Advances in Electrical and Computer Engineering **Volume 13, Number 4, 2013**

An Enhanced Binary Space Partitioning Algorithm for Indoor Radio Propagation

Abu Sulaiman Mohammad Zahid KAUSAR, Ahmed Wasif REZA, Kamarul Ariffin NOORDIN,
Mohammad Jakirul ISLAM, Harikrishnan RAMIAH
Department of Electrical Engineering, Faculty of Engineering, University of Malaya
50603 - Kuala Lumpur, Malaysia
wasif@um.edu.my

***Abstract*—Precise multipath propagation modeling is the fundamental prerequisite to design indoor wireless radio networks. In recent years, ray tracing based propagation prediction algorithms have been successfully used in prediction of indoor radio propagation. Even though these algorithms have its own noticeable benefits, it suffers from lack of accuracy and sluggish performance. To overcome these shortcomings, a new three dimensional (3D) ray tracing algorithm is presented here. This algorithm is based on balanced Binary Space Partitioning (BSP). For optimization purposes, novel concepts of Nearest Object Priority (NOP) and In Contact Surface (ICS) are combined with this BSP. Using of BSP as well as optimization techniques make the algorithm faster and more accurate. The obtained results show that, among all of the scenarios of five considered environments, the maximum accuracy increase can be 87.27% and the maximum computation time reduction can be 33.60% than the existing algorithms.**

***Index Terms*—Radiowave propagation, personal communication networks, wireless communications, wireless sensor networks.**

I. INTRODUCTION

The outburst in wireless communications has resulted in new technologies and new applications for the personal use of radio frequencies. For example, Personal Communication Systems (PCS), Wireless Local Area Network (WLAN), wireless PBX, paging, and so on are now developing worldwide and becoming very popular. These emerging technologies use different bandwidth of radio waves and are used for different indoor communication purposes. These cover a wide variety of situations, such as confidential data and voice communication between personnel's sitting in different rooms of a building, hospitals, factories, etc., communication between a group of people through WiFi hotspots inside a building, communication between central database and a group of sensors employed for some specific data collection purposes, etc. This type of indoor wireless system is established with some transmitters *Tx*(s) and receivers *Rx*(s). The signal transmission inside building between these *Tx*(s) and *Rx*(s) is known as the indoor radio propagation. The indoor radio propagation model predicts the way of radio waves that are propagated from a *Tx* to an *Rx* in order to predict the propagation path.

Accurate prediction in a complex indoor propagation environment is undoubtedly critical for the precise deployment of indoor wireless communication. This is attributed to the unquestionably remarkable amount of unpredictability within the indoor radio channel that poses great challenges to the network designer. A few years back, actual channel measurements were the main source of information related to indoor radio propagation characterization [1-5]. Many measurement-based statistical techniques were developed then to draw rational conclusions from the data collected during the itinerary of measurement. In general, these techniques do not express the details of a certain building under study, though they are very simple, fast, and using direct formula, but their site-specific accuracy is very poor and therefore, wideband parameter prediction is not possible. On the other hand, site-specific propagation techniques are built on the basis of electromagnetic wave propagation theory to describe the indoor radio propagation [4, 6-8]. Site-specific propagation techniques do not rely on the extensive measurement; nonetheless, they are accurate and can predict wideband parameters. They are able to grasp a better insight of the indoor environment and no doubt, this is an important ingredient for the accurate prediction of signal within a building. Researchers used a mixture of these techniques to assist them in improving the accuracy, broadening the generality, and reducing the amount of computational complexity.

One of the key shortcomings in designing ray tracing based propagation prediction algorithm is to store the details about environment efficiently. In most of the algorithms [4, 9-11], the researchers used a single list to serve this purpose. This consumes a lot of time during simulation to search for a specific object. Other researchers [12-14], on the other hand, used tree-based algorithm. However, this technique is not used for indoor propagation prediction modeling. Consumption of high time during ray-object intersection test is another problem for the propagation prediction algorithms. This test consumed the most execution time and resources in a ray tracing technique [9, 15]. Intersection test is performed every time, a new ray is generated and its goal is to determine whether there is a ray-object intersection or not. During intersection test, all of the objects present in the area of concern will be used to identify which one has the actual intersection. Hence, if all objects participate in this test, the ray tracing time consumed will be extremely high.

However, the existing models, such as Modified Shooting and Bouncing Ray (MSBR) [16], Ray Frustums (RF) [7], Space Division (SD) [17], Bi-Directional Path Tracing (BDPT) [18], Prior Distance Measure (PDM) [15], and Brick Tracing (BT) [19] techniques have their own limitations. In MSBR technique, for the *Rx*(s) of two successive cone areas, double ray counting error occurs.

Digital Object Identifier 10.4316/AECE.2013.04018

This erroneous ray counting results in lack of accuracy. In RF technique, for storing the large number of triangular frustums, a large amount of computer memory is required for complex environments, which results in sluggish performance. In SD technique, a single list is used to store the entire unique cells ID. This full list has to search for finding a specific cell during simulation, which consumes a lot of time and thus increases the execution time. The BDPT technique cannot be used for the indoor environments that have several rooms. In this case, this technique shows inaccurate results and takes a lot of time to create the ray paths. In PDM method, a preprocessing operation is needed for the environment. This type of preprocessing makes the overall process more complex. For BT technique, erroneous reflection and transmission coefficients are found due to the truncation of the slab of the corner bricks. In summary, the main problems in modeling the indoor radio propagation are: (i) high time consumption of the ray-object intersection test, (ii) lower ray prediction accuracy, and (iii) storing of the environmental details in an inefficient way.

