M. J. Islam, A. W. Reza*, K. A. Noordin, A. S. M. Z. Kausar and H. Ramiah

Red-Black Tree Based Fast and Accurate Ray Tracing for Indoor Radio Wave Prediction

Abstract: Ray tracing is a well-known technique to investigate the multipath propagation in the complex indoor environment. However, it still suffers from the computation time and ray prediction accuracy for the complex indoor environment. Therefore, a new three dimensional (3D) ray tracing based on the Red-Black tree along with object skipping and surface extraction techniques for the complex indoor environment is presented in this paper. The Red-Black tree data structure provides a faster object retrieval mechanism while object skipping technique prevents the unnecessary objects to take part in intersection tests, thus accelerates the ray tracing. Conversely, the surface extraction is a novel technique that makes the tracing of the ray paths easily and accurately, mostly for the complex 3D environment. In addition, the calculation of exact ray paths that reach the receiver after multiple reflections, refractions and/or diffractions is also considered in this paper. The obtained results show that the proposed method provides superior improvement of ray prediction time (75.65% on average) and ray prediction accuracy (27.78% on average) than the existing ray tracing methods.

Keywords: ray tracing, Red-Black tree, surface extraction, object skipping, radio wave propagation

PACS® (2010). 41.20.Jb

***Corresponding author: A. W. Reza:** Department of Electrical Engineering, Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia. E-mail: awreza98@yahoo.com
M. J. Islam, K. A. Noordin, A. S. M. Z. Kausar, H. Ramiah: Department of Electrical Engineering, Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia

1 Introduction

The use of wireless communication is increasing gradually for its simplicity that increases the demands of wireless communication services. Proper design of wireless networks needs deep understanding and accurate characterization of radio wave propagation in indoor environments. Radio wave propagation in outdoor and indoor environments is not identical. In case of outdoor environments, most of the propagated radio waves arrived at the receiver by simple line-of-sight propagation, whereas inside the buildings, the propagated radio waves reached the receiver by simple line-of-sight propagation and/or multiple reflections, refractions, diffractions or transmission. As a consequence, the efficient and accurate ray path prediction of propagating waves for describing the propagation characteristics still constitutes the most important component in designing and establishing a wireless indoor network.

Therefore, many radio wave propagation prediction models were developed over the past two decades for indoor environments. The most straightforward and commonly used method is a statistical propagation model, which is developed based on reference measurements. In [1], statistical models are developed for indoor radio propagation based on the results of extensive multipath propagation measurements in two office buildings. This model was not successful due to access-delay and lack of accuracy. Therefore, some numerical techniques have been proposed that can take into account the site-specific feature. The site-specific radio wave propagation models [2–4] are based on electromagnetic wave propagation theory. Site-specific models do not spend time on extensive measurements; they provide accurate prediction results using appropriate knowledge of the environment. Based on this, many ray tracing methods can be found in the literature, such as Image method [5–7], Shooting and Bouncing Ray (SBR) method [8–10], Brute-Force (BF) method [11], Conventional Ray Tracing (CRT) method [12], and Bi-Directional Path Tracing (BDPT) method [13]. The image method is accurate but not efficient for the complex environment where the number of objects is relatively large. Hence, its application is limited to simple environments. On the other hand, ray tubes are used in an SBR method to face the spherical wavefront at the receiving location. If a receiver is to be found in the overlapping region between the ray tubes then the receiver will receive two rays, resulting ray double counting error. Proper distribution of wavefronts [9] can improve the accuracy of the SBR method and remove the ray double counting error, otherwise, it might obtain inaccurate ray prediction results. The BDPT gives erroneous ray prediction results in some cases and fails to reduce computational time due to the huge amount of intersection tests. However, the BF method offers an easy way to predict the propagated ray than the image, SBR, CRT, and BDPT methods. The BF method launches rays in all feasible directions and obtains better accuracy. However, the main disadvantage of BF method is an increasing ray prediction time that grows geometrically with the analysis area, complexity of the environment, the number of considered effects, or the number of rays launched and tested. Therefore, ray tracing still suffers from computation time and accuracy for the complex indoor environment.

The purpose of this study is to develop a new three dimensional (3D) ray tracing model that can trace the propagated waves efficiently and accurately in the complex 3D indoor environment. The proposed 3D ray tracing method is based on the surface extraction and Red-Black tree data structure along with object skipping technique. The surface extraction is used to identify the correct hit point on the surface of an object and thus provides the exact propagation paths of the rays that reach the receiver after multiple reflections and/or refractions or diffractions. On the other hand, the Red-Black tree data structure provides a faster object retrieval mechanism so that the exact object can be found within a logarithmic time instead of linear search time during the intersection operation. Conversely, the object skipping technique is used to skip unnecessary objects during the intersection tests that reduced the significant amount of ray prediction of the proposed ray tracing method. The proposed method is compared with the existing ray tracing methods. The simulated results show that, the proposed model provides satisfactory results in terms of number of predicted rays and computational time. The proposed ray tracing method is described in Section 2 and results and performance evaluation are presented in Section 3. Finally, the conclusion is depicted in Section 4.

2 Proposed method

The ray tracing process in 3D [14] is quite similar to two dimensional ray tracing model [15–18], but it can take more time to predict the propagation paths since ray tracing comes into existence by extensive ray and object intersection test. Therefore, to overcome this drawback, the proposed method uses a well-known balanced binary tree, namely, Red-Black tree [19–21]. In this study, this tree is used to store the 3D objects of the simulation environment. In the next subsections, we firstly describe the important mechanisms of the proposed ray tracing method. Afterwards, the details of the proposed ray tracing method and its complexity analysis will be deliberated.

