Progress In Electromagnetics Research, PIER 82, 19–32, 2008

A NOVEL HYBRID APPROACH TO RAY TRACING ACCELERATION BASED ON PRE-PROCESSING & BOUNDING VOLUMES

N. Sedaghat Alvar, A. Ghorbani, and H. Amindavar

Amirkabir University of Technology
Hafez St. 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, novel methods are proposed in which the main focus is on reducing the number of ray-facet intersections. First a pre-processing method is proposed which reduces the number of ray-facet intersection tests dramatically. Later this method is combined with a volume bounding algorithm to make improvements in the speed of ray-tracing simulations, even more.

1. 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 [1-10]. In most of the cases, the modeling requires a large number of facets to be considered. On the other hand, even a simple mobile communications problem, calls for computation of the field for a large number of points along a predefined path, or on a user-defined mesh.

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]. The purpose of this test is 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 would be very slow and almost impractical.

Many modifications and acceleration techniques have been proposed to speed up the ray tracing procedure [3, 11–19]. Some of the most important techniques are Binary Space Partitioning (BSP) [3], Space Volumetric Partitioning (SVP) [3], etc.

In this paper, a novel method, called PDM (Prior Distance Measures), is proposed in which the main focus is on reducing the number of ray-facet intersection tests. No extra approximations are applied to the solution to gain the mentioned improvement. Instead, it eliminates the need to perform some part of the operation, which in fact is redundant. This is done by performing a light-weight pre-processing operation on the environment, before the main procedure is started. 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).

Although the basic PDM method makes great enhancements to the simulation performance, it is combined with volume bounding methods in the rest of this paper, forming a hybrid method, to make even more improvements. The detailed information for this procedure is described in the following sections.

2. THE MAIN PROPOSED METHOD

2.1. Prior Information Collection

In this method, prior information about the environment is collected at the initialization stage. To do this, all N facets in the environment are indexed in an arbitrary order. Then two $N \times N$ matrices are constructed. Let the names be D^1 and D^2. These matrices are filled in so that D^1_{ij} corresponds to the minimum distance between facets i & j and similarly D^2_{ij} corresponds to the maximum distance between facets i & j. These matrices are symmetric matrices with zeros on the main diagonal:

$$D^k : \left[\begin{array}{cccc} 0 & D^k_{1,2} & \cdots & D^k_{1,N} \\ D^k_{1,2} & 0 & & \\ \vdots & & \ddots & \\ D^k_{1,N} & & & 0 \end{array} \right]_{k=1,2} \tag{1}$$

Without loss of generality, facets are assumed to be positioned so that no crossing pair exists. This is true as we can divide each of the crossing facets to two distinct parts. Now, the minimum distance between two

facets, would be the minimum of a set containing distances between each facet and all the edges of the other one:

$$D^1_{i,j} = \min\left\{\left\|P^i_u, F^j\right\|, \left\|F^i, P^j_v\right\|; \quad u = 1, \ldots, n_i \;\&\; v = 1, \ldots, n_j\right\} \tag{2}$$

where n_i & n_j denote number of vertices/edges in ith & jth facet, respectively. $\|P^i_u, F^j\|$ represents the distance between the jth facet and uth vertex of the ith facet. For triangular facets, as an example, the set would contain 6 entries. Fig. 1 depicts a demonstration where the bolded dashed line shows the minimum distance between the two facets. Note that the triangles represent facets in 3D space.

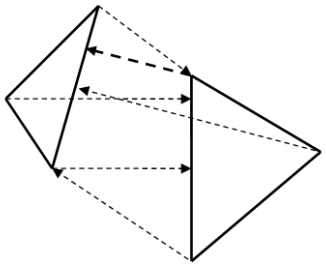

Figure 1. A demonstration of minimum distance measurement between two facets in 3D space.