Thus, in this paper, we have extended the use of tree-based algorithm, i.e., Binary Space Partitioning (BSP) to solve the data storing problem for indoor propagation prediction modeling. As will be seen later, BSP will decrease the data searching time tremendously. For solving the ray-object intersection test, first we emphasize on identifying the exact position of the object [20-22] and then it will be easy to minimize the testing time. To minimize this time, we have optimized the BSP technique by using the Nearest Object Priority (NOP) concept. This will decrease the searching time during ray intersection testing process. Intersection points found by the NOP technique will later be used for the object location identification to determine the exact location of an object.

Now, modeling in a 3D environment is much more complex than two dimension (2D). Electromagnetic rays can hit anywhere in a surface of the 3D object. It is a very complex idea to calculate all the different incident points during the simulation. To solve this problem, another new concept of In Contact Surface (ICS) is introduced as an additional optimization technique, which will minimize the computational complexity of the 3D environment. This paper is organized as follows. In Section 2, we describe how to achieve a balanced BSP tree. Details about the proposed method including the ICS and NOP concepts are explained in Section 3. Then, in Section 4, we elaborate the results obtained. Finally, a conclusion is made in Section 5.

II. ACHIEVING BALANCED BINARY SPACE PARTITIONING

The BSP is a technique that recursively splits the target space using random plane. The underlying principle of a BSP is to divide a space into two parts. Then, each of the parts under consideration will again be divided into two minor parts. This process continues until the parts become satisfactorily tiny for analysis.

Mathematically [23, 24], we can say that, if *O* be a group of *n* pair-wise separate objects in 3D space IR^3, a binary space division partition for *O* is a recursively demarcated convex subdivision of space acquired by dividing the space into two uncluttered regions *O1* and *O2* by a plane. Thus, it recursively constructs a BSP for $\{o \cap O1 | o \in O\}$ within *O1* and a BSP for $\{o \cap O2 | o \in O\}$ within *O2*. This process comes to an end when each cell of the BSP splits up at most single object of *O*. The word pair-wise means identical and the objects which poses the same types of characteristics and have no physical connection with each other, are defined as pair-wise separate objects. In an indoor environment, there are a lot of objects having the same characteristics. As an example, a number of objects in an indoor environment are made of wood. Therefore, all the objects made of wood will show the same types of propagation characteristics (same as for the case of plastic, steel, glass, etc.). For this reason, these types of objects are defined as pair-wise objects. Fig. 1 shows an example of pair-wise separate objects.

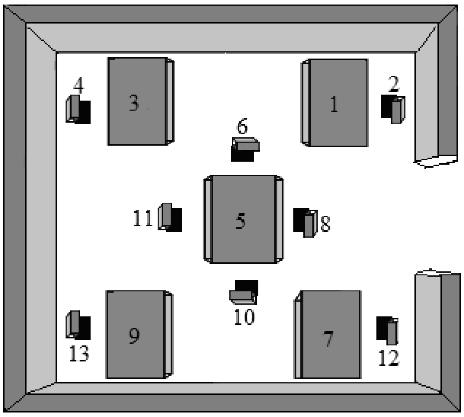


Figure 1. Pair-wise separate objects

In this figure, 13 objects have been shown. Among them, objects 1, 3, 5, 7, and 9 are made of wood and objects 2, 4, 6, 8, 10, 11, 12, and 13 are made of plastic. If we make a set of the same types of objects then we found,

Set of wooden objects, $S_W = \{1,3,5,7,9\}$

Set of plastic objects, $S_p = \{2,4,6,8,10,11,12,13\}$

Hence, we can say that S_w is the set of 5 pair-wise objects and S_p is the set of 8 pair-wise objects. As we see that the objects have no physical connection with each other, we can say that all of the objects are separate from each other.

Thus, the procedure we followed in creating the BSP tree is:

i. Create a list of all the walls and objects of the indoor environment *O*.

ii. Divide the space *O* into two parts, *O1* and *O2* by choosing one of the planes from the walls and objects as the splitter plane.

iii. Create a list named ‘front’ for the planes that are completely at the front of the splitter plane.

iv. Create a list named ‘back’ for the walls, which are completely at the back of the splitter plane.

v. When the splitter plane crosses any of the walls or objects, divide that wall or object into two parts. Then, the front part of the splitter will go to the front list and the other part into the back list.

vi. In the BSP tree construction process, the splitter plane is sited. The first plane is treated as the root of the entire tree. The objects, which are placed in front of the splitter in the environment, will go to the right side of the parent. The objects placed in the backside will go to the left side of the parent.

vii. The process (i)-(vi) will repeat for *O1* and *O2* separately until one line segment is left.

The challenge to execute the above procedure is to ensure that a balanced tree structure will be created. If the splitter intersects with the objects then the number of fragments of the scene will be increased. These may lead to potentially unbalanced tree structure. Thus, it is of utmost importance to choose the splitter wisely. Theoretically, a BSP tree can grow up to a size of $O(n^2)$, where n is the number of objects [25, 26]. In practice, however, the number of splits is smaller and it is about to the order of $O(n^{3/2})$ in our case. This is due to the principle of locality [27] used, which described that discrete geometric objects are bounded and their size is relatively small to the residing space.

With the above mentioned problem, the best solution is to avoid them completely, which is called perfect partitioning [28]. Perfect BSP is a BSP that can be acquired by daunting the limitation that each splitting plane is a plane that does not intersect at all. A perfect BSP for object set C in 3D is a binary tree T with the following properties [28]:

i. If $|C| \leq 1$ then T is a leaf; the set C is stored explicitly at this leaf.

ii. If $|C| > 1$ then the root v of T stores a plane p_v, such that the set of objects that are intersected by p_v is empty. The left (right) child of v is the root of a BSP tree $T^+(T^-)$ for the set $p_v{}^+ \cap C = s \in C : s \subset p_v{}^+ \left(p_v{}^- \cap C = s \in C : s \subset p_v{}^-\right)$, where $p_v{}^+ \left(p_v{}^-\right)$ is the region on the front (back) side of p_v and $p_v{}^+ \cap C \left(p_v{}^- \cap C\right)$ is non-empty.