2.1 Construction of Red-Black tree

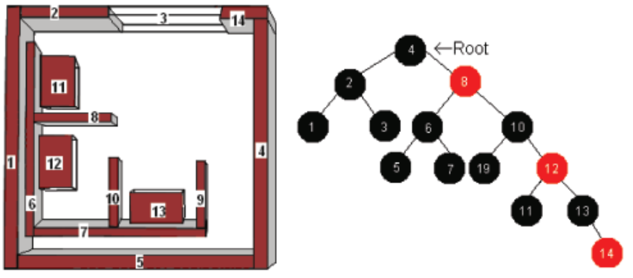
Red-Black tree is able to reduce the significant amount of ray prediction time by providing a faster object retrieval mechanism. The computational complexity for insertion, deletion or searching of an object for indoor scenario can be shown to be $O(\log_2 N)$, where N is the number of objects in an indoor scenario. In this study, the Red-Black tree is constructed according to the unique object ID of each object.

In Figure 1, the Red-Black tree constructed from fourteen objects is displayed and the corresponding flowchart of object insertion in Red-Black tree is given in Figure 2.

2.2 Surface extraction and calculation of intersection point

To easily simulate the indoor environment, each object is created using 3D cubes or cuboids based on the knowledge of 3D geometry, as shown in Figure 3(a).

During the simulation, each transmitted ray has to launch from the location of *Tx*. After that, an intersection test between the ray and the object surface is required. In a real situation, the signals emitted from the transmitter hit everywhere on the surface of 3D objects and reflect (or refract/diffract) back to other objects (door, window, wall,



(a) (b)

Fig. 1: (a) A 3D indoor scenario. (b) Corresponding Red-Black tree.

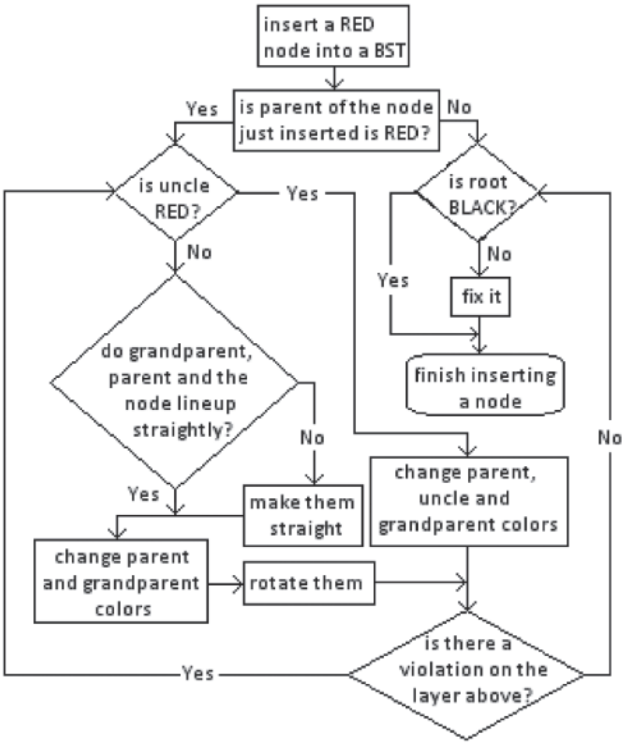

insert a RED node into a BST
is parent of the node just inserted is RED?
Yes
No
is uncle RED?
Yes
No
is root BLACK?
No
Yes
fix it
finish inserting a node
do grandparent, parent and the node lineup straightly?
No
Yes
make them straight
change parent, uncle and grandparent colors
No
change parent and grandparent colors
rotate them
is there a violation on the layer above?
Yes

Fig. 2: Flowchart of insertion of an object in a Red-Black tree.

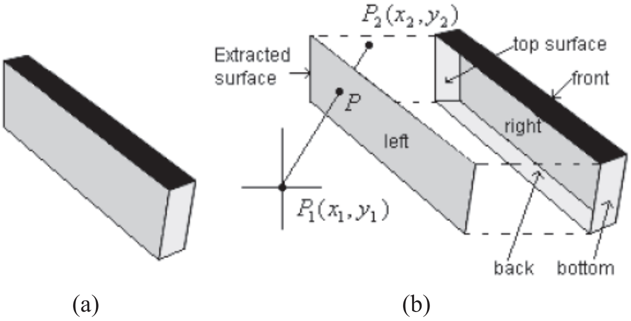


Fig. 3: (a) A 3D object. (b) Surface extraction and determination of intersection point *P*.

and furniture). However, in a simulation environment, it is quite difficult to determine such kind of random intersection points on a surface of a 3D object. To overcome this type of difficulties and to easily handle the 3D objects, the surface extraction technique is introduced here. According to the surface extraction technique and based on the quadrant of the ray, particular surfaces will take out from the 3D object for performing a ray-surface intersection test. Referring to Figure 3(b), a ray travels into the first quadrant with respect to its origin and left surface is extracted from the 3D object to find out the intersection point *P*. Therefore, this intersection point is calculated using Eqs. (1)–(4) as given below. Given a polygon (extracted surface) with *m* vertices $P_1(x_1,y_1)$ through $P_m(x_m,y_m)$ and surface normal is denoted by *N*, if $P_0(x_0,y_0)$ is the ray origin and $v(m,n)$ is the direction vector of that ray, then a point $P(x,y)$ on the ray is calculated by the following equations:

$$x = x_1 + t(x_2 - x_1) \tag{1}$$

$$y = y_1 + t(y_2 - y_1) \tag{2}$$

Therefore, the intersection point between the ray $P = P_0 + tv$ and the plane containing the polygon will be computed with the implicit equation

$$(P - P_1) \cdot N = 0 \tag{3}$$

Solving for *t* to get

$$t = \frac{(P_1 - P_0) \cdot N}{v \cdot N} \tag{4}$$

where "·" is a dot product. If $v \cdot N = 0$ then no intersection point exists between the ray and polygon, otherwise intersection point *P* is calculated by substituting the *t* value of Eq. (4) in Eqs. (1) and (2).

