Journal of Computers Vol. 31 No. 6, 2020, pp. 305-318
doi:10.3966/199115992020123106024

A Fast Calculation Method for Multiple Scattering of Rays Based on KD-Tree

Pei-Lei Zhang[1], Fu-Bing Li[1*], Wen-Jian Chen[2]

[1] School of Information & Communication Engineering, Beijing Information Science & Technology University, Beijing 100101, China
lifubing@bistu.edu.cn

[2] College of Underwater Acoustic Engineering, Harbin Engineering University, Harbin 150001, China

Received 15 August 2020; Revised 25 September 2020; Accepted 24 October 2020

Abstract. In this paper, a single-ended open rectangular parallelepiped cavity model is used as an example to calculate the multiple scattering propagation path of incident rays in the target. In order to improve the efficiency of ray tracing, a multi-layer KD tree structure of the target is constructed and the ray-triangle intersection is determined based on KD tree structure. To construct a KD tree, several steps are performed. First, the degree of triangle centroid dispersion of the target is separately calculated for three coordinate axes. The axis with the maximum dispersion is selected as splitting axis. Second, the 3D bounding box that containing the whole target (i.e., the root node) is divided into two sub-bounding boxes (i.e., the child nodes) along the splitting axis, each containing about half of target triangles. The above procedure is then repeated for each new-generated bounding box until the number of triangles in these boxes (i.e., the leaf nodes) satisfies the preset condition, and a KD tree is finally constructed. To calculate the ray-triangle intersection, the KD tree is firstly searched from the root node layer by layer to find the leaf node that intersects with the incident ray, and the ray-triangle intersection coordinate as well as the reflected ray can be calculated in the leaf box. Multiple scattering of the ray can be determined similarly by the steps above. Result indicates that under the parameter settings in this article, multiple scattering calculation based on KD tree can achieves a speedup of about 10 compared with the traditional method that calculates the ray-triangle intersection one surface by one surface. It also indicates that as the height of the KD tree increases, computation time gradually decreases and finally tends to stabilize.

Keywords: KD tree, multiple scattering, ray-triangle intersection algorithm, shooting-bouncing rays

1 Introduction

Ray-Tracing Method [1-2] is an approximate method for solving high-frequency electromagnetic field problems in a complex electromagnetic environment. It is mainly used for the prediction of radio wave propagation characteristics and the analysis of target electromagnetic scattering characteristics. The demands for the military and defense fields and civilian applications promote the research and development of electromagnetic scattering theory. Shooting and bouncing rays method [3-6] (SBR) is a kind of ray tracing technology, proposed by LING et al. in 1989. It was originally used to calculate the radar cross section [7-8] (RCS) inside an arbitrary cavity which was later extended to the calculation of electromagnetic scattering from complex targets with arbitrary shapes [9-10]. SBR combines geometric optics (GO) and physical optics (PO) methods. It has the advantages of clear physical concepts, high calculation accuracy, and simple implementation, so it has been widely used.

Shooting and bouncing rays method mainly includes two parts: ray tracing and electromagnetic calculation. In the process of ray tracing, SBR is similar to the ray tracing algorithm in computer graphics

[*] Corresponding Author

[11-12]. In the case of high frequency, the electromagnetic wave is abstracted as a line of rays. The launch point is regarded as the starting point of the ray and the propagation process follows the basic laws of geometric optics. Each ray propagates independently in a straight line, and specular reflection occurs when encountering a plane obstacle. The target object is composed of a large number of triangular facets, and the rays are scattered multiple times on the object. If no processing is done, each ray must intersect with all facets during the tracking process. There are multiple invalid intersection calculations in this process, resulting in low efficiency.

In recent years, domestic and foreign scholars have proposed a variety of acceleration algorithms for the ray tracing process [13-14], which are mainly divided into two categories. One is to optimize the performance of SBR by reducing the number of rays, such as the multi-resolution grid algorithm [15]; the other is to try to reduce the number of intersections between the ray and the facets. Jin et al. proposed a method based on adaptive space division using an octree structure [16], introducing the concept of bounding boxes, and evenly dividing the research scene area into 8 parts. This method achieves the effect of finely screening the intersection of rays and facets, but for unevenly distributed scenes, the acceleration effect is poor. Ren Jiamin and Tang Yaping introduced a ray tracing method based on visible walls [17-18], taking the radiation source point as the root node, and its visible surface as a child node. Each child node has its corresponding visible surface, which is then used as the next layer. Deeply traverse the visible walls tree to complete the calculation of the judgment of sifting out rays and invisible surfaces. This method relies on the relationship between the target surfaces and surfaces, and is only suitable for simple scenes. The dynamic partitioning method [19] proposed by Yuan Zhengwu is similar to the octree. It is converted from three-dimensional to two-dimensional, and has greater limitations on the scene. The mesh dissection method [20] introduced by Liu et al. stores two triangular facets in each mesh, and projects the rays to a two-dimensional plane for judgment calculation, which significantly reduces the judgment cost.

In computer graphics, Havran tested and compared the performance of different acceleration structures in various scenes, and concluded that the KD tree is the best acceleration structure in a static scene [21-22]. KD tree is a high-dimensional index tree data structure that divides data in K-dimensional space. It divides and reorganizes spatial geometric information and filters out combinations of rays and facets that cannot be intersected, so as to avoid invalid intersection calculations and increase the speed of the intersection test between the ray and the target object.

Compared with other acceleration algorithms for the ray tracing process, the partitioning method of the KD tree structure is more flexible and does not depend on the distribution of the triangle surfaces in the scene, and it has a good performance in various triangle surfaces distribution scenes. Comparing the complexity of different algorithms, it can be provided that the algorithm complexity of the KD tree structure is relatively low according to the nature of the binary tree.

The motivation of the work is to accelerate the ray tracing process, and shorten the time for ray-surface intersection. The designer can divide the target object by constructing a KD tree to quickly find the triangle surfaces that intersect the ray, and obtain the benefits from improving the efficiency of ray tracing. Using KD tree in SBR to accelerate the ray tracing process [23-25] might be the best choice for multiple scattering calculation at present.

In this paper, a clued KD tree structure is constructed, and the target object is divided into several sub-bounding boxes. According to the position where the ray intersects the target object and the division information in the bounding box, the facets that intersect the ray can be quickly found. In the example calculation, the ray tracing time under different KD tree heights was compared by changing the facet capacity in the leaf bounding box of the KD tree, and the influence of the KD tree structure on the calculation efficiency of the ray propagation path was analyzed on this basis. Experimental results show that the KD tree structure increases the efficiency of ray tracing by about 10 times.

