ones are the root nodes finally constructed to surround all leaf nodes.

c: TERMINATION ITERATION

Blank nodes may be obtained in the clustering process, the reason for which is that the number of primitives in a cluster may be 0 in the process of assigning to the corresponding cluster. In order to avoid such condition, we set iteration termination rules, that is, when the number of the primitives in the cluster to be processed is smaller than $m \cdot K$, the iteration process under this thread is terminated and the node marked as a leaf node; m represents the maximum number of primitives contained in the leaf node.

2) VERTICAL OPTIMIZATION

The K-ary tree after clustering has been obtained by the above-mentioned method. Next, the bottom-up clustering will be conducted on this K-ary tree by the simple parallel AC algorithm to construct intermediate nodes. This part is called post vertical optimization. Since the bottom-up construction method starts with the leaf node, a discriminating method must be introduced to judge whether the two given bounding boxes are more suitable for constructing the new parent node as shown in Figure 13. The method is the same as that of Wald [9], Wald *et al.* [10], and Ize *et al.* [11]. We also adopted the SAH cost function as the standard for merging two enclosing boxes. The specific SAH cost function is as follows:

$$C(i) = S_L(i)n_L(i) + S_R(i)n_R(i) \quad (7)$$

where, $S_L(i)$ and $S_R(i)$ represent the left and right sub bounding boxes of the bounding box i . That is, the bounding boxes of the cost need to be judged. $n_L(i)$ and $n_R(i)$ are the number of primitives in the two bounding boxes. At the beginning of the AC algorithm, the cost of any two clusters is calculated successively with each bounding box as a cluster. The pair of clusters with the lowest cost is selected to generate their parent node and generate a new cluster with the bounding box of the parent node. Afterwards, the algorithm is restarted until there is only one cluster in the collection. Since we have obtained all treelets and they are not correlated, they can be conveniently processed in parallel.

V. RESULTS AND ANALYSIS

A. EXPERIMENTAL SCHEME

Three different methods and the method proposed in this paper were adopted in the experiment to carry out the BVH construction experiment on the 6 groups of 3D models with different complexity. Since the SAH cost function can only express the approximate expected cost of the whole acceleration structure of ray traversal and the ray may exit the traversal process very early in the actual traversal process, the SAH cost cannot be used as single evaluation criteria. Therefore, the time of BVH construction, SAH cost, traversal speed, overlap and other metrics in this test are recorded in this section for detailed analysis. The BVH construction methods in contrast to our method are as follows: (1) the LBVH proposed by Bittner *et al.* [21]. This method is currently the method with the highest speed of BVH construction on the GPU, but its disadvantage is that the quality of the BVH constructed is too low. The quality here refers not only to SAH, but also to traversal speed and EPO and other EPO metrics. (2) The HLBVH method proposed by Pantaleoni and Luebke [15]. In order to ensure the consistency of the experiment, 60-bit coding is used in the calculation method based on Morton coding of the HLBVH and the method proposed by us, thus the probability of repetition of Morton coding can be effectively lowered. (3) The BVH construction method based on K-means proposed by Meister and Bittner [3]. In the experiment, we adopted the optimal parameter setting proposed by them in their paper. (4) Ours provided appropriate parameters by the experimental contrastive analysis of the performance of different parameters in a fixed scene and by measuring the construction time and traversal efficiency: clustering center $K = 8$, search radius $r = 50$ and the number of iterations $i = 5$. According to the error analysis in Section 4.4, the greater the search radius, the longer the time for calculating the local density. Since the error is very small when the search radius $r = 50$, we hold that $r = 50$ is appropriate. Besides, the maximum number of primitives contained in the lead node m is set to 4. That is, when the number of the primitives in the cluster to be processed is smaller than $m \cdot K$, the termination iteration process is initiated. All experiments are completed on computers, the CPU of which is Ryzen 1700 3.7Ghz and the computing power of GTX1060 GPU is 6.1.

