AN ACCURATE AND EFFICIENT SHOOTING-AND-BOUNCING-POLYGON RAY TRACER FOR RADIO PROPAGATION MODELLING

TEH CHIN HUI

DOCTOR OF PHILOSOPHY IN ENGINEERING

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN
APRIL 2019

AN ACCURATE AND EFFICIENT SHOOTING-AND-BOUNCING-POLYGON RAY TRACER FOR RADIO PROPAGATION MODELLING

By

TEH CHIN HUI

A Doctor of Philosophy in Engineering thesis submitted to
the Department of Electrical and Electronic Engineering,
Lee Kong Chian Faculty of Engineering and Science,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Engineering
April 2019

ABSTRACT

AN ACCURATE AND EFFICIENT SHOOTING-AND-BOUNCING-POLYGON RAY TRACER FOR RADIO PROPAGATION MODELLING

Teh Chin Hui

Radio propagation modelling is important for the design and deployment of wireless communication systems. Ray tracing is the state-of-the-art site-specific technique to modelling terrestrial and indoor propagation. In this project, we have proposed and implemented three important improvements to a 3D shooting-and-bouncing-polygon (SBP) ray tracer.

Firstly, we have derived delay correction factors to be used with a one-patch model of real building wall with thickness. The correction factors have been derived for reflection, transmission, and diffraction, based on a multilayer lossy wall model. The one-patch model and correction factors allow efficient and accurate treatment of real building walls with thickness in ray tracing.

Secondly, we have extended SBP to trace diffracted and diffracted-reflected ray beams using edge-fixed diffraction ray-polygons. Crucial to the solution is the ability to transform or project Cartesian polygons to edge-fixed polygons. We have described the associated problems and proposed an algorithm to perform the transformation or projection.

Thirdly, we have proposed a new spatial partitioning scheme to be used with the SBP ray tracer. The new scheme is named *convex cell partitioning* (CCP). It divides the simulated scene into a set of interconnected convex cells of any shapes. The interconnectivity between the convex cells is represented by a graph. Like binary-space-partitioning, CCP is able to sort objects by their visibility distance relative to any point, efficiently, incrementally, and adaptively. Better than binary-space-partitioning, in the context of SBP, CCP removes the need to perform polygon subtraction, a relatively expensive procedure. Traversal of the CCP graph is also simpler than the binary tree. To improve CCP usability, we have developed and implemented a tool to automatically construct the CCP graph from a given set of 3D walls.

The improved ray tracer is named SBP-CCP. We have evaluated its performance by making comparisons with commercial ray tracers, full-wave solutions, and published measurements. Results show that SBP-CCP is more accurate than the earlier version and it is about an order of magnitude faster than the binary-space-partitioning version in an urban canyon application. It is also shown that SBP-CCP outperforms commercial REMCOM Wireless InSite 3D ray tracers in terms of accuracy and time and memory efficiency in long tunnel, urban canyon, and indoor applications. In particular, SBP-CCP has very small memory footprint, 2 to 3 orders of magnitude smaller than REMCOM ray tracers.

SBP-CCP simulation results also show very good match (2 dB root-mean-square error) to full-wave solutions computed using CST Microwave

Studio time domain solver, in an indoor application with reflection, transmission, diffraction, and lossy walls with thickness. SBP-CCP is full 3D, fast, accurate, and memory efficient. It is a good candidate for simulating long tunnel, urban canyon, and indoor propagation environments. The current implementation does not include double diffraction and it is not suitable for urban environments with multiple over-roof-top diffractions. The current implementation is also not suitable for applications where diffuse scattering is an important propagation mechanism.

ACKNOWLEDGEMENT

First and foremost, I wish to thank my project supervisor, Prof. Chung Boon Kuan, and co-supervisor, Associate Prof. Lim Eng Hock. They have been very helpful and supportive. I wish to thank Universiti Tunku Abdul Rahman (UTAR) for the CST suite, high-end workstations, and relatively low tuition fees. I wish to thank Multimedia University (MMU) for the subscription of IEEE Xplore. I wish to thank Lee Kong Chian Faculty of Engineering and Science, UTAR and the Faculty of Engineering, MMU, for their understanding. It has been quite a long journey. I wish to thank Mr. Lo Yew Chiong for being the driver, in our trips to meet the supervisors. I wish to thank Mr. Phua Yeong Nam for helping me run CST simulations. I wish to thank the anonymous reviewers for their valuable comments and suggestions. I wish to thank REMCOM for the one-month full-featured free trial. Last but not least, I wish to thank my parents, my wife and children for their understanding. My sincere gratitude to all who have contributed directly or indirectly to the successful execution of this thesis. THANK YOU.

APPROVAL SHEET

This thesis entitled "**AN ACCURATE AND EFFICIENT SHOOTING-AND-BOUNCING-POLYGON RAY TRACER FOR RADIO PROPAGATION MODELLING"** was prepared by TEH CHIN HUI and submitted as partial fulfillment of the requirements for the degree of Doctor of Philosophy in Engineering at Universiti Tunku Abdul Rahman.

Approved by:

_____

(Prof. Dr. Chung Boon Kuan) Date:……..
Professor/Supervisor
Department of Electrical and Electronic Engineering
Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

_____

(Prof. Dr. Lim Eng Hock) Date:……..
Professor/Co-supervisor
Department of Electrical and Electronic Engineering
Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 24 April 2019

SUBMISSION OF DOCTOR OF PHILOSOPHY
IN ENGINEERING THESIS

It is hereby certified that Teh Chin Hui (ID No: 1007406) has completed this thesis entitled "An Accurate and Efficient Shooting-and-Bouncing-Polygon Ray Tracer for Radio Propagation Modelling" under the supervision of Prof. Dr. Chung Boon Kuan (Supervisor) from the Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, and Prof. Dr. Lim Eng Hock (Co-Supervisor) from the Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science.

I understand that University will upload softcopy of my thesis in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

_____

Teh Chin Hui

DECLARATION

I hereby declare that the dissertation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

Name Teh Chin Hui

Date 24 April 2019

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	Two dimensional
3D	Three dimensional
BSP	Binary space partitioning
CCP	Convex cell partitioning
GO	Geometrical optics
GPU	Graphics processing unit
PEC	Perfect electric conductor
rms	Root-mean-square
SBP	Shooting-and-bouncing-polygon
SBR	Shooting-and-bouncing-ray
UTD	Uniform geometrical theory of diffraction

CHAPTER 1

INTRODUCTION

1.1 Overview

The last 30 years have seen tremendous growth in wireless communication systems, from world-wide-area-network to body-area-network. The mobile phone access technology has developed from the first generation to the present fifth generation. Smart phone, Wi-Fi, Bluetooth, and soon IoT (Internet of Things) have become indispensable parts of our modern daily life. They have been made possible by the successful deployment of various radio frequency wireless communication systems.

The wireless systems operate by transmitting radio frequency signals into space. It is hoped that some of the signals, after interacting with the surrounding environment, will eventually arrive and be picked up by the intended receiver. The space over which the signals propagate, from the transmitter to the receiver, is commonly known as the communication channel or propagation channel. The propagation channel is not at all friendly to the signals. It will cause attenuation, delay, and distortion to the signals. Successful deployment of the wireless systems requires a good knowledge about the channel parameters, i.e. the nature and the amount of attenuation, delay, and distortion caused by the propagation channel. For example, if the attenuation is

known, various counter measures can be put in place, e.g. increase transmit power, increase transmitter height, increase coding gain, use directional antenna, or use lower order modulation scheme, to ensure that the signals can be picked up and decoded by the receiver. Other examples include determining availability, coverage, optimum base station locations, and achievable data rates.

Direct measurement of the channel parameters is costly and time consuming. More often, the channel parameters are estimated using a mathematical model of the propagation channel, known as propagation model. We shall limit our discussions to terrestrial and indoor propagation models. In general, there are 2 types of propagation models, namely empirical and theoretical. Empirical models are derived from extensive measurements. A very popular one is Okumura-Hata model (Okumura et al., 1968; Hata, 1980). A vast variety of empirical outdoor models are extensions or variants of Okumura-Hata model. The estimation is done based on general description of the propagation environment e.g. rural, sub-urban or urban. The formula is simple and fast to compute, but it is not able to deal with variation in the environment not accounted in the measurement studies.

Theoretical models are typically based on geometrical optics and geometrical theory of diffraction, where radio waves are treated as rays. Propagation of radio waves is modelled by direct ray, reflected ray, transmitted ray, and diffracted ray. Theoretical models are more computationally intensive but they are less dependent on measurements. They have evolved from earlier a few rays models (Longley and Rice, 1968; Edwards and Durkin, 1969; Walfisch

and Bertoni, 1988) to the state-of-the-art ray tracing models (Fuschini et al, 2015; Yun and Iskander, 2015; Leonor et al, 2019). Ray tracing models are also known as site-specific models for their ability to cope with details specific to the modelled environment. They are universal in the sense that they are able to estimate most of the channel parameters, narrowband or wideband. Central to the ray tracing models is the ray tracing engine or ray tracer. There are two-dimensional (2D) and three-dimensional (3D) ray tracers. 2D ray tracers are usually more efficient but they are also more restrictive, e.g. they are not suitable for indoor simulations. 3D ray tracers are computationally intensive. They are known for their flexibility and they have a wider range of applications (Egea-Lopes et al., 2019; Guan et al., 2019; Chen, Lai and Yu, 2019). The main objective of this project is to develop, implement, and validate an improved 3D ray tracer which is accurate and efficient, for radio propagation modelling.

1.2 Ray Tracing for Radio Propagation Modelling

The use of ray tracing for mobile radio propagation modelling has been studied extensively since early 1990s (Tam and Tran, 1995; Iskander and Yun, 2002; Sarkar et al., 2003; Bertoni, Torrico and Liang, 2005; Hrovat, Kandus and Javornik, 2014; Fuschini et al., 2015; Yun and Iskander, 2015). Ray tracing propagation models can be divided briefly into two types: image-based (Mckown and Hamilton, 1991) and shooting-and-bouncing-ray (SBR) (Seidel and Rappaport, 1994). In image-based ray tracing, an image is a point residing on the opposite side of a reflecting plane and having the same perpendicular

distance to the plane, relative to a source point which can be a transmitter or another image. Diffraction related images have the form of a line instead of a point. With the aid of the images, exact ray paths can be computed by tracing backward from the receivers towards the transmitters. Not all the ray paths are valid though. Various forms of illumination zones have been used to reduce invalid ray paths and improve computation efficiency (Tan and Tan, 1996; Fortune, 1996; Catedra et al., 1998; Son and Myung, 1999; Athanasiadou, Nix and McGeehan, 2000; Suzuki and Mohan, 2003; Teh and Chuah, 2003; Degli-Esposti et al., 2004; Tan, Su and Long, 2015).

In SBR ray tracing, many rays are shot from a transmitter in all directions. For each ray, the nearest ray-object intersection is computed. At each intersection, two new rays are spawned: a reflected ray and a transmitted ray. In the case of diffraction, a set of rays on the Keller's cone are spawned. Each ray is also checked for whether it arrives at the receivers, a process known as *reception test*. In the following discussions, a ray is denoted as *standard-ray* when only one ray is used to represent a ray cone. Three-dimensional (3D) reception test for a standard-ray is usually done by using a *reception sphere*, a sphere centred at the receiver and with diameter approximately equals to the ray separation. A standard-ray is said to have arrived at a receiver if it intersects the reception sphere.

The resultant ray path is not exact because it does not pass through the receiver. The error is a crucial one if ray contributions are summed coherently. The reception sphere method also assumes constant angular separation between

adjacent rays. The assumption is not met in practice and a reception sphere can be too big or too small (Novak, 2019). This leads to the well-known double counting and under counting errors. A popular solution to the double and under counting problem is to use ray-tubes instead of standard-rays (Chen and Jeng, 1997; Yang, Wu and Ko, 1998). A ray-tube is a pyramidal structure formed by 3 or more rays. A ray-tube is said to have arrived at a receiver if the receiver is inside the ray-tube. This greatly reduces double and under counting errors. However, it comes at a cost. Three or more rays are traced per ray-tube compared to one in the case of standard-ray. Usually, for computation efficiency, a non-exact ray path representative of the ray-tube, e.g. an average or median, is used to compute the ray contribution.

In both cases, namely the standard-ray and ray-tube, the ray separation has to be kept small enough so as not to miss an object and to reduce ray path error. Many rays will need to be traced. For example, at 0.25° ray separation (a default value used by a commercial suite like REMCOM Wireless InSite), there are about 600,000 rays to cover all directions in 3D space. Various methods of dynamic ray splitting have been used to improve computation efficiency and accuracy (Rajkumar et al., 1996; Suzuki and Mohan, 1997; Bernardi, Cicchetti and Testa, 2004). An alternative is to use ray-beams (Fortune, 1996; Catedra et al., 1998; Son and Myung, 1999; Athanasiadou, Nix and McGeehan, 2000; Suzuki and Mohan, 2003; Teh and Chuah, 2003; Degli-Esposti et al., 2004; Tan, Su and Long, 2015). A ray-beam is similar to a ray-tube except that it can be made bigger than the objects. Usually, a polygon-clipping algorithm is used to determine beam-object intersection. Most if not all ray-beam tracing

propagation models use the image method to compute exact ray path. Hence, they are generally regarded as image-based ray tracing models. Ideally, only one ray-beam needs to be shot from a transmitter, compared to hundreds of thousands of standard-rays or ray-tubes. The one ray-beam is adaptively split into smaller ray-beams as it intersects with objects. Tracing a ray-beam is equivalent to tracing an infinite number of rays with infinitesimally small separations. No doubt a beam-object intersection is much more computationally expensive than a ray-object intersection. In the case that a small but finite ray separation is adequate, ray-beams are more efficient when the beam sizes are big, such that the beam-object intersections are cheaper than the large amount of ray-object intersections, but gradually lose the advantage as the beam sizes become smaller.

On the other hand, ray-beam tracing is more complicated and less flexible. It has a hard time dealing with 3D diffraction and curved surface. Although 3D diffraction has been taken into account in various ways, 3D diffracted ray-beams have not been fully traced. The angular z-buffer method (Catedra et al., 1998) uses an edge-fixed coordinate system to track diffracted ray-beams, but only the bounding rectangle is tracked. Another method is described by Di Giampaolo and Bardati (2009), but they do not describe how diffracted-reflected (diffraction followed by reflection) ray-beams are handled. Tan, Su and Long (2015) have adopted Di Giampaolo and Bardati (2009) method and they do not trace diffracted-reflected ray-beam. Bernardi, Cicchetti and Testa (2004) use ray-tubes. Degli-Esposti et al. (2004) and Suzuki and Mohan (2003) do not trace diffracted ray-beam.

In an earlier work (Teh and Chuah, 2003), we have presented an image-based ray-beam tracer. It shoots and bounces polygons using perspective projection. Beam-object intersection is handled by polygon clipping. In the following discussions, it will be referred to as shooting-and-bouncing-polygon (SBP) ray tracer. Methods similar to SBP are described by Maurer (2000), Giampaolo and Bardati (2009) and Tao, Lin and Bao (2011). Unlike Catedra et al. (1998) and Degli-Esposti et al. (2004), SBP does not use spherical coordinates. The problem with spherical coordinates is that straight lines in Cartesian coordinates are not straight but they appear as curves in the θ - ϕ plane. Catedra et al. (1998) deal with the problem by tracking the bounding rectangle only. Saeidi and Hodjatkashani (2010) shows that the bounding rectangle defined by spherical coordinates of wall vertices is not accurate. Degli-Esposti et al. (2004) deal with the problem by adding more vertices to piecewise approximate the curves. Unlike Catedra et al. (1998), SBP does not use expensive image dependent data structures and it is capable of efficiently handling a high-order of multiple reflections and transmissions. SBP traces exact reflection and transmission ray-beams or illumination zones. There is no invalid ray path in this case as all ray paths computed are valid. The SBP ray tracer works very well with reflection and transmission but it too does not trace diffracted ray-beam and it is not capable of tracing diffracted-reflected rays.

Various spatial partitioning schemes have been used, mainly to accelerate ray-object intersection computation. Most of them are adopted from computer graphics literature. The key idea is to limit the intersection computation to objects that reside in the neighbourhood of the ray. Yun, Iskander

and Zhang (2000) have used a uniform rectangular grid. Yun, Zhang and Iskander (2002) have used a triangular grid. Catedra et al. (1998) have used angular sectoring. Kenny and Nuallain (2017) have used convex space partitioning. Many others have used binary space partitioning or kd-tree (Chen and Jeng, 1997; Teh and Chuah, 2003; Mohtashami and Shishegar, 2012; Tan, Su and Long, 2015). kd-tree is a special type of binary space partitioning where the dividing planes are axis-aligned. The selection of suitable partitioning scheme depends on the particular implementation of the ray tracer. For example, angular sectoring is a natural choice for angular z-buffer (Catedra et al., 1998); binary space partitioning is a natural choice for shooting-and-bouncing-polygon for its ability to sort objects relative to any point (Teh and Chuah, 2003).

A real wall has thickness. For accurate simulation of indoor environments, the two surfaces of a real wall are to be treated differently (Yang, Wu and Ko, 1998; Kenny and Nuallain, 2017). This effectively doubles the total number of walls which in turn increases the computation complexity. In an earlier work (Teh, Kung and Chuah, 2006), we have proposed a one-patch solution to the problem. A real wall with thickness is modelled as an infinitesimally thin patch in ray tracing. Delay correction factors are used to compensate the position errors of the wall surfaces. They are based on Burnside and Burgener (1983) formulation for one-layer lossless dielectric slab. Correction factor for diffraction has not been reported.

1.3 Project Scope

In this project, we will describe three important improvements to the shooting-and-bouncing-polygon (SBP) ray tracer. Firstly, we have extended the delay correction factors for walls with thickness to a more general multilayer lossy wall model. We have also derived the delay correction factor for diffraction. This improves the ray tracer's accuracy and flexibility when dealing with lossy walls with thickness. Secondly, we have extended SBP to trace diffracted and diffracted-reflected ray-beams. This improves the SBP ray tracer's accuracy when diffracted-reflected rays play an important role. Thirdly, we have proposed a better alternative to binary space partitioning for use with SBP, which greatly improves SBP computation efficiency. The resultant SBP ray tracer is full-3D, fast, accurate, and memory efficient. We will show its applications to long tunnel, urban canyon, and indoor environments. We will show its accuracy by comparing its simulation results against full-wave solutions and published measurements. We will compare its run time, accuracy, and memory consumption against commercial ray tracers from REMCOM Wireless InSite.

The current implementation of the SBP ray tracer does not handle double diffraction. Although double and higher order diffractions are important in some applications, e.g. over-roof-top diffraction, they are not required in many applications (Chen and Jeng, 1997; Catedra et al., 1998; Bernardi, Cicchetti and Testa, 2004; Remcom, 2018b). The project also does not address the problems of curved surface (Didascalou et al., 2000; Wang and Yang, 2006),

rough surface (Degli-Esposti et al., 2004; Cocheril and Vauzelle, 2007), and GPU (graphic processing unit) acceleration (Schmitz et al., 2011; Tan, Su and Long, 2015).

Based on some of the works in this project, we have published a paper entitled "An accurate and efficient 3D shooting-and-bouncing-polygon ray tracer for radio propagation modeling" in IEEE Transactions on Antennas and Propagation (in press). We appreciate valuable comments from the reviewers and we have incorporated many of their suggestions into this project.

1.4 Report Outline

CHAPTER 1 explains the project objective and scope. It also gives a general review of previous works. More specific reviews of previous works are given throughout the report as and when the needs arise. CHAPTER 2 reviews background information essential to the project execution and discussions. CHAPTER 3 to CHAPTER 5 detail the proposed improvements to the shooting-and-bouncing-polygon (SBP) ray tracer. CHAPTER 3 explains the derivation of the delay correction factors based on a multilayer lossy wall model, for reflection, transmission, and diffraction. CHAPTER 4 explains a solution to the 3D diffracted ray-beam problem. CHAPTER 5 explains a new spatial partitioning scheme, which we term *convex space partitioning*, for the SBP ray tracer. CHAPTER 6 discusses the performance of the improved SBP ray tracer through comparisons with commercial ray tracers, full-wave solutions, and

published measurements. CHAPTER 7 gives a summary of the project achievements and areas of future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

A general review of previous works has been given in CHAPTER 1. More specific reviews of previous works are given throughout the report as and when the needs arise. This Chapter reviews important background information required for the project execution, e.g. the implementation of the shooting-and-bouncing-polygon (SBP) ray tracer. This Chapter will serve as a reference for many of the subsequent discussions. Although the essence of the contents in this Chapter are not new, we have customized them for the project, made corrections, provided details to fill the gaps, and prepared original graphical illustrations.

2.2 Shooting-and-Bouncing-Polygon (SBP)

This Section describes the shooting-and-bouncing-polygon method presented by Teh and Chuah (2003). Flat polygonal objects will be denoted as *walls*. A simulated scene is described by a set of walls. Teh and Chuah (2003) have introduced *ray-polygon* that can be shot and traced. Ray-polygon is the cross sectional footprint of a polyhedral ray-beam on a view plane, as shown in Figure 2.1. Its shape on a view plane can be determined by perspective

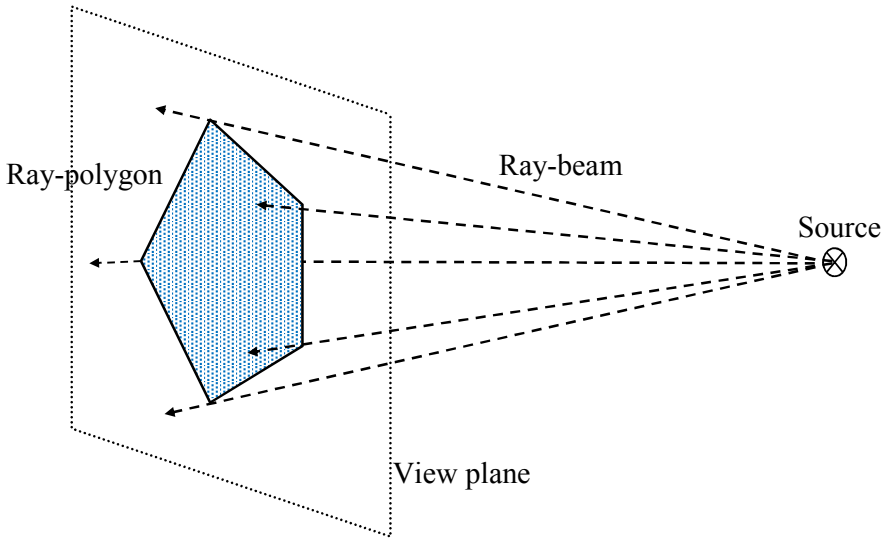


Figure 2.1 Ray-polygon

projection. The center of projection (COP) is the source of the ray-beam. Only 4 ray-polygons need to be shot from a transmitter to cover all directions in the 3D space. They form a tetrahedron enclosing the transmitter. They can be shot and traced independently on different threads or machines.