2 Construction of KD Tree

KD tree is a spatially accelerated data structure that divides the entire space into several subspaces and performs related search operations in specific subspaces. It is mainly used in large-scale data search and nearest neighbor search in multi-dimensional space. Fig. 1 shows the structure of a three-layer KD tree. Its essence is a binary tree. All triangles are stored in the leaf nodes, while the root node and intermediate nodes only store space division information, including split dimensions, split values and clue information, etc.

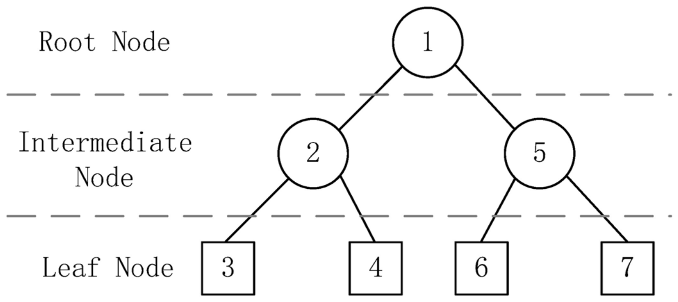

Fig. 1. Structure of a KD tree

2.1 The KD Tree Construction Process

The KD tree construction process is shown in Fig. 2. The specific steps are as follows:

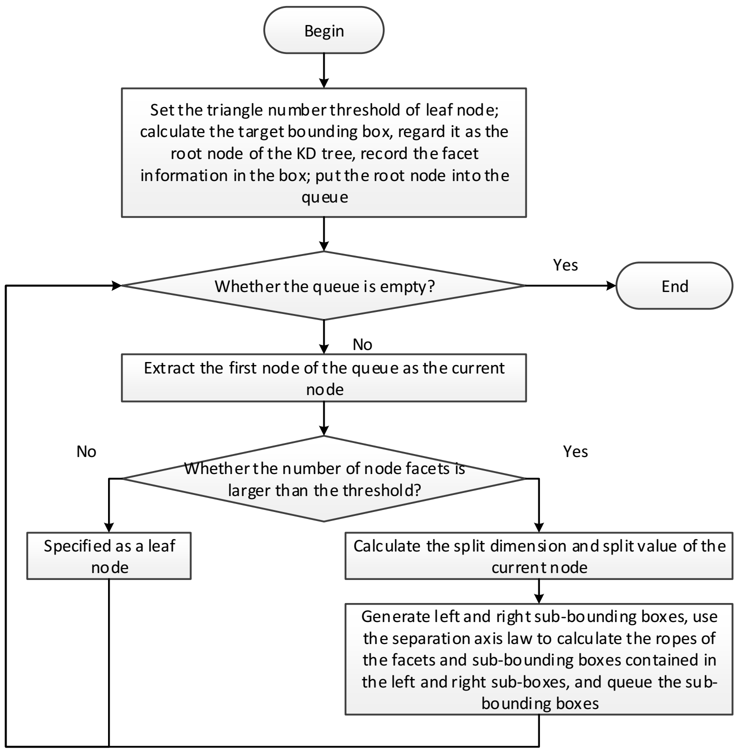
Begin
Set the triangle number threshold of leaf node; calculate the target bounding box, regard it as the root node of the KD tree, record the facet information in the box; put the root node into the queue
Whether the queue is empty?
Yes
End
No
Extract the first node of the queue as the current node
No
Yes
Whether the number of node facets is larger than the threshold?
Specified as a leaf node
Calculate the split dimension and split value of the current node
Generate left and right sub-bounding boxes, use the separation axis law to calculate the ropes of the facets and sub-bounding boxes contained in the left and right sub-boxes, and queue the sub-bounding boxes

Fig. 2. KD tree construction flowchart

(1) Construct an AABB (axis-aligned bounding box) that contains the entire target, regard it as the root node of the KD tree, record the facet information in the bounding box, and set the triangle number threshold of the leaf node.

(2) Put the root node into the queue and take out the first node of the queue as the current node. If the facet number of nodes is less than the threshold, the node can be labelled as a leaf node. Otherwise, calculate the split plane according to the node facet data, divide the current node into two sub-nodes, use the separation axis law to calculate facets contained in each sub-bounding box, add rope information, and put the sub-bounding boxes into the queue.

(3) Take out the next first node of the queue as the current node and repeat the above procedure. The KD tree will be constructed when the queue is empty.

During the process of constructing a KD tree, one “larger” node with too many triangles needs to be divided into two “smaller” nodes by a splitting plane. The splitting plane is usually perpendicular to one of three coordinate axes, which is called the splitting axis.

There are several methods to select the splitting axis, such as the method of alternating segmentation, maximum variance, or the surface area heuristic criterion (SAH) [26-27]. The SAH method reduces the space segmentation problem to a cost function to solve the problem, and determines which split plane is the optimal split plane by solving the minimum cost. Because the SAH criterion needs to meet several assumptions, such as that the distribution of rays is random uniform, the traversal cost and the intersection cost need to be known, et al, it takes a long time to evaluate and calculate a large number of split planes in practical applications.

This paper uses the method of maximum variance to select the splitting axis by performing the following steps. First, calculate the centroids for all the triangles in a bounding box. Second, calculate the variance of the centriods separately in x, y, and z axis, and select the axis with the largest variance as the splitting axis. This method uses the degree of dispersion as the criterion for selecting the splitting axis. The larger the variance is, the more severe the degree of triangle dispersion is.

To determine the splitting plane in the splitting axis, this paper adopts the method of mid-value segmentation. First, the spatial data is sorted in the split dimension using the quick sort algorithm. The mid-value is found as the split value, and the current bounding box is divided into two sub-boxes, each contains about half of triangles in the previous bounding box.

2.2 Separate Axis Law

As shown in Fig. 3, the whole bounding box is divided by the splitting plane into two sub-bounding boxes, i.e., box 1 and box 2. Triangular facet A is outside the bounding box. Triangular facet B is inside the sub-bounding box 1, and triangle C is just across the splitting plane and is divided into two parts, one is in sub-bounding box 1, the other is in box 2. Triangle D is partially included in sub-bounding box 2.

