A Novel Ray Tracing Acceleration Method Based On Bounding Volumes And Prior Environment Processing

Nima Sedaghat Alvar, Ayaz Ghorbani, and Hamidreza Amindavar

Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran

***Abstract* — Ray Tracing has been successfully used in prediction of wave propagation models in recent years. Although this method has its own obvious benefits, it suffers from a big problem: slow performance. In this paper, a novel method is proposed in which the main focus is on reducing the number of ray-facet intersections. It combines a volume bounding algorithm and a light-weight pre-processing operation on the environment which is completely independent of the locations of source and target and is performed once for each environment. In this step, measures for the distances between entities in the environment are computed and saved. Later in the main procedure, theses data will come to help reduce the number of ray-facet intersection tests dramatically.**

***Index Terms* — ray tracing, pre-processing, bounding volumes, UTD, wireless channel, radio propagation.**

I. Introduction

Radio propagation modeling in urban and indoor environments is a complicated electromagnetic problem. Ray tracing and Uniform Theory of Diffraction (UTD) are already widely applied to radio propagation modeling for wireless applications [1]-[4]. In most of the cases, the modeling requires a large number of facets to be modeled. In a mobile communications problem, it is necessary to compute the field for a large number of points along a predefined path [5].

The most time and resource consuming operation in ray tracing method, when used to model the propagation for a complex environment, is the ray-facet intersection test [3]. This is a test to determine if a ray intersects with the specified facet in the environment or not. This test should be performed each time a new ray is generated in the simulation process (after each reflection, diffraction, etc.). And each time, it should be tested for all the facets in the environment! So, when the number of involved facets is large, the ray tracing procedure is very slow and almost impractical.

Many modifications and acceleration techniques have been proposed to speed up the ray tracing procedure [3], [6]-[8]. Some of the most important techniques are Binary Space Partitioning (BSP) [3], Space Volumetric Partitioning (SVP) [3], etc.

One recent method, developed by authors of this paper, which is one of the most effective methods in this field, is the base of current paper [5]. The mentioned method focuses on acquiring measures of distances between pairs of all the facets in the scene. This is performed in the initialization stage, before the main process loop is started. These saved data will later come to help increase the speed of main processing loop by reducing the number of ray-facet intersection tests. This step is independent of the main ray tracing procedure and even the location of transmitter and receiver, and depends only on the environment. Thus this process is in fact performed when the environment data is being gathered & generated (not by the ray tracing software). In the main process, when a ray is launched from its source, only few facets are subjected to the ray-facet intersection test. The method has been shown to reduce the computation time even down to 1% of that in a standard shooting-and-bouncing ray method (SBR).

In the current paper, bounding spheres method has been combined with the mentioned method to improve the ray tracing speed even more. Bounding spheres has been used as a speed-up method in ray-tracing in the field of computer graphics [9].

In this paper, using spheres bounding all facets, two improvements will occur. First in the pre-processing step where the computation of distance measures between facets is enhanced (speed and simplicity). Second, in the main processing loop where the computations to choose the set of candidate facets are reduced a lot, due to the simpler decision-making algorithm. Finally and mainly, in the main processing loop the ray-facet intersection test, becomes a 2-phase intelligent test and will result in dramatic enhancements in speed of the ray-tracing algorithm.

In following sections, the algorithm will be described in detail.

II. The Proposed Method

A. Computation of Bounding Spheres

As described previously, the first step in this algorithm is computation of a bounding volume. Bounding spheres are found to be simple and effective enough to be reasonable for the purpose of this method. These spheres will be used in later steps to generate simple distance measures between facet pairs and also will enhance the ray-facet intersection test in the main processing loop.

In A 3D ray tracing problem, a facet in most cases is assumed to be a triangle. In this case, a volume enclosing all 3 vertices would enclose all the points belonging to the surface of the facet (Fig. 1).