B. EXPERIMENTAL SCENE MODEL

In order to ensure the accuracy and versatility of the final conclusion, we used 6 groups of scenes with different complexity in this experiment. Such scenes are widely applied to computer graphics, such as ray tracing. They are Sponza, Sibenik, Crytek Sponza, Conference, Soda Hall and San Miguel, respectively. Their number of primitives increases in sequence. The Sponza with the smallest number of primitives contains 66K primitives, while the San Miguel with the greatest number of primitives has 7880K primitives. Meanwhile, their structures are complex; the primitive size changes greatly; they are building scenes with untessellated

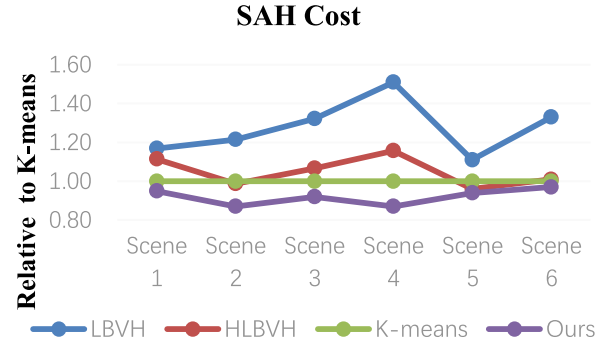


FIGURE 14. Broken line graph of SAH.

objects. See Table 3 for experimental results. Next, we will carry out contrastive analysis of the experimental results from construction speed, traversal speed, EPO and other metrics.

C. EXPERIMENTAL ANALYSIS

This paper compared the advantages and disadvantages of our method and the other three methods from the perspective of the four metrics: SAH cost, construction time, traversal time and EPO.

1) SAH COST

Our calculation formula for the SAH in the experiment is as follows:

$$c(N) = \frac{1}{S(N)} \left[c_T \sum_{N_i} S(N_i) + c_I \sum_{N_i} S(N_i) \right] \quad (8)$$

where, N_l and N_i represents the leaf node and the internal node, respectively; S represents the surface area; c_T and c_I represent the average time cost of the traversal of BVH by the ray and the average time cost of the intersection test between the ray and the primitive, respectively. In the test, we set c_T and c_I to 3 and 2 to ensure the consistency of experimental results. The experimental comparison diagram is shown in Figure 14.



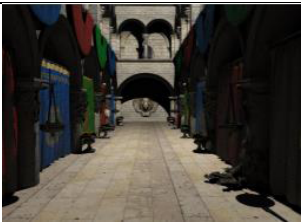

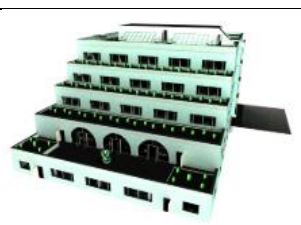
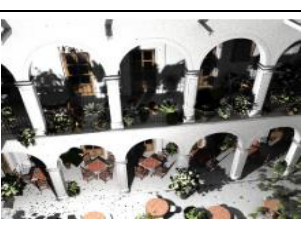
According to the diagram, our method has the lowest SAH cost in the six experimental scenes. The SAH cost of the LBVH is the highest in all test scenes. Our method has the best performance in scenes Sibenik and Conference. Its SAH cost is nearly 12% lower than that of K-means method, with an average cost reduction of 8%.

2) TRAVERSAL SPEED

The traversal speed reflects the rendering speed from the side and the traversal throughput of the ray in the whole scene. In the experiment, each pixel is sampled by emitting 16 rays in the same ray and from the same perspective; the total number of rays and the total time t from the emission of rays to the traversal of the nearest primitive by each ray are recorded to obtain the following traversal speed formula:

$$\text{Traversal_speed} = \frac{\text{number_of_rays}}{t} \quad (9)$$

TABLE 3. Performance comparison of the tested methods.

Scene	Metric	Builder			
		LBVH	HLBVH	K-means	Ours
 <p>Sponza 66k</p>	SAH cost	277	264	237	215
	EPO	12.26	8.39	20.19	6.28
	Trace speed	189	206	212	218
	Build time	1.8	11.7	9.0	12.6
 <p>Sibenik 75K</p>	SAH cost	192	156	158	140
	EPO	32.68	13.75	28.88	17.38
	Trace speed	175	192	190	200
	Build time	2.1	12.3	6.3	7.8
 <p>Crytek Sponza 262k</p>	SAH cost	279	225	211	195
	EPO	20.57	17.38	16.66	15.61
	Trace speed	140	155	167	169
	Build time	4.8	14.1	17.8	24.4
 <p>Conference 331K</p>	SAH cost	154	118	102	89
	EPO	21.68	9.79	11.51	7.46
	Trace speed	210	257	240	271
	Build time	5.8	13.5	8.7	11.5
 <p>Soda Hall 2169K</p>	SAH cost	252	219	228	215
	EPO	28.03	23.31	30.36	24.1
	Trace speed	189	197	191	195
	Build time	34.5	43.2	69.2	80.7
 <p>San Miguel 7880K</p>	SAH cost	267	204	201	195
	EPO	20.69	15.35	18.98	12.77
	Trace speed	81	96	108	110
	Build time	129.2	142.4	163.5	192.2