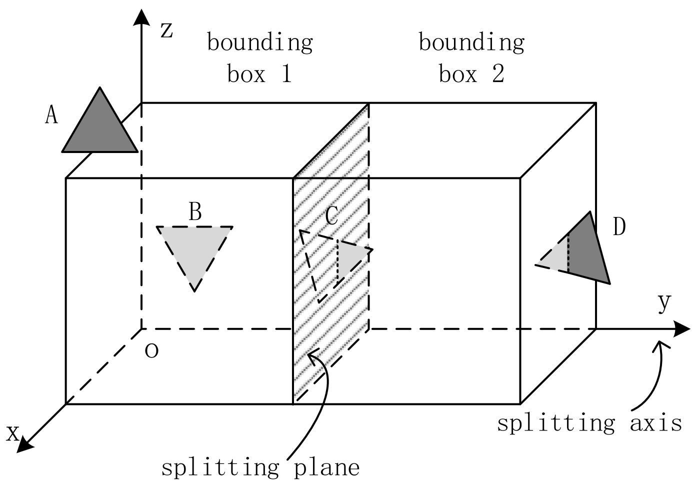

Fig. 3. Schematic diagram of the positional relationship between facets and bounding box

This paper uses the separate axis law to determine whether a triangle belongs to a bounding box. If two convex polygons or convex polyhedrons do not intersect, the projections of these two objects on one axis do not overlap, and this axis is called the separate axis. The choice of axis is generally the normal direction of each side of the convex polygon, the normal direction of each face of the convex polyhedron, etc. The intersection of the triangle facet and the bounding box tested in this paper is equivalent to detecting whether the convex polygon and the convex polyhedron intersect. There are 13 axes to be measured, including the 3 normal directions of the bounding box surface, the normal direction of the plane where the triangle facet is located, and vertical on the 3 normal directions of the bounding box surface and 9 axial directions of the 3 side vectors of the triangle facet. If the above 13 test results are all projection overlaps, it means that the triangle facet intersects the bounding box, and the facet should be stored in corresponding node. If there is a projection on one axis that does not overlap, the facet does not intersect the bounding box, and there is no need to save it in the child node.

Once the KD tree obtained, detailed triangle information for each leaf node should be calculated for ray-triangle intersection process, such as triangle number or triangle IDs inside in each node. Note that triangles across a splitting plane should be recorded in both sub-bounding boxes, such as triangle C in Fig. 3.

2.3 Bounding Box Rope

In the process of constructing the KD tree, rope information is added to the 6 faces of each bounding box to facilitate finding the bounding box adjacent to the face, so that the KD tree can be quickly traversed. The ropes may point to leaf nodes, intermediate nodes, or empty.

In this paper, the method in literature [28-29] is used for reference in the process of creating bounding box ropes. First, set the 6 ropes of the root bounding box as empty, that is, the neighboring nodes of the root bounding box are empty nodes, so that the ray tracing calculation is stopped after the ray passes through the root bounding box. In each subsequent division, the child bounding box inherits the rope information of its parent bounding box, and only the rope direction is updated at the splitting plane. The rope of the left sub-bounding box on the splitting plane points to the right sub-bounding box, and the rope of the right sub-bounding box on the splitting plane points to the left sub-bounding box. Traverse the KD tree to create ropes until the leaf bounding box, then save the boundary of the bounding box, the number of facets, face number and rope information.

3 Calculation of Ray Propagation Path Based on KD Tree

The ray path tracing process based on KD tree is shown in Fig. 4, which includes the following steps:

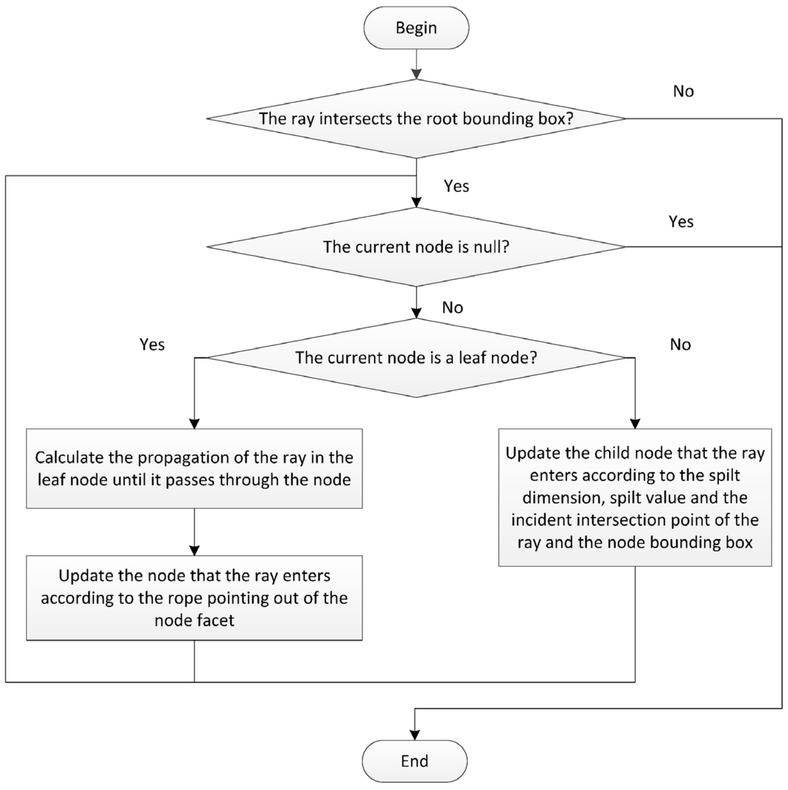
Begin
The ray intersects the root bounding box?
No
Yes
The current node is null?
Yes
No
The current node is a leaf node?
Yes
No
Calculate the propagation of the ray in the leaf node until it passes through the node
Update the node that the ray enters according to the rope pointing out of the node facet
Update the child node that the ray enters according to the spilt dimension, spilt value and the incident intersection point of the ray and the node bounding box
End

Fig. 4. Ray path tracing flowchart

(1) Make sure whether the ray intersects the root bounding box, and determine whether the current node is a leaf node under the premise that the current node is not empty.

(2) If the current node is not a leaf node, determine whether the ray enters the left child node or the right child node according to the splitting plane of the node and the incident intersection point of the ray and the node bounding box. Repeat the test until the leaf node is found.