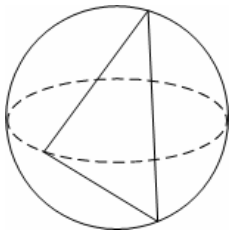

Fig. 1. A triangular facet in 3D space enclosed by a sphere.

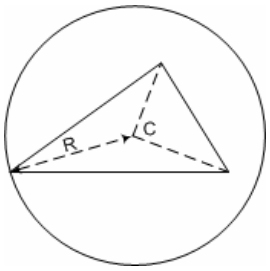

Fig. 2. A triangular facet in 3D space enclosed by a sphere.

There are algorithms for computing the optimized bounding sphere (enclosing sphere) for a set of points [10]. However the algorithm proposed in this paper is not theoretically optimized and is more considered to be simple and fast. It is described as follows. First the center point for vertices should be measured. This would be the average of associated points which in vector notation would become:

$$\vec{C} = \frac{\vec{V_1} + \vec{V_2} + \vec{V_3}}{3}. \qquad (1)$$

This will be the center point of the bounding sphere. Now the radius of the sphere would be the maximum distance from center point to vertices.

$$R = \max\left\{\left|\vec{V_i} - \vec{C}\right|\right\}. \qquad (2)$$

This way the bounding sphere is achieved in a simple and fast way. An example is depicted in Fig. 2.

B. Prior Information Collection

Prior information about the environment is collected at this step. To do this, all N facets in the environment are indexed in an arbitrary order. Then one NxN matrix is constructed. Let the name be D. D_{ij} corresponds to the distance between centers of spheres i & j. This is a symmetric matrix with zeros on the main diagonal. Also the radii of all spheres are saved in a vector (column matrix) named R.

C. Improved ray-facet intersection test

In a standard shooting-and-bouncing ray method, a ray, regardless of type of its source (a transmitter, a reflection point, etc.) is tested for intersection with all facets in the environment. After finding all intersecting facets, all distances between the source point and the intersection point on all facets should be computed and the nearest facet is selected [5]. In the method proposed in [5], however, a ray is tested with only a set of facets, resulting in improvements in speed. In this paper the approach is enhanced to increase the speed of algorithm even more.

Since there are two major enhancements due to the use of virtual spheres, for ease of understanding, they are described one by one and independently. The improvement in selection of candidate faces is described first.

The algorithm goes as follows. The first ray which is originated from the transmitter is tested just like before, and no optimization is performed. From this step on, all the sources are placed on some facet. So the optimization can be applied. Let the ray be originated from some point on facet number n. All existing facets are ordered with increasing distances (near to far) according to the entries of row number n of matrix D. This helps to perform the test on nearer facets before the others. This step is paused when the first facet having intersection with the ray is found. Let the index of this facet be k, so the corresponding entry in D would be D_{nk}.

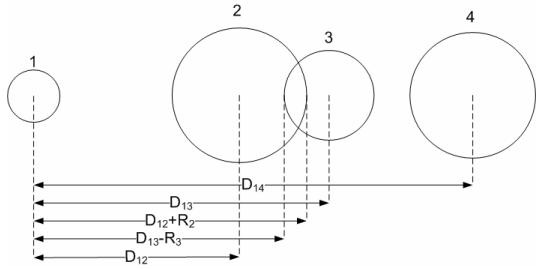

Fig. 3. Finding candidate spheres for “first intersecting facet”.

The next step is to find those few neighbors which are probable candidates for "nearest intersecting facet". Such candidates exist, due to the fact that the facets were ordered using distances between spheres, not the facets themselves. So there might be some other facet, with a farther bounding sphere, while the facet itself intersects the ray before the first candidate. These are the spheres which satisfy the following inequalities:

$$D_{ni} \geq D_{nk} \qquad (3)$$

$$D_{ni} - D_{nk} \leq R_i + R_k. \qquad (4)$$

Condition (3) is satisfied automatically because of the ascending nature of the algorithm. Condition (4) specifies the facets for which the sphere intersects with sphere number k. Fig. 3 displays a simple scenario. In this scene, the sphere #2 is the first candidate and #3 is the only neighbor candidate. There is no other facet that the ray might intersect with, before a member of this set (visible to the ray).