The maximum distance is similarly the maximum of a set which is constructed in a simpler way. This set contains the distances between vertices of the first facet and the vertices of the other one:

$$D^2_{i,j} = \max\left\{\left\|P^i_u, P^j_v\right\|; \quad u = 1, \ldots, n_i \;\&\; v = 1, \ldots, n_j\right\} \tag{3}$$

with the same notation. For the triangular facets example, it would contain 9 members. Fig. 2 depicts an example for this scenario.

2.2. Improved Ray-facet Intersection Test

As mentioned previously, 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 existing 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 [3]. In this new method, however, a ray is tested with only a few facets, resulting in a dramatic improvement in speed.

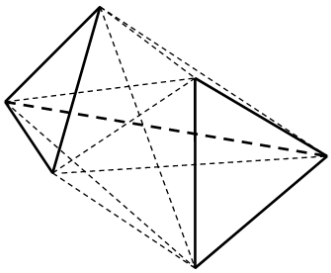

Figure 2. A demonstration of maximum distance measurement between two facets in 3D space.

The algorithm is 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 starting points of the rays will be located on some facet. So the optimization can be applied. Let the ray be originated from some point on facet number n. In this case row n of matrix D^1 gives some measure of distance between the facet containing the origin of the ray and all other facets. So all existing facets are ordered with ascending distances (near to far) according to D^1. 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^1 would be D^1_{nk}.

Up to here, the facet with index k is just a candidate for being "*the nearest intersecting facet*". This is due to the fact that the distance, d, between the origination point of the ray and the intersection point, fulfills the following limits:

$$D^1_{nk} \leq d \leq D^2_{nk} \tag{4}$$

So there are still other probable candidate facets, which are identified by:

$$D^1_{nx} \leq D^2_{nk} \tag{5}$$

There is no other facet that the ray might intersect with, before a member of this set (visible to the ray). So (5) designates the remaining facets for which the ray-facet intersection test should be performed.

We shall now consider a simplified example for this algorithm to clarify the procedure. Fig. 3 depicts a simple 2D scene. Note that in the 2D case, facets are replaced with lines. In this simple case, the ray originated from facet number 4 is to be considered. We have:

$$D^1_{4,5} = D^1_{4,6} = 0 \tag{6}$$

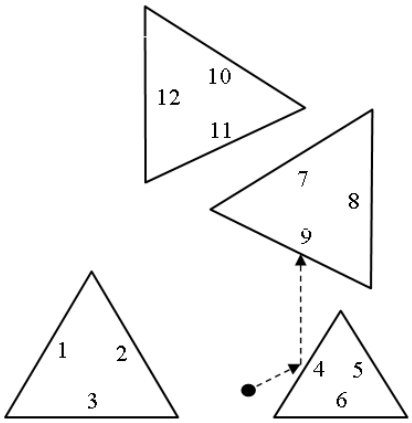

Figure 3. A simplified presentation of the method.

So the facets number 5 and 6 are tested before all other ones. Since no intersection is detected between the ray and these facets, one step farther facets will be tested, which in this example are the number 8 and 9. Number 9 intersects with the ray. So number 9 will be the "*first candidate*". Here the first step is completed and the algorithm moves on to the next step.

In this step, a set of facets, satisfying the following condition should be found and tested:

$$D^1_{4,x} \leq D^2_{4,9} \tag{7}$$

According to Fig. 3, facets 2, 3, 7 are the members of this set. Finally these facets are subjected to intersection test, and since there's no intersection, number 9 will be "*the first-intersecting facet*".

It can be seen that just some probable candidates had to be tested using the PDM method. This was for a simple environment containing only 12 facets and as it is discussed in the next sub-section, this method would show much more improvements in more complex environments.

2.3. Effectiveness & Optimization Degree

From the simple example of the previous sub-section, it is evident that the number of performed tests is reduced due to conditions (4) & (5) which limit the number of candidate facets to a really small set. Although in this simplified example, which contains only 12 facets, there is no big achievement; but the PDM method will have its best effect in crowded environments, such as urban areas.

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. Also 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 1.86 GHz, running Microsoft Windows XP Professional™ with 504 MB of RAM. Also none of the routine processes belonging to the OS have been stopped. The interesting result was that it only took about 21 seconds to generate both matrices!