(3) If the current node is a leaf node, test whether the ray intersects with triangles in the leaf bounding box. If so, calculate the ray-triangle intersection coordinates as well as the reflected vector, and go on tracing the reflected ray until it goes through the current bounding box (i.e., the current leaf node). By using the rope information of the box surface that the ray leaves, the ray will enter into the adjacent bounding box. Meanwhile, the adjacent bounding box will be updated to be the current box.

(4) When the ray goes through the root bounding box or the reflection number of the ray exceeds the threshold, ray tracing calculation stops.

3.1 Rays Intersect with the Bounding Box

In this article, an incident ray is defined by two points: firing point and aiming point. Each ray is tracked separately in the process of calculating the ray propagation path. Firstly, determine whether the ray intersects the root bounding box. If not, ray tracing will stop. If the ray intersects the root bounding box, determine which sub-bounding box the ray enters into according to the coordinates of the entry point and the splitting axis / splitting plane of the root bounding box. If the value of the entry point locates on the right of the plane (i.e., the value of the entry point on the splitting axis is larger than the splitting plane), the ray enters into the right child node. Otherwise, it enters into the left child node.

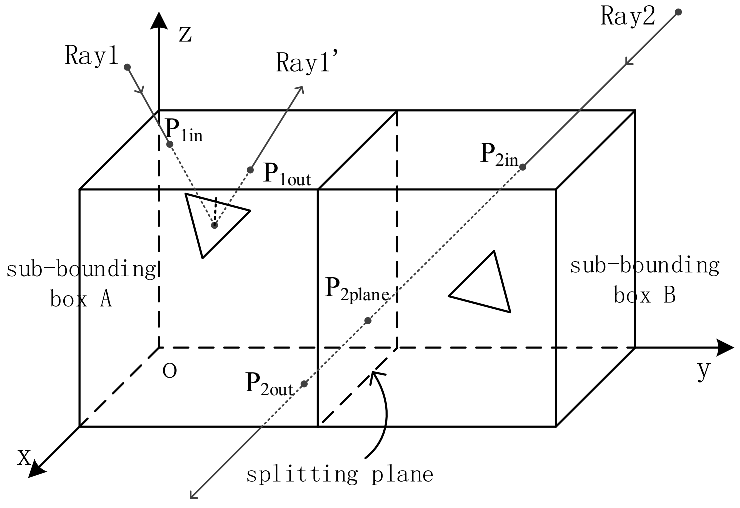

Fig. 5. Schematic diagram of ray intersection with bounding box and triangles

Fig. 5 shows that a big bounding box is divided into two adjacent sub-bounding boxes by the splitting plane. As can be seen from Fig. 5, Ray2 enters into the big bounding box at P_{2in}. As P_{2in} locates on the right of the splitting plane, Ray2 will enter into the sub-bounding box B and begin to search triangles that intersect with it. As there is no triangle intersects with it, the ray enters into its adjacent sub-bounding box A to search the triangles that intersect with it, and finally get out of the box. Different from the Ray2, Ray1 enters into the big bounding box at P_{1in}. As P_{1in} locates on the left of the splitting plane, Ray1 will enter into the sub-bounding box A, intersects with a triangle and reflects on it. The reflected ray finally goes out of the box enters into space.

3.2 Ray-triangle Intersection Algorithm

After the tracing ray enters the leaf bounding box, it is necessary to test whether the ray intersects the triangle facet. As shown in Fig. 5, the Ray1 enters the leaf bounding box A and intersects the facet, and the reflected Ray1' passes through the bounding box A. The Ray2 enters the leaf bounding box B without intersecting any facet. After passing through the 4th plane, the Ray2 enters the bounding box A. It still does not intersect any facet, and passes through the 4th plane.

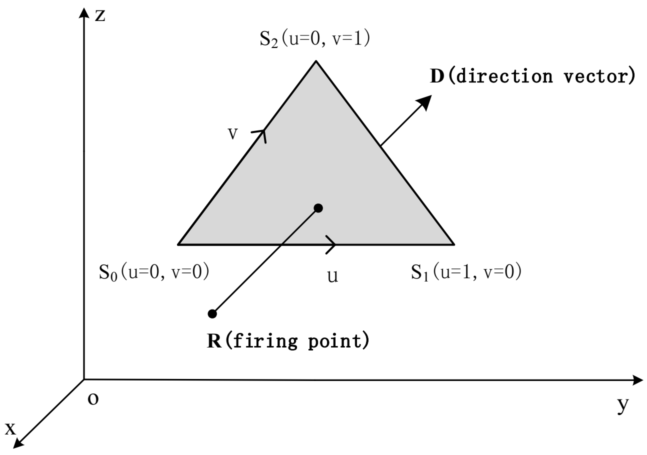

Fig. 6. Schematic diagram of a ray intersecting a triangle

As shown in Fig. 6, in Cartesian coordinate system, the parametric equation of a ray can be written as

$$\mathbf{R} + \mathbf{D}t \tag{1}$$

where **R** is the coordinate of the ray's firing point and **D** is direction vector and *t* is propagation distance. A point lies in a triangle can be represented as

$$(1-u-v)\mathbf{S}_0 + u\mathbf{S}_1 + v\mathbf{S}_2 \tag{2}$$

where $\mathbf{S}_0$, $\mathbf{S}_1$, and $\mathbf{S}_2$ are coordinates of three triangle vertices, respectively. *u* and *v* are weight parameters (shown in Fig. 6) and satisfies the requirements

$$\begin{cases} 0 \le u \le 1 \\ 0 \le v \le 1 \\ 0 \le u + v \le 1 \end{cases} \tag{3}$$

The intersection of a ray and a triangle facet is equivalent to solving the equation:

$$\mathbf{R} + \mathbf{D}t = (1-u-v)\mathbf{S}_0 + u\mathbf{S}_1 + v\mathbf{S}_2 \tag{4}$$

Among them, *t*, *u*, and *v* are unknowns. Let P=D×(S2-S0), Q=(R-S0)×(S1-S0). It can be solved by Kramer's rule and the mixed product formula:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{\left|-\mathbf{D},\ \mathbf{S}_1-\mathbf{S}_0,\ \mathbf{S}_2-\mathbf{S}_1\right|} \begin{bmatrix} \left|\mathbf{R}-\mathbf{S}_0,\ \mathbf{S}_1-\mathbf{S}_0,\ \mathbf{S}_2-\mathbf{S}_1\right| \\ \left|-\mathbf{D},\ \mathbf{R}-\mathbf{S}_0,\ \mathbf{S}_2-\mathbf{S}_1\right| \\ \left|-\mathbf{D},\ \mathbf{S}_1-\mathbf{S}_0,\ \mathbf{R}-\mathbf{S}_0\right| \end{bmatrix} \tag{5}$$

If the above formula has a unique solution, it means that the ray intersects the triangle. The reflected ray can be obtained using the law of reflection, and be traced according to the precious method until it does not intersect any facet and goes out of the root bounding box.