2.3 Identification of closest intersection point

A ray may intersect more than one object when it travels in the complex environment and thus, multiple intersections could occur.

Hence, we have to find out the closest intersection point. In this study, the closest intersection point is identified based on the distance from the ray's emanating point. For instance, given two coordinates of a ray, R_{start} and R_{end} intersects with two objects at P_1 and P_2. If *d* is the distance between R_{start} and P_1, d_1 is the distance between R_{start} and P_2, and the distance between R_{start} and R_{end} is d_2, then the closest intersection point according to Figure 4 is $P_{1,}$

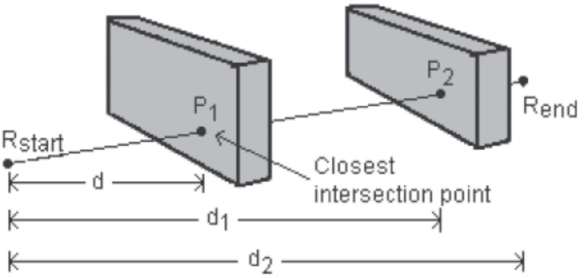


Fig. 4: Closest intersection point P_1 identified from multiple ray-surface intersections.

```
Algorithm : FindClosestIntersectPoint(minDistance, RB_Tree , Ray, Reactangle)
Input: Distance between start and end location of Ray (minDistance), Red-Black tree and emanated
       ray (Ray).
Output: Ray with new end location (closest intersection point)

if (Rectangle.Containes(RB_Tree.Object) && RB_Tree.Object!=null) then
  IntersectionPoint ← FindIntersectionPoint(Ray, Extract_Surface(RB_Tree.Object, Ray.quadrant))
  if (IntersectionPoint != Empty) then
    distance ← GetDistance(Ray.StartLocation, IntersectionPoint)
    if (distance < minDistance) then
      Ray.EndLocation ← IntersectionPoint;
       minDistance ← distance
    end if
  end if
end if
// else skip current object
if (RB_Tree.Left != NULL) then
   FindClosestIntersectPoint (minDistance RB_Tree.LeftNode, Ray, Rectangle)
end if
if (RB_Tree.Right != NULL) then
  FindClosestIntersectPoint (minDistance, RB_Tree.RightNode, Ray, Rectangle)
end if
return Ray
```

Fig. 5: A recursive algorithm for finding the closest intersection point using the object skipping technique.

because $d < d_1 < d_2$. In this paper, a recursive Red-Black tree traversal algorithm along with object skipping technique is shown in Figure 5, which is able to search the closest object and the intersection point on an object surface with the minimal computation time.

2.4 Reflection, refraction, transmission, and diffraction of the emanated ray

After finding the closest ray-surface intersection point, the obtained intersection point is used as the origin of the next ray that might be reflected, refracted, diffracted or transmitted through an object. In this study, when a ray in air hits on any glass object, a refracted ray will be generated as shown in Figure 6(a). On the other hand, if the object is made of any other materials, the reflected ray is generated as presented in Figure 6(b). The diffracted ray is generated when a ray hits on an edge of an object, as shown in Figure 6(c).

Indoor radio propagation is influenced by building materials that can change the entire propagation paths of reflected, refracted, and diffracted rays. Therefore, an accurate calculation of direction of such rays can only achieve the accurate path of predicted rays. Referring to Figure 6(b), a ray R_i incident at point P on an object surface. If the angle of incident and reflected ray is θ_i and θ_r, respectively, then according to the law of reflection, the angle of the reflected ray is equal to the angle of the incident ray, that is, $\theta_r = \theta_i$. Let $\vec{n}_i$ and $\vec{n}_r$ be the unit vectors along the direction of incident and reflected ray, respectively, with $\vec{S}$ being the normal vector at point P. Further, let $(\vec{n}_i)_{\parallel}$ and $(\vec{n}_i)_{\perp}$ denote the component vector of $\vec{n}_i$, parallel and perpendicular to the surface at P, respectively. Similarly, $(\vec{n}_r)_{\parallel}$ and $(\vec{n}_r)_{\perp}$ denote the component vector of $\vec{n}_r$, parallel and perpendicular to the surface at P. The

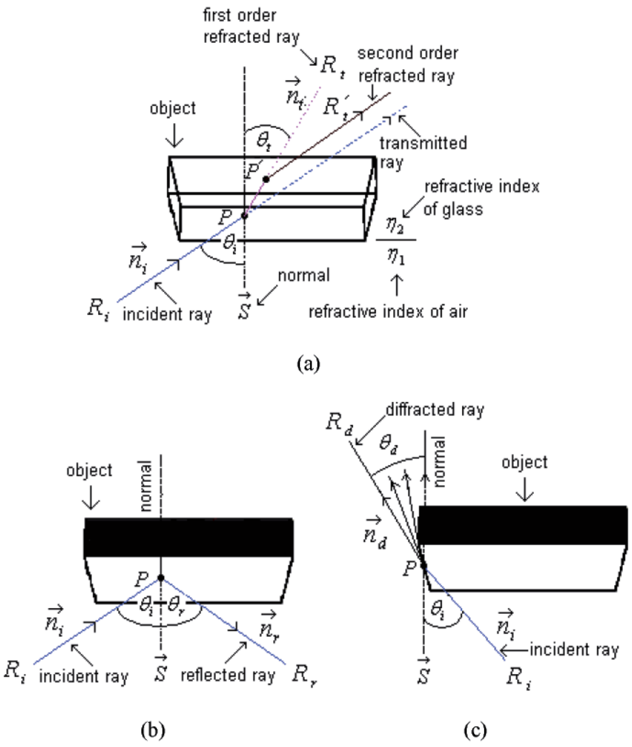


Fig. 6: Illustration of different rays: (a) refraction and transmission of incident ray, (b) reflection of incident ray, (c) diffraction of incident ray.