Now the other advantage of bounding spheres is considered: the 2-phase ray-facet intersection test. Using enclosing spheres lets a fast ray-facet intersection test in 2 phases. In phase 1, a ray is intersected with spheres. This is a test which needs less computation time than the standard ray-facet intersection. The procedure moves on to phase 2, only if the ray has intersected the bounding sphere. This way, the rigorous computations will be performed only for facets which are more likely to intersect the ray, and the others are simply subject to a fast test.

III. A Simplified Presentation Of The Method

In this section a simple example environment is assumed and the algorithm is applied to it. Fig. 4 displays the environment. A ray is originated from facet #1. In a standard SBR method, 5 ray-facet intersections tests would be performed and the minimum distance intersection point would be chosen at the end. However in this new method, the test is performed as follows.

The virtual spheres are ordered as 6,3,2,5,4 according to their distances from sphere #1. Sphere #6 is not in the path of the ray. So the algorithm moves on and tests sphere #3. The sphere intersects with the ray. Now the facet itself should be tested for intersection. If the ray and the facet intersect, which is assumed to be so in this example, #3 would become the first candidate. In this level, the only sphere satisfying (3) & (4) is #5 which fails in intersection test and #3 becomes the only candidate and is chosen as the "nearest intersecting facet".

In this simple example, 3 ray-facet intersection tests were performed. Considering that only one of the tests was a real ray-facet intersection test and the others were ray-sphere intersections, time & computation resources are saved much more than 40%.

It should be noted that this method shows its best performance in crowded environments like urban areas where time saving is much more than this example. This is discussed in next section in more details.

IV. Effectiveness Of The Method

A. Pre-processing Time

The preprocessing step, is performed only once for an environment. For example, if the ray tracing software is to be applied to a 3D city map, there's no need to perform the preprocessing each time. However this step is really light-weight. To investigate it, a simulation has been performed. 1000 facets have been distributed in 3D space uniformly. The preprocessing algorithm has been coded in C++, using Object Oriented coding schemes, without any special optimizations. The personal computer used for this simulation was an Intel® Pentium® Mobile processor 1.86GHz, running Microsoft Windows XP Professional™ with 504MB of RAM. Also none of the routine processes belonging to the OS have been stopped. The interesting result was that it only took 1 millisecond to generate both matrices! The time to generate pre-processing matrices in [5] was at least 20 seconds in the same situation.

B. Main Procedure Speed Improvements

The proposed approach has variable effects on speed of ray tracing procedure, according to the geometry of the scene. It appears in its best performance when applied to crowded scenes such as urban areas. To view this, a simulation has been run and 1000 rays have been traced in an environment containing 1000 uniformly distributed facets. The computation times have been observed and compared for standard SBR, simple distance measures proposed in [5], and the method proposed in this paper.

The results are as follows. Using the standard SBR method, it took about 7.7 seconds to trace all the rays. Using the method in [5], it took about 340 milliseconds to trace the roots. And finally using the new proposed method, it took only 43 milliseconds!

So, in comparison with the SBR method, more than 99% of time is saved. And in comparison with the method in [5], 87% improvement is achieved.