4 Example Analysis

The model used in this paper is a single-ended open rectangular parallelepiped cavity. The length, width and height of the cavity are 2m, 0.5m and 0.5m, respectively. The model consists of 1,896 vertices and 3,750 triangular surfaces. As shown in Fig. 7, a ray-source is firstly placed inside and then outside the cavity. A total of 15,376 rays are traced with a limitation of up to 30 reflections will be traced for a single ray. The total ray-tracing time using the traditional method and the method based on KD tree mentioned in this article are compared.

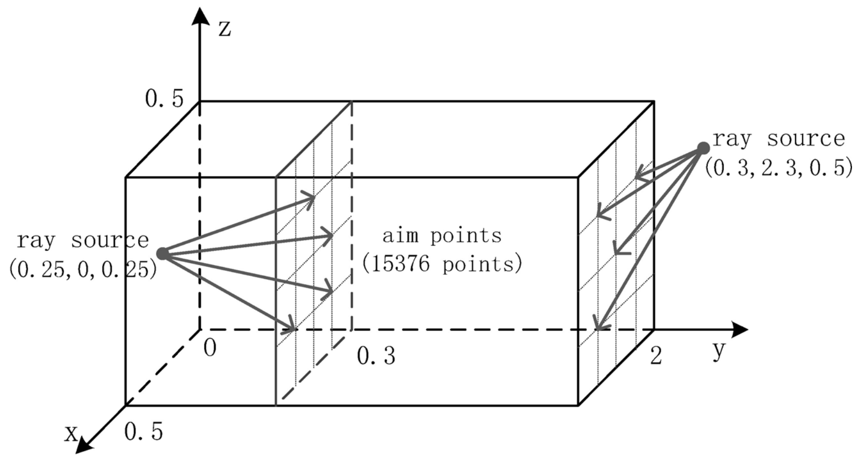

Fig. 7. Schematic diagram of the emission source outside and inside the cavity

The hardware platform PC used in this experiment, 2.80GHz Intel Core i5-4200H CPU, 4.00GB memory. The software environment is Visual Studio Community 2013.

4.1 Emission Source inside the Cavity

As shown in Fig. 7, the emission source is located on the left side of the cavity, and the coordinate is (0.25, 0, 0.25). The cross section of the cavity at the plane y=0.3 is equally divided into many square grids, producing 124*124=15376 intersection points. The emitted rays are assumed to fly via these points and the corresponding propagation paths are calculated.

Fig. 8 illustrated the calculated transmission path of a ray. The ray is reflected for three times on the inner surface of the cavity before it flies into the space. The short blue lines are norms of the surfaces. The reflection process of the ray proves that our experiment is correct.

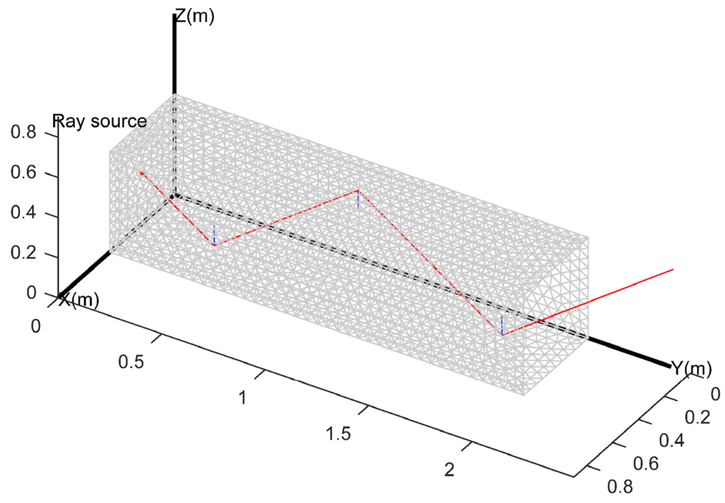

Fig. 8. The propagation path of a ray emitted from a source inside the cavity (number of reflections: 3)

As we know, the more facets in a leaf bounding box, the longer it takes to search the ray-triangle intersection in the box. If the object is completely in the leaf bounding box, the method mentioned in this article will degenerate into the traditional method. If the facet number in the current bounding box exceeds the facet capacity, it needs to be divided into two sub-bounding boxes. In some cases, although the current box is divided into sub-boxes, the facet number in a sub-box still can’t satisfy the facet limit because that the facets cut by the splitting plane needs to be included in both two sub-boxes. Change the facet capacity of the leaf bounding box and set it to 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, and 4 respectively. Run the program successively with other parameters unchanged, calculate and record the average facet number of leaf bounding boxes, KD tree height and ray tracing time, as shown in Table 1.

Table 1. Ray tracing results based on KD tree

Leaf bounding box facet capacity	Average facet number of leaf bounding box	The height of KD tree	Ray tracing time (s)
4096	3750	1	15.55
2048	1919	2	10.34
1024	1004	3	7.21
512	275	5	3.01
256	149	6	2.27
128	86	7	1.51
64	56	8	1.33
32	34	9	1.32
16	16	10	1.28
8	8	12	1.27
4	7	12	1.32