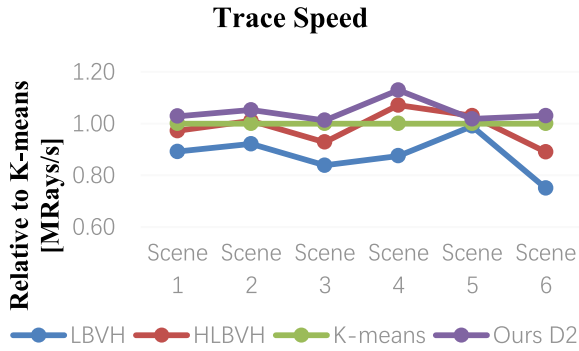


FIGURE 15. Broken line graph of trace speed.

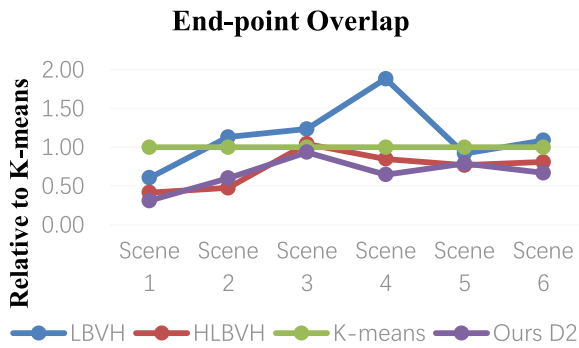


FIGURE 16. Broken line graph of EPO.

Figure 15 provides the comparison data of the traversal speed among all methods. It can be concluded from the experimental data that our method has the highest traversal speed in most scenes. Traversal speed is 13% higher than that of K-means method, with an average increase of about 4.5%.

3) EPO

EPO is the overlap metric of the BVH; it was first proposed by Alia *et al.* [4]. EPO is more descriptive than the metric proposed by Stich *et al.* [28] and Popov [29]. The calculation formula for EPO is as follows:

$$EPO = \sum_{n \in N} \frac{A((n \setminus Q(n)) \cap n)}{A(n)} \quad (10)$$

where, n is a certain node in the BVH, $Q(n)$ is a brother node of this node, $n \setminus Q(n) \cap n$ represents the collection that does not belong to the treelet n but overlaps with it; $A()$ is the corresponding volume. When bounding boxes of the branch nodes of the tree, the degree of overlap is zero. The smaller the EPO, the lower the overlap of the nodes in the BVH. Figure 16 is the broken line graph for the comparison between our method and other methods on the EPO.

Our method shows an average reduction of about 34% than the K-means method on the EPO and has the best performance in the EPO in the scene Sponza. This indicates that the overlap in the BVH constructed by our method is significantly lower and that the BVH is more compact.

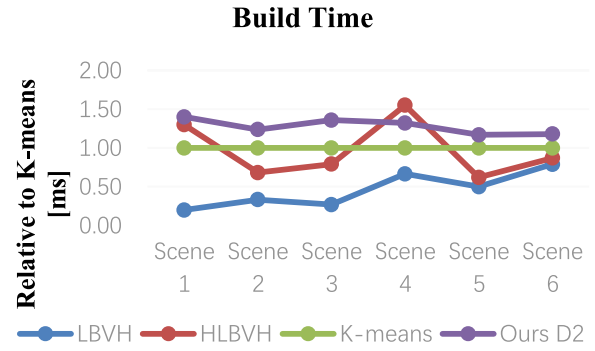


FIGURE 17. Broken line graph of build time.

4) CONSTRUCTION TIME

Construction time is an important standard for measuring a BVH. Figure 17 provides the comparison data of the construction time among all methods. This figure strongly shows that the speed of the LBVH is the highest among all methods; our method is slightly slower than the K-means method; the construction speed of our method in the scene Conference is slightly higher than that of the HLBVH. The main reason for this is that the computation overhead of the local density increases with the increase of scene complexity. Our method takes 192ms to construct the BVH in the scene San Miguel with the greatest number of primitives, and only 12 ms in the scene Sponza with the smallest number of primitives.