V. Conclusion

In this paper, a novel fast ray tracing method was presented. The method's main focus is on reducing the number of ray-facet intersection tests in a ray tracing procedure. This is done using a combination of the method proposed in [5] and the known volume bounding method. A simulation was performed

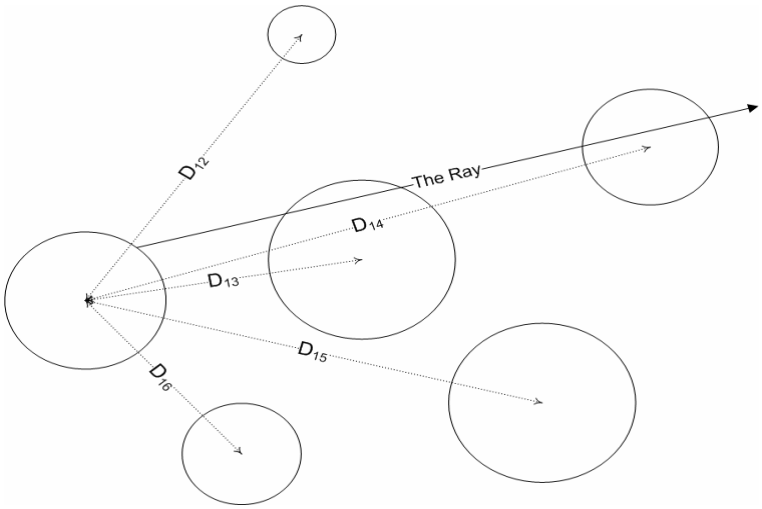

Fig. 4. A simplified presentation of the method.

and the results stated that the preprocess step takes a very short time for a scene containing 1000 facets (about 1 millisecond).

Also, another simulation was designed to consider the efficiency of the method in the main process time. It was shown that this method can bring the process time down to less than 1% of that in standard SBR method. And depending on the geometry of the environment, can make more than 80% improvements to the method proposed in [5].

REFERENCES

[1] M. C. Lawton, and J. P. McGeehan, "The application of a deterministic ray launching algorithm for the prediction of radio channel characteristics in small-cell environments," *IEEE Trans. Veh. Techno.l, vol. 43, no. 4, pp. 955-969, 1994.*

[2] C. F. Yang, B. C. Wu, and C. J. Ko, "A ray-tracing method for modeling indoor wave propagation and penetration," *IEEE Trans. Antennas Propagat., vol. 46, no. 6, pp. 907-919, 1998.*

[3] Catedra et al., "Cell Planning for Wireless Communications," *pp. 7-107, Feb. 1999.*

[4] T. Fügen, J. Maurer, W. Sörgel and W. Wiesbeck, "Characterization of Multipath Clusters with Ray-tracing in Urban MIMO Propagation Environments at 2 GHz," *IEEE Proceedings of the International Symposium on Antennas and Propagation, Washington DC, USA, July 2005.*

[5] N. Sedaghat Alvar, A. Ghorbani and H. Amindavar, "A Novel Ray Tracing Acceleration Method for Radio Propagation Modeling Based on Prior Environment Processing," *Manuscript submitted for publication.*

[6] F. A. Agelet, A. Formella, J. M. H. Rábanos, F. I. Vicente, and Fernando Pérez Fontán, "Efficient Ray-Tracing Acceleration Techniques for Radio Propagation Modeling," *IEEE Trans. Veh. Tech., vol. 49, no. 6, 2000.*

[7] Z. Yun, M.F. Iskander and Z. Zhang, "Fast ray tracing procedure using space division with uniform rectangular grid," *IEEE Electronics Letters, vol. 36, no. 10, pp. 895-897, May 2000.*

[8] I. Wald, H. Friedrich, G. Marmitt, P. Slusallek and H. Seidel, "Faster Isosurface Ray Tracing Using Implicit KD-Trees," *IEEE Trans. Vis. Comput. Graph. vol. 11, no. 5, pp. 562-572 , 2005.*

[9] H. Weghorst, G. Hooper, and D.P. Greenberg, "Improved Computational Methods for Ray Tracing," *IEEE Trans. Vis. Comput. Graph. vol. 11, no. 5, pp. 562-572 , 2005.*

[10] S. Skyum, "A simple algorithm for computing the smallest enclosing circle," *Inform. Process. Lett. 37 (1991) 121–125.*