Figure 2.2 and Figure 2.3 show a 2D illustration of the shooting-and-bouncing-polygon process. Without loss of generality, the ray-polygons and walls are perpendicular to the paper, and hence they appear as straight lines. After a ray-polygon is shot from the transmitter, reception tests are performed on receivers residing between the transmitter and the nearest wall, by projecting the receivers onto the ray-polygon. Receivers whose projections lie in the ray-polygon are said to have received the ray (see Figure 2.2(a)). Then, the nearest wall is projected onto the ray-polygon (see Figure 2.2(b)). The portion obstructed by the wall is removed from the ray-polygon using a generic polygon

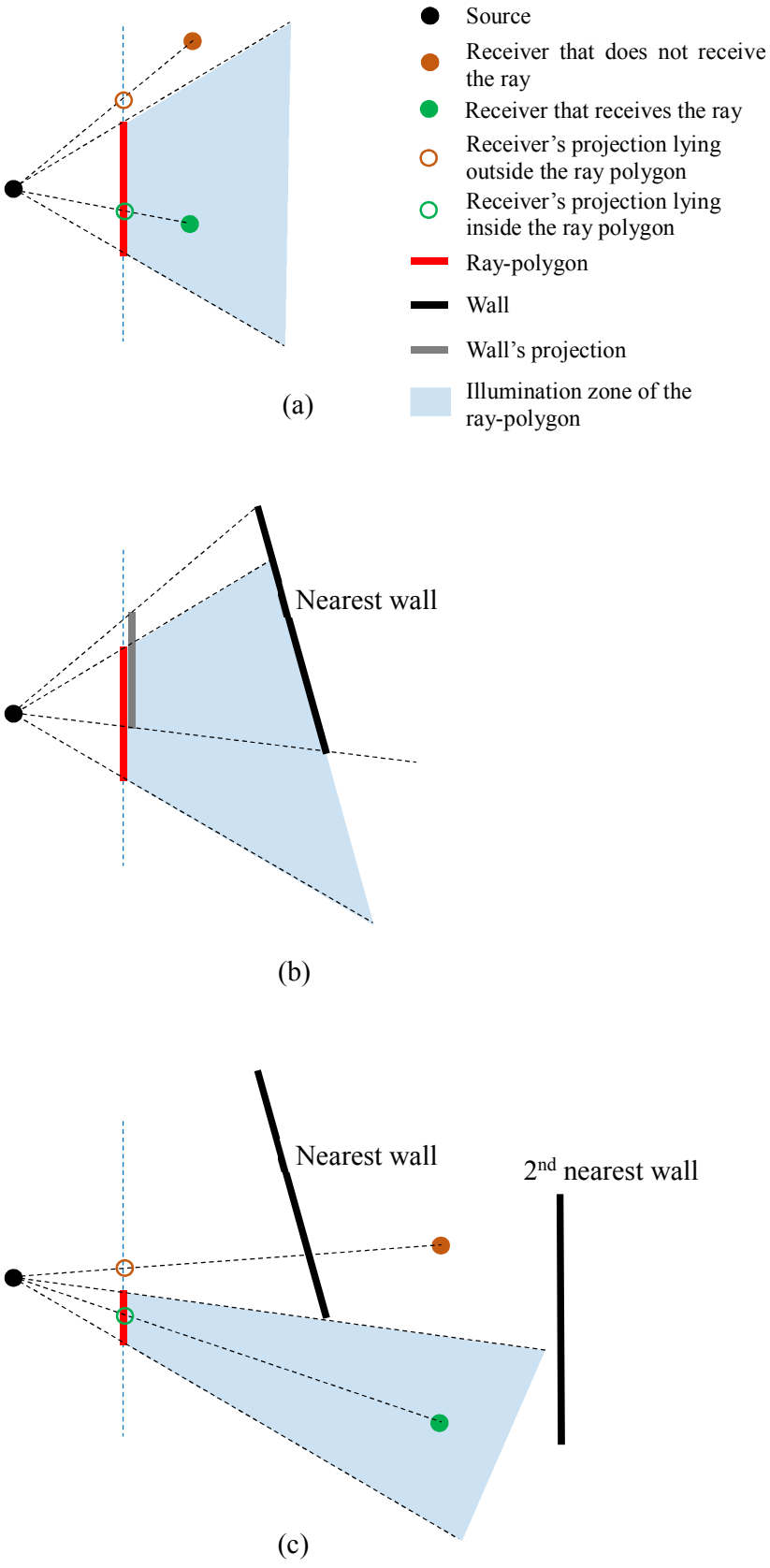
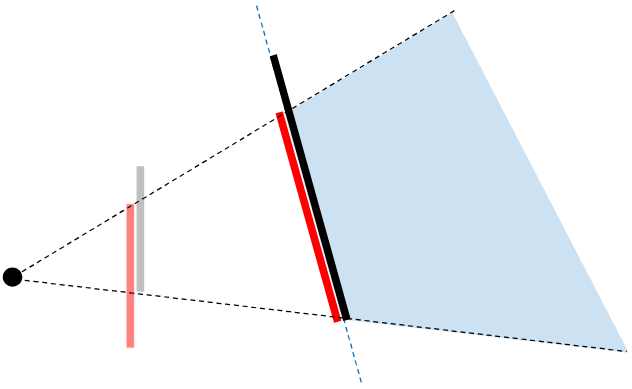
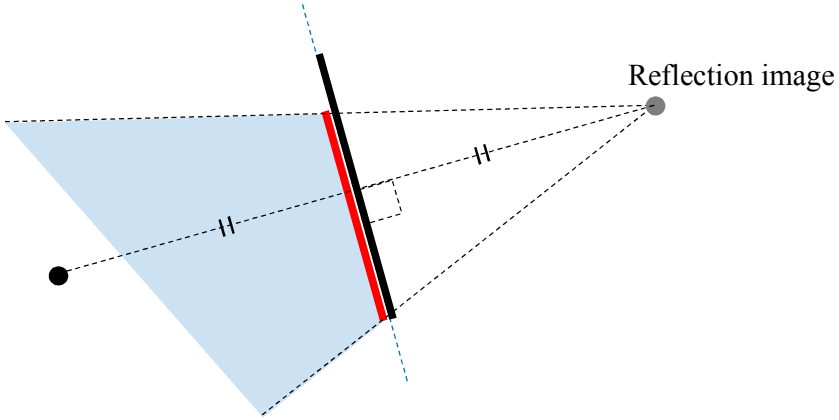


Figure 2.2 Shooting and clipping ray-polygon



(a) Transmitted ray-polygon



(b) Reflected ray-polygon

Figure 2.3 Transmitted and reflected ray-polygons

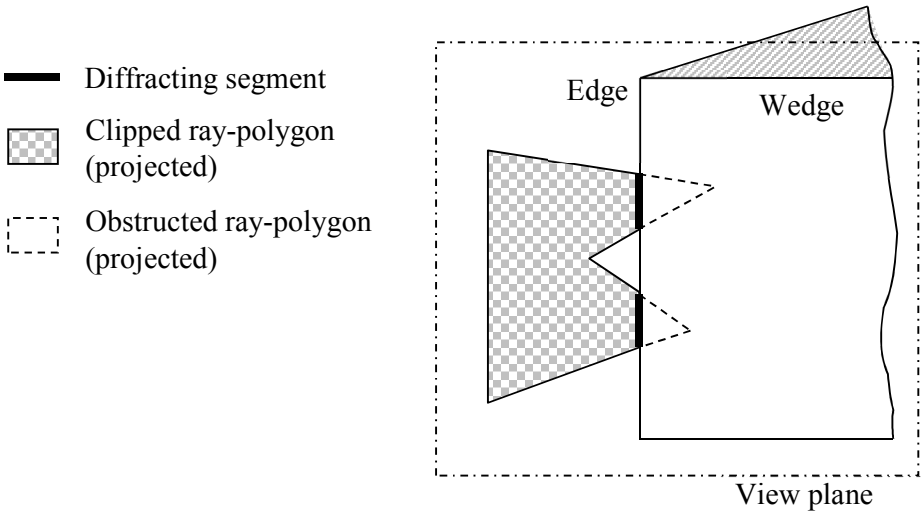


Figure 2.4 Clipped ray-polygon and diffracting edge

clipping algorithm (Vatti, 1992; Murta, 2017). The clipped ray-polygon is then used for reception tests on receivers residing between the current wall and the next nearest wall and so on (see Figure 2.2(c)). The process repeats until the ray-polygon is empty (totally obstructed) or it exits the simulated scene.

The obstructed portion of the ray-polygon defines both the reflected ray-polygon and transmitted ray-polygon (see Figure 2.3). For reflection, the COP is the reflection image of the transmitter. For transmission, the COP is the transmitter or it can be shifted to simulate refraction. These ray-polygons are traced in the same way as described above. A diffracting edge is said to have given rise to diffraction if it coincides with one of the edges of the clipped ray-polygon (projected), as shown in Figure 2.4. Teh and Chuah (2003) have not extended the ray-polygon concept to diffraction due to the difficulty in projecting diffracted ray-polygon. A non-ray-polygon method is used to handle diffraction but it is not able to trace diffracted-reflected rays. A summary of the SBP processes is given in Figure 2.5.

Upon reception, backward tracing is performed to reveal the exact ray paths. The backward tracing only requires N line-plane intersections for N interactions (reflection / transmission) because the specific walls giving rise to the interactions are known and it is also known that the ray path is not shadowed by other walls. Ray contributions are computed using geometrical optics (GO) and uniform geometrical theory of diffraction (McNamara, Pictorius and Malherbe, 1990; Bernadi, Cicchetti and Testa (2002); see Sections 2.5 and 2.6).

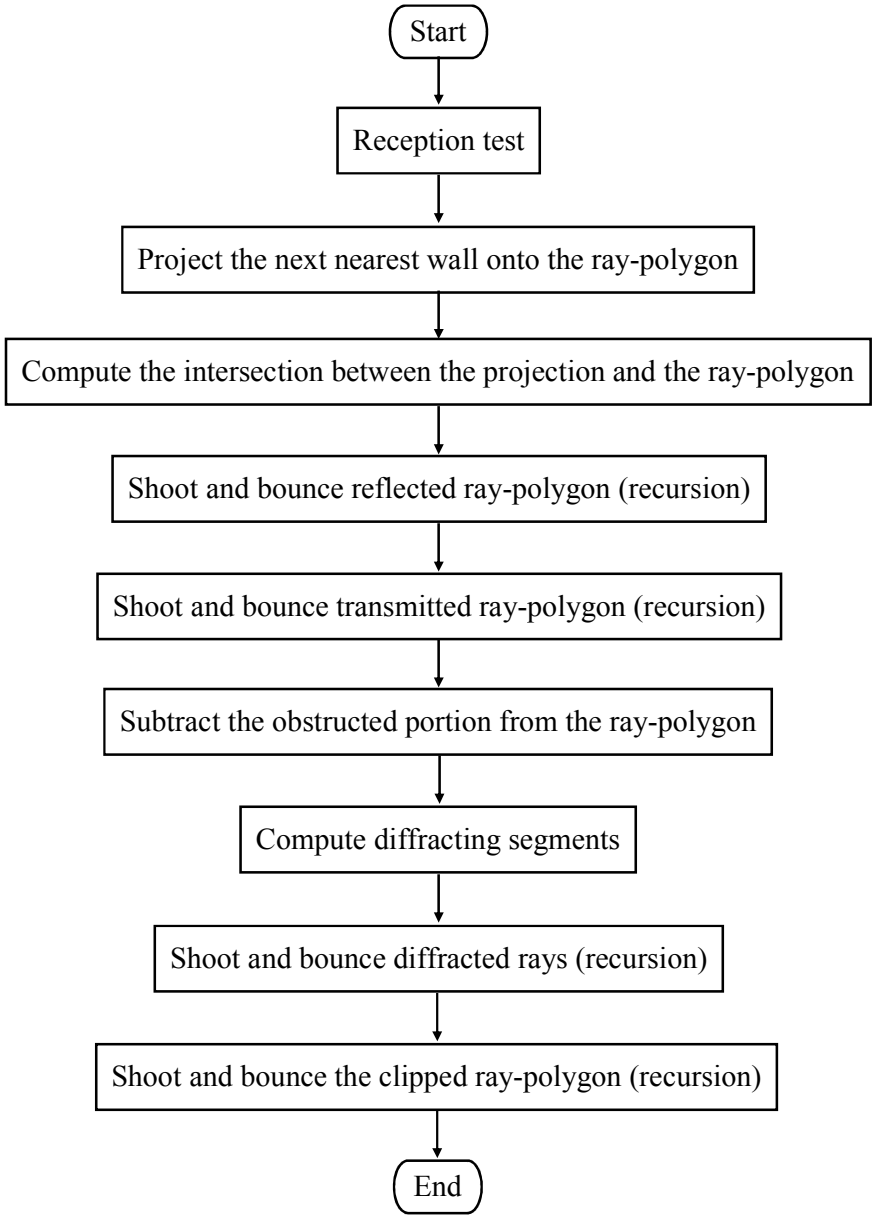


Figure 2.5 Shooting-and-bouncing-polygon flowchart

For narrowband simulations, multiple rays arriving at a receiver are summed coherently.

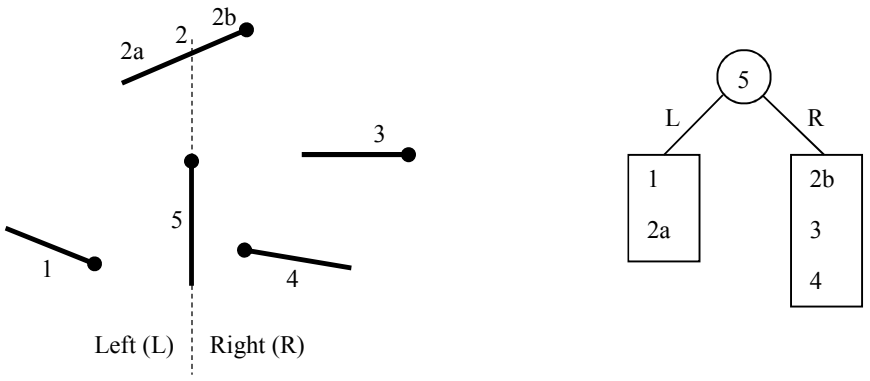
Obviously, the above requires that the walls and receivers are sorted by their distances relative to the COP. This is achieved efficiently using binary-space-partitioning (Foley et al., 1996). The sort can be done incrementally, adaptively, and with little effort by traversing a binary tree in a specific way (see Section 2.3). The construction of the binary tree is a one-time job for a simulated scene. The binary tree is independent of the COP.

The SBP algorithm allows ray contributions to be computed alongside generation of images (COP of ray-polygons). Images may be discarded as their contributions have been accounted for. Use of image or visibility tree (Athanasiadou, Nix and McGeehan, 2000; Degli-Esposti et al., 2004) is optional. An image tree can be stored and reused for future simulations, and it saves the cost of generating images. However, an image tree can grow huge for complex environments, greatly increasing the memory requirement and often leading to slow execution due to high memory usage. For simpler environments, with SBP, the cost of generating images is often negligible compared to ray path and field computations. Hence, in most cases, an image tree offers little advantage to SBP.

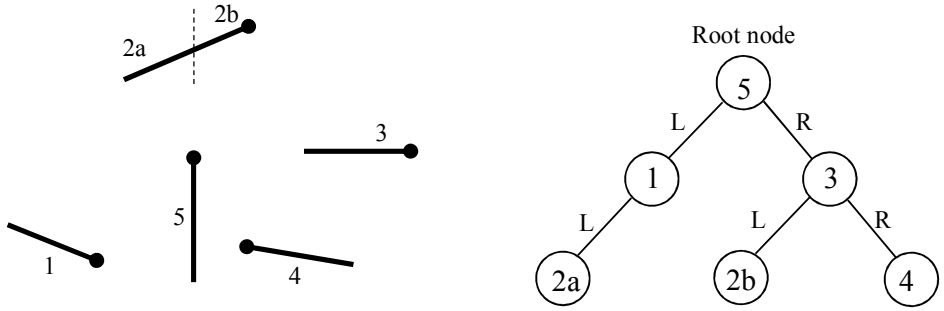
2.3 Binary Space Partitioning (BSP)

Figure 2.6 shows a 2D example that will be used in the following discussions. The scene is shown on the left, the binary tree on the right. Without loss of generality, the walls are perpendicular to the paper and so they appear as straight lines. The orientations of the walls are indicated by the small solid circles attached to one end of the lines. The left and right sides of the lines (walls) are defined with the solid circles pointing upward. In Teh and Chuah (2003), BSP begins by partitioning the simulated scene with a plane which coincides with one of the walls. It then classifies the walls according to their positions relative to the dividing plane, i.e. on the left or on the right. Those coincide with the dividing plane form a BSP node while those crossing the plane are split into two, one on the left and one on the right (see Figure 2.6(a)). The partitioning process is repeated recursively on the left and the right set of walls, generating a binary tree (see Figure 2.6(b)).

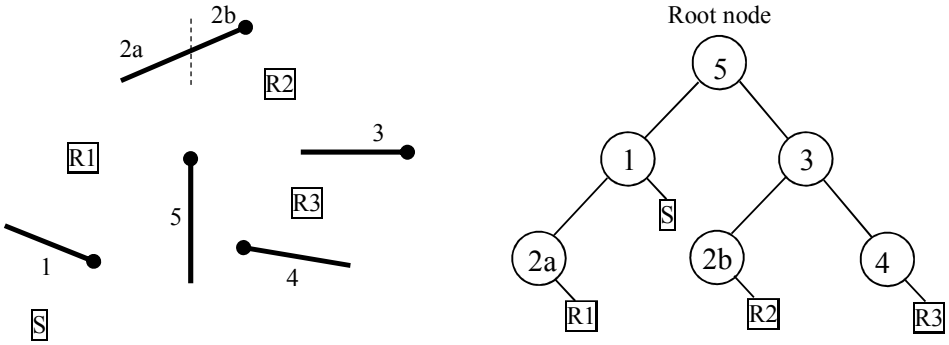
Sources and receivers are pushed onto the binary tree in a similar way, based on their positions relative to the BSP nodes. They always settle in an empty branch not having a BSP node. Take Figure 2.6(c) for example. Relative to source S, the nearest wall is the wall forming the parent node of S, i.e. node or wall 1. Receivers, if any, closer to S than the nearest wall are those settled at the same branch as S. To determine the next wall, S is pushed through the opposite branch of node 1. S is on the right of wall 2a and it will settle at the right branch of node 2a together with receiver R1. Hence, R1 is closer to S than the second wall, i.e. R1 lies in between the first and second wall relative to S.



(a) First division



(b) Complete binary tree



(c) Binary tree with transmitter and receiver

Figure 2.6 Binary-space-partitioning

After both branches of node 1 is considered, the next (third) wall is at the parent node of node 1, i.e. node or wall 5. The same algorithm is used to determine the sort order at the right branch of node 5. S is pushed through the branch and settle on the left of node 4. So, the fourth wall is wall 4 followed by R3 on the right of node 4. Returning to node 3 (the parent node of node 4), the fifth wall is wall 3. Then, S is pushed through the left branch of node 3 and settle on the right of node 2b together with R2. So, R2 comes before the sixth wall, wall 2b. The sort result is wall 1 – R1 – wall 2a – wall 5 – wall 4 – R3 – wall 3 – R2 – wall 2b. Similar discussions can be found on Wikipedia (2018) except that they are in computer graphics terms.

In Teh and Chuah (2003), other than sorting, BSP is also exploited to simplify the search for possible ray-wall interactions. Take Figure 2.6(c) for example. For first order reflection off wall 1, reflected rays only exist on the same side as the source S, no reflected ray on the other side of wall 1. So, the left branch of node 1, i.e. wall 2b and R1, need not be considered for second order interaction or reception test. Arriving at node 5, the shooting-and-bouncing-polygon process requires the projection of the reflected ray-polygon onto node 5 (its dividing plane). From the scene configuration shown in Figure 2.6, it is obvious that the reflected ray-polygon has no valid projection on node 5. This implies that the reflected ray-polygon originated from the left of node 5 will not reach node 5. This further implies that the reflected ray-polygon will not reach the right side of node 5. Hence, the right branch of node 5, i.e. wall 2b, wall 3, wall 4, R2 and R3, also need not be considered for second order

interaction or reception test. In this example, only one ray-polygon projection is required to arrive at the conclusion that all the walls will not give rise to second order reflection after the first reflection from wall 1, and the first reflection from wall 1 will not reach the receivers. It is important to note that the sorting done by traversing the binary tree is not only incremental but it is also adaptive.

2.4 Channel Parameters

In this Section, we describe the definition and computation of some common channel parameters. The electric field intensity of a ray originated from a transmitter and ended at an observation point after going through multiple reflections, multiple transmissions, and single diffraction from planar objects is given by (McNamara, Pictorius and Malherbe, 1990)

$$\mathbf{E}_r = \frac{\hat{\mathbf{e}}_t(\theta_t, \phi_t) E_0 e^{-jk(s'+s)} \sqrt{g_t(\theta_t, \phi_t)}}{\sqrt{ss'(s+s')}} \cdot \prod_{i=1}^{N} \mathbf{C}_i \qquad (2.1)$$

where

$\hat{\mathbf{e}}_t(\theta_t, \phi_t)$ is the unit vector of the transmitted electric field in the transmit direction,

E_0 is the electric field intensity at 1 m from the transmit antenna in the direction of maximum radiation,

$g_t(\theta_t, \phi_t)$ is the normalized transmit antenna gain in the transmit direction,

$\mathbf{C}_i$ is the dyadic reflection, transmission or diffraction coefficient of *i* th interaction,

N is the total number of interactions,

s and s' are the path lengths before and after diffraction, respectively,

θ_t and ϕ_t are the spherical vertical and horizontal angles of the transmit direction, respectively.

The coherently summed narrowband power received at a receiver is given by

$$P_r = P_t G_t G_r \left(\frac{\lambda}{4\pi}\right)^2 \left| \sum_{j=1}^{n} \frac{\mathbf{E}_{rj}}{E_0} \cdot \hat{\mathbf{e}}_{rj}(\theta_{rj}, \phi_{rj}) \sqrt{g_r(\theta_{rj}, \phi_{rj})} \right|^2 \qquad (2.2)$$

where

P_t is the transmit power,

G_t and G_r are the peak gains of the transmitting and receiving antennas, respectively,

λ is the signal wavelength,

$\mathbf{E}_{rj}$ is the electric field intensity, as given in Equation (2.1), due to j th ray,

n is the total number of rays received by the receiver,

$\hat{\mathbf{e}}_r(\theta_r, \phi_r)$ is the unit vector of the transmitted electric field in the receive direction when the receiving antenna is in the transmit mode,

$g_t(\theta_r, \phi_r)$ is the normalized receive antenna gain in the receive direction,

θ_r and ϕ_r are the spherical vertical and horizontal angles of the receive direction, respectively.