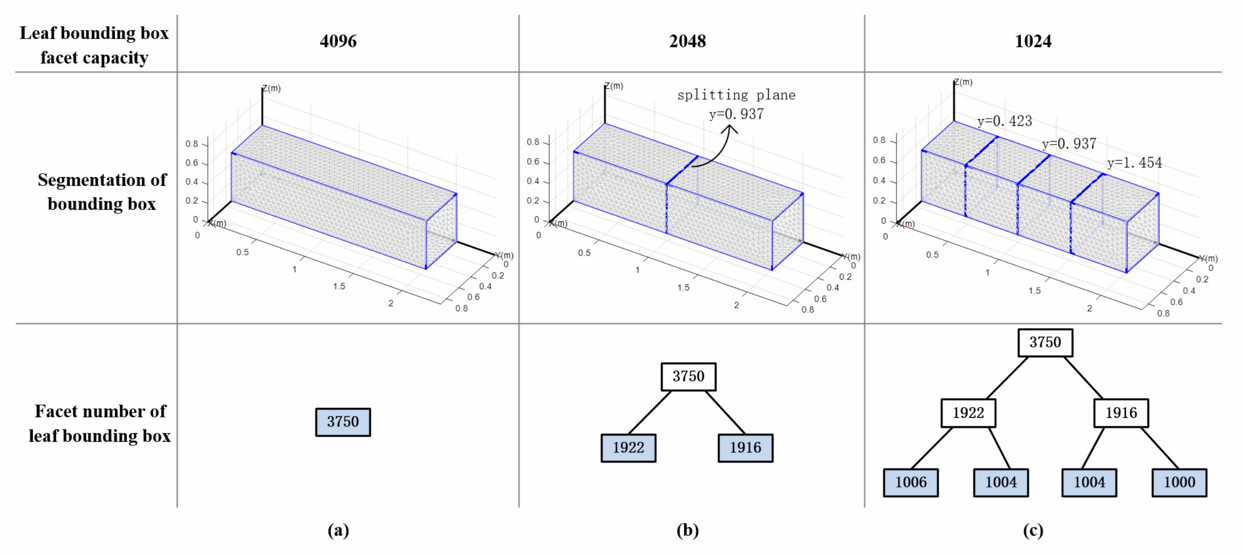
Leaf bounding box facet capacity
4096
2048
1024
Segmentation of bounding box
splitting plane y=0.937
y=0.423
y=0.937
y=1.454
Facet number of leaf bounding box
3750
1922
1916
1006
1004
1004
1000
(a)
(b)
(c)

Fig. 9. KD tree division under different leaf bounding box facet capacity

As can be seen from Fig. 9(a), when facet limit is set to be 4096, the whole object is totally contained in the root/leaf bounding box. In such case, facets in the bounding box have to be judged one by one to find out those that intersect with the ray, thus producing the longest ray-tracing time. As illustrated in Fig. 9(b), when the leaf bounding box facet capacity is reduced from 4096 to 2048, the root bounding box has to be divided into two leaf bounding boxes by the splitting plane y=0.937. If the ray intersects with the cavity in the left leaf bounding box, then it is only necessary to perform the ray-surface intersection in the left leaf node. There is no need to judge the triangular surfaces in the right leaf node, which shortens the time for ray- surfaces intersection. When the leaf bounding box facet capacity was set as to be 1024, the two sub-bounding boxes in Fig. 9(b) were respectively divided into two leaf bounding boxes by the splitting plane y=0.423 and y=1.454, as shown in Fig. 9(c). As the facet number of leaf bounding box decreases, the ray tracing time was further reduced.

It can be seen from Table 1 that as the facet capacity of the leaf node decreases, the height of the KD tree increases and the corresponding tracking time is significantly reduced. When the facet capacity of the leaf bounding box is reduced from 8 to 4, although the average facet number in the bounding box continues to decrease (from 8 to 7 per leaf box), however, the facet limit can’t be realized, and the ray tracing time even slightly increases.

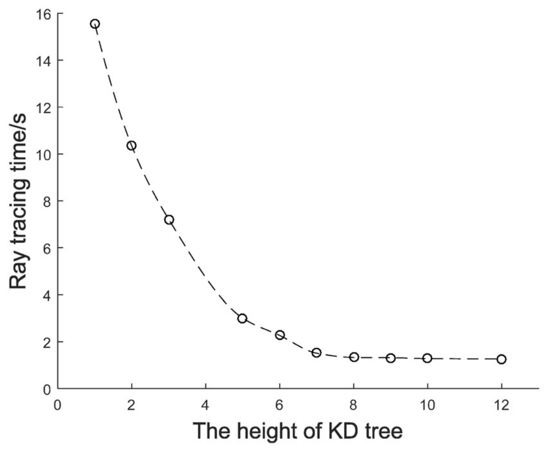

Fig. 10. Ray-tracing time varies with the height KD tree

Fig. 10 illustrates the ray-tracing time as a function of the KD tree height. Result indicates that the application of the KD tree can significantly accelerate the speed of the ray-tracing process. In this example, when the height of the KD tree increases to 12, the ray tracing time reaches a minimum of 1.270 seconds. Compared with the case without KD tree structure (i.e., the height of KD tree is 1), the calculation efficiency is improved by about 12 times.

The average reflection number of the emitted rays is 3.29. Fig. 11 also illustrates the histogram of the ray reflection times. As can be seen in Fig. 10, under the current parameter settings, most of the rays reflect about 2 to 5 times. No ray reflects over 6 times.

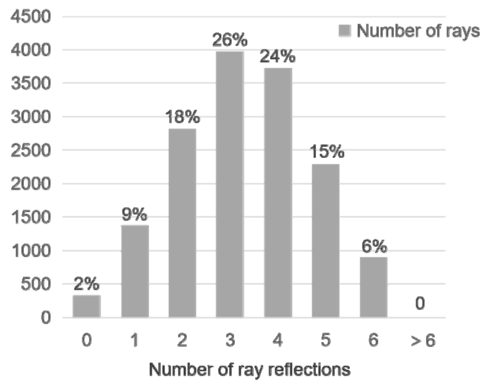

Fig. 11. Histogram of ray reflections

It can be seen from Fig. 12 that both Ray1 and Ray2 pass directly out of the cavity without intersects with the box, which corresponds to the reflection number 0 in Fig. 11. Under the parameter settings in Section 4.1, the maximum number of reflections of rays does not exceed 6 times. Ray3 shows a possible propagation path with a reflection number of 6.

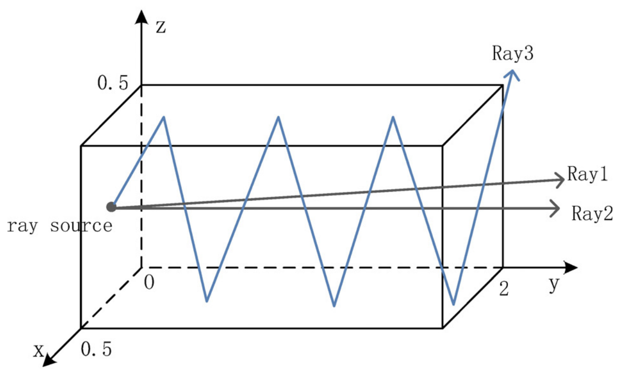

Fig. 12. Histogram of special ray reflections

4.2 Emission Source outside the Cavity

In this case, the coordinate of the emission source is (0.3, 2.3, 0.5), located outside the cavity (Fig. 7). The aim points of the incident rays are on the cross section of the cavity at plane y=2. The propagation path for each ray is traced and the computation time for all the 15376 rays is recorded. Fig. 13 illustrated the calculated transmission path of a ray. The ray reflects 14 times on the inner surface of the cavity before it flies into the space.