This kind of perfect BSP tree is however difficult to occur in any practical indoor environment. Only for a very simple environment, it may exist. The good thing is, this perfect BSP can be exploited for partial tree generation. In our case, for example, if a portion of the complex environment is relatively simple, then the perfect partitioning will be exploited. This will not only minimize the splitting time but also decrease the execution time.

Since perfect BSP is not suitable to be used in a complex environment, we resort to hybrid heuristics. A single hybrid heuristic is composed of six heuristics as presented in [29]. For evaluating a sub-problem P, a digraph $G(P)$ is used. Fig. 2 represents a digraph of a portion of a complex environment. The graph contains a set of nodes. These nodes signify the objects of the sub-problem. The existence of an edge between x and y nodes is possible if and only if the fundamental plane of x and object of y criss-cross each other. The six heuristics [29] are for choosing the candidate splitter S:

(i) Select the minimum outdegree node S from $G(P)$, where the number of tail endpoints adjacent to a node is the outdegree of the node.

(ii) Select the maximum indegree node S from $G(P)$, where the number of head endpoints adjacent to a node is the indegree of the node.

(iii) If there is more than one solution of (i), then tie-break using (ii).

(iv) Select the node S from $G(P)$ that minimizes – $|IN(S) - OUT(S)|$.

(v) Select the node S from $G(P)$ that minimizes - $(Outdegree(S) \times |IN(S) - OUT(S)|)$.

(vi) Select the node S from $G(P)$ that minimizes - $(N \times Outdegree(S) \times |IN(S) - OUT(S)|)$.

where $IN(S)$ is the amount of objects in the half-space containing S that includes the source, $OUT(S)$ is the amount of objects in the half-space containing S that excludes the source, and N is a positive integer. This hybrid heuristic will help us choosing the proper splitter planes to create a balanced and relatively small tree.

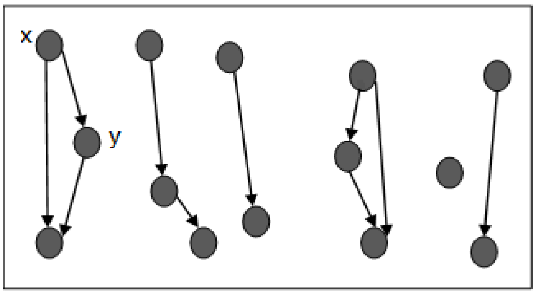
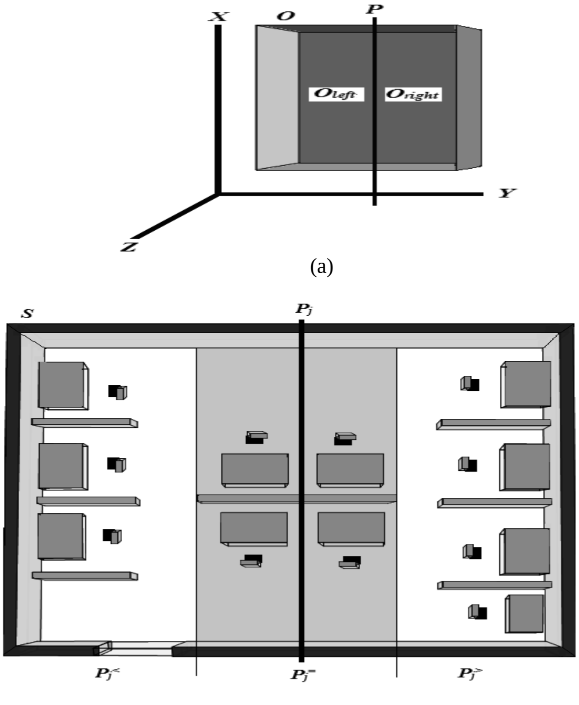


Figure 2. Illustration of digraph $G(P)$ for a portion of environment *P*

The algorithm explained so far is used for constructing a balanced tree structure and it is for a situation whereby there is no intersection between the splitter and the object. To address the situation of intersection between the splitter and the object, we used a 'window list'. It is a simple data structure introduced by Chazzelle [30, 31]. An axis parallel obstacle *O* in a 3D space IR^3 is defined as [31]:

$$O = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3] \quad (1)$$



(a)

(b)

Figure 3. Illustration of the intersection between splitter and object and creation of window list (a) Single object (b) Multiple object

Any plane *P* normal to any of the 3 axes divides IR^3 into two halves as shown in Fig. 3. Any object *R* intersected by plane *P* splits into two parts, which are non-overlapping. More specifically, for any $p \in (a_j, b_j)$, the plane $x_j = p$ splits the object $O = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$ into two parts:

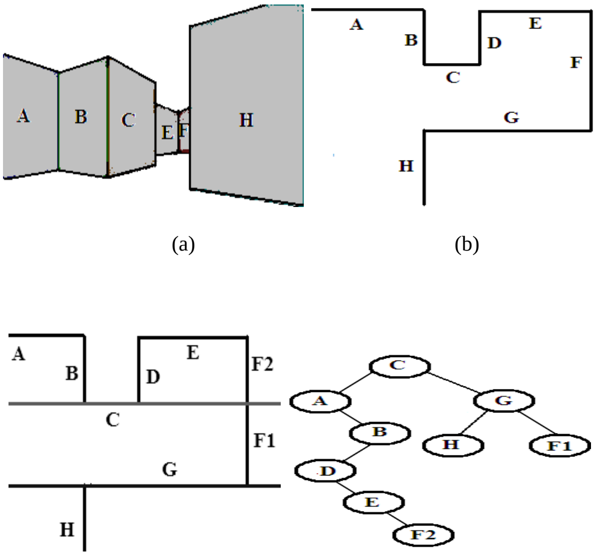
$$O_{left} = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3] \ldots \times [a_j, p] \quad (2)$$

and,

$$O_{right} = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3] \ldots \times [p, b_j] \quad (3)$$