Often, path loss is used in place of received power, which is defined by

$$PL = 10 \log_{10}\left(\frac{P_t}{P_r}\right) \qquad (2.3)$$

The power delay profile received at a receiver is given by

$$PDP(t) = P_t G_t G_r \left(\frac{\lambda}{4\pi}\right)^2 \cdot \left| \sum_{j=1}^{n} \left(\frac{\boldsymbol{E}_{rj}}{E_0} \cdot \hat{\mathbf{e}}_{rj}(\theta_{rj}, \phi_{rj}) \sqrt{g_r(\theta_{rj}, \phi_{rj})} \right) h_{BPF}(t - t_{dj}) \right|^2 \quad (2.4)$$

where

$h_{BPF}(t)$ is the impulse response of the receive bandpass filter,

t_{dj} is the propagation delay of j th ray.

Root-mean-square (rms) delay spread is generally regarded to have correlation with inter-symbol interference (Chuang, 1987). It is the variance of power delay profile, i.e.

$$\tau_{rms} = \int (t - \tau_m)^2 PDP(t)\, dt \quad (2.5)$$

where τ_m is the mean delay given by

$$\tau_m = \int t \cdot PDP(t)\, dt \quad (2.6)$$

2.5 Reflection and Transmission Coefficients

Reflection or transmission coefficient is the ratio of the reflected or transmitted field to the incident field, respectively. The expressions of the coefficients depend on the definitions of the vectors used in the dyadic tensor. In this Section, to be consistent with Balanis (1989) and other textbooks, the dyadic reflection and transmission coefficients are defined as follows:

$$\mathbf{R} = \hat{\mathbf{e}}_{si}\hat{\mathbf{e}}_{sr}R_s + \hat{\mathbf{e}}_{hi}\hat{\mathbf{e}}_{hr}R_h \quad (2.7)$$

$$\mathbf{T} = \hat{\mathbf{e}}_{si}\hat{\mathbf{e}}_{st}T_s + \hat{\mathbf{e}}_{hi}\hat{\mathbf{e}}_{ht}T_h \quad (2.8)$$

$$\hat{\mathbf{e}}_{si} = \hat{\mathbf{e}}_{sr} = \hat{\mathbf{e}}_{st} = \hat{\mathbf{s}}' \times \hat{\mathbf{n}} \tag{2.9}$$

$$\hat{\mathbf{e}}_{hi} = \hat{\mathbf{e}}_{si} \times \hat{\mathbf{s}}' \tag{2.10}$$

$$\hat{\mathbf{e}}_{hr} = \hat{\mathbf{s}}_r \times \hat{\mathbf{e}}_{sr} \tag{2.11}$$

$$\hat{\mathbf{e}}_{ht} = \hat{\mathbf{e}}_{st} \times \hat{\mathbf{s}}_t \tag{2.12}$$

where

R_s and R_h are soft (perpendicular) and hard (parallel) reflection coefficients, respectively,

T_s and T_h are soft (perpendicular) and hard (parallel) transmission coefficients, respectively,

$\hat{\mathbf{s}}'$, $\hat{\mathbf{s}}_r$ and $\hat{\mathbf{s}}_t$ are the unit vectors of the incident, reflected, and transmitted directions, respectively,

$\hat{\mathbf{n}}$ is the unit normal of the reflecting surface.

Figure 2.7 shows a graphical illustration of the vectors. For a plane interface, the reflection and transmission coefficients are given by Fresnel reflection and transmission coefficients (Balanis, 1989). For non-magnetic materials, the Fresnel coefficients are

$$R_s' = \frac{\cos\theta_i - \sqrt{\frac{\varepsilon_2}{\varepsilon_1} - \sin^2\theta_i}}{\cos\theta_i + \sqrt{\frac{\varepsilon_2}{\varepsilon_1} - \sin^2\theta_i}} \tag{2.13}$$

$$R_h' = \frac{-\frac{\varepsilon_2}{\varepsilon_1}\cos\theta_i + \sqrt{\frac{\varepsilon_2}{\varepsilon_1} - \sin^2\theta_i}}{\frac{\varepsilon_2}{\varepsilon_1}\cos\theta_i + \sqrt{\frac{\varepsilon_2}{\varepsilon_1} - \sin^2\theta_i}} \tag{2.14}$$

$$T_s' = \frac{2\cos\theta_i}{\cos\theta_i + \sqrt{\frac{\varepsilon_2}{\varepsilon_1} - \sin^2\theta_i}} \tag{2.15}$$

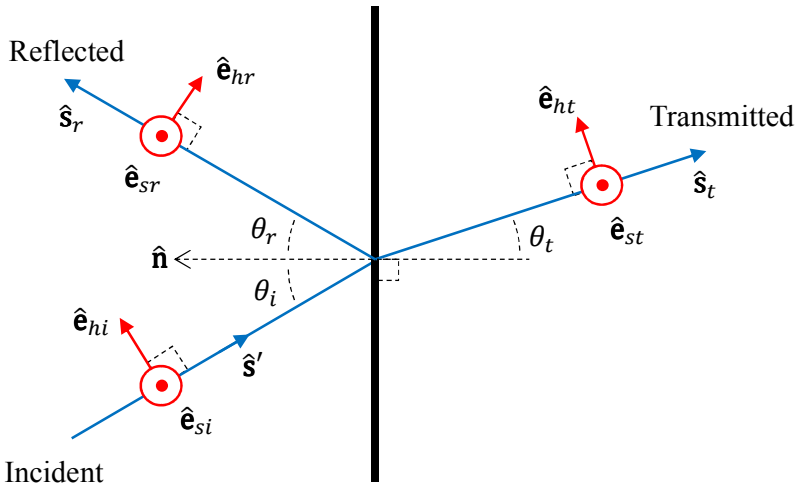


Figure 2.7 Vectors in dyadic reflection and transmission coefficients

$$T_h' = \frac{2\sqrt{\frac{\varepsilon_2}{\varepsilon_1}}\cos\theta_i}{\frac{\varepsilon_2}{\varepsilon_1}\cos\theta_i + \sqrt{\frac{\varepsilon_2}{\varepsilon_1} - \sin^2\theta_i}} \tag{2.16}$$

where

ε_1 and ε_2 are the electric permittivity of the first and second medium, respectively,

θ_i is the incident angle measured from the interface normal, as shown in Figure 2.7.

Equations (2.13) to (2.16) are valid for lossy materials with complex valued permittivity and angle (Orfanidis, 2016). A method to compute reflection and transmission coefficients of a thin lossless dielectric slab (wall) is described by Burnside and Burgener (1983). The coefficients have been conditioned to work with ray tracing, i.e. they can be used directly with Equation (2.1). They are

$$R_{s,h} = \frac{R_{s,h}'(1 - P_d^2 P_a)}{1 - R_{s,h}' P_d^2 P_a} \tag{2.17}$$

$$T_{s,h} = \frac{\left(1 - {R'_{s,h}}^2\right) P_d P_t}{1 - {R'_{s,h}}^2 P_d^2 P_a} \tag{2.18}$$

where

$$P_a = e^{j2k_0 d \sin\theta_i \tan\theta_t} \tag{2.19}$$

$$P_d = e^{-jk\, d \sec\theta_t} \tag{2.20}$$

$$P_t = e^{jk_0 d \cos(\theta_i - \theta_t) \sec\theta_t} \tag{2.21}$$

$R'_{s,h}$ are Fresnel reflection coefficients,

k_0, k are the wave numbers in air and the slab, respectively,

d is the slab thickness.

A more general approach to deal with a lossy slab or multilayer wall is to use a multilayer wall model. We have used the multilayer wall reflection and transmission coefficients given in Balanis (1989). Other equivalent coefficients can be found in other textbooks e.g. Kong (1985), Ishimaru (1991), and Orfanidis (2016). The transmission coefficients, however, have not been conditioned to be used directly with Equation (2.1). The required correction is explained in Section 3.2. The coefficients as given in Balanis (1989) are: (there were typo errors which we have corrected below)

$$R_s = \frac{B_0}{A_0} \tag{2.22}$$

$$R_h = \frac{D_0}{C_0} \tag{2.23}$$

$$T_s = \frac{1}{A_0} \tag{2.24}$$

$$T_h = \frac{1}{C_0} \tag{2.25}$$

A_0, B_0, C_0 and D_0 are computed using the recursive formulas:

$$A_{M+1} = C_{M+1} = 1 \tag{2.26}$$

$$B_{M+1} = D_{M+1} = 0 \tag{2.27}$$

$$A_p = \frac{e^{\psi_p}}{2}\left[A_{p+1}\left(1 + Y_{p+1}\right) + B_{p+1}\left(1 - Y_{p+1}\right)\right] \tag{2.28}$$

$$B_p = \frac{e^{-\psi_p}}{2}\left[A_{p+1}\left(1 - Y_{p+1}\right) + B_{p+1}\left(1 + Y_{p+1}\right)\right] \tag{2.29}$$

$$C_p = \frac{e^{\psi_p}}{2}\left[C_{p+1}\left(1 + Z_{p+1}\right) + D_{p+1}\left(1 - Z_{p+1}\right)\right] \tag{2.30}$$

$$D_p = \frac{e^{-\psi_p}}{2}\left[C_{p+1}\left(1 - Z_{p+1}\right) + D_{p+1}\left(1 + Z_{p+1}\right)\right] \tag{2.31}$$

where

$$Y_{p+1} = \frac{\cos\theta_{p+1}}{\cos\theta_p}\sqrt{\frac{\omega\varepsilon_p - j\sigma_p}{\omega\varepsilon_{p+1} - j\sigma_{p+1}}} \tag{2.32}$$

$$Z_{p+1} = \frac{\cos\theta_p}{\cos\theta_{p+1}}\sqrt{\frac{\omega\varepsilon_p - j\sigma_p}{\omega\varepsilon_{p+1} - j\sigma_{p+1}}} \tag{2.33}$$

$$\psi_p = d_p\gamma_p\cos\theta_p \tag{2.34}$$

$$\gamma_p = \sqrt{-\omega\mu_p\left(\omega\varepsilon_p - j\sigma_p\right)} \tag{2.35}$$

$$\cos\theta_p = \sqrt{1 - \left(\frac{\gamma_0}{\gamma_p}\right)^2 \sin^2\theta_0} \tag{2.36}$$

θ_0 is the incident angle.

ω is the angular frequency.

M is the total number of layers.

$d_p, \mu_p, \varepsilon_p,$ and σ_p are the thickness, permeability, permittivity, and conductivity of the p th layer, respectively.

2.6 Diffraction Coefficient

Diffraction coefficient is the ratio of the diffracted field to the incident field. A common way to compute diffraction coefficient is given by Kouyoumjian and Pathak (1974) who have introduced uniform geometrical theory of diffraction (UTD). Figure 2.8 shows the edge-fixed coordinate system and vectors used in the diffraction calculations. They require a priori knowledge about the diffraction point. The diffraction point (DP) can be computed from known locations of the source ($\overrightarrow{OS}$), the receiver ($\overrightarrow{OR}$), and the diffracting edge ($\overrightarrow{OE}$ and $\hat{\mathbf{e}}$), using the law of edge diffraction or generalized Fermat's principle (Keller, 1962), i.e. $\beta = \beta'$, as shown in Figure 2.9. More specifically:

$$h_s = \overrightarrow{ES} \cdot \hat{\mathbf{e}} \tag{2.37}$$

$$h_r = \overrightarrow{ER} \cdot \hat{\mathbf{e}} \tag{2.38}$$

$$\mathbf{s}_t' = -\overrightarrow{ES} + h_s\hat{\mathbf{e}} \tag{2.39}$$

$$\mathbf{s}_t = \overrightarrow{ER} - h_r\hat{\mathbf{e}} \tag{2.40}$$

$$h_d = h_r - \frac{(h_r - h_s)d_r}{d_s + d_r} = \frac{h_r d_s + h_s d_r}{d_s + d_r} \tag{2.41}$$

$$\overrightarrow{OD} = \overrightarrow{OE} + h_d\hat{\mathbf{e}} \tag{2.42}$$

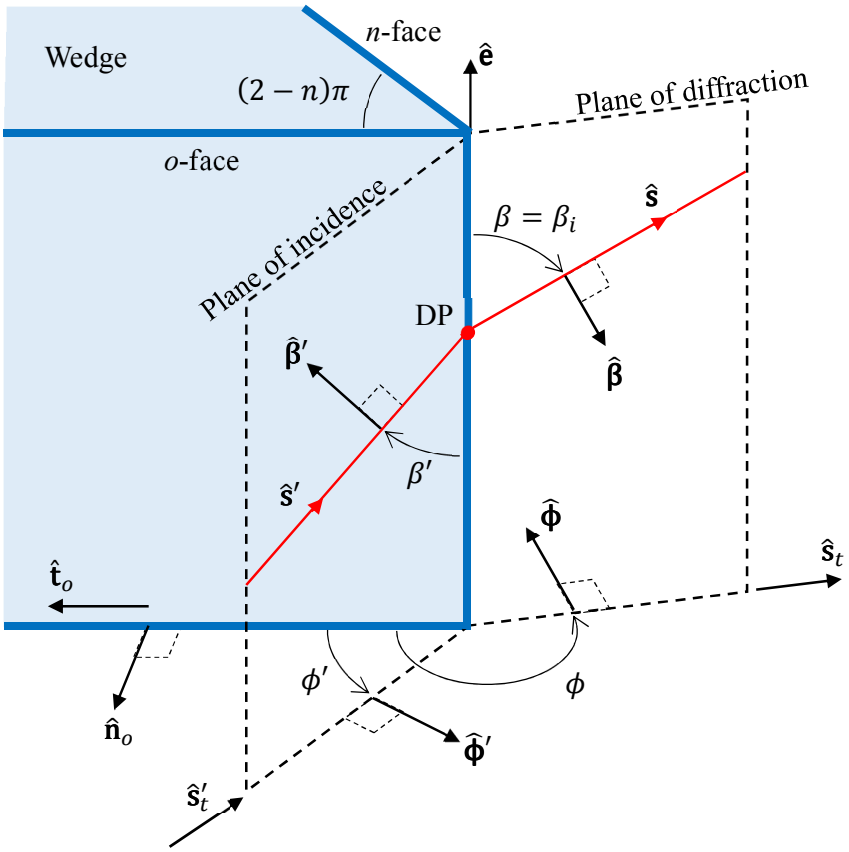


Figure 2.8 Angles and vectors for diffraction calculations

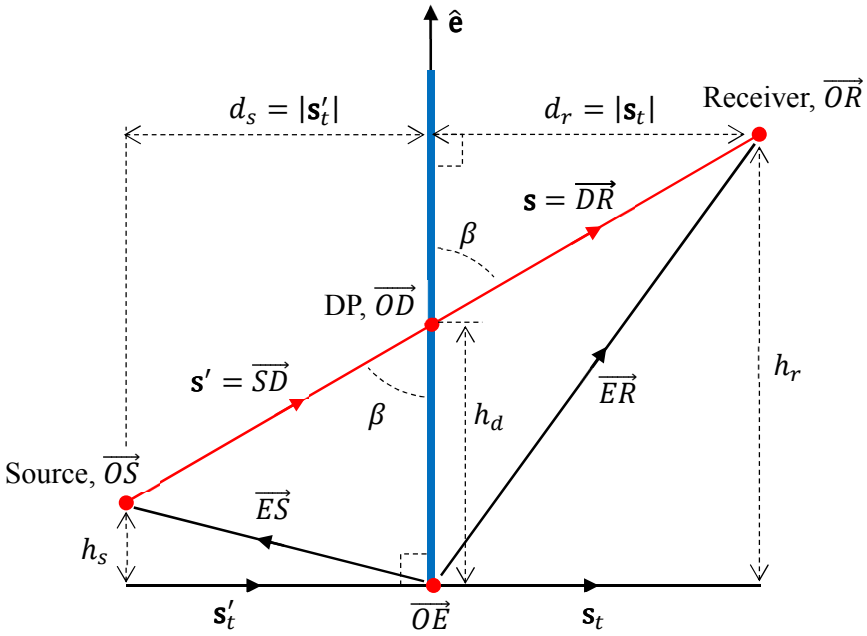


Figure 2.9 Diffraction point

The dyadic diffraction coefficient is (McNamara, Pictorius and Malherbe, 1990):

$$\mathbf{D} = -\hat{\boldsymbol{\beta}}'\hat{\boldsymbol{\beta}}\, D_s - \hat{\boldsymbol{\phi}}'\boldsymbol{\phi}\, D_h \tag{2.43}$$

where

$$\hat{\mathbf{e}} = \hat{\mathbf{t}}_o \times \hat{\mathbf{n}}_o \tag{2.44}$$

$$\hat{\boldsymbol{\phi}}' = \frac{\hat{\mathbf{s}}' \times \hat{\mathbf{e}}}{|\hat{\mathbf{s}}' \times \hat{\mathbf{e}}|} \tag{2.45}$$

$$\hat{\boldsymbol{\phi}} = -\frac{\hat{\mathbf{s}} \times \hat{\mathbf{e}}}{|\hat{\mathbf{s}} \times \hat{\mathbf{e}}|} \tag{2.46}$$

$$\hat{\boldsymbol{\beta}}' = \hat{\boldsymbol{\phi}}' \times \hat{\mathbf{s}}' \tag{2.47}$$

$$\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\phi}} \times \hat{\mathbf{s}} \tag{2.48}$$

$$\phi' = \begin{cases} \arccos(-\hat{\mathbf{s}}_t' \cdot \hat{\mathbf{t}}_o), & -\hat{\mathbf{s}}_t' \cdot \hat{\mathbf{n}}_o \geq 0 \\ 2\pi - \arccos(-\hat{\mathbf{s}}_t' \cdot \hat{\mathbf{t}}_o), & -\hat{\mathbf{s}}_t' \cdot \hat{\mathbf{n}}_o < 0 \end{cases} \tag{2.49}$$

$$\phi = \begin{cases} \arccos(\hat{\mathbf{s}}_t \cdot \hat{\mathbf{t}}_o), & \hat{\mathbf{s}}_t \cdot \hat{\mathbf{n}}_o \geq 0 \\ 2\pi - \arccos(\hat{\mathbf{s}}_t \cdot \hat{\mathbf{t}}_o), & \hat{\mathbf{s}}_t \cdot \hat{\mathbf{n}}_o < 0 \end{cases} \tag{2.50}$$

$$\beta = \beta' = \arccos(\hat{\mathbf{s}}' \cdot \hat{\mathbf{e}}) \tag{2.51}$$

$$D_{s,h} = D_1 + D_2 \mp (D_3 + D_4) \tag{2.52}$$

$$D_0 = \frac{-e^{-j\pi/4}}{2n\sqrt{2\pi k}\sin\beta} \tag{2.53}$$

$$D_1 = D_0 \cot\left[\frac{\pi + (\phi - \phi')}{2n}\right] F\left[kL^i a(\phi - \phi')\right] \tag{2.54}$$

$$D_2 = D_0 \cot\left[\frac{\pi - (\phi - \phi')}{2n}\right] F\left[kL^i a(\phi - \phi')\right] \tag{2.55}$$

$$D_3 = D_0 \cot\left[\frac{\pi + (\phi + \phi')}{2n}\right] F[kL^{rn} a^+(\phi + \phi')] \tag{2.56}$$

$$D_4 = D_0 \cot\left[\frac{\pi - (\phi + \phi')}{2n}\right] F[kL^{ro} a(\phi + \phi')] \tag{2.57}$$

$$a(\phi \pm \phi') = 2\cos^2\left(\frac{\phi \pm \phi'}{2}\right) \tag{2.58}$$

$$a^+(\phi+\phi') = 2\cos^2\left(\frac{2n\pi-(\phi+\phi')}{2}\right) \tag{2.59}$$

$$L^i = L^{ro} = L^{rn} = L = \begin{cases} \dfrac{ss'}{s+s'}\sin^2\beta, & \text{point source} \\ \dfrac{ss'}{s+s'}, & \text{line source} \\ s\sin^2\beta, & \text{plane wave} \end{cases} \tag{2.60}$$

$$F(x) = 2j\sqrt{x}e^{jx}\int_{\sqrt{x}}^{\infty} e^{-ju^2}du \tag{2.61}$$

$$F(x) \approx \begin{cases} F^*(|x|), & x < 0 \\ \left(\sqrt{\pi x} - 2x\,e^{j\pi/4} - \dfrac{2x^2e^{-j\pi/4}}{3}\right)e^{j(x+\pi/4)}, & 0 \le x < 0.3 \\ 1 + j\dfrac{1}{2x} - \dfrac{3}{4x^2} - j\dfrac{15}{8x^3} + \dfrac{75}{16x^4}, & x > 5.5 \\ \text{Lookup table}, & 0.3 \le x \le 5.5 \end{cases} \tag{2.62}$$

The UTD diffraction coefficient is said to be valid only when the largeness parameter $\kappa = kL\sin^2\beta > 1$ ($\kappa > 3$ if $n \approx 1$). k is the wave number for the medium outside the wedge. Equation (2.60) is valid only for straight wedges with flat faces. More general treatment of curved edge and curved surface can be found in Kouyoumjian and Pathak (1974) and McNamara, Pictorius and Malherbe (1990). The lookup table in Equation (2.62) can be constructed from the transition function plot given in Kouyoumjian and Pathak (1974). Equation (2.52) is valid only for perfect electric conductors (PEC). Its rigorous extension to non PEC edge is yet to be found. There are rigorous solutions for non PEC edge based on Sommerfeld-Maliuzhinets integrals and surface impedance boundary condition approximation (Volakis, 2013), but they are complicated and difficult to use.

A heuristic solution for 3D diffraction from a thin lossless dielectric slab is given by Burnside and Burgener (1983). The idea is to modify Equation (2.52) to ensure that the total field is continuous at the shadow boundaries using reflection and transmission coefficients computed for the dielectric slab, e.g.