The PDM method has variable effects on speed of the main ray tracing procedure, according to the geometry of the scene and as stated before, it appears in its best performance when applied to complex environments. It can be seen well in Fig. 4. In this simplified 2D example, there are 280 facets (lines) in the scene. As it is shown, for this particular ray, only 36 facets should be tested. This is only about 13% of the standard case. Which proves the great improvement achieved using PDM.

It should be noted that, in a real example, there are much more facets in the scene; thousands and even more. In such cases, the number of tested facets will remain equal to 36 which leads to interesting results. The computation time can be reduced even down to 1%!

It is evident that the time of simulation goes up with increasing the complexity of the environment, ONLY up to some specific level. After that point, the simulation time will remain almost constant and this is a great improvement in ray-tracing performance which will be investigated at the end of this paper.

3. COMBINATION WITH BOUNDING SPHERES

In this section, the so called "*bounding spheres method*" has been combined with the basic PDM 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 [20].

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-

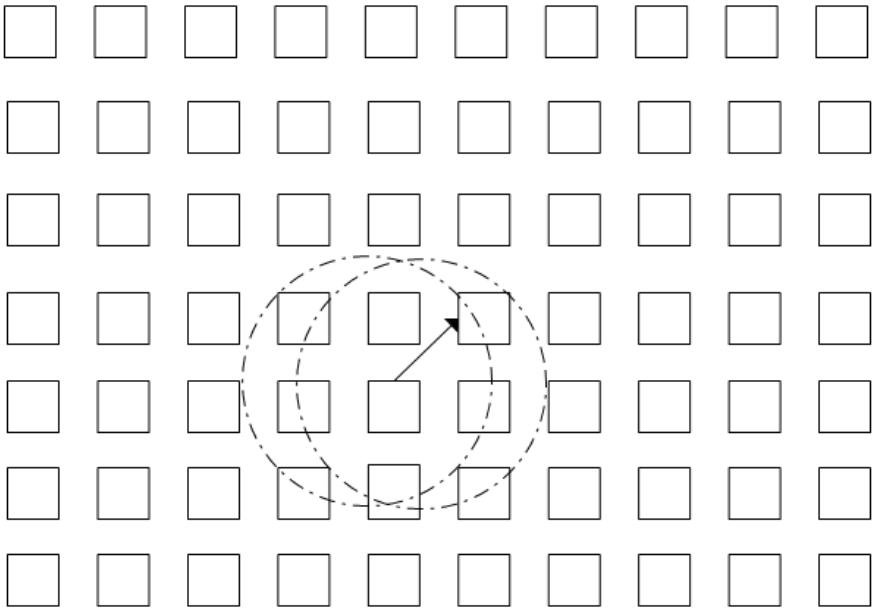

Figure 4. A simple 2D environment to evaluate the achieved optimization.

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 even more enhancements in speed of the ray-tracing algorithm. The algorithm is described in detail in following subsections.

3.1. Computation of Bounding Spheres

The first step in the new hybrid method would be 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. 5).

There are algorithms for computing the optimized bounding sphere (enclosing sphere) for a set of points [21]. However the algorithm proposed in this paper is not theoretically optimized and is more

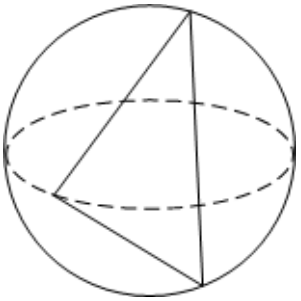

Figure 5. A triangular facet in 3D space enclosed by a sphere.

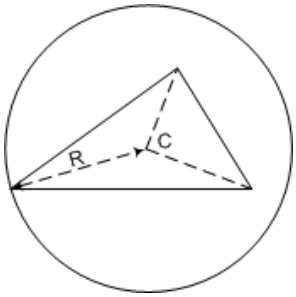

Figure 6. Computation of the center point and radius of the bounding sphere.