direction of the reflected ray can be calculated using Equation (5) by applying vector geometry [22, 23].

$$\vec{n}_r = \vec{n}_i - 2\left(\vec{n}_i \cdot \vec{S}\right)\vec{S} = \vec{n}_i - 2\cos\theta_i\,\vec{S} \tag{5}$$

where $\cos\theta_i = \vec{n}_i \cdot \vec{S}$. Similarly, $(\vec{n}_t)_{\parallel}$ and $(\vec{n}_t)_{\perp}$ denote the component vector of $\vec{n}_t$, parallel and perpendicular to the surface at P. If θ_t and η_1 and η_2 be the angle of refracted ray and refractive index of air and glass, respectively then the direction of the refracted ray can be calculated as follows:

$$\vec{n}_t = \frac{\eta_1}{\eta_2}\vec{n}_i + \left(\frac{\eta_1}{\eta_2}\cos\theta_i - \sqrt{1-\sin^2\theta_t}\right)\vec{S} \tag{6}$$

where $\sin^2\theta_t = \left(\frac{\eta_1}{\eta_2}\right)^2 \sin^2\theta_i = \left(\frac{\eta_1}{\eta_2}\right)^2\left(1-\cos^2\theta_i\right)$.

By the same way, the direction of the diffracted ray as shown in Figure 6(c) is calculated by the following equation [24]:

$$\overrightarrow{PR_i}\cdot\overrightarrow{S} = \overrightarrow{PR_d}\cdot(\overrightarrow{-S}) \tag{7}$$

where R_i is the origin of the incident ray, R_d is the end point of the diffracted ray, and $\vec{S}$ is the direction vector. Here, the direction vector $\vec{S}$ is oriented such that $\vec{S}\times\vec{W} = \vec{N}$, where $\vec{W}$ is a vector that lies in the plane of one of the two surfaces of the wedge and $\vec{N}$ is the unit vector.

2.5 Ray tracing procedure

For ray tracing, this study considers a 3D indoor environment in Figure 7(a) and a Red-Black tree of that environment in Figure 7(b).

There are thirty one objects exist in this environment, which are marked by numerical values (1–31). At the beginning, ray *TxZ* has to launch from the *Tx* and build it using the knowledge of triangular geometry. At the next step, find all object intersections (*A*, *W*, *X*, and *Y*) with this ray. Among these, the intersection point *A* is the closest intersection point, as shown in Figure 7. To calculate this intersection point, first we should determine in which quadrant the ray *TxZ* is travelling. A ray will be traveled either in the first, second, third, and fourth quadrant, if the incident angle of a ray is in between 0–90, 91–180, 181–270, and 271–360 degrees, respectively. It can be

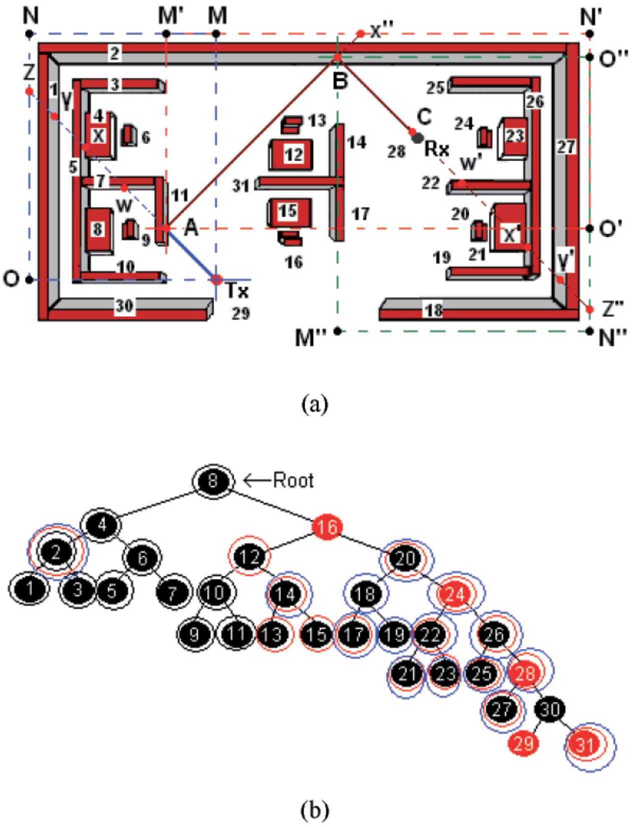


Fig. 7: (a) Illustration of an indoor scenario where each object is labeled by its own ID. (b) A Red-Black tree is constructed from the objects of that environment.

observed from Figure 7(a) that the ray *TxZ* travels in the second quadrant and thus, a rectangle *TxMNO* is needed to form in the second quadrant to identify the objects belong to this rectangle. Referring to Figure 7(b), the object nodes marked by the black circles lie in the rectangle *TxMNO*. To this point, intersection tests between the ray *TxZ* and objects in the rectangle *TxMNO* are applied to find out the closest object for the intersection point *A*. After getting the desired object from the Red-Black tree, particular surfaces (right/bottom surface of the second quadrant) are extracted from the 3D object to perform a ray-surface intersection test, namely, surface ignoring technique. The whole operation is done by the algorithm as stated in Figure 5. Based on the object properties, either ray *TxZ* reflects, refracts, transmits or diffracts from the closest intersection point *A*. The direction of these rays can be calculated using Eqs. (5)–(7). It can be seen in Figure 7 that the ray *TxZ* is reflected from the object because the intersection point *A* found on the object surface made of brick. Therefore, from the location of *A*, a new ray *AX''* is needed to build in the first quadrant. Afterwards, a rectangle *AM'N'O'* is needed to form in the first quadrant. Only the objects marked by the red circles in the tree nodes (Figure 7(b)) that lie in the rectangle *AM'N'O'* are taking part in intersection test and thus, the closest intersection point *B* can be found using the algorithm shown in Figure 5. Similarly, the objects marked by the blue circles in the tree nodes that lie in the rectangle *BM''N''O''* are taking part in intersection test to find out the closest intersection point *C*, and so on. That is, in each intersection test, only the objects that lay on a particular quadrant are taking part in intersection test and the objects in three other quadrants are completely skipped. Referring to Figure 7, a total of 55 objects (20 + 16 + 19) is skipped during an intersection test for the ray segments *TxZ*, *AB*, and *BC*, respectively. Hence, skipping unnecessary objects during intersection test can reduce huge amount of ray prediction time of the proposed method. Moreover, the computational time is also reduced by ignoring unnecessary object surfaces while an intersection test is performed with an object and thus, further acceleration is possible by the proposed ray tracing method.