Now let, $S = \{O_1, O_2, \ldots, O_n\} = \{[a_1, b_1], [a_2, b_2], \ldots, [a_n, b_n]\}$ be a set of *n* feasibly overlapping objects in a 3D space IR^3, and let, P_j be a random plane orthogonal to the *j*-th coordinate path. Let $P_j^<$ and $P_j^>$ indicates the set of objects lying in the part below the cut P_j and above the cut P_j, respectively. Next, let $P_j^=$ indicates the set of objects, which are intersected by the plane P_j. Let, the maximum number of objects intersected by the plane P_j is k_j^*. The profile k^* of *S* is demarcated to be minimum of all k_j^*, i.e.,

$$k^* = \min\{k_1^*, k_2^*, k_3^*\} \quad (4)$$



(a) (b)

(c) (d)

Figure 4. Creation of BSP tree from sample environment (a) Part of an environment, (b) 2D representation of (a), (c) Partitioning of (b), (d) BSP tree of (a)

Let, $l = \min_i a_i$, $r = \max_i b_i$ for $1 \le i \le n$, and $x^{\equiv}$ be the set of objects in *S* close to *x* for each $x \in [l, r].x^{\equiv}$ is named the profile of *S* at *x*. Now, $[l, r]$ will split into a non-overlapping set of *T* in parts of $I_1, \ldots, I_m$ and it is satisfied by the following [30]:

$$\left|W_j\right| \le \delta k^*\left(W_j\right) \quad (5)$$

where $\delta \ge 2$ is a constant and $k^*\left(W_j\right) = \max\left(1, \min_{x \in I_j} \left|x^{\equiv}\right|\right)$ is the minimum profile of the set W_j. Each set W_j, $1 \le j \le m$, is termed as window and the systematic sequence $W = (W_1, W_2 \ldots, W_m)$ is called the window list $W(\psi)$.

In Fig. 4, an illustration for creating the BSP tree from a sample environment is shown. In this illustration, plane *C* is chosen as the first splitter. This splitter plane cuts wall *F* into two parts, *F1* and *F2*. These two parts are now in different branches. No further intersection occurs and hence, the normal BSP procedure will continue.

III. PROPOSED ALGORITHM

In the section, we will deliberate on our proposed method. The algorithm for the object location identification will be described first and then followed by faster ray tracing algorithm. Our proposed method is based on the balanced BSP as elaborated in the previous section. For reducing the execution time and complexities, we introduced two new concepts: ICS and NOP. The ICS concept is used for searching the position of the objects and also for the ease of the 3D simulation, while the NOP concept is used for the ray-object intersection validation purpose.

First, we will discuss the position searching of an object relative to the transmitter (*Tx*). This process is carried out by determining the intersection point between the ray generated by the *Tx* and the object. The intersection point (x, y, z) is the position of the object from the *Tx*.

The ICS shown in Fig. 5 is defined as the effective surface within an object, which is used for calculating ray-object intersection. In this concept, a new invisible or imaginary surface is created inside the 3D object at a point where the ray contacted with the object. This surface will be a 2D surface and it is created at the middle of the object. It is introduced with the aim of simplifying the algorithm and hence, reducing the computational complexity. The 3D objects we used in this study are made of cubes or cuboids, which consist of six faces and eight vertices. Each vertex will be assigned a unique address or coordinate point. Using these points, the system developed in this study will calculate the coordinates of the ICS.

Assuming *k*, *l*, *m*, and *n* are the four vertices of the ICS and *C*1, *C*2, *C*3, *C*4, *C*5, *C*6, *C*7, and *C*8 are the eight vertices of the cube or cuboids, then the coordinate of *k* of the ICS is defined as:

$$abscissa\_of\_k = \frac{(abscissa\_of\_C1 - abscissa\_of\_C3)}{2} \quad (6)$$

$$ordinate\_of\_k = ordinate\_of\_C1 = ordinate\_of\_C3 \quad (7)$$

and *l*, *m*, and *n* coordinates are determined in the same manner as the *k* coordinate. When a ray is launched, the algorithm will first determine the existence of a line-of-sight (LOS) path between the *Tx* and the receiver (*Rx*). If there is a LOS path, there will be no need to find the object position. In this regard, the ray tracer will trace a source ray in a particular direction and test whether an object and the ray intersection occur.

Referring to Fig. 5, suppose *L1* is the source ray and *L2* is the ICS generated after the source ray incident to the 3D object. Here, (x_1, y_1, z_1) with vector (a_1, b_1, c_1) represents *L1*, while (x_2, y_2, z_2) with vector (a_2, b_2, c_2) represents *L2* or the ICS. If *L1* and *L2* are the two lines represented by vectors $\overline{L1}$ and $\overline{L2}$, then we have to first ascertain whether these lines are intersecting or not. For this, we used vector algebra, where the cross product of $\overline{L1}$ and $\overline{L2}$ are [32]:

$$\overline{L1} \times \overline{L2} = \left|\overline{L1}\right|\left|\overline{L2}\right| \sin \theta . \hat{\eta} \tag{8}$$

According to vector algebra, we know that if two vectors are in parallel then their cross product will be zero. Now, the algorithm will check first that (8) is zero or not. If it is not zero then the algorithm will conclude the two lines as intersecting and determine the intersecting point. Hence, we can write the parametric equations for the two lines as:

$$x = x_1 + a_1 * t_1, y = y_1 + b_1 * t_1, z = z_1 + c_1 * t_1 \tag{9}$$

and,

$$x = x_2 + a_2 * t_2, y = y_2 + b_2 * t_2, z = z_2 + c_2 * t_2 \tag{10}$$

where t_1 and t_2 are two unknown parameters.