According to experimental results and compared with the other three advanced methods, the BVH constructed by our method is characterized with low SAH cost, low EPO and high quality in scenes with different complexity, can effectively reduce the traversal and intersection times of the ray and improve the traversal speed and the construction time, and is competitive with other methods.

VI. CONCLUSION

A novel GPU-oriented method to construct a BVH was proposed in this paper. The primitive density was first defined by this method. It used this property to optimize the quality of the BVH. This method is divided into three phases: calculation of primitive density, iterative top-down clustering and bottom-up AC. In the phase of calculation of primitive density, we used the sorted Morton coding for local search to effectively calculate the approximate density and the initial clustering center. Later, the top-down clustering and the bottom-up AC were combined to complete the BVH construction process on the GPU. Our algorithm was compared with LBVH, HLBVH, K-means and other methods in scenes with different complexity. It can be seen from experimental analysis that our method has the lowest SAH cost and EPO. The traversal speed is 13% higher than that of the K-means method at most. Meanwhile, it is competitive with other methods in the construction method. In the future, we will construct the multi-frame optimized BVH in dynamic scenes in virtue of the advantage of the primitive density in BVH

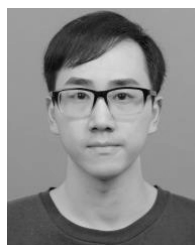
construction and the combination with the object motion information in such scenes.