Figure 8 shows some significant signals obtained by the proposed simulation model. Here, the circles labeled by *Tx* and *Rx* represent the transmitter and receiver, respectively. The circles labeled with *A*, *B*, and *C* represent reflection, refraction, and diffraction of ray, respectively.

The path of the wave from *Tx* to *Rx* is composed of the connected line (ray) segment; if the last ray segment of a wave path is intersected with any *Rx* then this ray path is considered as a significant ray path. If it passes away the indoor environment then this ray path is discarded. Afterwards, another ray is needed to launch from the position of *Tx* to trace the rest of the rays in the indoor environment. The overall ray prediction algorithm is included in Figure 9.

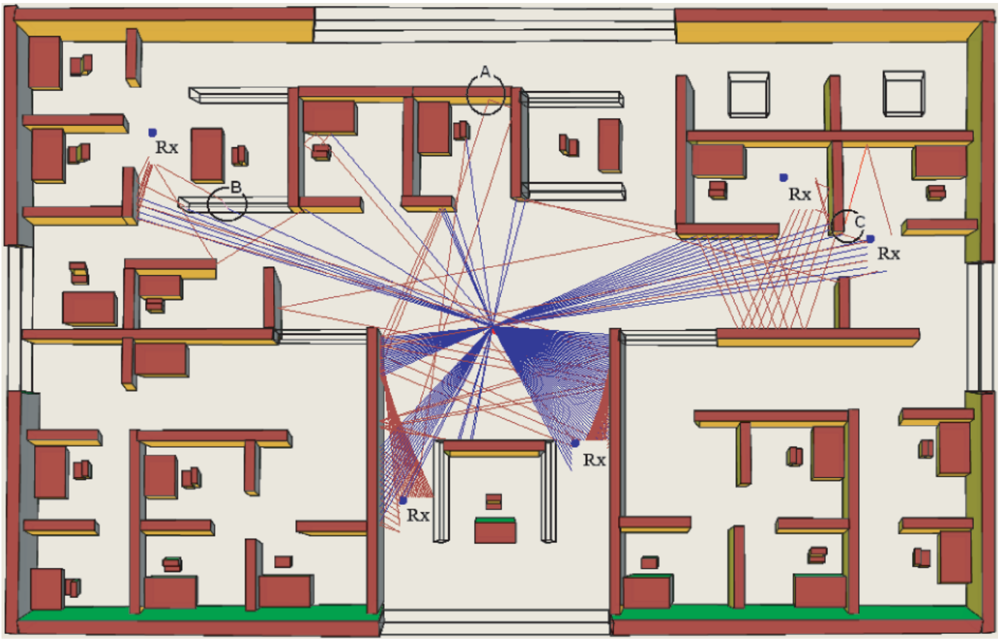


Fig. 8: Simulated results in a complex 3D indoor environment.

Algorithm: Ray_ Tracer (StartingAngle, EndingAngle, RB_Tree, Tx)

Input: Red-Black tree, Start location of Tx, start and end angle of s ource ray

Output: Draw only the significant Ray’s

```
angle = StartingAngle
Ray.startLocation = Tx.startLocation
repeat
    Quadrant ← FindQuadrant (angle)
    Ray ← CreateRay(Quadrant, Ray.startLocation)
    ListOfRay.Add(Ray)
    Distance ← GetDistance(Ray.StartLocation, Ray.EndLocation )
    Ractangle ← CreateRectangle (Ray.St artLocation, Quadrant )
    Ray ← FindClosestIntersectPoint (Distance, RB_Tree, Ray, Rectangle)
    ListOfRay.Add(Ray)
    angle = CalculateAngleOfNextRay(Ray.incidentAngle)
    Ray.startLocation  = Ray.endLocation
until (Ray.Received = “Yes”  or Ray.Lost = “Yes”)
if (ListOfRay[ ListOfRay.Count - 1].IncidentAt = “Receiver”)  then
      DrawOnlySignificantRays(ListOfRay)
end if
StartingAngle ← StartingAngle+1
if (StartingAngle <= EndingAngle) then
    Ray_ Tracer ( StartingAngle , EndingAngle, RB_Tree, Tx);
end if
```

Fig. 9: Proposed recursive ray tracing algorithm.

2.6 Complexity analysis

In a Red-Black tree, each node v has one extra bit of its color $v.color \in \{Red, Black\}$ and the following three properties are satisfied [20]:

i. The root node is black and every leaf node is also black.
ii. If a node is red, then both of its child nodes are also black.
iii. For each node v, all paths from v to its leaves have the same number of black nodes.

Let, h be the height of the Red-Black tree. According to property (ii), at least half of the nodes on any simple path from the root node to a leaf node must be black. Therefore, the black-height of the root must be at least $h/2$ and thus,

$$N \geq 2h/2 - 1 => (N+1) \geq 2h/2 \tag{8}$$

After solving Eq. (8) and taking logarithms on both sides yields:

$$h \leq 2\log_2(N+1) \tag{9}$$

Hence, the search operation of a Red-Black tree can be implemented in $O(\log_2 N)$ time according to Eq. (9). Based on this equation, the time complexity of the proposed method is described below.