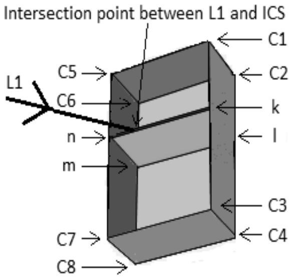


Figure 5. Creation of ICS and intersection with the source ray

At the intersection point, the (x, y, z) values of both equations will be same. Now, after equating the values of *x* and *y*, we will get:

$$x_1 + a_1 * t_1 = x_2 + a_2 * t_2 \tag{11}$$

$$y_1 + b_1 * t_1 = y_2 + b_2 * t_2 \tag{12}$$

By solving (11) and (12), we found the unknown values of t_1 and t_2 as follows:

$$t_1 = \frac{b_2(x_2 - x_1) + a_2(y_2 - y_1)}{a_2 b_1 + a_1 b_2}, t_2 = \frac{b_1(x_1 - x_2) + a_1(y_2 - y_1)}{a_2 b_1 + a_1 b_2} \tag{13}$$

After putting these values of t_1 and t_2 into (9) and (10), we can determine the intersection point (x, y, z) of *L1* and *L2*. This point is the relative position of the object from the *Tx* and hence, the position of the object has been identified. It should be noted here that the algorithm will check the position of all objects that interact with the *Tx*.

After having identified the relative position of the object with respect to the *Tx*, the next task is to identify the intersecting object. This is done by the NOP concept. With NOP, all the prerequisites are: the pre-calculated normal of the plane, the nearest pre-calculated distance from the source to the plane, and a vector going from the origin to the object. To formulate it, we first assume the following:

-*q* is a vector, which starts from the object to the origin.

-*s* is the distance from the origin to the plane.

-*l* is a vector, which is opposite to the plane’s normal direction and its length is one.

-*Q* is the angle between *q* and *l*.

From the vector analysis [32], we know that, a dot product is equal to the product of the absolute length of the two vectors, multiplied by the angle between them or sum of the product of the components. Mathematically, it can be expressed as follows:

$$dot\,(q,l) = |q||l| \cos Q = q.x * l.x + q.y * l.y + q.z * l.z \tag{14}$$

Since, the length of the normal is one to:

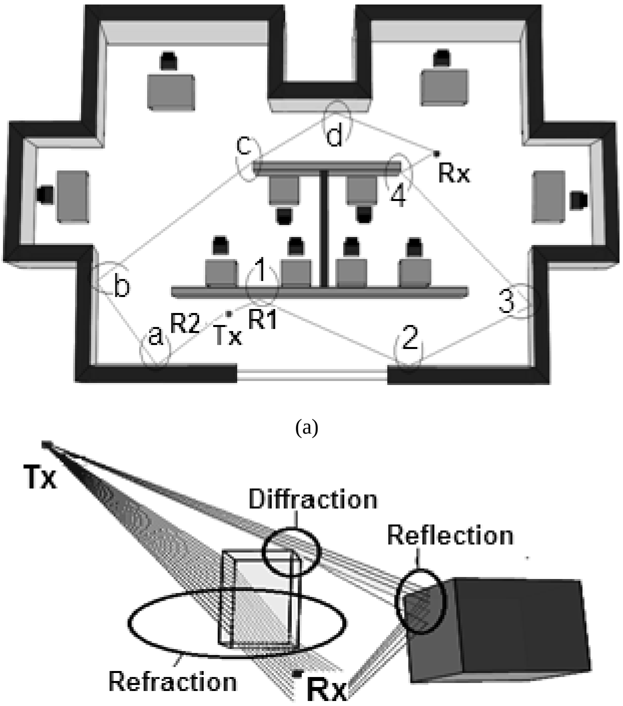
$$dot\,(q,l) = |q| \cos Q = q.x * l.x + q.y * l.y + q.z * l.z \tag{15}$$

However, one of the elements of vector *q*, $|q|\cos Q$ is in the direction of vector *l*. That means, if $dot(q,l)$ is a less significant quantity than *s*, top of the plane is denoted and if $dot(q,l)$ is superior in respect to *s*, bottom of the plane is denoted. Therefore, by subtracting them, we found:

$$Side = \operatorname{sgn}(s - dot(q,l)) \tag{16}$$

$$Side = \operatorname{sgn}(s - (q.x * l.x + q.y * l.y + q.z * l.z)) \tag{17}$$

Now, we can quickly determine which side of any object is facing the *Tx* by using (17).



(a)

(b)

Figure 6. (a) Multipath propagation in a 3D environment (b) Propagation mechanism

In an indoor environment, such as the one shown in Fig. 6(a), there will be many objects within the environment, in which most of them are overlapping. The target here is to predict the path of the significant rays emanated from *Tx* to *Rx*. In this figure, only two rays *R1* and *R2* are significant. Inside the environment, multipath propagation [33] will take place due to typical propagation mechanisms, such as reflection, refraction, and diffraction as indicated in Fig. 6(b). If the rays traverse through two or more objects at any one time, then it needs to find out the nearest object or the intersecting object. For finding the nearest object, we used the NOP technique as described before.

In the normal ray tracing method, data of the indoor environment are stored in a single list [4]. This is to say that if there are one thousand objects and we need the 1000th object for ray tracing, then it searches through the whole list. This will increase the computation time. To decrease this time, we used the balanced BSP tree as explained in Section 2 for building the database. The time required for data searching from a single list is in the order of $O(n)$, where *n* is the number of objects. In our case, it is reduced to $O(\log n)$, which is significantly smaller than $O(n)$. Note that, when an object is divided by the splitter plane, then we have to go to both of the branches, for this reason, sometimes it can be slower than $O(\log n)$.