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:

$$\vec{C} = \frac{\vec{V}_1 + \vec{V}_2 + \vec{V}_3}{3} \tag{8}$$

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:

$$R = \max\left\{\left|\vec{V}_i - \vec{C}\right|\right\} \tag{9}$$

This way the bounding sphere is achieved in a simple and fast way. An example is depicted in Fig. 6.

3.2. Prior Information Collection

After applying bounding spheres to the basic PDM method, prior information collection undergoes some changes. This step is now performed as follows. At first, all the N existing facets in the environment (or their corresponding spheres) are indexed in an arbitrary order. The result would be an $N \times N$ matrix. Let the name be D. D_{ij} corresponds to the distance between centers of spheres I & j. This would be a symmetric matrix with zeros on the main diagonal:

$$D : \begin{bmatrix} 0 & D_{1,2} & \cdots & D_{1,N} \\ D_{1,2} & 0 & & \\ \vdots & & \ddots & \\ D_{1,N} & & & 0 \end{bmatrix} \tag{10}$$

This matrix is a new one and is different with those presented in (1)–(3). It is seen that there is only one distance matrix needed for this enhanced approach, when compared to the main method proposed in the previous section. Although the radii of all the spheres are saved in a simple vector (column matrix) named R:

$$R : [R_i]_{i=1,\ldots,N} \tag{11}$$

3.3. Improved Ray-facet Intersection Test

The basic PDM method proposed in previous section, improved the speed of ray-tracing simulation by limiting the number of facets which should be tested for intersection with the ray. In this section 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 considered 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, as stated in the main algorithm, 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 n of matrix D. This helps to perform the test on nearer facets before the others, just like before. 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}. Up to now, all the steps seem similar to the main PDM. However the big difference is seen in the following step:

The next step is to find those few neighbors which are probable candidates for "*the 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:

$$D_{ni} \geq D_{nk} \tag{12}$$

$$D_{ni} - D_{nk} \leq R_i + R_k \tag{13}$$

Condition (12) is satisfied automatically because of the ascending nature of the algorithm. Condition (13) specifies the facets for which the sphere intersects with sphere number k. Fig. 7 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).

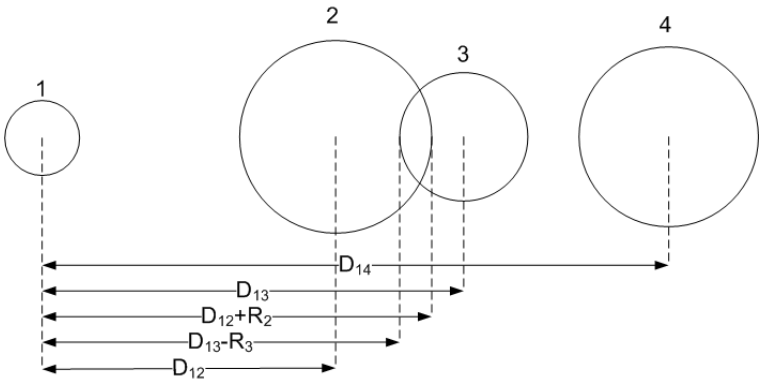

Figure 7. Finding candidate spheres for "*the first intersecting facet*".

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. In fact it is a simple intersection test between a half-line and a sphere in which the heaviest computation job is checking of the following inequality:

$$\|L, C_i\| \leq R_i \tag{14}$$

where L indicates the half-line which represents the path of the ray and C_i & R_i indicate the center of the ith sphere and its radius respectively. It is evident that this test is performed much easier than the ray-facet intersection test in which the half-line is to be intersected with a bounded plane!

The procedure moves on to the 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.