Let N be the number of objects, S be the total number of surfaces of each 3D object, and T be the intersection testing time for the proposed method. If M number of intersections are required to predict each significant ray then the total intersection testing time is calculated by the following equation:

$$T = M \times S \times O(\log_2 N) \tag{10}$$

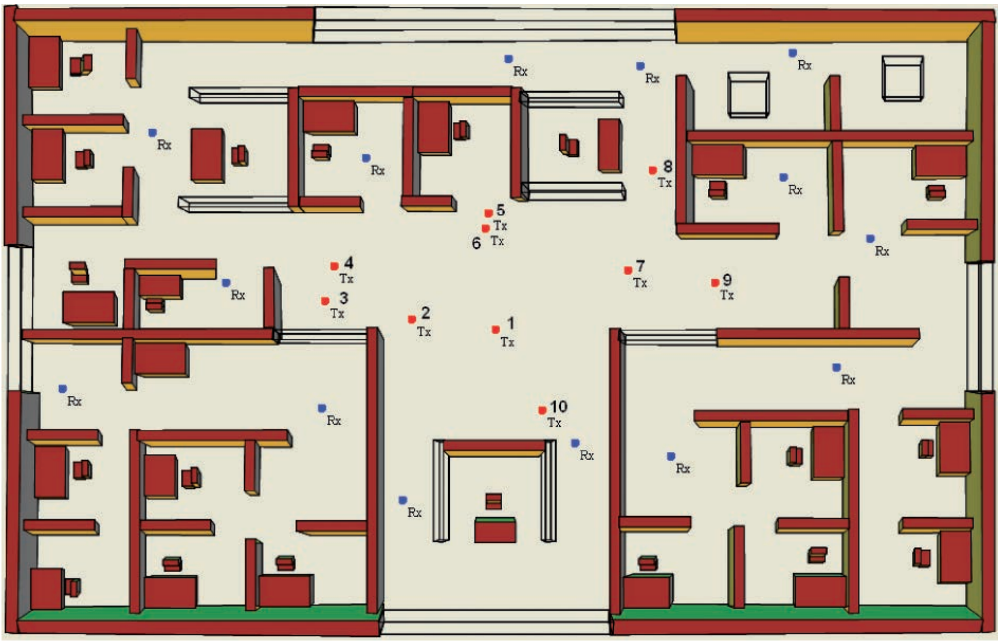


Fig. 10: Illustration of a complex indoor environment with fourteen *Rx* and labels beside the *Tx* represent different locations of same *Tx*.

Moreover, according to the object skipping technique, the proposed method can skip the significant amount of object during each intersection test. Let N' be the average number of skipped objects, then Eq. (10) becomes:

$$T = M \times S \times O(\log_2(N - N')) \tag{11}$$

Furthermore, the surface ignoring technique is applied to ignore the unnecessary object surface and thus, it reduces the computational time once again. If S' number of surfaces out of a total S number of surfaces takes part in an intersection test, then we can rewrite Eq. (11) as:

$$T = M \times (S - S') \times O(\log_2(N - N')) \tag{12}$$

However, the intersection testing time for the existing methods [9], [12], and [13] would be

$$T = M \times S \times O(N) \tag{13}$$

where $O(N)$ is the worst case object searching time and N be the number of objects in an indoor environment.

3 Results and discussion

The simulation environment as illustrated in Figure 10 is implemented over Visual Studio 2008 using C Sharp Object Oriented Programming and the experimental settings of various scenarios are kept equivalent for performance evaluation.

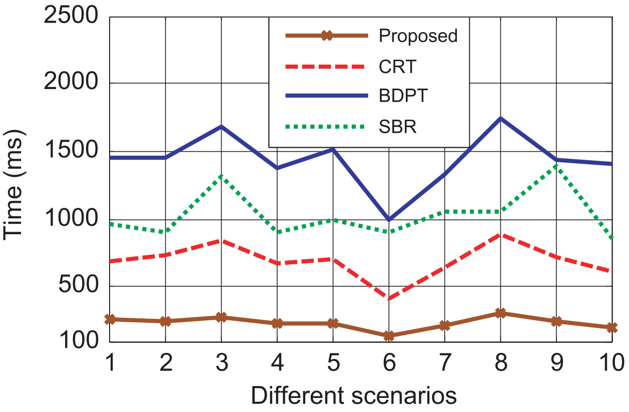


Fig. 11: Comparison of different ray tracing techniques in terms of ray prediction time.

The proposed method is compared with SBR [9], CRT [12], and BDPT [13] methods. The results are taken by moving one *Tx* in 10 different locations while keeping 14 *Rx* steady in the simulation environment. The numbers beside the *Tx* represent different scenarios. Figure 11 shows the comparison, which only illustrates the computation time that considers 10 different scenarios as presented in Figure 10.

The computational time is recorded in millisecond (ms). It can be observed from Figure 11 that the proposed method gives a higher computational efficiency of about

Table 1: Intersection time of different ray tracing methods.

Ray tracing method	Intersection time
SBR [9]	$S \times O(N)$
CRT [12]	$S \times O(N)$
BDPT [13]	$S \times O(N)$
Proposed method	$(S - S') \times O(\log_2(N - N'))$

75.65% on average compared to existing ray tracing methods as described in [9], [12], and [13]. This is because the ray-surface intersection test consumes the significant amount of ray prediction time in a ray tracing method. This test is performed each time a new ray segment is generated in the simulation process. Therefore, when the number of objects in a complex indoor environment is relatively large, the ray tracing procedure would be very slow (required $O(N)$ time) for [9], [12], and [13]. Conversely, the proposed method smartly reduces the ray prediction time by using Red-Black tree data structure and object skipping technique in which ray-object intersection time reduced from $O(N)$ to $O(\log_2(N-N'))$. In addition, only two object surfaces are taking part in the ray-surface intersection test, thus further enhancing the ray prediction time of the proposed method. Table 1 shows the comparison, which is based on the computational complexity of the proposed and existing ray tracing techniques.