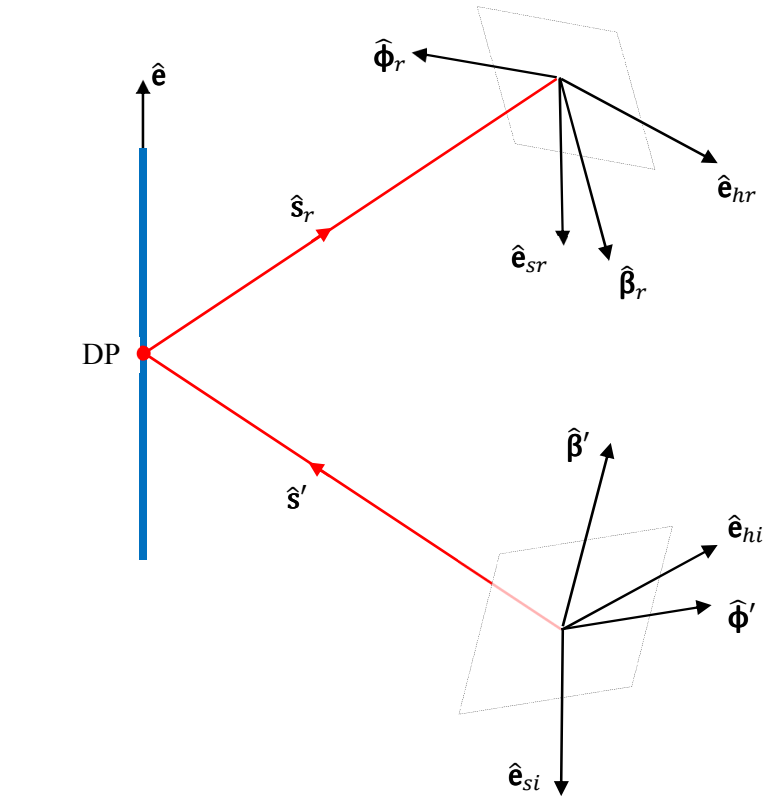
$$D_{s,h} = \left(1 - T_{s,h}\right)(D_1 + D_2) + R_{s,h}(D_3 + D_4) \tag{2.63}$$

which is valid for 2D or normal incidence ($\beta = \frac{\pi}{2}$). A shadow boundary is a boundary at which there is a discontinuity in the incident, reflected or transmitted field due to the loss of the incident, reflected or transmitted ray across the boundary. The problem becomes more complicated with 3D oblique incidence because the soft and hard polarizations for reflection and transmission differ from those for diffraction, as shown in Figure 2.10. Uniform plane wave is assumed and hence the polarization vectors are on the transverse plane perpendicular to the ray directions. To be consistent with Burnside and Burgener (1983) formulation, the hard polarization of reflected wave has been redefined as

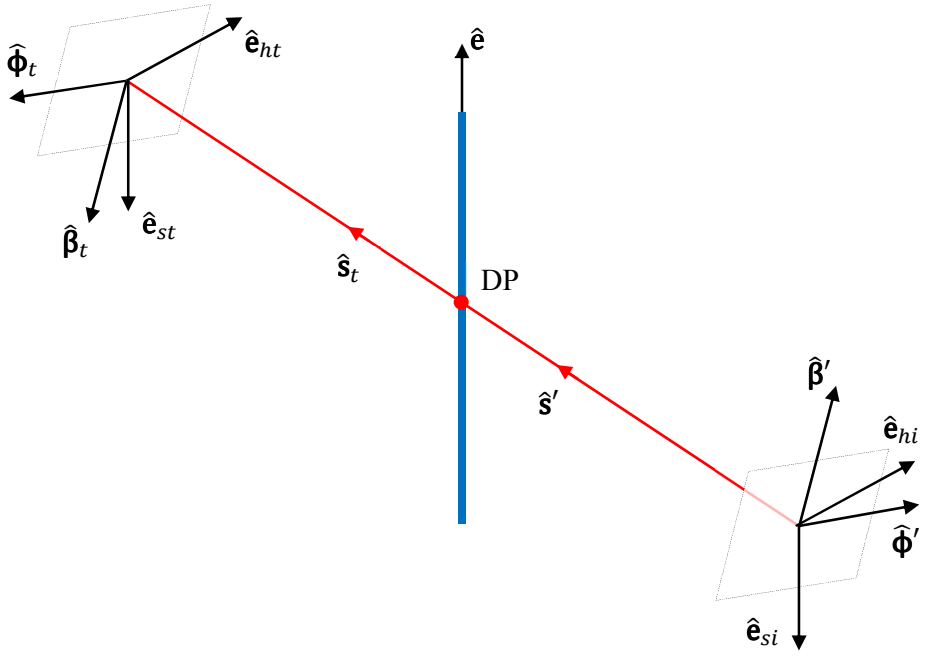
$$\hat{\mathbf{e}}_{hr} = \hat{\mathbf{e}}_{sr} \times \hat{\mathbf{s}}_r \tag{2.64}$$

The effect is reflection coefficients for hard polarization is the negative of those given in Section 2.5.

Figure 2.11 redraws the polarization vectors in 2D such that the transverse planes coincide with the paper and the ray directions are pointing outward. The vectors have been rotated such that the hard polarizations for ray-fixed planes of incident and reflection are pointing upward. Figure 2.11 is similar to "Fig. 6" in Burnside and Burgener (1983). We believe there are errors in the definitions of ray-fixed polarizations given in Burnside and

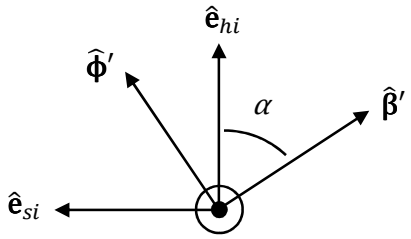


(a) Reflection shadow boundary

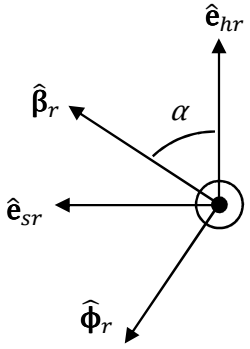


(b) Incident shadow boundary

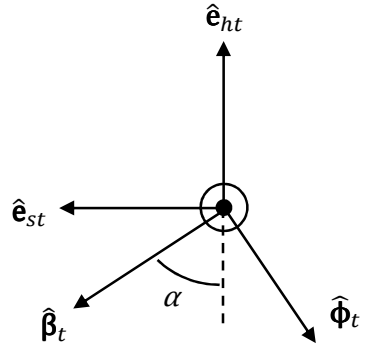
Figure 2.10 Ray-fixed and edge-fixed basis vectors (3D)



(a) Incident ray



(b) Reflected ray



(c) Transmitted ray

Figure 2.11 Ray-fixed and edge-fixed basis vectors (2D)

Burgener (1983) because they are not consistent with "Fig. 6", from which the formulas are derived. At different incident angles, the angle α changes but $\hat{\mathbf{e}}_s$ and $\hat{\boldsymbol{\phi}}$ are always leading $\hat{\mathbf{e}}_h$ and $\hat{\boldsymbol{\beta}}$ by 90° (in counterclockwise direction), respectively. It can be written that:

$$\begin{bmatrix} E_{\beta i} \\ E_{\phi i} \end{bmatrix} = \mathbf{T}(-\alpha) \begin{bmatrix} E_{hi} \\ E_{si} \end{bmatrix} \tag{2.65}$$

$$\begin{bmatrix} E_{\beta r} \\ E_{\phi r} \end{bmatrix} = \mathbf{T}(\alpha) \begin{bmatrix} E_{hr} \\ E_{sr} \end{bmatrix} \tag{2.66}$$

$$\begin{bmatrix} E_{\beta t} \\ E_{\phi t} \end{bmatrix} = -\mathbf{T}(-\alpha) \begin{bmatrix} E_{ht} \\ E_{st} \end{bmatrix} \tag{2.67}$$

where

$$\mathbf{T}(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \tag{2.68}$$

$$\mathbf{T}^{-1}(\alpha) = \mathbf{T}(-\alpha) \tag{2.69}$$

$$\sin\alpha = -\hat{\mathbf{e}}_{hi} \cdot \hat{\boldsymbol{\phi}}' \tag{2.70}$$

$$\cos\alpha = \hat{\mathbf{e}}_{si} \cdot \hat{\boldsymbol{\phi}}' \tag{2.71}$$

$E_{\beta i}$, $E_{\phi i}$, E_{hi} and E_{si} are the incident electric field components in $\hat{\boldsymbol{\beta}}'$, $\hat{\boldsymbol{\phi}}'$, $\hat{\mathbf{e}}_{hi}$ and $\hat{\mathbf{e}}_{si}$ directions, respectively.

$E_{\beta r}$, $E_{\phi r}$, E_{hr} and E_{sr} are the reflected electric field components in $\hat{\boldsymbol{\beta}}_r$, $\hat{\boldsymbol{\phi}}_r$, $\hat{\mathbf{e}}_{hr}$ and $\hat{\mathbf{e}}_{sr}$ directions, respectively.

$E_{\beta t}$, $E_{\phi t}$, E_{ht} and E_{st} are the transmitted electric field components in $\hat{\boldsymbol{\beta}}_t$, $\hat{\boldsymbol{\phi}}_t$, $\hat{\mathbf{e}}_{ht}$ and $\hat{\mathbf{e}}_{st}$ directions, respectively.

$E_{\beta r}$, $E_{\phi r}$, $E_{\beta t}$ and $E_{\phi t}$ are related to $E_{\beta i}$ and $E_{\phi i}$ by:

$$\begin{bmatrix} E_{\beta r} \\ E_{\phi r} \end{bmatrix} = \mathbf{B} \begin{bmatrix} E_{\beta i} \\ E_{\phi i} \end{bmatrix} \tag{2.72}$$

$$\begin{bmatrix} E_{\beta t} \\ E_{\phi t} \end{bmatrix} = \mathbf{A} \begin{bmatrix} E_{\beta i} \\ E_{\phi i} \end{bmatrix} \tag{2.73}$$

where

$$\mathbf{B} = \mathbf{T}(\alpha)\begin{bmatrix} R_h & 0 \\ 0 & R_s \end{bmatrix}\mathbf{T^{-1}}(-\alpha) \tag{2.74}$$

$$\mathbf{A} = -\mathbf{T}(-\alpha)\begin{bmatrix} T_h & 0 \\ 0 & T_s \end{bmatrix}\mathbf{T^{-1}}(-\alpha) \tag{2.75}$$

The resultant dyadic diffraction coefficient is

$$\mathbf{D} = \left(-D_a\hat{\boldsymbol{\beta}}' - D_b\hat{\boldsymbol{\phi}}'\right)\hat{\boldsymbol{\beta}} + \left(-D_c\hat{\boldsymbol{\beta}}' - D_d\hat{\boldsymbol{\phi}}'\right)\hat{\boldsymbol{\phi}} \tag{2.76}$$

where

$$\begin{bmatrix} D_a & D_b \\ D_c & D_d \end{bmatrix} = (\mathbf{I}_2 + \mathbf{A})(D_1 + D_2) - \mathbf{B}(D_3 + D_4) \tag{2.77}$$

$\mathbf{I}_2$ is 2×2 identity matrix, $\mathbf{A}$ and $\mathbf{B}$ are the *ray transfer* matrices defined in Equations (2.74) and (2.75). Equations (2.76) and (2.77) reduce to Equations (2.43) and (2.63) at normal incidence, i.e. $\alpha = \pm\frac{\pi}{2}$.

A popular heuristic solution for diffraction from a lossy wedge at normal incidence is proposed by Luebbers (1984). The idea is very similar to that of Burnside and Burgener (1983) except that there is no transmission through the wedge and two different reflection coefficients are used instead of one. The second reflection coefficient is introduced to make the diffraction coefficient reciprocal. The resultant diffraction coefficients are:

$$D_{s,h} = (D_1 + D_2) + R_{ns,h}(n\pi - \phi)D_3 + R_{os,h}(\phi')D_4 \tag{2.78}$$

where $R_{ns,h}(n\pi - \phi)$ and $R_{os,h}(\phi')$ are the reflection coefficients for *n*-face and *o*-face at incident angles (measured from the wedge faces) $n\pi - \phi$ and ϕ', respectively. It is assumed that *o*-face is the wedge face closer to the source (smaller incident angle). Luebbers' diffraction coefficients work well when the receiver sees *n*-face, i.e. $\phi > (n-1)\pi$, this includes the shadow region.

Conversely, the error is large (Aïdi and Lavergnat, 2001) but the effect is usually small because at those locations the fields are dominated by the direct ray and reflected ray. Demetrescu, Constantinou and Mehler (1997) have shown that the error is big in the shadow region but we agree with Aïdi and Lavergnat (2001) that Demetrescu et al. may have made a mistake with Luebbers' coefficients. Refinements to Leubbers' heuristic solution can be found in Booysen and Pistorius (1992), Rouviere, Douchin and Combes (1999), Holm (2000), Aïdi and Lavergnat (2001), El-Sallabi, Rekanos and Vainikainen (2002), Schettino et al. (2007). Extension to 3D oblique incidence should be possible using Burnside and Bergener (1983) method described above (Vandamme, Baranowski and Mariage, 1995; Soni and Bhattacharya, 2010).

Another heuristic approach is presented by Bernadi, Cicchetti and Testa (2002). It differs from Burnside and Bergener (1983) and Luebbers (1984) variants in that it does not treat the diffraction terms $D_1, D_2, D_3,$ and D_4 as distinct terms, one for each real or virtual shadow boundary. Instead, it treats D_1 and D_4 as a pair for one real shadow boundary and D_2 and D_3 as another pair for another real shadow boundary. Additional pairs are added if there are more real shadow boundaries, one pair per real shadow boundary. The pairs are multiplied by the respective reflection or transmission coefficients for the real shadow boundaries. The real shadow boundaries are those fall outside the wedge. They are termed *characteristic optic rays* in Bernadi, Cicchetti and Testa (2002). Co-located incident and transmission shadow boundaries, if both exists, are considered two real shadow boundaries. Bernadi et al. formulation does not include virtual shadow boundaries which fall inside the wedge. To compute

diffraction inside the wedge, the definition of the wedge is reversed, i.e. *o*-face becomes *n*-face and vice versa and the new wedge angle parameter is $n' = 2 - n$. Burnside and Burgener (1983) method is used for 3D oblique incidence. The resultant dyadic diffraction coefficient is:

$$\mathbf{D} = \sum_{p=1}^{P} S_p \mathbf{d}_p \cdot \mathbf{T}_p \tag{2.79}$$

where

P is the total number of real shadow boundaries,

S_p is the jump indicator which assumes a value of +1 if the shadow boundary exhibits null to non-null discontinuity in the positive ϕ direction; Otherwise, it assumes a value of –1;

$$\mathbf{d}_p = d_{ps}\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}_p + d_{ph}\hat{\boldsymbol{\phi}}\hat{\boldsymbol{\phi}}_p \tag{2.80}$$

$$d_{ps,h} = D_0 \left\{ \cot\left(\frac{\phi - \phi_p}{2n}\right) F\left[2kL_p \sin^2\left(\frac{\phi - \phi_p}{2}\right)\right] \pm \cot\left(\frac{\phi + \phi_p}{2n}\right) F\left[2kL_p \sin^2\left(\frac{\phi + \phi_p - g(\phi + \phi_p)}{2}\right)\right]\right\} \tag{2.81}$$

ϕ_p is the ϕ coordinate of the shadow boundary,

$\phi, \phi_p \in [0, n\pi]$, i.e. no virtual shadow boundaries,

L_p takes the form of Equation (2.60) for straight wedge with flat faces,

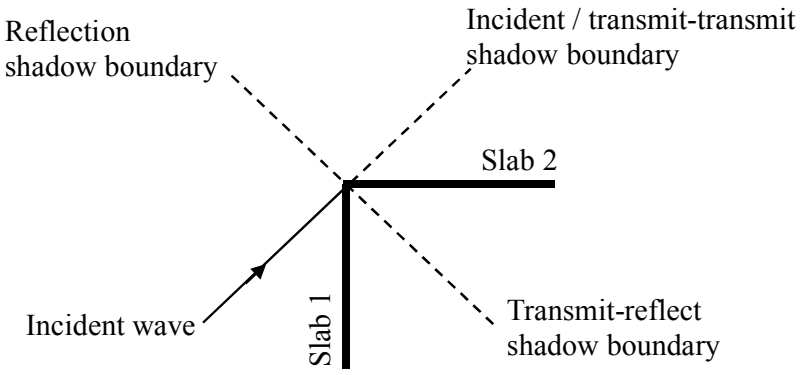
$$g(x) = \begin{cases} 0, & x < n\pi \\ 2n\pi, & x \geq n\pi \end{cases} \tag{2.82}$$

$$\mathbf{T}_p = T_{11}\hat{\boldsymbol{\beta}}_p\hat{\boldsymbol{\beta}}' + T_{12}\hat{\boldsymbol{\beta}}_p\hat{\boldsymbol{\phi}}' + T_{21}\hat{\boldsymbol{\phi}}_p\hat{\boldsymbol{\beta}}' + T_{22}\hat{\boldsymbol{\phi}}_p\hat{\boldsymbol{\phi}}' \tag{2.83}$$

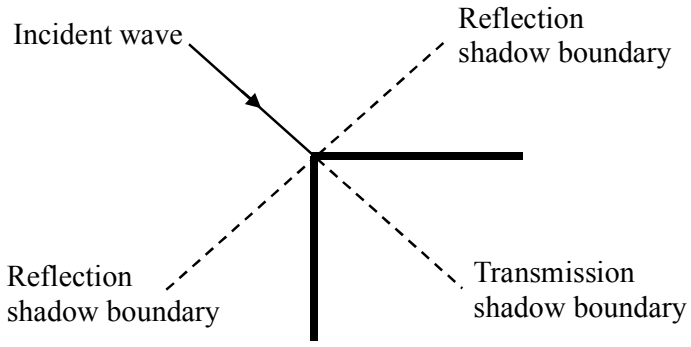
$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}$ is the matrix similar to Equations (2.74) and (2.75).

Equation (2.79) reduces to Equations (2.43) and (2.52) for PEC wedges. Figure 2.12 shows the shadow boundaries for a transmissive hollow wedge (Bernadi, Cicchetti and Testa, 2004). The ray transfer matrices for the reflection and transmission shadow boundaries are as given in Equations (2.74) and (2.75). The ray transfer matrices for the transmit-transmit and transmit-reflect shadow boundaries, respectively, are

$$\mathbf{U} = -\mathbf{T}(-\zeta)\begin{bmatrix} T_{h2} & 0 \\ 0 & T_{s2} \end{bmatrix}\mathbf{T}(-\delta)\begin{bmatrix} T_{h1} & 0 \\ 0 & T_{s1} \end{bmatrix}\mathbf{T^{-1}}(-\chi) \qquad (2.84)$$



(a) $\phi' < (n-1)\pi$ or $\phi' > \pi$



(b) $\pi < \phi' < (n-1)\pi$

Figure 2.12 Shadow boundaries for diffraction from a transmissive hollow wedge

$$\mathbf{V} = \mathbf{T}(\zeta)\begin{bmatrix} R_{h2} & 0 \\ 0 & R_{s2} \end{bmatrix}\mathbf{T}(-\delta)\begin{bmatrix} T_{h1} & 0 \\ 0 & T_{s1} \end{bmatrix}\mathbf{T^{-1}}(-\chi) \tag{2.85}$$

where

$$\sin\chi = -\hat{\mathbf{e}}_{hi1}\cdot\hat{\boldsymbol{\phi}}' \tag{2.86}$$

$$\cos\chi = \hat{\mathbf{e}}_{si1}\cdot\hat{\boldsymbol{\phi}}' \tag{2.87}$$

$$\sin\delta = -\hat{\mathbf{e}}_{hi1}\cdot\hat{\mathbf{e}}_{hi2} \tag{2.88}$$

$$\cos\delta = \hat{\mathbf{e}}_{si1}\cdot\hat{\mathbf{e}}_{hi2} \tag{2.89}$$

$$\sin\zeta = -\hat{\mathbf{e}}_{hi2}\cdot\hat{\boldsymbol{\phi}}' \tag{2.90}$$

$$\cos\zeta = \hat{\mathbf{e}}_{si2}\cdot\hat{\boldsymbol{\phi}}' \tag{2.91}$$

2.7 REMCOM Wireless InSite

REMCOM Wireless InSite® is a commercial suite of propagation simulators (Remcom, 2018a), recommended by an anonymous reviewer. A basic suite costs about RM 180,000 per license (as of 2018), not including yearly maintenance fees. It has four ray tracing simulators, three of them are shooting-and-bouncing-ray (SBR) ray tracers. The fourth one is named *Eigen ray method* which we believe is a simple image-based ray tracer. We do not use it for our comparisons because it is limited to 3 reflections whereas we need at least 6 reflections in our simulations. The three SBR ray tracers are, as REMCOM has named them, Urban Canyon (UC), Full 3D (F3D), and X3D ray tracers. The UC ray tracer is a 2D ray tracer. It uses the image technique to compute 3D ground reflected ray paths (Remcom, 2018b). As its name suggests, it is meant for urban canyon simulations, i.e. outdoor simulations without over-roof-top diffraction.

The F3D ray tracer is a traditional 3D SBR ray tracer which can be used for outdoor and indoor simulations. The X3D ray tracer is an enhanced version of F3D. The two major enhancements are GPU acceleration and exact path calculation. X3D is supposed to be faster and more accurate than F3D. All three SBR ray tracers limit the maximum number of interactions (reflection+transmission+diffraction) to 30. They have used a technique to overcome the well-known double counting and under counting problem (Remcom, 2018b). An oversized reception sphere is used for reception test. The received rays are stored, sorted, and then used to identify and remove duplicate rays. As we see it, this greatly increases the memory requirement and the per-receiver computation complexity.

CHAPTER 3

CORRECTION FACTORS FOR MULTILAYER WALL

3.1 Introduction

A real wall has thickness. For accurate simulation of indoor environments, the two surfaces of the wall are to be treated differently because they have different positions (Yang, Wu and Ko, 1998; Kenny and Nuallain, 2017). Our approach is to model the wall as one infinitesimally thin patch when computing ray paths. Errors in the ray paths are corrected using correction factors. The advantage of the one-patch approach is it is more efficient than the other methods that treat a wall as two patches instead of one. In a previous work (Teh, Kung and Chuah, 2006), we have derived the correction factors for reflected and transmitted ray paths, based on Burnside and Burgener (1983) formulation. In this Chapter, we will derive the correction factors for reflected, transmitted, and diffracted ray paths, based on a more general multilayer wall model (Balanis, 1989). Uniform plane wave is assumed, as it is done in Sections 2.5 and 2.6 (pp. 24-29). It is reasonable to assume that errors in the field amplitude and polarization are negligible (Burnside and Burgener, 1983). Hence, we only make corrections to the path delay which in turn corrects the field phase angle.

3.2 Reflection and Transmission

Figure 3.1 shows the relationship between the infinitesimally thin patch used for ray path computation and the multilayer wall it represents. The red lines are the correct ray paths to be used in conjunction with reflection and transmission coefficients given by the multilayer wall model (see Section 2.5, p. 24). The blue lines are ray paths computed using the infinitesimally thin patch. Based on the shown geometry, the delay correction factor for both the reflected and transmitted ray paths is

$$DCF_r = \frac{-d\cos\theta_i}{c} \tag{3.1}$$

where c is the speed of light in air.