Here a simple example environment is assumed and the algorithm is applied to it to clarify the steps of the algorithm. Fig. 8 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:

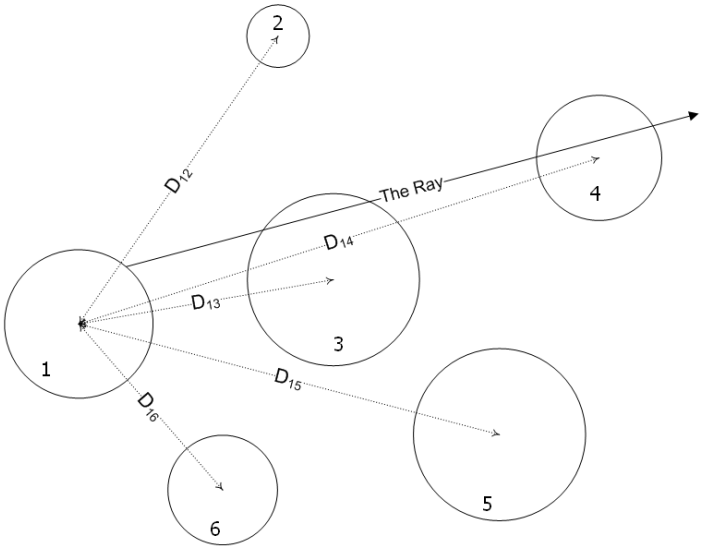

Figure 8. A simplified presentation of the method.

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 (12) & (13) 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, as stated before, 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 the next sub-section in more details.

3.4. Effectiveness & Optimization Degree

To investigate the performance of the pre-processing step (for the bounding spheres case), a simulation has been performed on the same 1000-facet environment introduced in the previous section. Hardware,

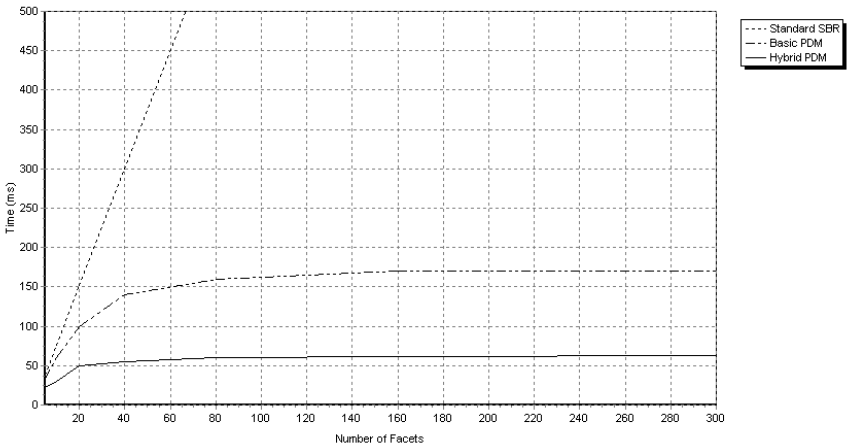

Figure 9. Simulation time versus complexity of the environment, for standard SBR, basic PDM and hybrid PDM methods.

software and OS had the same conditions as before. The interesting result was that it only took 1 millisecond to generate both matrices! This time was at least 20 seconds for the case in which no bounding spheres were used.

The enhanced approach still has variable effects on speed of ray tracing procedure, according to the geometry of the scene and as before, it appears in its best performance when applied to crowded scenes. To view this, a simulation has been run and 1000 rays have been traced in an environment. The simulation times versus the number of facets present in the environment are plotted in Fig. 9. This plot shows the great improvement in the time of simulation. It can be seen that the PDM method, even without the sphere bounding enhancement, approaches a constant time and the simulation time is not increased proportional to the complexity of the environment after that. So it is obvious that in complex environments such as urban areas, even up to 99% of simulation time could be saved, when compared to a standard SBR method.

4. CONCLUSIONS

In this paper, a novel fast ray tracing method, called PDM, was presented. The method's main focus is on reducing the number of ray-facet intersection tests in a ray tracing simulation. This method has a basic version which relies on distance measures between facets

which improves the performance very nice. However a complementary method is provided to enhance the basic method and gain more advance in performance. As the results of the simulation section state, we could even reach to more than 99% percent of time saving in comparison to standard SBR method.