Here, N is the number of objects and S is the number of surfaces of each object. Referring to Table 1, SBR, CRT, and BDPT required more computation time than that of the proposed ray tracing method because of too much involvement of objects and object surfaces during the intersection test. On the other hand, the proposed method uses Red-Black tree to store the objects whose searching time complexity (worst case) is $O(\log_2 N)$. According to the object skipping and surface ignoring techniques, if on average N' number of objects and S' number of surfaces are skipped during intersection test, the proposed technique will require much less ray prediction time compared to the existing ray tracing methods.

Moreover, as the accuracy is same with the CRT, Figure 12 shows a comparison between SBR [9] and BDPT [13] methods, which is based on the number of predicted rays.

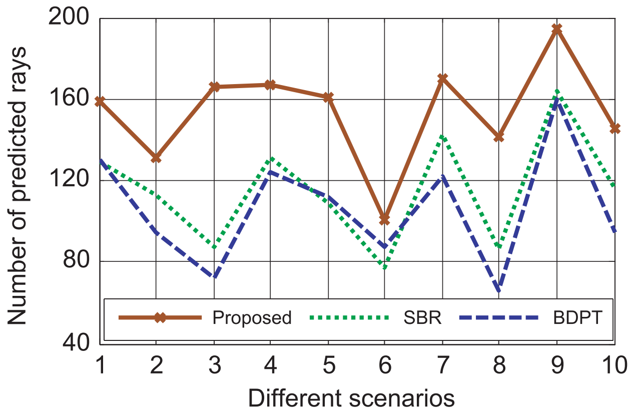


Fig. 12: Comparison of different ray tracing techniques in terms of number of predicted rays.

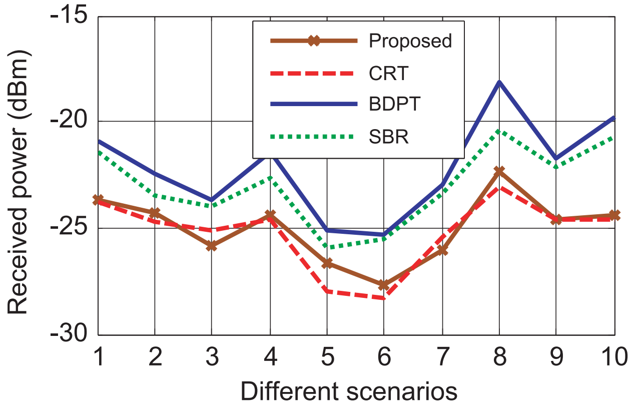


Fig. 13: Comparison of different ray tracing techniques in terms of the received power.

Referring to Figure 12, the proposed method predicts more rays of about 27.78% on average than the existing ray tracing methods. The main reason is that, the proposed method considers the followings: (i) rays are launched in all possible directions in the simulation space that considers refraction, penetration, and diffraction along with reflection, (ii) the exact and accurate intersection point can be determined using surface extraction technique, and (iii) there is no possibility to ignore any valid objects during ray-object intersection test.

In addition, a comparison with other methods (SBR, CRT, and BDPT) based on the received power is shown in Figure 13. This result is taken from the simulation environment, as presented in Figure 10.

A semi-spherical antenna operating at 2.44 GHz with 15 dB gain and radiating 10 mW is considered as the transmitter (*Tx*). At the receiver side, same type of antenna is used. The Uniform Theory of Diffraction (UTD) [24] is used for the calculation of diffraction coefficient. Whereas, the reflection and transmission coefficient is calculated from [25]. Refraction is considered only for the transparent objects of the simulation environment. The electrical properties of the materials [26] used in the simulation are listed in Table 2.

It can be seen from Figure 13 that, the proposed method predicts almost the same amount of received power as the CRT method [12]. However, there is a slight difference in the received power is observed (about

Table 2: Material properties.

Carrier frequency	2.44 GHz
Height of the walls	2.8 m
Thickness of the outer glass and brick walls	0.21 m
Thickness of the inner glass and brick walls	0.12 m
Thickness of the wooden table	0.04 m
Permittivity of the brick	5.2
Permittivity of the glass	3.0
Permittivity of the wood (table and chair)	3.0
Height of transmitter and receiver	1.4 m
Reflection	Activate
Transmission	Activate
Diffraction	Activate

2.42 dBm) in the proposed method compared to the SBR [9] and BDPT [13] methods. Because, the energy reaches the destination through multiple reflections, refractions, and diffractions of the rays and contributes to the received power in the indoor environment. These rays possibly will follow different propagation paths based on the approaches used in the different ray tracing algorithms. Therefore, differences on the received rays are observed for the different ray tracing algorithms (as shown in Figure 12) that also makes a difference on the received power (as demonstrated in Figure 13). The obtained results in Figure 13 show satisfactory performance and good agreement.

4 Conclusion

In this study, a new 3D ray tracing technique for indoor scenario is introduced where Red-Black tree along with object skipping technique is used for acceleration purpose. For further acceleration, surface ignoring is also introduced here. The obtained results confirm that the proposed method is faster than the existing ray tracing methods because Red-Black tree along with object skipping and surface ignoring performed the searching operation within a logarithmic time $O(\log_2(N-N'))$ instead of linear searching time $O(N)$. Conversely, the surface extraction technique together with an accurate calculation of direction of reflected, refracted, and diffracted rays provides the exact propagation paths of the transmitted waves. Therefore, the proposed method offers better accuracy in comparison with SBR and BDPT ray tracing methods. The proposed method provides a reliable indoor ray tracing tool, which can be used for predicting electromagnetic waves (at any frequency range) in multiple rooms of the single storied building.