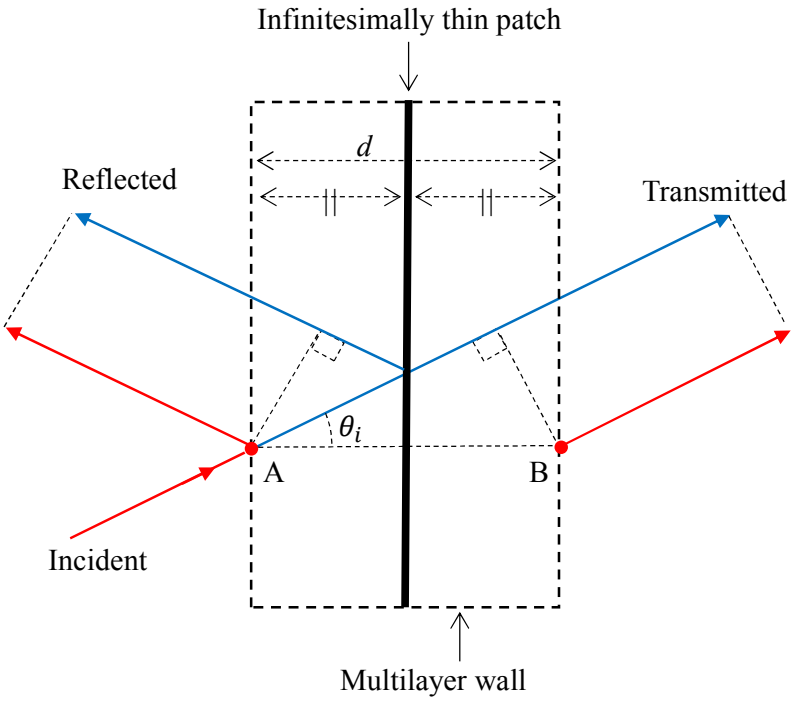


Figure 3.1 An infinitesimally thin patch model of a multilayer wall for ray path computation

The transmission coefficients given in Equations (2.24) and (2.25) (p. 27) include the phase change due to propagation delay from point A to point B (see Figure 3.1). To accurately simulate the propagation delay, the delay information should be removed from the transmission coefficients and added to the delay correction factor for the transmitted ray path. The negative phase change due to propagation delay from one interface to the next interface of a multilayer wall is given by the imaginary part of ψ_p in Equation (2.34) (p. 28). The total negative phase change is:

$$\Delta\phi = \sum_{p=1}^{M} \mathrm{Im}(\psi_p) \tag{3.2}$$

Hence, the modified transmission coefficients and delay correction factor for the transmitted ray path are:

$$T_s' = \frac{1}{A_0} e^{j\Delta\phi} \tag{3.3}$$

$$T_h' = \frac{1}{C_0} e^{j\Delta\phi} \tag{3.4}$$

$$DCF_t = \frac{\Delta\phi}{\omega} - \frac{d\cos\theta_i}{c} \tag{3.5}$$

where

ω is the angular frequency,

A_0 and C_0 are as defined in Equations (2.24) and (2.25) (p. 27)

The transmitted ray path correction is not specific to the one-patch approach nor the multilayer formulas given in Section 2.5 (p. 24). The same correction is required for a two-patch approach or other multilayer formulas.

We have not seen such correction being described in the literature. The implementation of the multilayer formulas has been verified by comparing its results to those of Orfanidis (2016) implementation which uses a different set of multilayer formulas. The implementation of the correction factors has been verified by comparing its results to those of Teh, Kung and Chuah (2006). The comparison has also shown that Burnside and Burgener (1983) formulas implemented by Teh, Kung and Chuah (2006) is not accurate for lossy walls but the error is small.

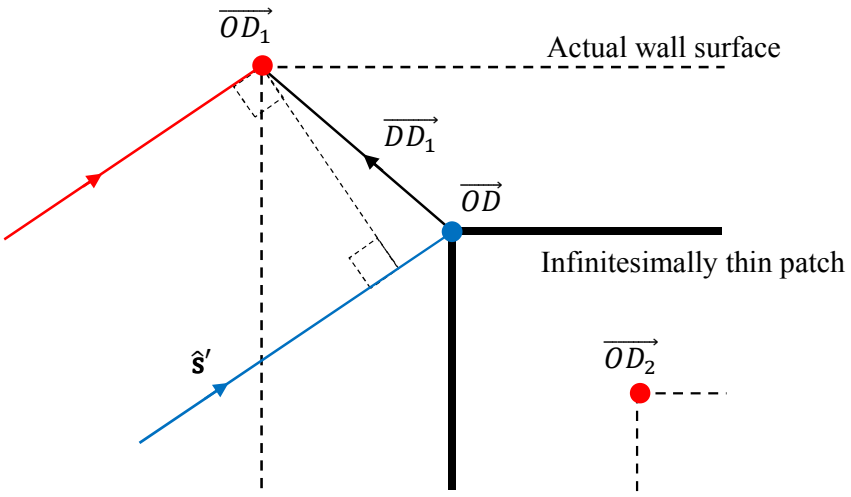
3.3 Diffraction

Figure 3.2 shows the problem geometry involving diffraction. $\overrightarrow{OD_1}$ and $\overrightarrow{OD_2}$ are the position vectors of diffraction points on the actual edges. The actual ray paths are in red. The ray paths computed using the one-patch approach are in blue. In general, the plane containing the incident ray and $\overrightarrow{DD_1}$ is not the same as the plane containing the diffracted ray and $\overrightarrow{DD_1}$. Hence, we have plotted them separately. Also, the planes are not perpendicular to the wall, in general, i.e. the distance between the interior and exterior wall surfaces shown in Figure 3.2 is not the same as the wall thickness. From Figure 3.2, the delay correction factor for diffraction from the actual exterior edge is

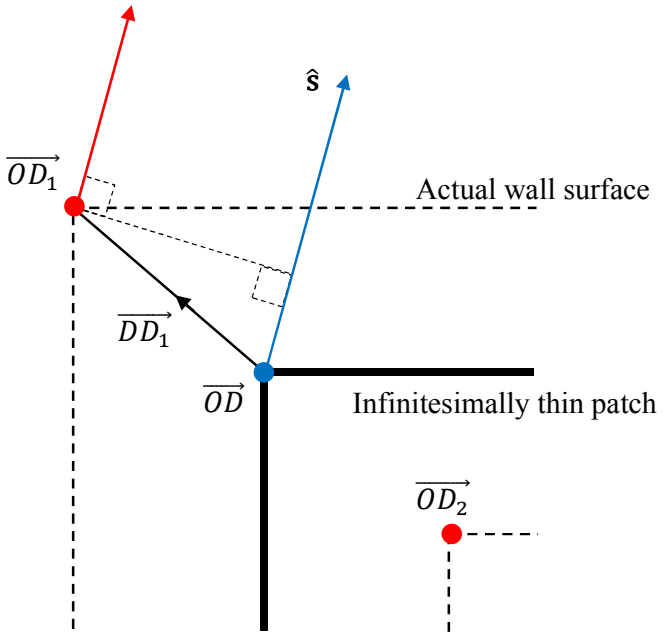
$$DCF_{d1} = \overrightarrow{DD_1} \cdot (\hat{\mathbf{s}}' - \hat{\mathbf{s}}) \tag{3.6}$$

Similarly, the delay correction factor for diffraction from the actual interior edge is

$$DCF_{d2} = \overrightarrow{DD_2} \cdot (\hat{\mathbf{s}}' - \hat{\mathbf{s}}) \tag{3.7}$$



(a) Incident ray



(b) Diffracted ray

Figure 3.2 Ray path correction for diffraction

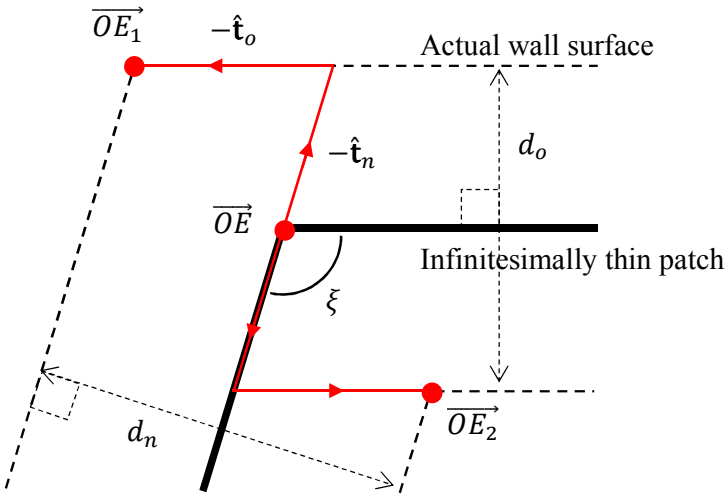


Figure 3.3 Actual exterior and interior edges of a wedge

$\overrightarrow{OD_1}$ and $\overrightarrow{OD_2}$ can be computed from Equations (2.37) to (2.42) (pp. 29) using shifted $\overrightarrow{OE}$. From Figure 3.3, the shifted $\overrightarrow{OE}$ are

$$\overrightarrow{OE_1} = \overrightarrow{OE} - \frac{\hat{\mathbf{t}}_o d_n + \hat{\mathbf{t}}_n d_o}{2\sin\xi} \tag{3.8}$$

$$\overrightarrow{OE_2} = \overrightarrow{OE} + \frac{\hat{\mathbf{t}}_o d_n + \hat{\mathbf{t}}_n d_o}{2\sin\xi} \tag{3.9}$$

The reflection and transmission coefficients used in diffraction calculation also need to be corrected, in accordance to the corrections made to the reflected, transmitted, and diffracted ray paths, to ensure field continuity across the shadow boundaries. Without any path correction, the reflected and diffracted fields along a reflection shadow boundary, respectively, are

$$E_r = Re^{-jk(s+s')} \tag{3.10}$$

$$E_d = -\frac{E_r}{2} = -\frac{R}{2}e^{-jk(s+s')} \tag{3.11}$$

After the path corrections, they become

$$E_r' = Re^{-jk(s+s')-j\omega \cdot DCF_r} \tag{3.12}$$

$$E_d' = -\frac{R}{2}e^{-jk(s+s')-j\omega \cdot DCF_d} \tag{3.13}$$

To maintain field continuity across the shadow boundary, the diffracted field needs to be corrected to

$$\begin{aligned} E_d'' &= -\frac{E_r'}{2} \\ &= -\frac{R}{2}e^{-jk(s+s')-j\omega \cdot DCF_r} \\ &= -\frac{Re^{j\omega(DCF_d - DCF_r)}}{2}e^{-jk(s+s')-j\omega \cdot DCF_d} \end{aligned} \tag{3.14}$$

Hence, the phase correction factor to be applied to the reflection coefficient used in diffraction calculation is:

$$\varphi_r = \omega(DCF_d - DCF_r) \tag{3.15}$$

Similarly, the phase correction factor to be applied to the transmission coefficient used in diffraction calculation is:

$$\varphi_t = \omega(DCF_d - DCF_t) \tag{3.16}$$

CHAPTER 4

DIFFRACTION RAY-POLYGON

4.1 Introduction

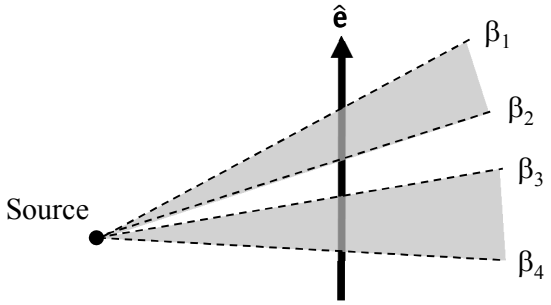
In an earlier work (Teh and Chuah, 2003), we have proposed a shooting-and-bouncing-polygon (SBP) ray tracer for radio propagation modelling (see Section 2.2, p. 12). The SBP ray tracer works very well with reflection and transmission. However, it is handicapped when dealing with diffraction. Due to the difficulty in projecting diffraction ray-polygon, a non-ray-polygon method has been used to handle diffraction. The non-ray-polygon method is not capable of tracing diffracted-reflected rays (diffraction followed by reflection). In this Chapter, we will describe a way to project and trace diffraction ray-polygon. With the full implementation of diffraction ray-polygon, the improved SBP ray tracer will be able to trace diffracted-reflected rays. It will produce more accurate results when diffracted-reflected rays play an important role.

4.2 Edge-Fixed Diffraction Ray-Polygon

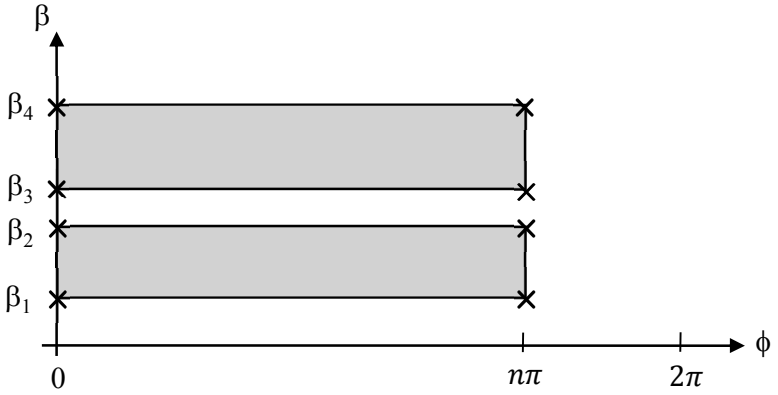
Diffraction ray-polygon is to be projected following the law of edge diffraction. Performing the projection in a Cartesian coordinate system is cumbersome and has no obvious advantage. We have adopted an edge-fixed

coordinate system for diffraction ray-polygon. Details about the edge-fixed coordinate system and the computation of edge-fixed β and ϕ coordinates can be found in Section 2.6 (p. 29). The face of the wedge from which ϕ is measured is known as *o-face*; the other face is known as *n-face*. The edge vector $\hat{\mathbf{e}}$ and the direction of ϕ are related by a right hand rule. The β coordinate is measured from $\hat{\mathbf{e}}$. Diffracted ray-polygons, including diffracted-reflected and diffracted-transmitted ray-polygons, are defined in the β - ϕ plane. Upon diffraction, a diffracted ray-polygon is spawned. Its ϕ coordinate extends from 0 to $n\pi$ where $n\pi$ is the exterior angle of the diffracting wedge. The span of the ϕ coordinate can be increased, e.g. extending from 0 to 2π to include diffraction into the wedge, or reduced to cover the shadowed regions only. The span of the β coordinate is defined by the illuminated edge segments. An example is shown in Figure 4.1. Shaded area in Figure 4.1(a) is the illumination zone. The example has 2 illuminated edge segments. The ray-polygon consists of 2 contours (shaded in Figure 4.1(b)), each is defined by 4 vertices marked by × markers.

The diffracted ray-polygon is traced in a way similar to Figure 2.5 (p. 17). Reception test is done by computing the (β, ϕ) coordinates of the receivers. A receiver is said to have received the ray if its position in the β - ϕ plane is inside the diffracted ray-polygon. To compute the beam-wall intersection, the wall is projected onto the β - ϕ plane. A wall's projection in the β - ϕ plane is curved in general and is approximated by a polygon. The intersection between the diffracted ray-polygon and the wall's projection defines new diffracted-reflected and diffracted-transmitted ray-polygons. The diffracted ray-polygon is



(a) Illuminated edge segments



(b) Ray-polygon in the β - ϕ plane

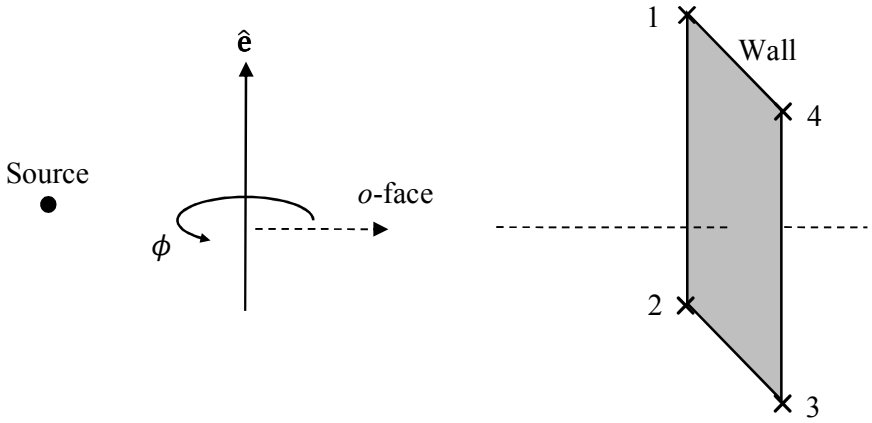
Figure 4.1 Initial diffracted ray-polygon

clipped by subtracting from it the wall's projection. Double and higher order diffraction by parallel edges can be handled in the same way because the computation of (β, ϕ) coordinates is the same. However, we have not implemented double and higher order diffraction in this work. Although double diffraction is important in some applications, e.g. over-roof-top diffraction, it is not required in many applications (Chen and Jeng, 1997; Catedra et al., 1998; Bernardi, Cicchetti and Testa, 2004; Remcom, 2018b).

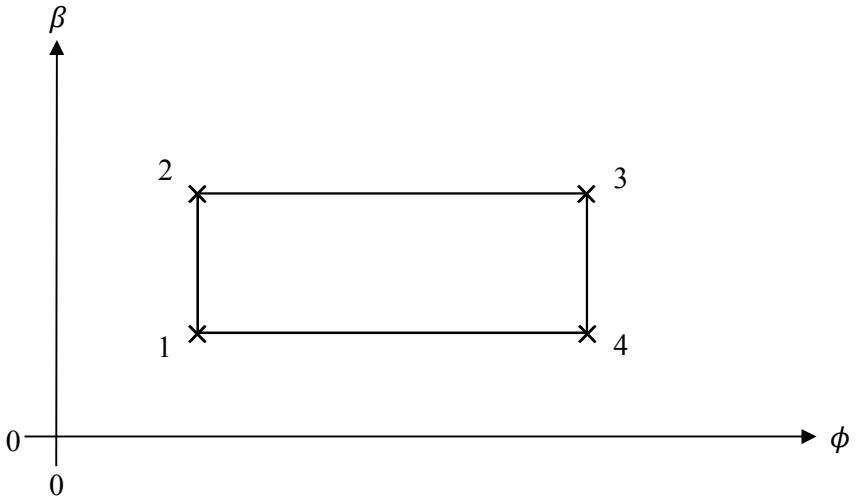
4.3 Projection Problems

The above relies on the ability to compute the β - ϕ projection of Cartesian walls. Catedra et al. (1998) have also used an edge-fixed coordinate system in their angular z-buffer algorithm to keep track of diffracted rays. They have assumed that a wall's projection in the β - ϕ plane is approximated by its vertices. The assumption is not always valid, however. Figure 4.2 and Figure 4.3 illustrate two scenarios where the assumption is not valid. Wall vertices are marked by × markers and are numbered. The actual wall's projection is shaded in (c). Its polygon approximation is given by the red dashed contours. The red square markers mark the additional vertices required to define the red dashed contours.

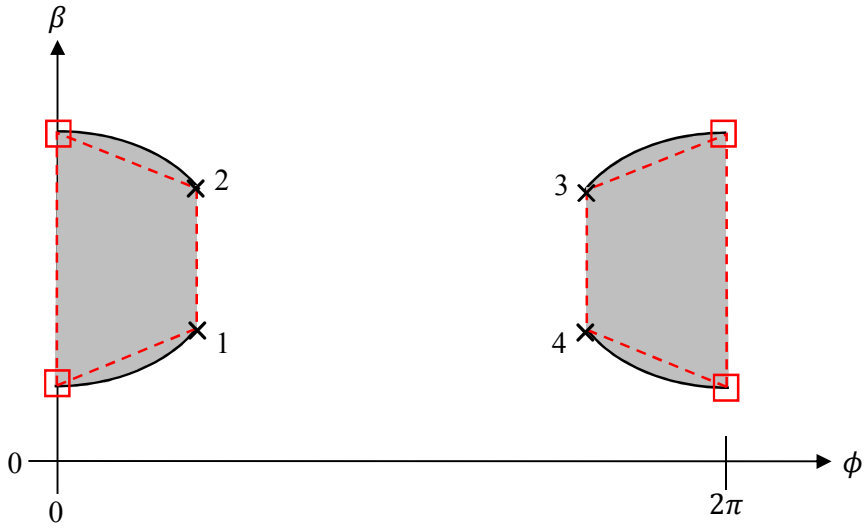
In Figure 4.2, the wall spans the *o*-face (ϕ = 0) of the diffracting wedge. It is obvious that the wall contains ϕ = 0 and its β - ϕ projection is as defined in Figure 4.2(c) instead of Figure 4.2(b). The actual wall's projection is curved and it is approximated by the red dashed polygon in Figure 4.2(c). In Figure 4.3, a square wall perpendicular to the diffracting edge intersects the edge (or its extension) at the wall's center. The four vertices of the wall have the same β coordinate and fail to approximate the wall's projection. The wall's projection can be visualized by drawing radial lines on the wall, from the edge-wall intersection point to the wall sides, as shown in Figure 4.3(d). Points lying on one radial line have the same ϕ coordinate. Their β coordinates extend from a maximum at the wall side to a minimum at the edge-wall intersection point. Each radial line corresponds to a vertical line in the β - ϕ plane. Thus, the wall's