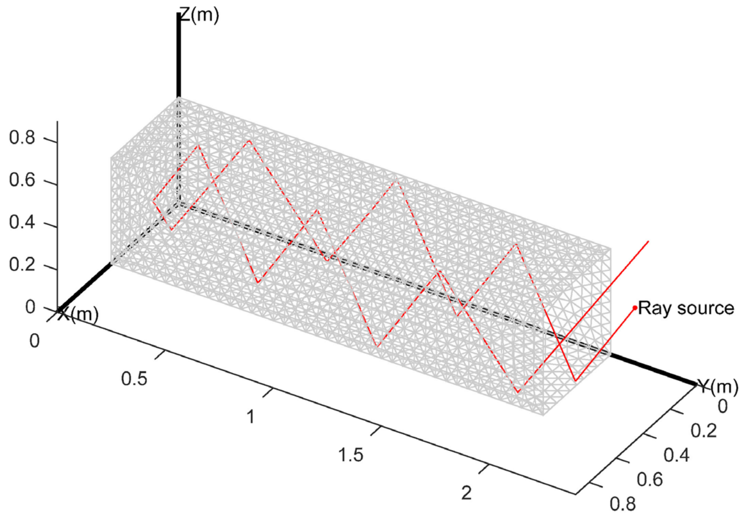

Fig. 13. The propagation path of a ray emitted from a source outside the cavity (number of reflection: 14)

Set the leaf node facet capacity to 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, 4, and run the program separately with other parameters unchanged. The facet capacity of leaf bounding box, the average facet number of leaf bounding boxes and the height of the KD tree are the same as the previous example. The ray tracing time is recorded, as shown in Table 2.

Table 2. Ray tracing results based on KD tree

Leaf bounding box facet capacity	Average facet number of leaf bounding box	The height of KD tree	Ray tracing time(s)
4096	3750	1	44.63
2048	1919	2	27.39
1024	1004	3	18.23
512	275	5	7.91
256	149	6	5.60
128	86	7	4.73
64	56	8	3.78
32	34	9	3.31
16	16	10	3.20
8	8	12	3.22
4	7	12	3.35

The curve of the ray tracing time varying with the height of the KD tree is shown in the solid curve in Fig. 14. In this example, when the height of the KD tree increases to 10, the ray tracing time reaches a minimum of 3.20 seconds. Compared with the case without KD tree structure (i.e., the height of KD tree is 1), the calculation efficiency is improved by about 13 times.

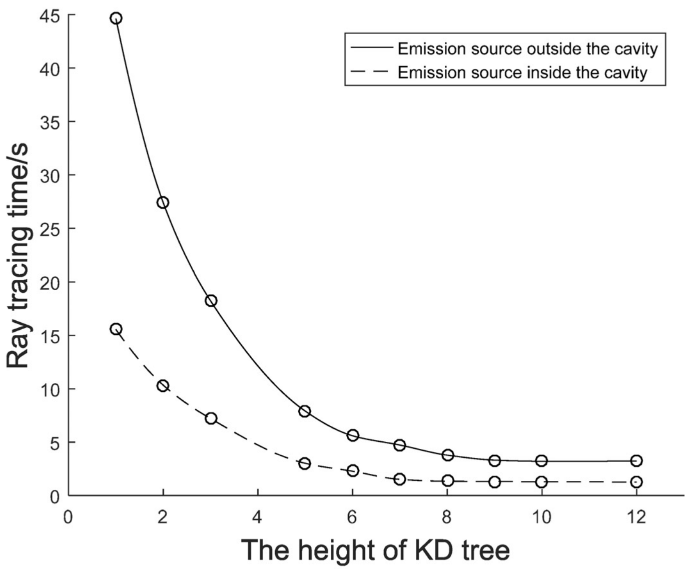

Fig. 14. Curves of ray tracing time varying with KD tree height

The dotted curve in Fig. 14 is the situation where the emission source is inside the cavity. Compared with the dotted curve, the corresponding time becomes longer due to the more reflection numbers. The reason for the increase in the number of reflections is that the rays enter the cavity first and then exit from the outside. It is worth noting that the efficiency improvement in the two cases is similar. The time does not increase linearly with the increase in the average number of reflections.

Fig. 15 illustrates the histogram of the ray reflection times. Under the current parameter settings, most of the rays reflect about 6 to 20 times. No ray reflects over 25 times. The average reflection number of the emitted rays is 11.34. The time does not increase linearly with the increase in the average number of reflections.

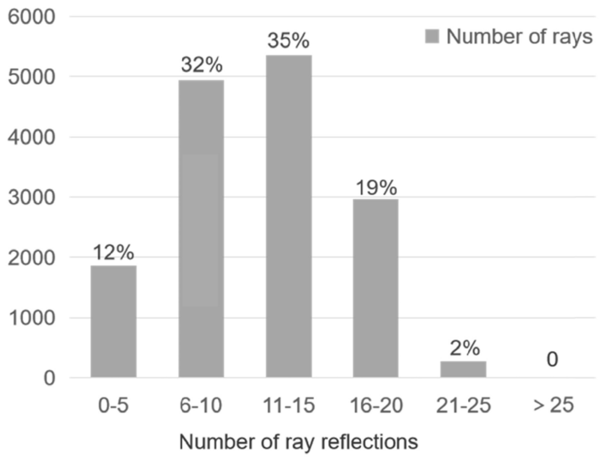

Fig. 15. Histogram of ray reflection frequency distribution

5 Conclusion

This paper constructs a KD tree structure for an object with amounts of triangle surfaces. Based on the KD tree, a fast calculation method is established to accelerate the ray-tracing process. A single-ended open rectangular parallelepiped cavity model is used as an example to calculate the path of incident rays emitted from a source inside and outside the cavity. The time consumed for ray-tracing is recorded. Results indicate that the KD tree structure improves the efficiency of ray-tracing time. The ray path propagation calculation obtains a considerable speedup, and the efficiency is increased by about 10 times.