For finding the exact object position, we used ICS. It is easier to represent one side of an object with two different points. However, in the case of electromagnetic ray, we can define the starting point, but not the finishing point. Fortunately, we know the direction of the ray. For this reason here, we represented the ray with a point and a directional vector. Thus, we can get the exact object position.

```
Input : Root splitter of BSP tree and transmitter (Tx) location
Input : Start and stop angles of source ray shooting
Input : Ray shooting angle, θ is from 0° to 259° and angle increment is 1° each time
angle = StartingAngle
Ray.initialLocation = Tx.initialLocation
repeat
    Quadrant ← FindQuadrant (angle)
    Ray ← CreateRay(Quadrant, Ray.initialLocation)
    ListOfRay.Add(Ray)
    Distance ← GetDistance(Ray.InitialLocation, Ray.FinalLocation)
    Ractangle ← CreateRectangle (Ray.InitialLocation, Quadrant )
    Ray ← FindNearestIntersectPoint (Distance, BSP_Tree, Ray, Rectangle)
    ListOfRay.Add(Ray)
    angle=CalculateAngleOfNextRay(Ray.incidentAngle)
    Ray.initialLocation = Ray.finalLocation
until (Ray.Received = “Yes” or Ray.Lost = “Yes”)
if (ListOfRay[ ListOfRay.Count - 1].IncidentAt = ”Receiver”) then
    DrawSignificantRays(ListOfRay)
end if
InitialAngle ← InitialAngle+1
if (InitialAngle <= FinalAngle) then
    Ray_ Tracer (InitialAngle, FinalAngle, BSP_Tree, Tx);
end if
```

Figure 7. Overall ray tracing algorithm

The accuracy of an algorithm depends on the number of significant rays, i.e., a higher number of significant rays means higher accuracy. For this, we have to ensure that all the rays generated by the *Tx* are reaching the *Rx*. To achieve this, the smallest ray shooting angle is needed. However, if the angle is very small then the computation time will increase. Hence, a trade-off between the shooting angle and the time is required. In our work, we have chosen one degree (1°) difference between two consecutive rays. This will ensure that the most number of significant rays are captured, while at the same time ensuring less computation time. Thus, the accuracy will also increase. The overall signal prediction algorithm is shown in Fig. 7.

IV. RESULTS AND DISCUSSION

To assess the performance of the proposed technique, experiments are done in real test bed. All the environments taken for these experiments are practical environments. Total five environments (Fig. 8 shows one of the five environments) are used for the performance evaluation of the proposed technique in comparison with the existing techniques. The simulation environments consist of multiple objects that typically exist in any indoor environment and the measurement was carried out for ten different scenarios of each environment. In the simulation, the transmitter and the receiver are represented by *Tx* and *Rx*, respectively. The difference between one scenario and the other is the position of the *Tx* and *Rx* inside the indoor environment. No specific rule is followed for placing the *Tx*(s) and *Rx*(s). They are placed in a random manner. This is because, in an indoor environment, the receiver can be anywhere and the *Tx* position is also changed for optimizing its performance. The number of *Tx*(s) and *Rx*(s) are also kept similar for all of the scenarios (we have just changed the positions of them). For fair comparison with the existing algorithms, we used the same 3D environments and all experimental settings are kept equivalent. The results for the environment of Fig. 8 have been represented in the later part of this section and the detailed results for all five environments are given as well.

For object position detection, we have compared our proposed algorithm with Multiple Time Sample (MTS) method [34] and hybrid localization method [6]. The MTS method for localization needs some experimental data to locate the object in which sometimes it is not possible to carry out these experiments. In that case, this method can be inefficient. On the other hand, the hybrid localization method needs a large database for storing the data and hence, it takes more time for computation. Fig. 9 represents the results on how accurately an object can be localized by different methods evaluated. It can be seen from Fig. 9 that although the accuracy is close to each other but the proposed method reaches the highest. Thus, we can conclude that the

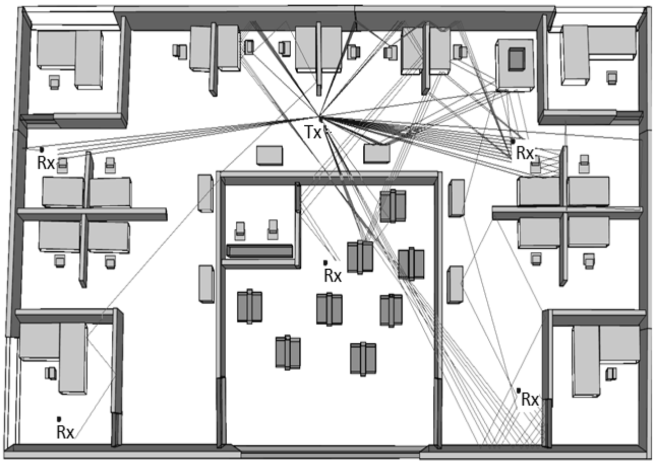


Figure 8. 3D simulation environment

proposed method is the most accurate method among the three methods evaluated.

In terms of computation time for calculating the position of an object, it can be seen from Fig. 10 that the proposed method requires the least. This is attributed to the simple vector algebra used in the proposed method, which resulted in more efficient computation time.