(a) Scenario

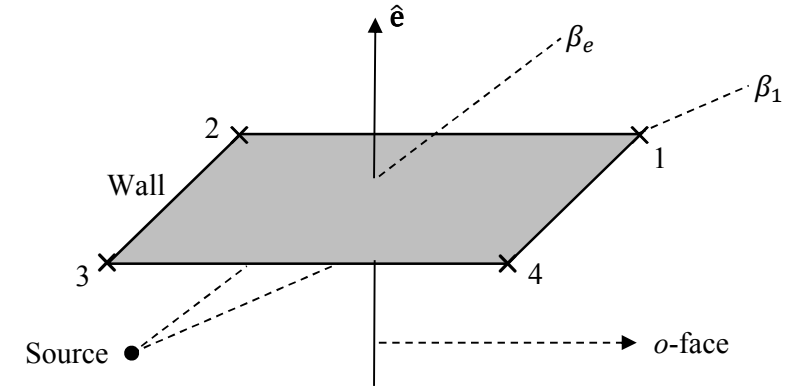


(b) Polygon formed by wall vertices

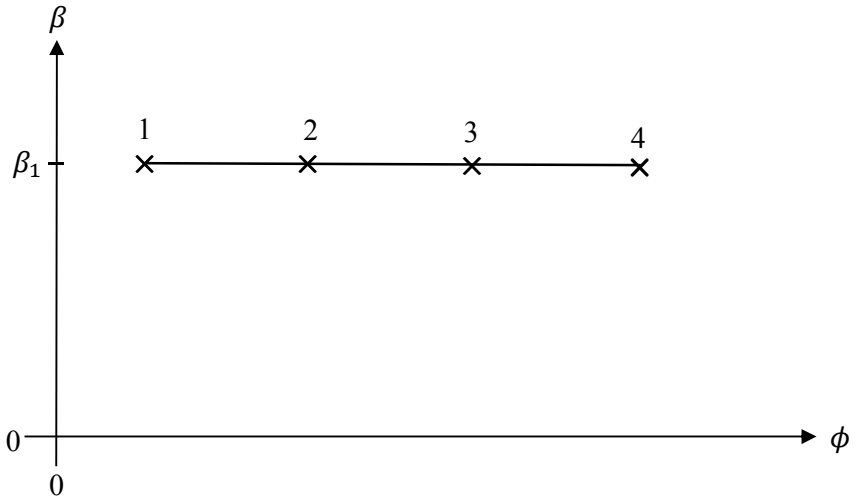


(c) Correct wall's projection

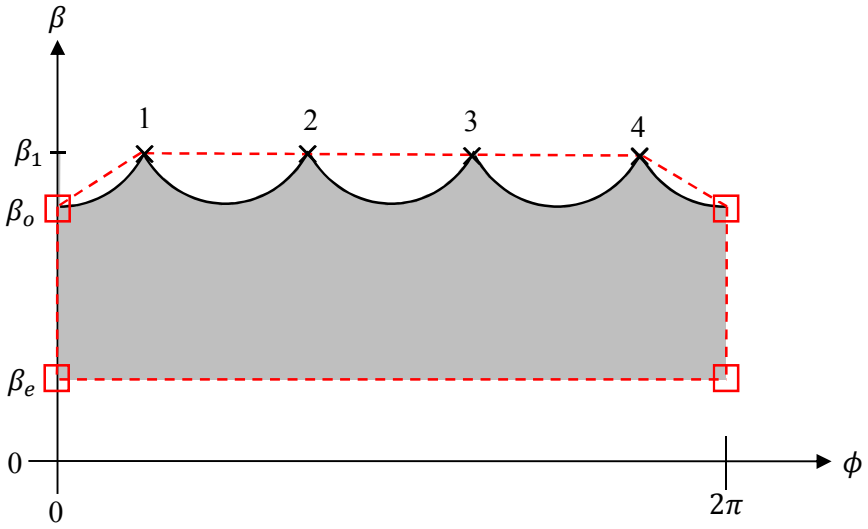
Figure 4.2 Wall's projection in the β - ϕ plane: wall spans *o*-face



(a) Scenario

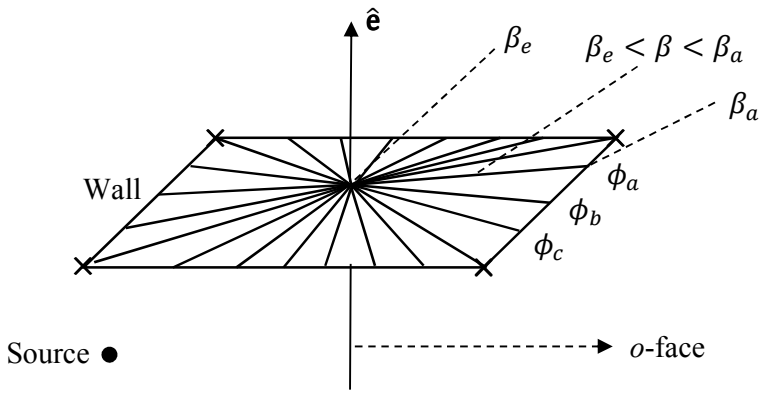


(b) Polygon formed by wall vertices



(c) Correct wall's projection

Figure 4.3 Wall's projection in the β - ϕ plane: wall intersects edge



(d) Radial lines

Figure 4.3 Wall's projection in the β - ϕ plane: wall intersects (continued)

projection is as shown in Figure 4.3(c). The same arguments apply to any wall intersecting the diffracting edge (or its extension).

4.4 Projection Algorithm

We describe below an algorithm to compute the correct β - ϕ projections of walls. It consists of 2 basic steps: 1) to identify the wall configuration and 2) to add vertices or split the output contour based on the wall configuration. A flowchart of the algorithm is given in Figure 4.4. Let a wall be defined by an array of vertices where adjacent vertices define the wall sides. Let line(k, i) be the line extending from vertex k to vertex i.

1) If the wall and the edge are on the same plane, the β - ϕ projection is merely a straight line with no area and is ignored.
2) Compute the β coordinate of wall-edge intersection point, β_e .

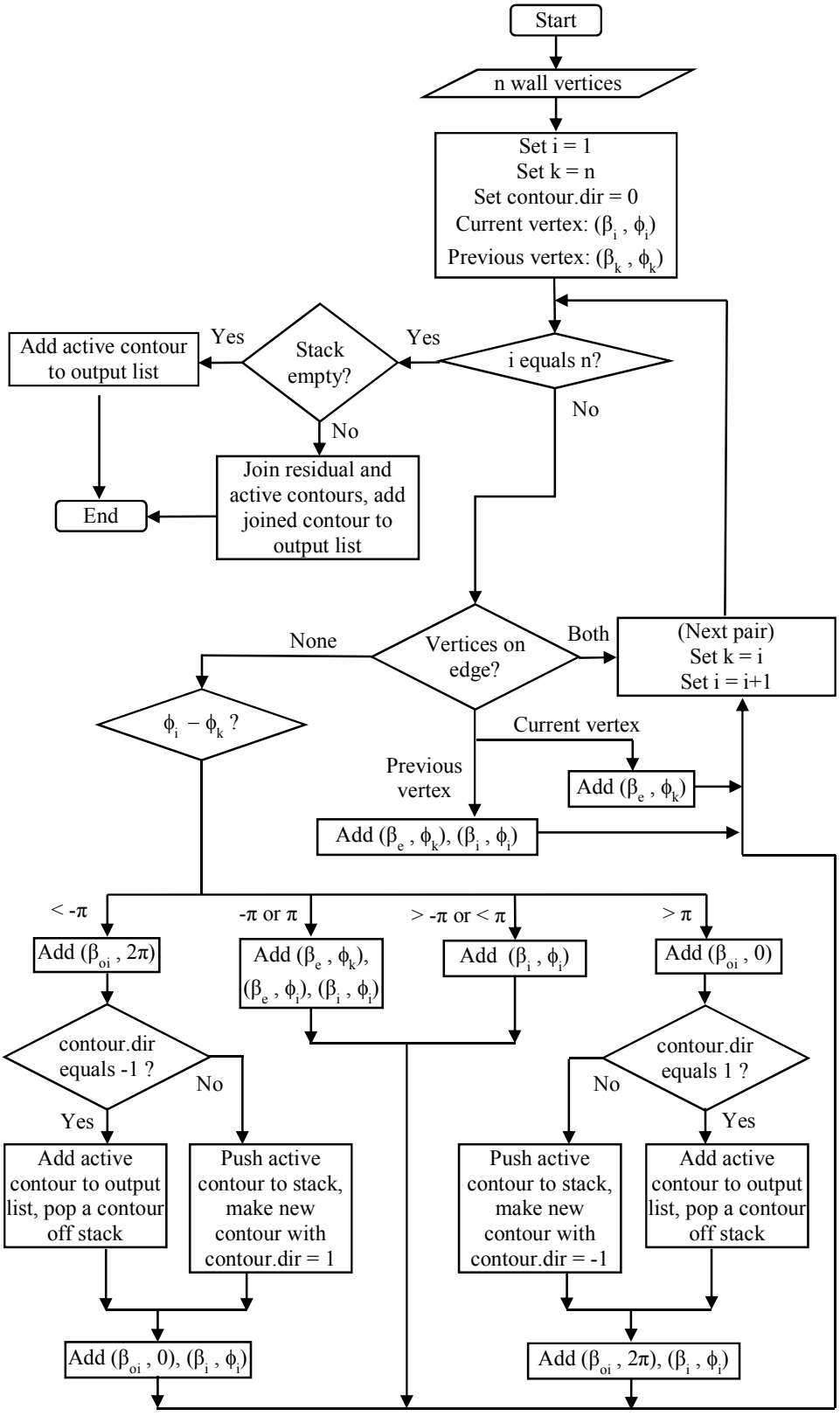


Figure 4.4 Edge-fixed β - φ projection of Cartesian polygons

3) Compute the (β, ϕ) coordinates of wall vertices.

4) Create an empty stack of incomplete contours.

5) Create an empty output list of complete contours.

6) Create an empty contour with direction flag contour.dir = 0.

7) Do once for each wall vertex, in sequence:

   a) If both the previous vertex (β_k, ϕ_k) and the current vertex (β_i, ϕ_i) are not on the edge:

      i) If $|\phi_i - \phi_k| < \pi$, the wall side i.e. line(k,i) does not span the *o*-face. The current wall vertex (β_i, ϕ_i) is appended to the active contour.

      ii) If $\phi_i - \phi_k > \pi$, from vertex k to vertex i, the wall side spans the *o*-face in the $-\phi$ direction. Contours crossing $\phi = 0$ are to be split into two (see Figure 4.2). There are two possible scenarios. If contour.dir = 1, the contour starts from $\phi = 0$ and grows in the $+\phi$ direction. If it crosses the *o*-face in the $-\phi$ direction, it has made a complete loop and returns to the previous contour stored on the stack. On the other hand, if contour.dir $\neq 1$ and the contour crosses the *o*-face in the $-\phi$ direction, the contour is incomplete. It is stored on the stack and a new contour is created. The new contour starts from $\phi = 2\pi$ and grows in the $-\phi$ direction. In either scenario, new vertices on the *o*-face are to be appended to the contours accordingly. A summary is given here:

- Compute the intersection point (β_{oi}, 0) between line(k, i) and the *o*-face. Append (β_{oi}, 0) to the active contour.
- If contour.dir = 1, add the active contour to the output list and pop an incomplete contour off the stack. Otherwise, push the active contour onto the stack and create a new empty contour with flag contour.dir = -1.
- Append (β_{oi}, 2π) and (β_i, ϕ_i) to the active contour.

iii) If $\phi_i - \phi_k < -\pi$, from vertex k to vertex i, the wall side spans the *o*-face in the $+\phi$ direction. The discussion is similar to (ii) above except for the change in contour direction. A summary is given here:

- Compute the intersection point (β_{oi}, 0) between line(k, i) and the *o*-face.
- Append (β_{oi}, 2π) to the active contour.
- If contour.dir = -1, add the active contour to the output list and pop an incomplete contour off the stack. Otherwise, push the active contour onto the stack and create a new empty contour with flag contour.dir = 1.
- Append (β_{oi}, 0) and (β_i, ϕ_i) to the active contour.

iv) If $|\phi_i - \phi_k| = \pi$, the wall side intersects the diffracting edge. The edge divides the wall side into two radial lines. The projection of the first radial line extends from (β_k, ϕ_k) to (β_e, ϕ_k). The projection of the second radial line extends

from (β_e, ϕ_i) to (β_i, ϕ_i). In summary, it is to append (β_e, ϕ_k), (β_e, ϕ_i) and (β_i, ϕ_i) to the active contour.

b) If the previous vertex (β_k, ϕ_k) is not on the edge but the current vertex (β_i, ϕ_i) is on the edge, line(k, i) is a radial line. Its projection extends from (β_k, ϕ_k) to (β_e, ϕ_k). In summary, it is to append (β_e, ϕ_k) to the active contour.

c) If the previous vertex (β_k, ϕ_k) is on the edge but the current vertex (β_i, ϕ_i) is not on the edge, line(k, i) is a radial line. Its projection extends from (β_e, ϕ_i) to (β_i, ϕ_i). In summary, it is to append (β_e, ϕ_i) and (β_i, ϕ_i) to the active contour.

d) If both the previous vertex (β_k, ϕ_k) and the current vertex (β_i, ϕ_i) are on the edge, do nothing because the current vertex is a duplicate.

8) If the stack has no residual contour, the wall does not intersect the diffracting edge (or its extension). The active contour is a complete contour and it is added to the output list.

9) If the stack has one residual contour, the wall intersects the diffracting edge (or its extension). The active contour and residual contour are joined to form one complete contour extending from $\phi = 0$ to 2π. Let the first vertex of the active contour and the last (back) vertex of the residual contour be (β_f, ϕ_f) and (β_{br}, ϕ_{br}), respectively. Note that $\phi_f = 0$ or 2π, $\phi_{br} = 2\pi$ or 0 and $\beta_f = \beta_{br} = \beta_o$, where β_o is the β coordinate of the intersection between one of the wall sides and the *o*-face. From the discussions on Figure 4.3, it is known that, at $\phi = 0$ or 2π, the β coordinates of the

projection should extend from β_o to β_e. Hence, the two contours are joined by appending (β_e, ϕ_{br}), (β_e, ϕ_f) and the active contour to the residual contour. The resultant contour is added to the output list.

We describe below applications of the algorithm to Figure 4.2 and Figure 4.3. Let the first previous-current vertices pair be vertex 4 and vertex 1 (k = 4, i = 1). For Figure 4.2, the first pair has $\phi_i - \phi_k < -\pi$. The first contour is appended and it contains $\{(\beta_{o1}, 2\pi)\}$. It is pushed onto the stack. The second contour is appended and it contains $\{(\beta_{o1}, 0), (\beta_1, \phi_1)\}$. For the second pair, $|\phi_i - \phi_k| = 0 < \pi$. The second contour is appended and it contains $\{(\beta_{o1}, 0), (\beta_1, \phi_1), (\beta_2, \phi_2)\}$. For the third pair, $\phi_i - \phi_k > \pi$, the second contour is appended and it contains $\{(\beta_{o1}, 0), (\beta_1, \phi_1), (\beta_2, \phi_2), (\beta_{o3}, 0)\}$. It is added to the output list. It defines the left red dashed contour in Figure 4.2(c). The first contour is popped off the stack. It is appended and it contains $\{(\beta_{o1}, 2\pi), (\beta_{o3}, 2\pi), (\beta_3, \phi_3)\}$. For the fourth pair, $|\phi_I - \phi_k| = 0 < \pi$. The first contour is appended and it contains $\{(\beta_{o1}, 2\pi), (\beta_{o3}, 2\pi), (\beta_3, \phi_3), (\beta_4, \phi_4)\}$. The stack has no residual contour. The first contour is added to the output list. It defines the right red dashed contour in Figure 4.2(c). The resultant $\beta - \phi$ projection is given by the output list which contains the two red dashed contours in Figure 4.2(c). The same can be achieved with any order of wall vertices as long as the adjacent vertices have defined the wall sides.

For Figure 4.3, the first pair has $\phi_i - \phi_k < -\pi$. The first contour is appended and it contains $\{(\beta_{o1}, 2\pi)\}$. It is pushed onto the stack. The second contour is appended and it contains $\{(\beta_{o1}, 0), (\beta_1, \phi_1)\}$. For the second, third and fourth

pairs, $|\phi_i - \phi_k| < \pi$. The second contour is appended and it contains $\{(\beta_{o1}, 0), (\beta_1, \phi_1), (\beta_2, \phi_2), (\beta_3, \phi_3), (\beta_4, \phi_4)\}$. The stack has one residual contour, i.e. the first contour. The residual contour is appended and it contains $\{(\beta_{o1}, 2\pi), (\beta_e, 2\pi), (\beta_e, 0), (\beta_{o1}, 0), (\beta_1, \phi_1), (\beta_2, \phi_2), (\beta_3, \phi_3), (\beta_4, \phi_4)\}$. It is added to the output list. It correctly defines the red dashed contour in Figure 4.3(c). Note that the additional computations involved are:

1) One line-plane intersection and one coordinate conversion to compute β_e .
2) One line-plane intersection and one coordinate conversion to compute β_{o1} .
3) Some flow controls.

The algorithm is applicable to all simple (non-self-intersecting) concave walls. For the concave walls, the output list may contain holes. To determine which contours are holes, the farthest intersections (from the edge) between the contours and the *o*-face are tracked. The intersections are sorted and numbered by their distance from the edge in descending order, e.g. intersection #1 is the farthest followed by #2. Contours having even numbered intersections are holes. The sort does not require explicit distance computation. It can be done using the β coordinates of the intersections. Depending on the wall orientation relative to the diffracting edge, β_e is always bigger or smaller than the β coordinates. If β_e is smaller, the β coordinates are proportional to the distance from the edge. If β_e is bigger, the β coordinates are inversely proportional to the distance.

If desired, the polygon approximation of the β - ϕ projection can be made more accurate by interpolation. Performing interpolation in the β - ϕ domain is

difficult. A simple but inefficient way is to interpolate the wall sides prior to projection. A close look at the projections will reveal that curves occur only when there are changes in the ϕ coordinates. We describe below a better interpolation algorithm. Before appending a new point to a contour, the change in the ϕ coordinate is computed. If the change is greater than a preset threshold, the (β, ϕ) coordinates of the midpoint are computed and appended to the contour. The same procedure applies recursively when appending the midpoint.

CHAPTER 5

CONVEX CELL PARTITIONING

5.1 Introduction

The main reason Teh and Chuah (2003) have used binary-space-partitioning is that it allows efficient sorting of objects by their distances from any point. We describe in this Chapter a better alternative to binary-space-partitioning for shooting-and-bouncing-polygon (SBP). The new scheme will be denoted as *convex cell partitioning* (CCP). CCP divides the simulated scene into a set of interconnected convex cells bordered by walls. Similar convex space methods have been used by Zhang, Yun, Iskander (2001) and Kenny and Nuallain (2017). Zhang, Yun and Iskander (2001) have used triangular pyramids (tetrahedrons) bordered by walls. Kenny and Nuallain (2017) have used rectangular cuboids; walls are also modelled as rectangular cuboids. Both Zhang and Kenny have used the convex space methods to accelerate ray-object intersection. Our CCP uses general convex shapes. Walls are treated as infinitesimally thin patch bordering the convex cells. The effect of wall thickness is taken care of when computing ray contribution (see CHAPTER 3, p. 43). In this work, the function of CCP is more than to accelerate ray-object intersection. It is used to efficiently sort objects by their distances from any point, which is essential for SBP. And, for the first time, a convex space method is used to facilitate the image generation process. Our results show that the CCP

version is about an order of magnitude faster than the binary-space-partitioning version in some applications (see Section 6.3, p. 92).

5.2 Basic Ideas

Take the indoor scene in Figure 5.1 for example. The indoor scene can be divided into 3 convex cells, e.g. EFGH (Cell #1), ABCHG (Cell #2) and CDEH (Cell #3). Wall GH is an empty or virtual wall connecting Cells #1 and #2. Wall EH connects Cell #1 and Cell #3. Wall CH connects Cell #2 and Cell #3. The interconnectivity between the cells can be represented by a graph, as shown in Figure 5.2. There is one diffracting wedge, i.e. CHE. The transmitter (4.4, 1.4, 2.4) is located in Cell #1, marked by a solid circle. Rays originated from the transmitter must hit the borders of Cell #1 before reaching the other cells. In other words, the 6 walls (including ceiling and floor) bordering Cell #1 are walls closest to the transmitter. If rays entering the other cells are treated as transmitted rays, direct rays from the transmitter only interact with the 6 walls bordering Cell #1, irrespective of the total number of walls the simulated scene might have. Because the cell is convex, the 6 walls will not shade each other. For the same reason, it can be concluded that receivers residing inside Cell #1 will receive the direct ray from the transmitter without any intersection test. For shooting-and-bouncing-ray (SBR), this means the direct rays need only be tested against the 6 walls (intersection test) and the receivers (reception test only) inside the cell. For image-based ray tracing, only the 6 walls will give rise to first order reflection or transmission. The same applies to rays transmitted

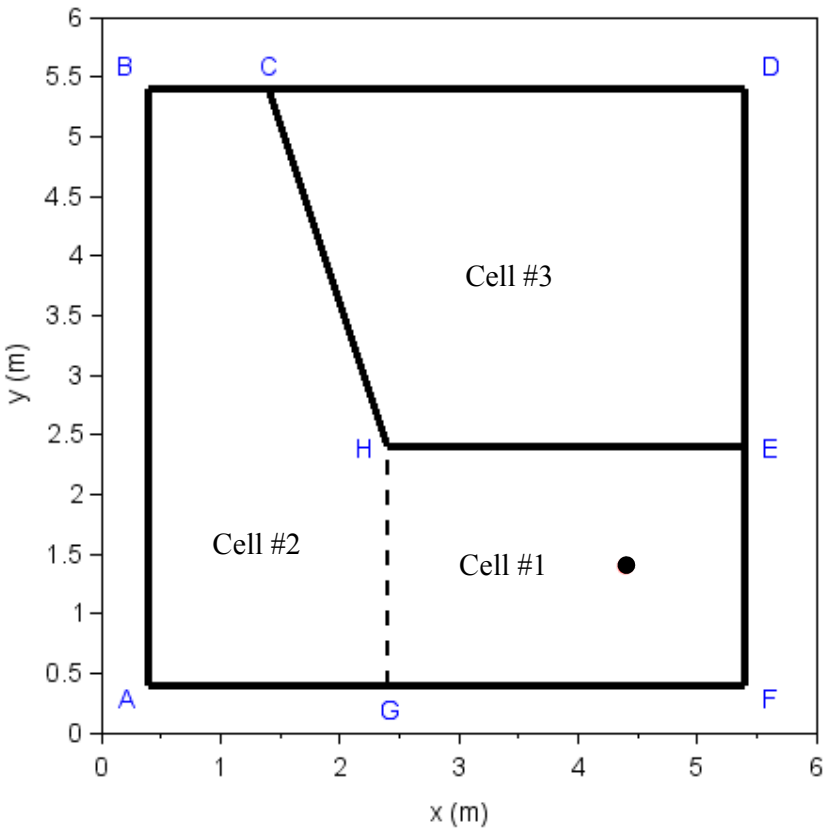


Figure 5.1 Convex cell partitioning of an indoor scene

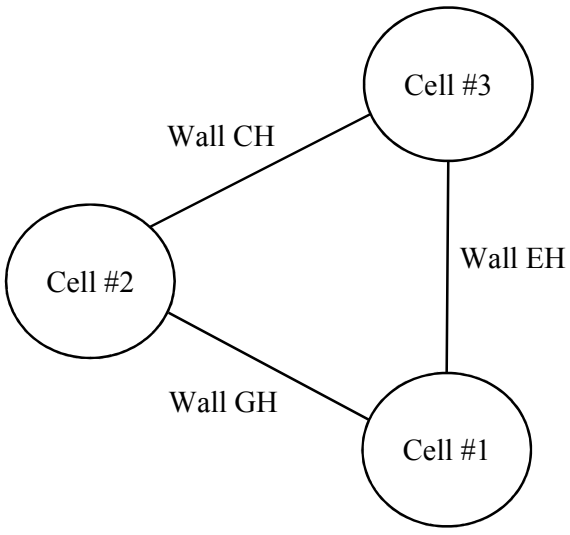


Figure 5.2 CCP graph

into a cell, the rays only interact with the borders of the cell and the receivers inside the cell. In other words, CCP has effectively reduced and localized the scattering problem to the cell containing the rays.