REFERENCES

- [1] J. D. MacDonald and K. S. Booth, "Heuristics for ray tracing using space subdivision," *Vis. Comput.*, vol. 6, no. 3, pp. 153–166, May 1990.
- [2] R. Weller, D. Mainzer, A. Srinivas, M. Teschner, and G. Zachmann, "Massively parallel batch neural gas for bounding volume hierarchy construction," in *Proc. Workshop Virtual Reality Interact. Phys. Simulation*, 2014, pp. 1–9.
- [3] D. Meister and J. Bittner, "Parallel BVH construction using k -means clustering," *Vis. Comput.*, vols. 6–8, no. 32, pp. 977–987, Jun. 2016.
- [4] T. Aila, T. Karras, and S. Laine, "On quality metrics of bounding volume hierarchies," in *Proc. 5th High-Perform. Graph. Conf.*, Anaheim, CA, USA, 2013, pp. 101–108.
- [5] S. M. Rubin and T. Whitted, "A 3-dimensional representation for fast rendering of complex scenes," *Comput. Graph.*, vol. 14, no. 3, pp. 110–116, Jul. 1980.
- [6] H. Weghorst, G. Hooper, and D. P. Greenberg, "Improved computational methods for ray tracing," *ACM Trans. Graph.*, vol. 3, no. 1, pp. 52–69, Jan. 1984.
- [7] T. L. Kay and J. T. Kajiya, "Ray tracing complex scenes," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, pp. 269–278, Aug. 1986.
- [8] J. Goldsmith and J. Salmon, "Automatic creation of object hierarchies for ray tracing," *IEEE Comput. Graph.*, vol. 7, no. 5, pp. 14–20, May 1987.
- [9] I. Wald, "On fast construction of SAH-based bounding volume hierarchies," in *Proc. IEEE Symp. Interact. Ray Tracing*, 2007, pp. 33–40.
- [10] I. Wald, S. Boulos, and P. Shirley, "Ray tracing deformable scenes using dynamic bounding volume hierarchies," *ACM Trans. Graph.*, vol. 26, no. 1, Jan. 2007, Art. no. 6.
- [11] T. Ize, I. Wald, and S. G. Parker, "Asynchronous BVH construction for ray tracing dynamic scenes on parallel multi-core architectures," in *Proc. 7th Eurograph. Conf. Parallel Graph. Vis.*, 2007, pp. 101–108.
- [12] W. Hunt, W. R. Mark, and D. Fussell, "Fast and lazy build of acceleration structures from scene hierarchies," in *Proc. Symp. Interact. Ray Tracing*, Sep. 2007, pp. 47–54.
- [13] H. Dammertz, J. Hanika, and A. Keller, "Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays," *Comput. Graph. Forum*, vol. 4, no. 17, pp. 1225–1233, Jun. 2010.
- [14] C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha, "Fast BVH construction on GPUs," *Comput. Graph. Forum*, vol. 28, no. 2, pp. 375–384, Apr. 2009.
- [15] J. Pantaleoni and D. Luebke, "HLBVH: Hierarchical LBVH construction for real-time ray tracing of dynamic geometry," in *Proc. Conf. High-Perform. Graph.*, Saarbrücken, Germany, 2010, pp. 87–95.
- [16] C. Apetrei, "Fast and simple agglomerative LBVH construction," in *Proc. Comput. Graph. Vis. (CGVC)*, 2014, pp. 41–44.
- [17] M. Vinkler, J. Bittner, and V. Havran, "Extended Morton codes for high performance bounding volume hierarchy construction," in *Proc. Conf. High-Perform. Graph.*, Los Angeles, CA, USA, 2017, Art. no. 9.
- [18] B. Walter, K. Bala, M. Kulkarni, and K. Pingali, "Fast agglomerative clustering for rendering," in *Proc. IEEE Symp. Interact. Ray Tracing*, Los Angeles, CA, USA, Aug. 2008, pp. 81–86.
- [19] Y. Gu, Y. He, K. Fatahalian, and G. Blueloch, "Efficient BVH construction via approximate agglomerative clustering," in *Proc. 5th High-Perform. Graph. Conf.*, Anaheim, CA, USA, 2013, pp. 81–88.
- [20] D. Meister and J. Bittner, "Parallel locally-ordered clustering for bounding volume hierarchy construction," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 3, pp. 1345–1353, Mar. 2018.
- [21] J. Bittner, M. Hapala, and V. Havran, "Fast insertion-based optimization of bounding volume hierarchies," *Comput. Graph. Forum*, vol. 32, no. 1, pp. 85–100, Jan. 2013.
- [22] T. Karras and T. Aila, "Fast parallel construction of high-quality bounding volume hierarchies," in *Proc. 5th High-Perform. Graph. Conf.*, Anaheim, CA, USA, 2013, pp. 89–100.
- [23] D. Meister and J. Bittner, "Parallel reinsertion for bounding volume hierarchy optimization," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 463–473, May 2018.
- [24] J. Hendrich, D. Meister, and J. Bittner, "Parallel BVH construction using progressive hierarchical refinement," *Comput. Graph. Forum*, vol. 36, no. 2, pp. 487–494, May 2017.
- [25] V. Havran, R. Herzog, and H.-P. Seidel, "On the fast construction of spatial hierarchies for ray tracing," in *Proc. IEEE Symp. Interact. Ray Tracing*, Salt Lake City, UT, USA, Sep. 2006, pp. 71–80.
- [26] M. Ernst and G. Greiner, "Early split clipping for bounding volume hierarchies," in *Proc. IEEE Symp. Interact. Ray Tracing*, Ulm, Germany, Sep. 2007, pp. 73–78.
- [27] H. Dammertz and A. Keller, "The edge volume heuristic—Robust triangle subdivision for improved BVH performance," in *Proc. IEEE Symp. Interact. Ray Tracing*, Los Angeles, CA, USA, Aug. 2008, pp. 155–158.
- [28] M. Stich, H. Friedrich, and A. Dietrich, "Spatial splits in bounding volume hierarchies," in *Proc. Conf. High-Perform. Graph.*, New Orleans, LA, USA, 2009, pp. 7–13.
- [29] S. G. Popov, "Object partitioning considered harmful: Space subdivision for BVHs," in *Proc. Conf. High Perform. Graph.*, New Orleans, LA, USA, 2009, pp. 15–22.



YINGSONG HU received the Ph.D. degree in computer science and technology from the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2011, where he is currently an Associate Professor with the School of Computer Science and Technology. His research interests include image processing and computer vision.



WEIJIAN WANG received the bachelor's degree from Anhui University, in 2017, and the master's degree in computer science and technology from the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2019. His research interests include image processing and computer vision.



DAN LI received the B.E. and M.S. degrees in mechanical design, manufacturing and automation and the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1998, 2002, and 2008, respectively, where she is currently an Associate Professor with the School of Computer Science and Technology. Her research interests include computer graphics, multimedia, and intelligence.



QINGZHI ZENG received the bachelor's degree from Chongqing University, in 2018. He is currently pursuing the master's degree in computer science and technology with the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China. His research interests include computer vision and image processing.



YUNFEI HU received the bachelor's degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2018, where he is currently pursuing the master's degree in computer science and technology with the School of Computer Science and Technology. His research interests include computer vision and image processing.

...