REFERENCES

1. Lawton, M. C. 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. Tech.*, Vol. 43, No. 4, 955–969, 1994.
2. Yang, C. F., 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, 907–919, 1998.
3. Catedra, M. F., *Cell Planning for Wireless Communications*, Artech House, 1999.
4. Chen, C. H., C. L. Liu, C. C. Chiu, and T. M. Hu, "Ultra-wide band channel calculation by SBR/Image techniques for indoor communication," *J. of Electromagn. Waves and Appl.*, Vol. 20, No. 1, 41–51, 2006.
5. El-Sallabi, H. M. and P. Vainikainen, "Radio wave propagation in perpendicular streets of urban street grid for microcellular communications. Part I: Channel modeling," *Progress In Electromagnetics Research*, PIER 40, 229–254, 2003.
6. Wang, N., Y. Zhang, and C. H. Liang, "Creeping ray-tracing algorithm of UTD method based on NURBS models with the source on surface," *J. of Electromagn. Waves and Appl.*, Vol. 20, No. 14, 1981–1990, 2006.
7. Mphale, K. and M. Heron, "Ray tracing radio waves in wildfire environments," *Progress In Electromagnetics Research*, PIER 67, 153–172, 2007.
8. Li, L. W., T. S. Yeo, P. S. Kooi, M. S. Leong, and J. H. Koh, "Analysis of electromagnetic wave propagation in forest environment along multiple paths," *Progress In Electromagnetics Research*, PIER 23, 137–164, 1999.
9. Liu, Y.-J., Y.-R. Zhang, and W. Cao, "A novel approach to the refraction propagation characteristics of UWB signal waveforms," *J. of Electromagn. Waves and Appl.*, Vol. 21, No. 14, 1939–1950, 2007.
10. Fügen, T., 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.

11. Agelet, F. A., A. Formella, J. M. H. Rábanos, F. I. Vicente, and F. P. Fontán, "Efficient ray-tracing acceleration techniques for radio propagation modeling," *IEEE Trans. Veh. Tech.*, Vol. 49, No. 6, 2000.
12. Yun, Z., M. F. Iskander, and Z. Zhang, "Fast ray tracing procedure using space division with uniform rectangular grid," *IEEE Electronics Letters*, Vol. 36, No. 10, 895–897, May 2000.
13. Tao, Y. B., H. Lin, and H. J. Bao, "KD-tree based fast ray tracing for RCS prediction," *Progress In Electromagnetics Research*, PIER 81, 329–341, 2008.
14. Bang, J.-K. and B.-C. Kim, "Time consumption reduction of ray tracing for RCS prediction using efficient grid division and space division algorithms," *J. of Electromagn. Waves and Appl.*, Vol. 21, No. 6, 829–840, 2007.
15. Jin, K.-S., "Fast ray tracing using a space-division algorithm for RCS prediction," *J. of Electromagn. Waves and Appl.*, Vol. 20, No. 1, 119–126, 2006.
16. Cocheril, Y. and R. Vauzelle, "A new ray-tracing based wave propagation model including rough surfaces scattering," *Progress In Electromagnetics Research*, PIER 75, 357–381, 2007.
17. Teh, C. H., F. Kung, and H. T. Chuah, "A path-corrected wall model for ray-tracing propagation modeling," *J. of Electromagn. Waves and Appl.*, Vol. 20, No. 2, 207–214, 2006.
18. Liang, C., Z. Liu, and H. Di, "Study on the blockage of electromagnetic rays analytically," *Progress In Electromagnetics Research B*, Vol. 1, 253–268, 2008.
19. Wald, I., 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, 562–572, 2005.
20. Weghorst, H., G. Hooper, and D. P. Greenberg, "Improved computational methods for ray tracing," *IEEE Trans. Vis. Comput. Graph.*, Vol. 11, No. 5, 562–572, 2005.
21. Skyum, S., "A simple algorithm for computing the smallest enclosing circle," *Inform. Process. Lett.*, Vol. 37, 121–125, 1991.