To be more specific, the rays only interact with the inner surface (relative to the cell) of the walls. This means an image can only give rise to higher order reflection from a wall if it is on the inner side of the wall. For example, the image of first order reflection from wall FG is at (4.4, -0.6, 2.4). The target cell for rays reflected from wall FG is Cell #1. The image is on the left of wall EF, i.e. the inner side of wall EF. Hence, the image will give rise to second order reflection from wall EF. The second order image is at (6.4, -0.6, 2.4). The new target cell is still Cell #1. The inner side of wall FG is above wall FG. Because the second order image is below wall FG, it will not give rise to third order reflection from wall FG.

The connectivity between the cells gives a sense of order to the walls and cells. For example, walls CD, CH and DE are borders of Cell #3. For a ray to interact with them, it must be in Cell #3. A ray originated from the transmitter must hit wall EH before entering Cell #3. Hence, wall EH is closer to the transmitter than walls CD, CH and DE. The relative order of walls CD, CH and DE is not important because they will not shade each other from the transmitter.

In summary, the basic ideas of CCP include:

1) Rays interact only with the cell they are in. This greatly simplifies the search for ray-object interaction.
2) The convexity of cell ensures that its walls will not shade each other. This removes the need for intra-cell shadowing test.
3) A wall is visible to an image only if the image resides in the same half-space as the cell relative to the wall, i.e. the inner surface of the wall is facing the image. This further simplifies the search for ray-object interaction.
4) Rays leave the cell only through one of its borders (walls) to the adjacent cell. This gives a sense of order to walls.

5.3 SBP-CCP

In this Section, we explain the use of CCP for shooting-and-bouncing-polygon (SBP). Following the basic ideas above, a ray-polygon only interacts with its target cell (the cell it is in) and the walls will not shade each other. There is no need to keep track the unobstructed part of the ray-polygon. Hence, a notable advantage of CCP over binary-space-partitioning is that CCP removes the need to clip the ray-polygon, a relatively expensive process. Figure 5.3 shows the updated flowchart for SBP-CCP. Traversing from one cell to another is also simpler and more intuitive than traversing a binary tree which requires the computation of the location of the source relative to a wall (left or right) (see Section 2.3, p. 19).

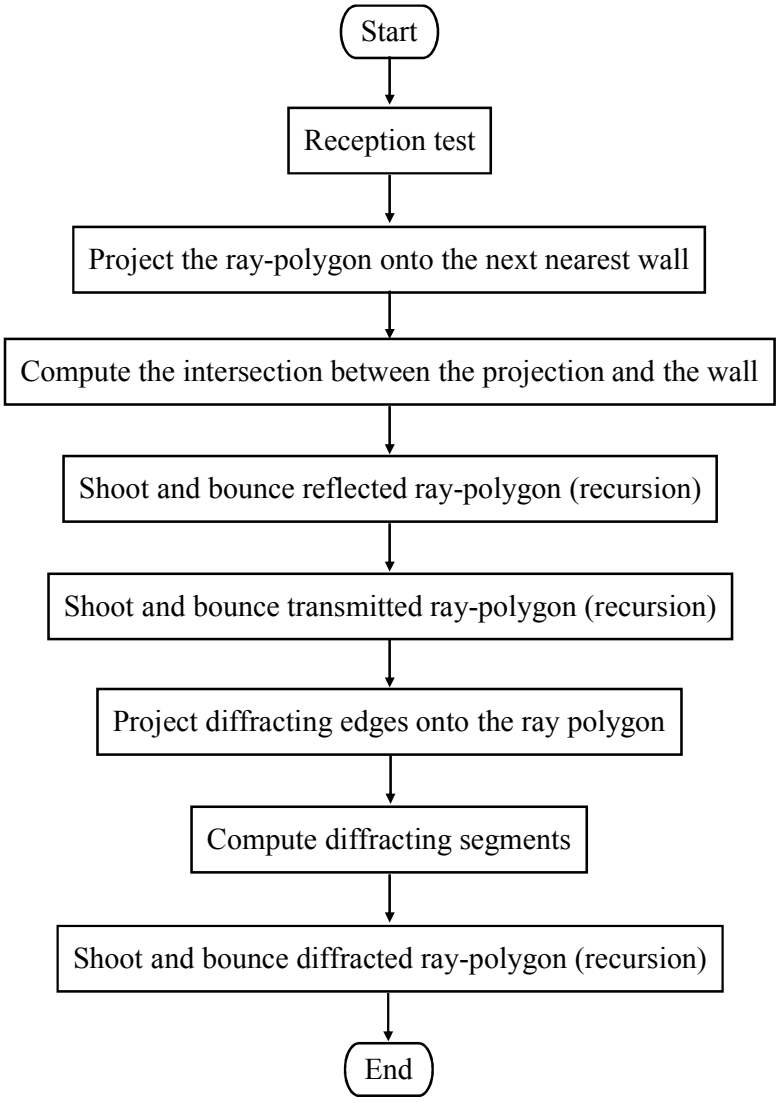


Figure 5.3 SBP-CCP flowchart

With reference to Figure 5.3, receivers subject to reception test are those residing in the target cell of the ray-polygon. Walls and edges subject to interaction are those bordering the target cell. Take the indoor scene in Figure 5.1 for example. Only one ray-polygon needs to be shot from the transmitter to cover all directions in the 3D space. This first ray-polygon is the universal set and it does not have a physical shape. Its target cell is Cell #1. Reception test

for the universal set is a trivial task, the result is always positive. All receivers in cell EFGH will receive the direct rays. Walls and edges subject to interaction are walls EF, FG, GH, HE, the floor and ceiling, and wedge CHE. For the universal set, the ray-polygon projections and intersections are trivial tasks, all walls and edges of the target cell will be illuminated in whole.

Take first order reflection off wall EF for example (see Figure 5.4). The reflection image is at (6.4, 1.4, 2.4). The reflection ray-polygon (grey line) is the entire wall EF. The target cell is Cell #1. Reception tests are performed on all receivers residing in Cell #1. Receivers whose projections on wall EF lie inside the reflection ray-polygon are said to have received the reflected rays. Upon reception, the exact ray path is computed by tracing backward from the receiver towards the transmitter. The point of reflection on wall EF is the intersection between wall EF and the line joining the receiver and the reflection image. The resultant ray path is transmitter $\rightarrow$ reflection point on wall EF $\rightarrow$ receiver. Note that the backward tracing only involves N line-plane intersections for N reflections. There isn't any other intersection or shadowing test. Walls and edges subject to second order interaction are again those bordering Cell #1.

Take first order transmission through wall EH for example (see Figure 5.5). The transmission image is the transmitter itself. The transmission ray-polygon (grey line) is the entire wall EH. The target cell is the adjacent cell on the other side of wall EH, i.e. Cell #3. The subsequent processing is similar to that of reflection. Transmission through empty walls is treated in the same way except that empty walls are omitted in backward tracing. In other words, empty

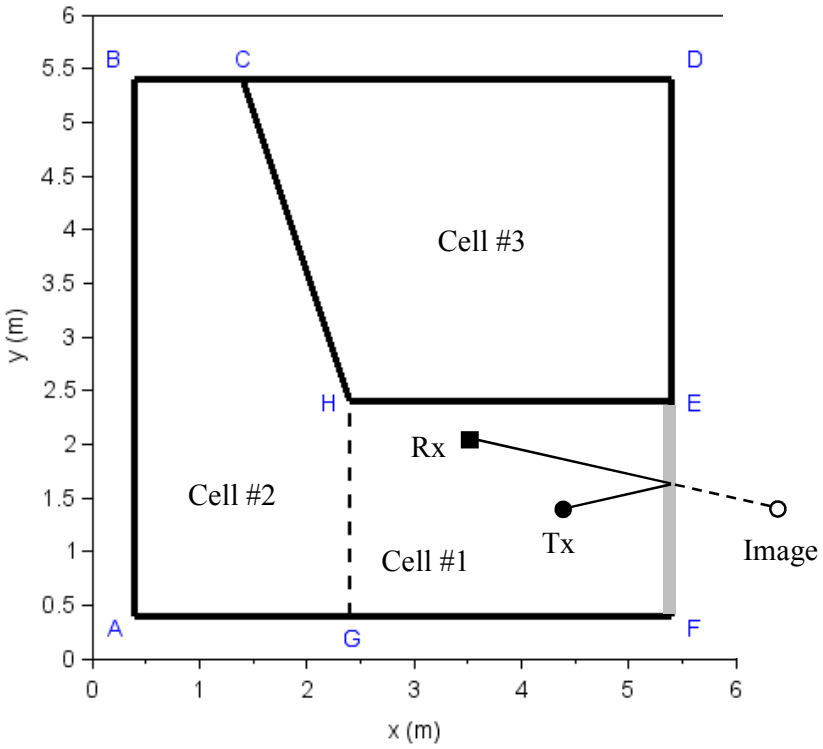


Figure 5.4 CCP: reflection

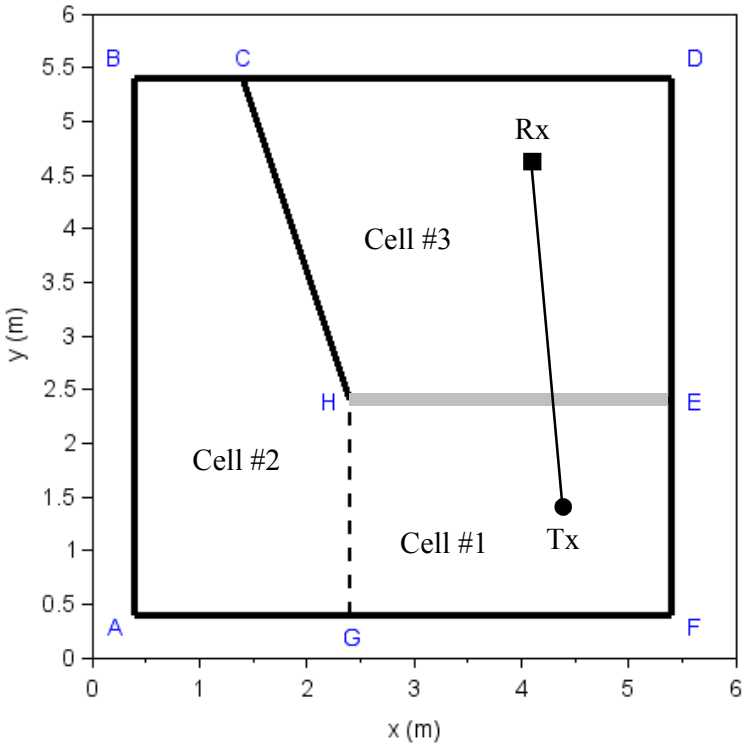


Figure 5.5 CCP: transmission

walls do not slow down backward tracing. Diffraction images are also treated in the same way except that edge-fixed diffracted ray-polygons are used and each is associated to multiple target cells. For example, diffraction off wedge CHE has 2 target cells, i.e. Cells #1 and #2. Cell #3 can be included as a target cell if diffraction into the wedge is desired. Recursive extension to higher order reflection and transmission is straight forward.

Note how CCP has helped to systematically generate the images. If walls ABCDEFGA are marked as non-transmissive, additional walls and cells appended to the model will have no effect on image generation. In the search of potential ray-wall or ray-receiver interactions, CCP does the first level of filtering by limiting the search scope to the target cells only. The location of the image relative to the wall (inside or outside) does the second level of filtering. Lastly, SBP does the third level of filtering, which is also the most expensive and restrictive but worthwhile. CCP can be used without SBP. CCP alone can achieve substantial acceleration in the image generation and backward tracing processes. However, our experience reveals that, without the additional filtering done by SBP, CCP-only will run considerably slower than SBP-CCP.

5.4 Implementation

Data of the walls, cells, diffracting wedges, transmitters, receivers, antennas and wall materials are stored in arrays. Individual item is referenced by its array index. The wall data structure consists of an ordered set of coplanar

patches. Each patch is associated with a material. The first patch always defines the full extent of the wall. It is the only patch used for SBP. The other smaller patches are used when computing ray contribution. For example, a concrete wall with glass windows and wooden doors will have the concrete wall as the first patch and smaller patches of windows and doors. Walls can be individually flagged as transmissive, reflective or empty. An empty wall is used to connect two adjacent cells not separated by a physical wall. The surface normal of a wall is related to the order of wall vertices by a right hand rule. It distinguishes the two half-spaces separated by the wall, one in the direction of the surface normal, one in the opposite direction. Data structure of a diffracting wedge holds indices to walls forming the wedge and cells containing the wedge, in addition to other wedge information like the edge vector and wedge angle. Data structure of an image holds a ray-polygon, the center of projection and a pointer to the parent image from which the current image is spawned. A transmitter is treated as a special image without ray-polygon and parent image. The core of CCP is the cell data structure. A cell holds indices to the bordering walls and diffracting wedges, the adjacent cells and the receivers residing in the cell. A cell also holds flags about the half-spaces it occupies, relative to the bordering walls.

CCP is recursive in nature. However, a recursive implementation is not suitable for an imperative programming language like C/C++. We describe below an iterative implementation of CCP-SBP:

1) Create an empty stack of trace-nodes. A trace-node holds an image, the index of the target cell, the trace-mode, and the current wall or edge

sequence number in the target cell. There are 3 trace-modes: reception test, reflection and diffraction.

2) Create a trace-node for each transmitter and push it onto the stack. The image is the transmitter. The target cell is the cell containing the transmitter. The initial trace-mode is reception test for all new trace-nodes. The wall sequence number is irrelevant in reception test mode.

3) Execute a trace-node from the top of the stack (without removing it from the stack, unless stated otherwise below) depending on the trace-mode until the stack is empty (see Figure 5.6):

   (a) Reception test mode: Perform reception test on all receivers in the cell. Upon reception, compute the ray path and ray contribution. Then, proceed to reflection mode with wall sequence number 0 (the first wall). In the case of a transmitter, all receivers in the cell will receive the direct ray. Other than that, the computation is similar to those described in Sections 2.2 (p. 12), 4.2 (p. 50) and 5.3 (p. 68).

   (b) Reflection mode: If the current wall sequence number $\geq$ the total number of walls bordering the cell, all walls bordering the cell have been considered, proceed to diffraction mode with edge sequence number 0 (the first edge). Otherwise, if the image and the cell are not in the same half-space of the current wall, the image will not illuminate the wall, proceed to the next wall, i.e. increment the wall sequence number and repeat (b). Otherwise, project the active ray-polygon held by the image onto the wall. The intersection between the projection and the wall defines new reflected ray-polygon and transmitted ray-polygon. If the active ray-polygon is a diffracted ray-polygon, use the β - ϕ

projection instead (see CHAPTER 4, p. 50). In the case of a transmitter, the entire wall defines new reflected and transmitted ray-polygons. If the wall is reflective, create a new trace-node for the reflection image and push it onto the stack. The target cell for the reflection image is the current cell. Do the same for the transmission image if the wall is transmissive. The target cell for the transmission image is the adjacent cell on the other side of the wall. Increment the wall sequence number of the active trace-node and repeat step 3 to execute the new trace-nodes.

(c) Diffraction mode: If the current edge sequence number ≥ the total number of diffracting edges in the cell, all edges in the cell have been considered, remove the current trace-node from the top of the stack and repeat step 3 to execute the next trace-node. Otherwise, compute the illuminated edge segments. All cells containing the edge will get the direct diffracted rays. They will be the target cells for the diffraction image. Create new trace-nodes, one per target cell, for the diffraction image and push them onto the stack. Increment the edge sequence number of the active trace-node and repeat step 3 to execute the new trace-nodes.

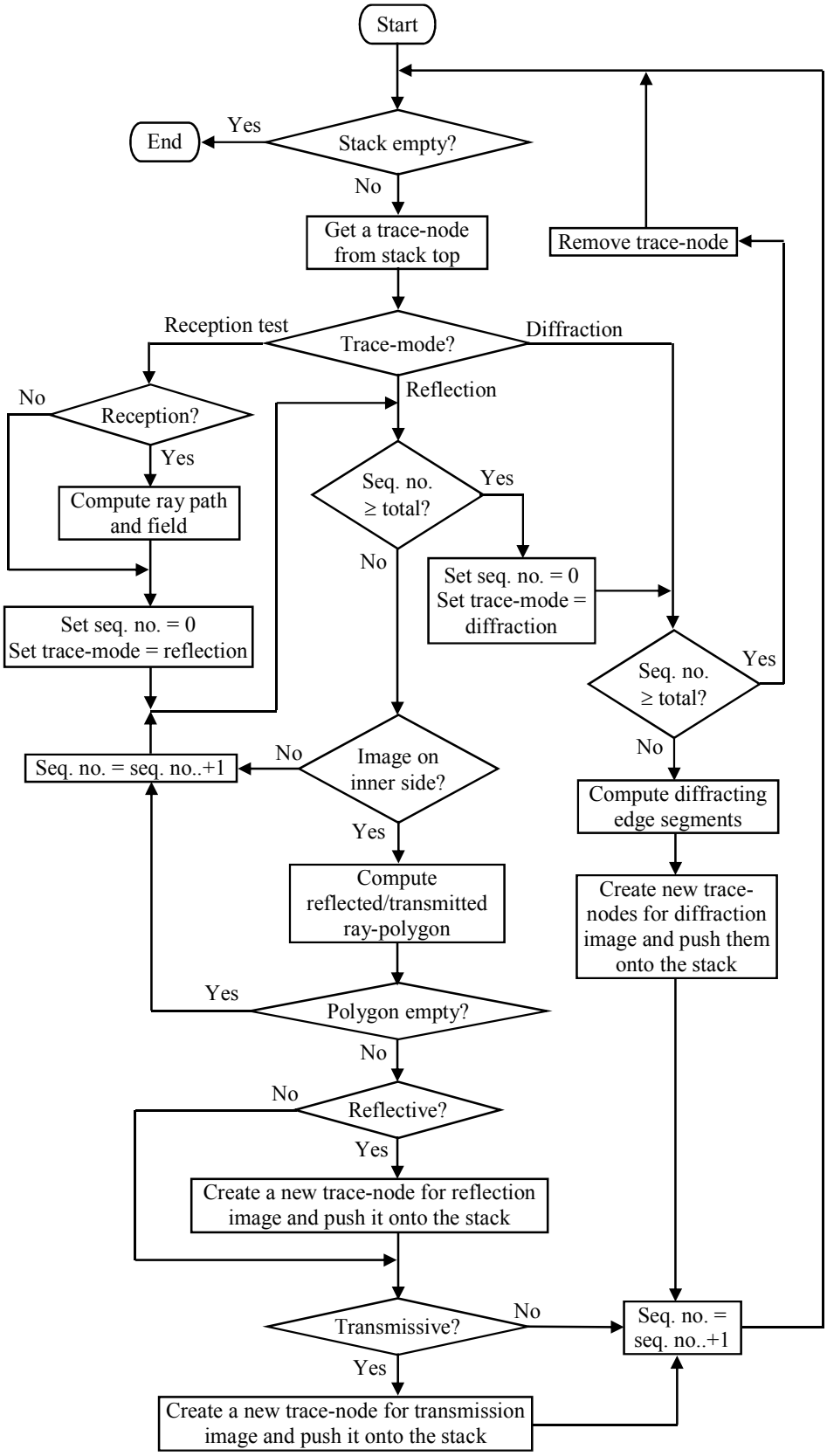
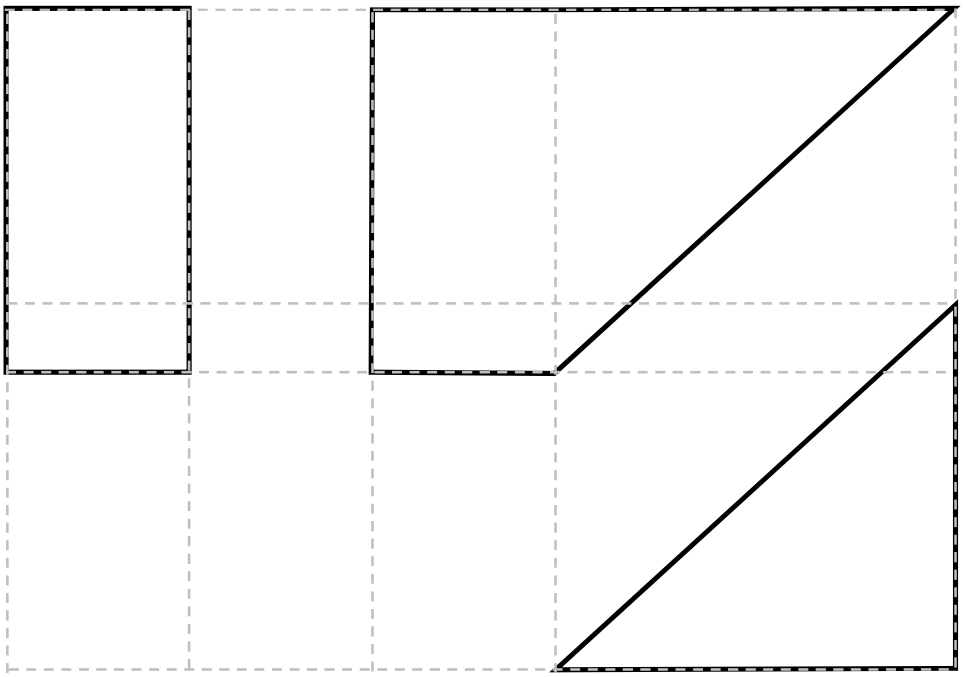


Figure 5.6 CCP trace-node execution

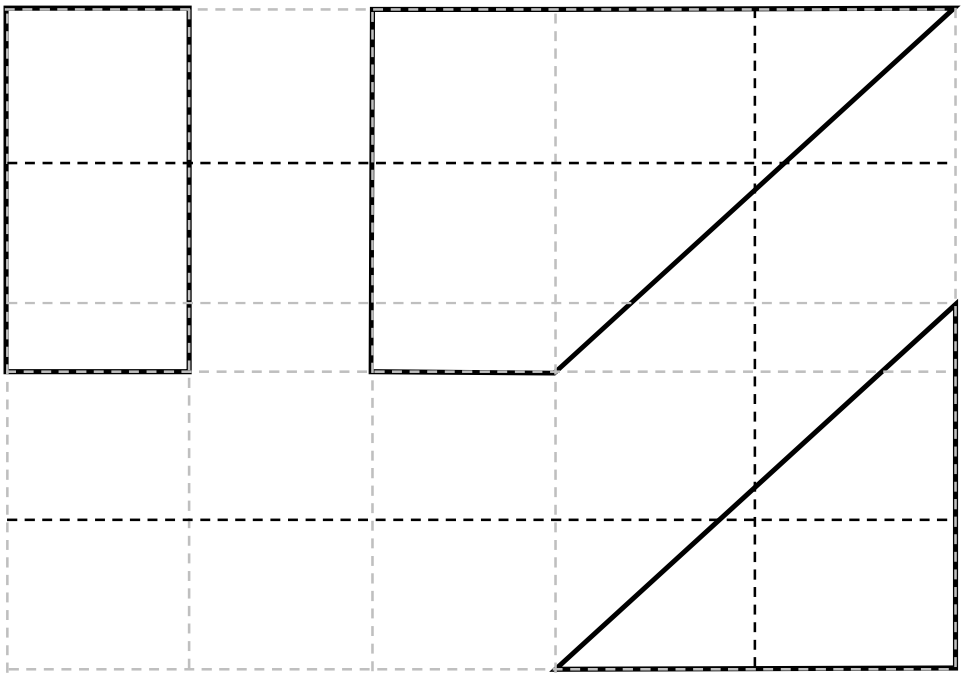
5.5 Generation of Convex Cell Data Structures

Unlike the binary tree, which must be computer generated, the formation of convex cells is much more intuitive and it can be defined manually as part of the scene description process. It is a one-time job per simulated scene. For large complex scenes, it is desirable that the convex cell data structures be computer generated, especially if the set of walls defining the scene is already available in digital form. In this section, we describe a way to automatically generate the convex cell data structures from a set of walls, which we have developed. Other possible methods can be found in computer graphics literature on *cell-and-portal generation* (Haumont, Debeir and Sillion, 2003; Walsh, 2008). A major difference being their cells are linked by portals (virtual walls) only, whereas our cells are linked by both portals and real walls.