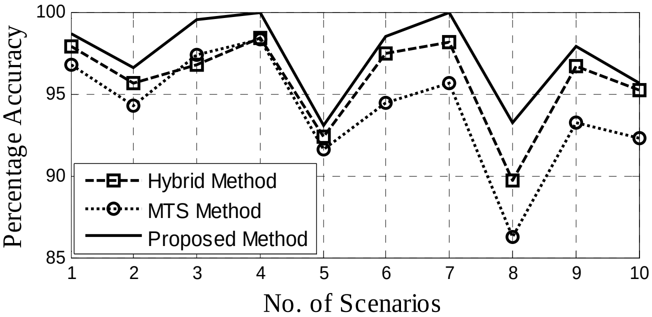


Figure 9. Comparison in terms of positioning accuracy

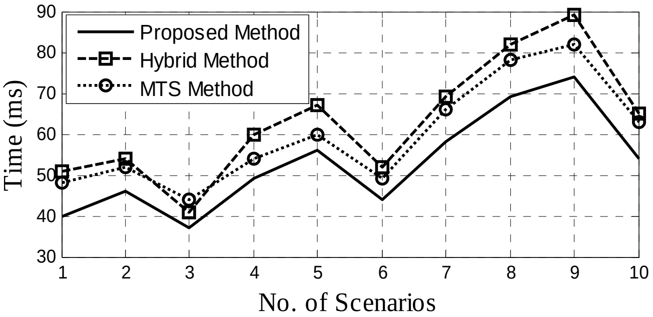


Figure 10. Comparison in terms of detection time

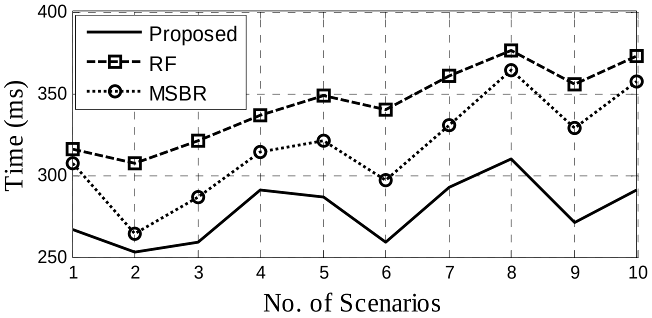


Figure 11. Comparison between different signal prediction algorithms in terms of computation time

For the propagation prediction mode, we compare the proposed method with the RF technique [7] and the MSBR technique [16]. Fig. 11 shows a comparison based on computational time for ten different scenarios as mentioned earlier. It can be observed that our proposed method gives 23.87% superior computational efficiency than the RF technique and 18.48% than the MSBR technique. Furthermore, the comparison in terms of ray prediction accuracy is presented in Fig. 12. It was found that our proposed method has a superior accuracy of 88.75% than the MSBR technique and 65.93% than the RF technique.

It can be observed from Table 1 that our proposed method gives 26% superior computational efficiency than the RF technique and 21.16% than the MSBR technique. It is also found that our proposed method has a superior prediction accuracy of 66.50% than the MSBR technique and 42.44% than the RF technique. Here, the superiority values are averages of the five environments.

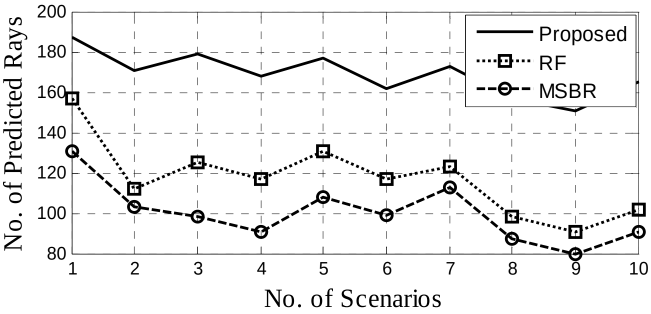


Figure 12. Comparison between different signal prediction algorithms in terms of prediction accuracy

TABLE I. OVERALL RESULTS FOR ALL FIVE ENVIRONMENTS

	Time (ms)			**No. of predicted rays**		
	Proposed	RF	MSBR	Proposed	RF	MSBR
1	278	344	317	169	117	100
2	279	365	352	123	88	71
3	207	305	287	147	105	99
4	164	247	231	187	129	121
5	338	432	402	103	72	55

V. CONCLUSION

In a ray tracing algorithm, a huge amount of time is required for storing and retrieving of object information and ray-object intersection test execution. The algorithm based on BSP tree, which is optimized by ICS and NOP, is presented in this paper. By embedding a balanced BSP tree structure, the data storing and retrieving time is minimized. The optimization techniques of ICS and NOP reduced the ray-object intersection test time by calculating the accurate and the nearest position of the object. Analysis of the results proved that the proposed algorithm has lower time complexity and higher efficiency than the existing techniques. The average results obtained from the five environments reveal that the computational efficiency of the proposed algorithm has been improved of about 26% and the accuracy has been increased up to 66.50%. The proposed ray tracing technique has a great potential to be used for planning indoor wireless networks and with some minor changes it can also be used for outdoor environments.

ACKNOWLEDGMENT

Our sincere thanks go to the University of Malaya for offering extensive financial support under the University of Malaya Research Grant (UMRG) scheme (RG098/12ICT).

REFERENCES

[1] M. F. Iskander, Y. Zhengqing, “Propagation prediction models for wireless communication systems,” IEEE Transaction on Microwave Theory and Techniques, vol. 50, no. 3, pp. 662-673, 2002.

[2] T. K. Sarkar, Ji. Zhong, K. Kyungjung, A. Medouri, M. Salazar-Palma, “A survey of various propagation models for mobile communication,” IEEE Antennas and Propagation Magazine, vol. 45, no. 3, pp. 51-82, 2003.

[3] N. Yarkony, N. Blaunstein, “Prediction of propagation characteristics in indoor radio communication environments,” Progress In Electromagnetics Research, vol. 59, 151-174, 2006.

[4] M. S. Sarker, A. W. Reza, K. Dimyati, “A novel ray -tracing technique for indoor radio signal prediction,” Journal of Electromagnetic Waves and Application, vol. 25, pp. 1179-1190, 2011.

[5] C. H. Liang, Z.-L. Liu, H. Di, “Study on the blockage of electromagnetic rays analytically,” Progress In Electromagnetics Research B, vol. 1, pp. 253-168, 2008. [Online]. Available: http://www.jpier.org/pierb/pier.php?paper=07102902

[6] A. Tayebi, J. Gomez, F. M. Saez de Adana, O. Gutierrez, “The application of ray-tracing to mobile localization using the direction of arrival and received signal strength in multipath indoor environments,” Progress In Electromagnetics Research, vol. 91, pp. 1-15,2009.

[7] H. Kim, H.-S. Lee, “Accelerated three dimensional ray tracing techniques using ray frustums for wireless propagation models,” Progress In Electromagnetics Research, vol. 96, pp. 21-36, 2009.