This method is might helpful to improve the efficiency of the path tracing of a ray with multiple reflections in a complex environment. The limitation of the method in this paper is that it can only track multiple reflections of rays at present, and has not been extended to beam tracking of acoustics. Our future research will focus on the conditions of the acoustic beam splitting and how to split the beam in order to solve the divergence problem with acoustic beam tracking.

Acknowledgements

This work is supported by Qin Xin Talents Cultivation Program, Beijing Information Science & Technology University (QXTCP C201905), Science and Technology Projects of Beijing Municipal Education Commission (KM201911232012), and National Natural Science Foundation of China (12074089).

References

[1] Z.-Z. Wu, Mobile Communication Radio Wave Propagation, Posts & Telecom Press, Beijing, 2002.

[2] W.-H. Wang, Ray tracing method based on simplicse for calculating electric field intensity in indoor environments, [dissertation] Nanjing: Nanjing University of Posts and Telecommunications, 2013.

[3] H. Ling, R.-C. Chou, S.-W. Lee, Shooting and bouncing rays: calculating the RCS of an arbitrarily shaped cavity, IEEE Transactions on Antennas and Propagation 37(2)(1989) 194-205.

[4] J.-Z. Li, S. Li, A high precision shooting and bouncing rays method without mesh for scattering calculation of complex target, International Journal of Numerical Modelling: Electronic Networks, Devices and Fields 32(6)(2019) 1-13.

[5] J.-Y. Zhou, Y.-P. Han, Analyzing the electromagnetic scattering characteristics for plasma targets based on shooting and bouncing ray method, AIP Advances 9(6)(2019) 065106 .

[6] H. Yan, Y. Chen, S. Li, L.-P. Hu, H.-M. Li, H.-C. Yin, A fast algorithm for establishing 3-D scattering center model for ship targets over sea surface using the shooting and bouncing ray technique, Journal of Radars 8(1)(2019) 107-116.

[7] P.-K. Huang, H.-C. Yin, X.-J. Xu, Radar Target Characteristics, Publishing House of Electronics Industry, Beijing, 2006.

[8] Y. Deep, P. Held, S.-S. Ram, D. Steinhauser, A. Gupta, F. Gruson, A. Koch, A. Roy, Radar cross-sections of pedestrians at automotive radar frequencies using ray tracing and point scatterer modeling, IET Radar, Sonar & Navigation 14(6)(2020) 833-844.

[9] J. Baldauf, S.-W. Lee, High frequency scattering from trihedral corner reflectors and other benchmark targets: SBR vs. experiments, IEEE Trans, Antennas Propagation 39(9)(1991) 1345-1351.

[10] Y.-B. Tao, Fast electromagnetic computing on graphics hardware, [dissertation] Hangzhou: Zhejiang University, 2009.

[11] J.-D. Macdonald, K.-S. Booth, Heuristics for ray tracing using space subdivision, The Visual Computer 6(3)(1990) 153-166.

[12] X.-K. Guo, Ray-tracing algorithms and accelerate research, [thesis] Xi’an: Xidian University, 2008.

[13] P.-C. Gao, Parallel acceleration technique for electromagnetic scattering problems based on GPU computing platforms, [dissertation] Hangzhou: Zhejiang University, 2013.

[14] X.-P. Tang, Research on the acceleration method of ray tracing algorithm in the application of complex electromagnetic environment, [thesis] Beijing: Beijing University of Posts and Telecommunications, 2018.

[15] S. Suk, T. Seo, H. Park, H. Kim, Multi resolution grid algorithm in the SBR and its application to the RCS calculation, Microwave and Optical Technology Letters 29(6)(2001) 394-397.

[16] K.-S. Jin, T.-I. Suh, S.-H. Suk, B.-C. Kim, H.-T. Kim, Fast ray tracing using a space-division algorithm for RCS prediction, J. Electromagnet. Waves Applicate 20(1)(2006) 119-126.

[17] J.-M. Ren, The research on path loss prediction of radio wave propagation in urban areas, [thesis] Nanjing: Nanjing University of Aeronautics and Astronautics, 2012.

[18] Y.-P. Tang, Research on path loss prediction method and application of radio wave propagation, [thesis] Nanjing: Nanjing University of Aeronautics and Astronautics, 2014.

[19] Z.-W. Yuan, Y.-C. Li, L. Li, W. Mu, Acceleration method of ray tracing based on dynamic zoning, Computer Engineering and Applications 46(27)(2010) 77-79.

[20] Z. Liu, D. Shi, Y.-G. Gao, C. Yuan, J.-J Bi, Z.-L. Tan, A new ray tracing acceleration technique in the simulation system of electromagnetic situation, in: Proc. Seventh Asia-Pacific Conference on Environmental Electromagnetics (CEEM’2015), 2015.

[21] V. Havran, Heuristic ray shooting algorithms, [dissertation] Prague: Czech Technical University, 2000.

[22] J. Zhang, H.-L. Yu, C. Qin, J. Zhao, H. Wei, Ray Tracing Dynamic Scenes Using Multiple Kd-trees, in: Proc. 2nd International Conference on Applied Mathematics, Modeling and Simulation, 2018.

[23] Y.-B. Tao, H. Lin, H.-J. Bao, Kd-tree based fast ray tracing for RCS prediction, Progress in Electromagnetics Research (81)(2008) 329-341.

[24] C.-Y. Zhang, Fast Algorithm of Target Electromagnetic Scattering Based on KD-tree, [thesis] Xi’an: Xidian University, 2014.

[25] W.-J. Chen, H. Sun, A shooting and bouncing beams method for calculating the acoustic scattering field of concave targets, Acta Acustica 38(2)(2013) 147-152.

[26] W.-S. Fan, B. Wang, A fast KD-tree construction method by probing the optimal splitting plane heuristically, Chinese Journal of Computers 32(2)(2009) 185-192.

[27] J.-F. Li, Y.-H. Tan, S.-H. Liao, Highly parallel SAH-KD-tree construction for raytracing, Journal of Hunan University (Natural Sciences) 45(10)(2018) 148-154.

[28] S. Popov, J. Gunther, H. P. Seidel, P. Slusallek, Stackless KD-tree traversal for high performance GPU ray tracing, Computer Graphics Forum 26(3)(2007) 415-424.

[29] Y. Zhang, S.-Z. Li, C.-Y. Zhang, P. Zhou, M. Zhang, Efficient computational method of coupled targets electromagnetic scattering based on KD-tree, Acta Armamentarii 36(S2)(2015) 173-177.