Received: December 28, 2012. Accepted: March 10, 2013.

Acknowledgments: Our sincere thanks go to the University of Malaya for offering extensive financial support under the University of Malaya Research Grant (UMRG) scheme.

References

[1] Hashemi, H., Lee, D., Ehman, D. (1992). Statistical modeling of the indoor radio propagation channel ii. In *IEEE 42nd Vehicular Technology Conference*, **2**, 839–843.

[2] Zhong, J., Bin-Hong, L., Hao-Xing, W., Hsing-Yi, C., Sarkar, T.K. (2001). Efficient ray-tracing methods for propagation prediction for indoor wireless communications. *IEEE Antennas and Propagation Magazine*, **43**(2), 41–49.

[3] Frank, L. (2000). Prediction of in building multipath radio channels for future radio services. *Frequenz*, **54**(7–8), 165–170.

[4] Jacob, M., Piesiewicz, R., Kurner, T. (2008). Propagation modeling and system analysis for future multi gigabit THz communication. *Frequenz*, **62**(5–6), 132–136.

[5] Liu, Z.Y., Guo, L.X. (2011). A quasi three-dimensional ray tracing method based on the virtual source tree in urban microcellular environments. *Progress In Electromagnetics Research*, **118**, 397–414.

[6] Pechac, P. (2002). Electromagnetic wave propagation modelling using the ant colony optimization algorithm. *Radioengineering*, **11**(3).

[7] Tiberi, G., Bertini, S., Malik, W.Q., Monorchio, A., Edwards, D.J., Manara, G. (2009). Analysis of realistic ultrawideband indoor communication channels by using an efficient ray-tracing based method. *IEEE Antennas and Propagation Magazine*, **57**(3), 777–785.

[8] Tao, Y.B., Lin, H., Bao, H.J. (2008). Kd-tree based fast ray tracing for rcs prediction. *Progress In Electromagnetics Research*, **81**, 329–341.

[9] Mohtashami, V., Shishegar, A.A. (2010). Efficient shooting and bouncing ray tracing using decomposition of wavefronts. *IET Microwaves, Antennas and Propagation*, **4**(10), 1567–1574.

[10] Ding, J., Chen, R.-S., Zhou, H., Fan, Z.H. (2011). An improvement for the acceleration technique based on monostatic bistatic equivalence for shooting and bouncing ray method. *Microwave and Optical Technology Letters*, **53**(5), 1178–1183.

[11] Dersch, U., Zollinger, E. (1994). Propagation mechanism in microcell and indoor environments. *IEEE Transactions on Vehicular Technology*, **43**(4), 1058–1066.

[12] Iskander, M.F., Yun, Z. (2002). Propagation prediction models for wireless communication systems. *IEEE Transactions on Microwave Theory and Techniques*, **50**(3), 662–673.

[13] Cocheril, Y., Vauzelle, R. (2007). A new ray-tracing based wave propagation model including rough surfaces scattering. *Progress In Electromagnetics Research*, **75**, 357–381.

[14] Maurer, J., Fügen, T., Knörzer, S., Wiesbeck, W. (2006). A ray-optical approach to model the inter-vehicle transmission channel. *Frequenz,* **60**(5–6), 95–98.

[15] Reza, A.W., Dimyati, K., Noordin, K.A., Kausar, A.S.M.Z., Sarker, M.S. (2012). A comprehensive study of optimization algorithm

for wireless coverage in indoor area. *Optimization Letters*, doi:10.1007/s11590-012-0543-z

[16] Sarker, M.S., Reza, A.W., Dimyati, K. (2011). A novel ray-tracing technique for indoor radio signal prediction. *J. of Electromagn. Waves and Appl.*, **25**, 1179–1190.

[17] Reza, A.W., Sarker, M.S., Dimyati, K. (2010). A novel integrated mathematical approach of ray-tracing and genetic algorithm for optimizing indoor wireless coverage. *Progress In Electromagnetics Research,* **110**, 147–162.

[18] Reza, A.W., Dimyati, K., Noordin, K.A., Sarker, M.S. (2011). Intelligent ray-tracing: an efficient indoor ray propagation model. *IEICE Electronics Express*, **8**(22).

[19] Wood, D. (1993). *Data Structures, Algorithms, and Performance*. Addison-Wesley Publishing Co.

[20] Thomas, H., Cormen, C., Ronald, E.L., Stein, L.R.C. (2001). *Introduction to Algorithms, second edition*. The MIT Press, Cambridge, Massachusetts and London, England.

[21] Wiener, R. (2005). Generic red-black tree and its C# implementation. *Journal of Object Technology*, **4**, 59–80.

[22] Gatland, I.R. (2002). Thin lens ray tracing. *American J. Phys.*, **70**(21), 1184.

[23] Smith, C.J. (1963). *A Degree Physics, Part III, Optics*, pp. 125–133. Edward Arnold Publishers, London.

[24] Tsingos, N., Funkhouser, T., Ngan, A., Carlbom, I. (2001). Modelling acoustics in virtual environments using the uniform theory of diffraction. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 545–552.

[25] Pena, D., Feick, R., Hristov, H.D., Grote, W. (2003). Measurement and modeling of propagation losses in brick and concrete walls for the 900-MHz band. *IEEE Transactions on Antennas and Propagation*, **51**(1), 31–39.

[26] Mohtashami, V., Shishegar, A.A. (2010). Accuracy and computational efficiency improvement of ray tracing using line search theory. *IET Microwaves, Antennas & Propagation*, **4**(9), 1290–1299.