[8] A. W. Reza, M. S. Sarker, K. Dimyati, “A novel integrated mathematical approach of ray-tracing and genetic algorithm for optimizing indoor wireless coverage,” Progress In Electromagnetics Research, vol. 110, pp. 147-162, 2010.

[9] C. H. Teh, F. Kung, H. T. Chuah, “A path-corrected wall model for ray-tracing propagation modeling,” Journal of Electromagnetic Waves and Applications, vol. 20, no. 2, pp. 207-214, 2006.

[10] C. Wang, M. T. Thai, Y. Li, F. Wang, W. Wu, “Optimization scheme for sensor coverage scheduling with bandwidth constraints,” Optimization Letters, vol. 3, no. 1, pp. 63-75, 2009.

[11] A. W. Reza, K. Dimyati, K. A. Noordin, A. S. M. Z. Kausar, M. S. Sarker, “A comprehensive study of optimization algorithm for wireless coverage in indoor area,” Optimization Letters, DOI: 10.1007/s11590-012-0543-z, 2012.

[12] Y. B. Tao, H. Lin, H. J. Bao, “Kd-tree based fast ray tracing for RCS prediction,” Progress In Electromagnetics Research, vol. 81, pp. 329-341, 2008.

[13] J.-K. Bang, B.-C. Kim, “Time consumption reduction of ray tracing for RCS prediction using efficient grid division and space division algorithms,” Journal of Electromagnetic Waves and Applications, vol. 21, no. 6, pp. 829-841, 2007.

[14] K. S. Jin, “Fast ray tracing using a space-division algorithm for RCS prediction,” Journal of Electromagnetic Waves and Applications, vol. 21, no. 1, pp. 119-126, 2006.

[15] N. S. Alvar, A. Ghorbani, H. R. Amindavar, “A novel hybrid approach to ray tracing acceleration based on pre-processing & bounding volumes,” Progress In Electromagnetics Research, vol. 82, pp. 19-32, 2008.

[16] V.Mohtashami, A. A. Shishegar, “Modified wavefront decomposition method for fast and accurate ray-tracing simulation,” IET Microwaves, Antennas & Propagation, vol. 6, no.3, pp. 293-304, 2012.

[17] Y. Zhengqing, M.F. Iskander, Z. Zhijun, “Fast ray tracing procedure using space division with uniform rectangular grid,” Electronics Letters, vol. 36, no. 10, pp. 895-897, 2000.

[18] Y. Cocheril, R. Vauzelle, “A new ray-tracing based wave propagation model including rough surfaces scattering,” Progress In Electromagnetics Research, vol. 75, pp. 357-381, 2007.

[19] M. Thiel, K. Sarabandi, “A hybrid method for indoor wave propagation modeling,” IEEE Transactions on Antennas and Propagation, vol. 56, no. 8, pp. 2703-2709, 2008.

[20] A. W. Reza, S. M. Pillai, K. Dimyati, K. G. Tan, “A novel positioning system utilizing zigzag mobility pattern,” Progress In Electromagnetics Research, vol. 106, pp. 263-278, 2010.

[21] A. W. Reza, T. W. Yun, K. Dimyati, K. G. Tan, K. A. Noordin, “Deployment of a 3D tag tracking method utilizing RFID,” International Journal of Electronics, vol. 99, no. 4, pp. 557-573, 2011.

[22] H. Kwon, D. J. Pack, “Cooperative target localization by multiple unmanned aircraft systems using sensor fusion quality,” Optimization Letters, vol. 6, no. 8, pp. 1707-1717, 2012.

[23] C. K. Chow, S. Y. Yuen, “A multiobjective evolutionary algorithm that diversifies population by its density,” IEEE Trans. on Evolutionary Computation, vol. 16, no. 2, pp. 149-172, 2012.

[24] A. Dumitrescu, J. S. B. Mitchell, M. Sharir, “Binary space partitions for axis-parallel segments, rectangles, and hyperrectangles,” Discrete & Computational Geometry, vol. 31, no. 2, pp. 207-227, 2004.

[25] H. Radha, M. Vetterli, R. Leonardi, “Image compression using binary space partitioning trees,” IEEE Trans. on Image Processing, vol. 5, no. 12, pp. 1610 - 1624, 1996.

[26] Toth, C. D., “Binary space partitions : recent developments,” Combinatorial and Computational Geometry, MSRI Publication, vol. 52, pp. 529-555, 2005.

[27] Z. Lining, W. Lipo, L. Weisi, “Generalized biased discriminant analysis for content-based image retrieval,” IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 42, no. 1, pp. 282-290, 2012.

[28] M. de Berg, M. M. de Groot, M. H. Overmars, “Perfect binary space partitions,” Computational Geometry: Theory and Applications, vol. 7, no. 1-2, pp. 81-91, 1997.

[29] G. S. Alijani, R. Krishnaswamy, “On constructing binary space partitioning trees,” ACM Annual of Computer Science Proceedings, vol. 18, pp. 230-235, 1990.

[30] B.Chazelle, “Filtering search: A new approach to Query-Answering,” SIAM Journal of Computing, vol. 15, pp. 703-724, 1986.

[31] M. I. Shamos, F. P. Preparata, Computational Geometry: An Introduction, Springer-Verlag, New York, 1985.

[32] T. P. Humphreys, A Reference Guide to Vector Algebra, Jain Pub Co; Pap/Cdr Edition, Fremont, CA, 2010.

[33] M. Carvalho, A.S., V. Boginski, B. Balasundaram, “Topology design for on-demand dual-path routing in wireless networks,” Optimization Letters, DOI: 10.1007/s11590-012-0453-z, 2012.

[34] C. Xu, D. Schonfeld, A. A. Khokhar, “Localization and Trajectory Estimation of Mobile Objects Using Minimum Samples,” IEEE Trans. on Vehicular Technology, vol. 58, no. 8, pp. 4439-4446, 2009.