Our method consists of 3 stages. The first stage creates axis-aligned rectangular grids based on the given set of walls. It consists of 2 steps. The first step creates the grids at all vertices of the walls. A 2D illustrative example is given in Figure 5.7(a). Solid lines are real walls (perpendicular to the paper). Grey dashed lines are grids at wall vertices. The second step adds additional grid lines to ensure that the separation between two adjacent grid lines is smaller than a threshold. The objective is to ensure that rectangular cells formed by the grids is cut by at most one non-axis-aligned wall. This is to simplify the splitting of cell due to the cut. Take Figure 5.7(a) for example. With the original grids at wall vertices, the grid separation is too big that the rightmost middle cell is cut by two non-axis-aligned walls. In Figure 5.7(b), new grid lines (black dashed



(a) Grids at wall vertices



(b) Finer grids

Figure 5.7 Cell-and-portal generation: creating grids (stage 1)

lines) are added to reduce the grid separation when it is too big (step 2). With the finer grids, each of the cells are cut by at most one non-axis-aligned wall. The two new horizontal grid lines are indeed not necessary. They are an undesired side-effect of the threshold. A good choice of the threshold is the smallest distance between two disjoint non-axis-aligned walls. In the case that all walls are axis-aligned, the threshold can be set big to avoid creating finer grids. Our implementation allows the threshold for each of the 3 axes to be set independently.

The second stage builds the convex cell data structures from the grids. A cell will be rectangular as defined by the grids if it is not cut by a non-axis-aligned wall, else it will be split into two convex cells. The split is done using a lookup table based on the type of split. The intersections between the cell borders and real walls are determine using a polygon clipping algorithm (Vatti, 1992; Murta, 2017). Only non-axis-aligned walls coinciding with the splitting plane need to be considered for intersection with the non-axis-aligned border. The intersections for axis-aligned borders are accelerated by the grouping of axis-aligned walls based on the grids. The intersections define real walls bordering the cell. Unfilled areas of the borders define the portals (virtual walls). Interconnectivity between the cells can be determined easily from the grids. An illustrative example for the second stage is given in Figure 5.8. The dashed lines are virtual walls. At this point, there are already working convex cell data structures. However, there are more convex cells than necessary.

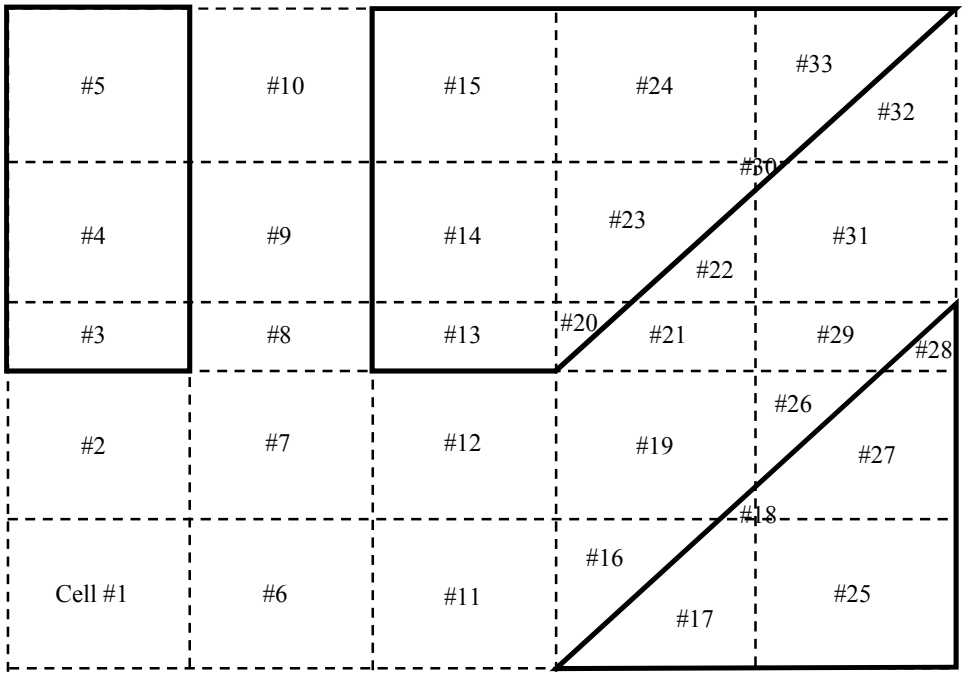


Figure 5.8 Cell-and-portal generation: forming convex cells (stage 2)

The third stage is to merge small convex cells into bigger convex cells. A cell can only merge with an adjacent cell if it has an axis-aligned border covered by virtual walls only and it is connected to the same adjacent cell through the border. The merge cannot happen across a non-axis-aligned border because it is at least covered in part by a real wall which is the reason for its existence. Once the merge criteria are met, all axis-aligned walls of both cells will not deter the merge. Only non-axis-aligned walls need to be checked for convexity. This is done by checking if the non-axis-aligned walls are on the inner side of all other borders of the merged cell. Take Cell #1 for example (see Figure 5.8). Let the right border (implementation dependent) be the first to be considered. The right border meets the merge criteria because it is fully virtual, it connects to Cell #6 only, and both Cells #1 and #6 have no non-axis-aligned

wall. After the merge, the right border (implementation dependent) of the merged cell is considered. It meets the merge criteria and the merge continues to merge Cells #1, #6, #11, and #16. Only the first cell number is retained. After that, the top border of the merged cell is considered (the left and bottom borders have no adjacent cell). It does not meet the merge criteria because it connects to multiple adjacent cells (#2, #7, #12, and #19). The merged cell cannot grow further for the moment. The merge process is repeated with the other cells that have not been merged. Figure 5.9 shows the merge results after the first round of merging. Multiple rounds of merging are required to merge into larger cells. The merging process stops when the round achieves no merging. Figure 5.10 shows the merge results after the final round of merging. The cell and wall numbers are renumbered when writing the output files (input files for the ray tracing engine).

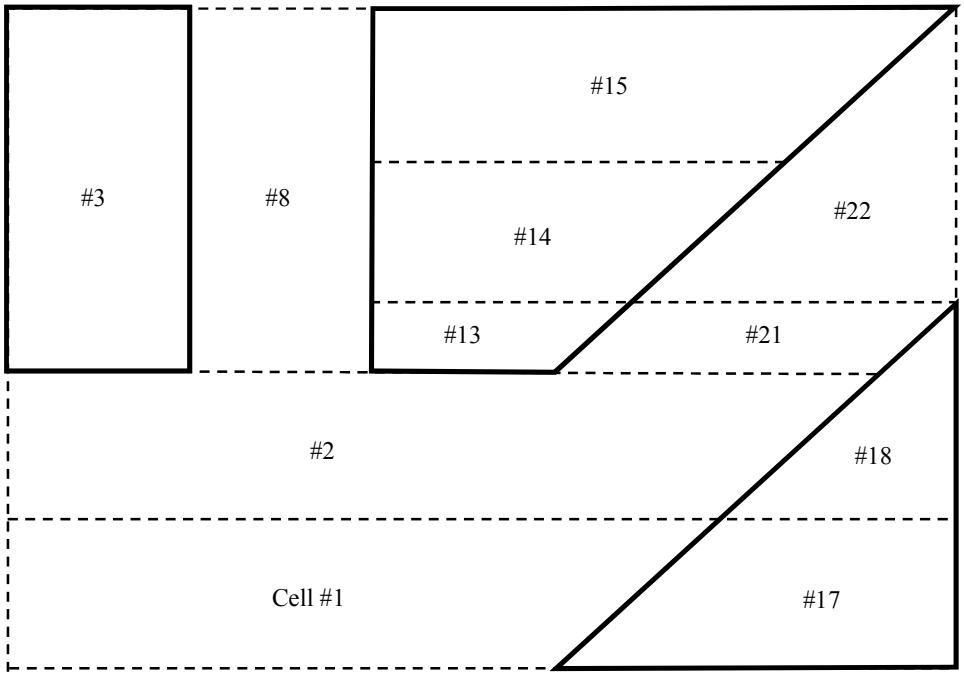


Figure 5.9 Cell-and-portal generation: first round of merging

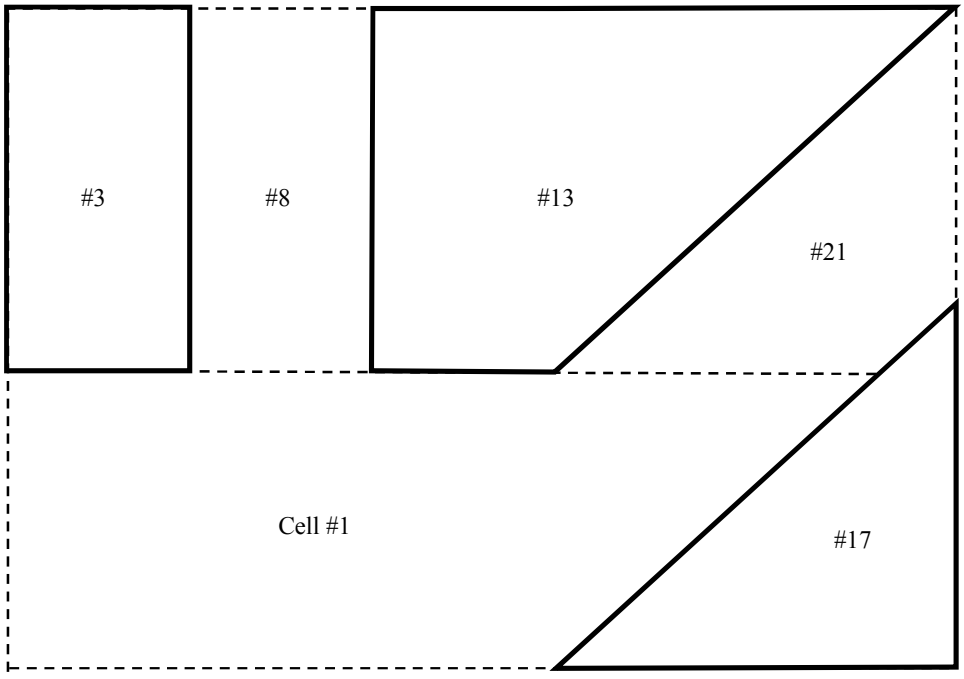


Figure 5.10 Cell-and-portal generation: merging convex cells (stage 3)

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 Introduction

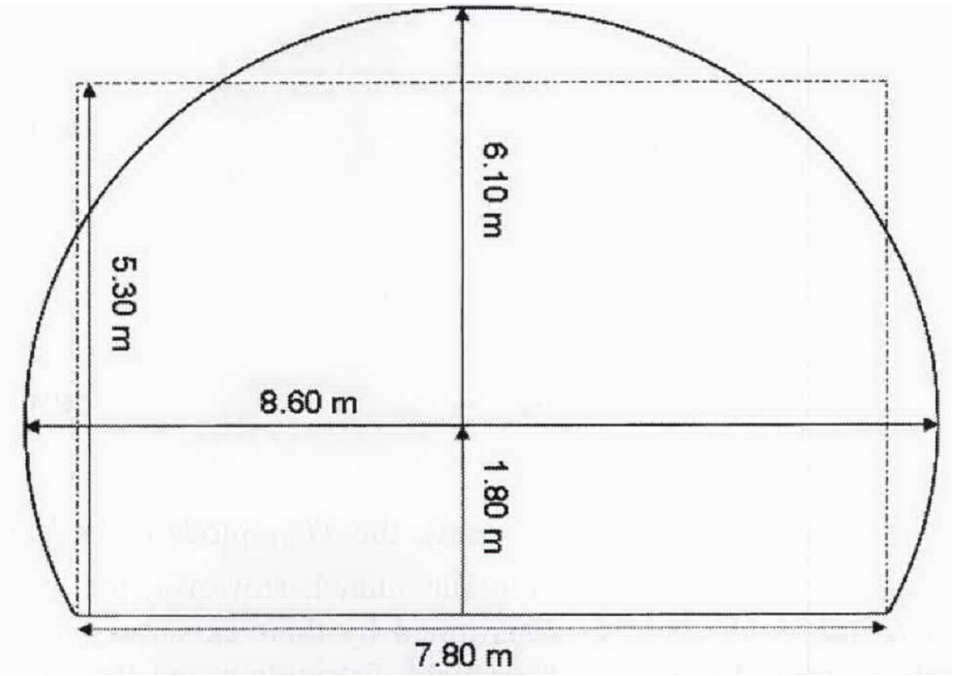
Two shooting-and-bouncing-polygon (SBP) ray tracers are implemented: one with binary space partitioning (SBP-BSP) and the other with convex cell partitioning (SBP-CCP). They are implemented in C/C++ programming language. In the following, their performance in terms of accuracy and efficiency is evaluated by comparing their simulation results against published measurements and simulation results from commercial REMCOM Wireless InSite®'s ray tracers and CST MICROWAVE STUDIO®'s full-wave solvers. An evaluation of the Big-O time and memory complexity of the SBP-CCP ray tracer is also presented at the end of this Chapter. All simulations are run on a laptop computer (Intel® CoreTM i7-5500U, 12 GB RAM, GeForce 840M) powered by the mains without GPU parallel computing unless specified otherwise. It is shown that the proposed SBP-CCP ray tracer is applicable to long tunnel, urban canyon, and indoor propagation environments. Also, it is superior to REMCOM Wireless InSite's 3D ray tracers in terms of accuracy, time efficiency, and memory efficiency, in those applications.

6.2 Massif Central Tunnel

We present here the application of the SBP ray tracers to a long straight tunnel, i.e. the Massif Central Tunnel in France. For a long tunnel, there are two challenges. Firstly, rays have large incident angles and hence reflection coefficients close to 1 at the tunnel's walls. A large number of reflections are required to achieve convergence in the ray tracing results. Secondly, at a distance, a tunnel wall subtends a small angle at the transmitter, which requires a very small ray separation to achieve convergence in the ray tracing results. General purpose ray tracers are generally not efficient at handling a large number of reflections and very small ray separation.

Figure 6.1(a) shows a photo of the simulated Massif Central Tunnel (Molina-Garcia-Pardo et al., 2012). The dimension details are illustrated in Figure 6.1(b). Also shown in Figure 6.1(b) is the equivalent rectangular tunnel model proposed by Dudley et al. (2007), which is valid only at sufficiently large distances from the transmitter. The equivalent rectangular tunnel model is used in the ray tracing simulation. The dielectric constant and conductivity of the tunnel walls and floor are 5 and 0.01 S/m, respectively, as proposed by Dudley et al. (2007). There are one transmitter and 1250 receivers (observation points). They are 2 m above ground and 1.95 m from one of the vertical walls of the equivalent rectangle. The first receiver is 2 m from the transmitter in the longitudinal direction. Subsequent receivers are placed at 2 m intervals up to 2500 m from the transmitter. The transmit power is +34 dBm. For 900 MHz, the transmitting and receiving antennas are +7 dBi horn antennas.

(a) Photo



(b) Dimension details

Figure 6.1 Massif Central Tunnel

For 450 MHz, the transmitting and receiving antennas are +2.15 dBi half-wave dipole antennas.

The ideal number of rays arriving at a receiver in a rectangular tunnel can be determined analytically and is given by $1+2N+2N^2$ where N is the maximum number of reflections. Our SBP ray tracers are able to capture all of them, i.e. they produce ideal reflection-only GO solutions for the rectangular tunnel. Figure 6.2 shows that, at large distances, the SBP simulation results (CCP and BSP versions) at 900 MHz are nearly identical to those presented by Fuschini and Falciasecca (2012) using a mixed rays-modes method. The simulation results are able to predict the pseudo-periodic fading pattern at large

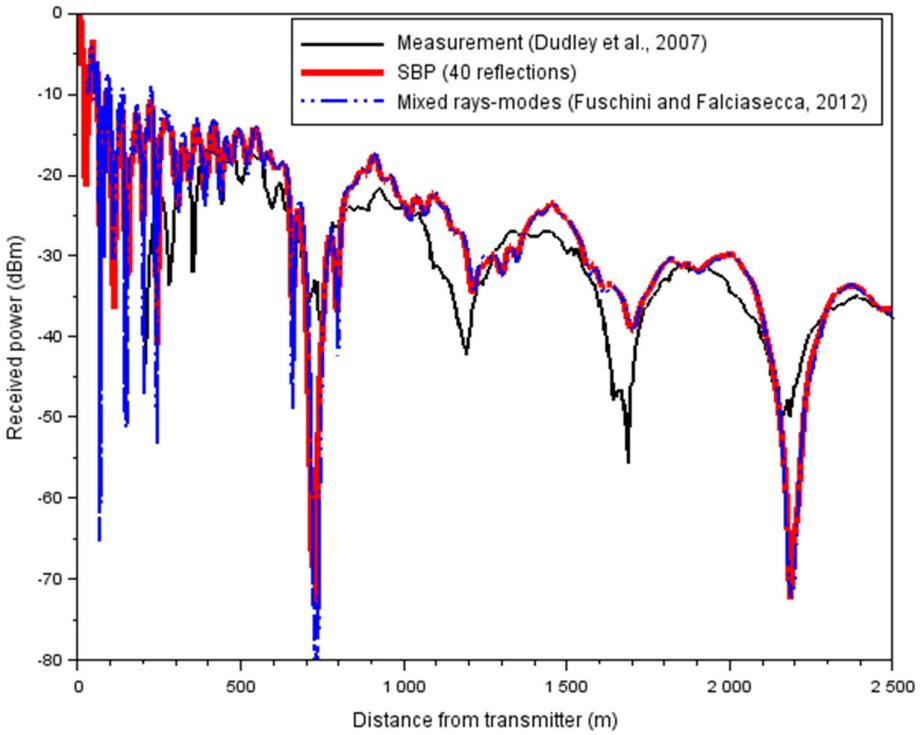


Figure 6.2 Received power at 900 MHz in Massif Central Tunnel

distances due to the lack of higher order propagation modes. They match fairly well with the measurements (Dudley et al., 2007) except that they predict much deeper fades at 720 m and 2200 m, excluding which the root-mean-square (rms) error is 4.4 dB for distances greater than 500 m. Similar observations are seen in Figure 6.3 at 450 MHz, except that a much larger number of reflections is required to achieve convergence in the ray tracing results due to the much higher attenuation. It is important to note that, in this application, the discrepancy of the ray tracing results from measurements is not due to the ray tracers' limitations as they produce ideal GO solutions. The discrepancy is due to the limitations of GO and the equivalent rectangle in modelling the real world semi-circular tunnel.

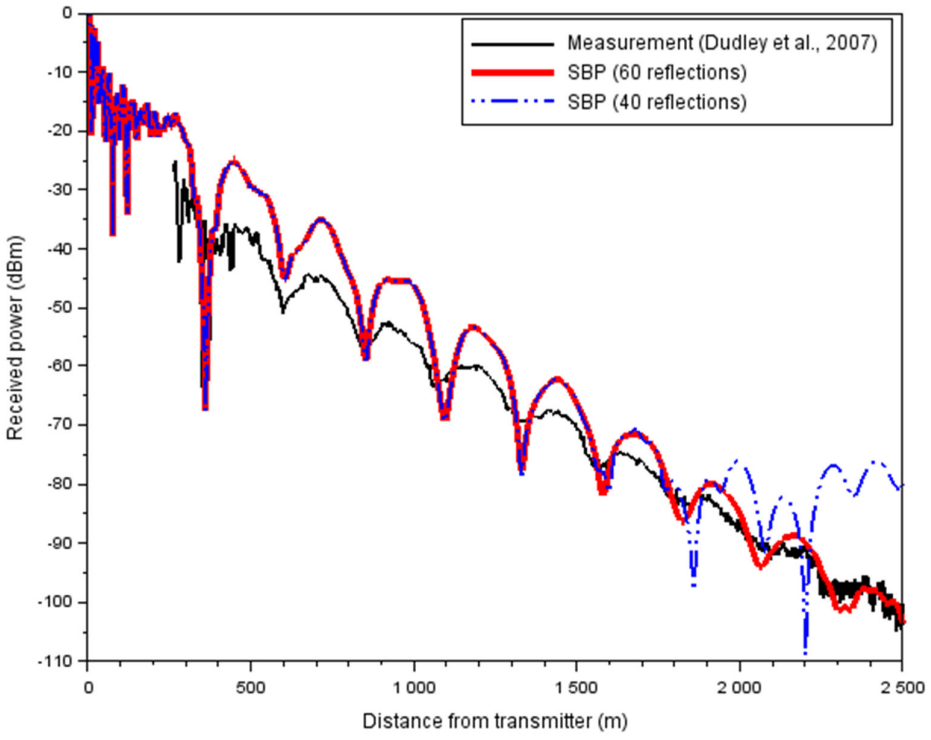


Figure 6.3 Received power at 450 MHz in Massif Central Tunnel

For the same simulation at 25 reflections, Zhang et al. (2016) reported 13.8 hours of computation time on a workstation using a 3D image-based ray tracer. Figure 6.4 shows the comparison between their simulation results and ours at 900 MHz. We have added an offset of +7 dB to bring their results to the correct level. Using our results (which have been shown above to be ideal GO solutions) as a benchmark, the discrepancy (after the offset), though small, shows that their ray tracer did not manage to capture all the rays.

At 25 reflections, Wireless InSite's (a commercial suite) X3D and Full 3D (F3D) ray tracers take 342 s (with GPU) and 273 s, respectively, but they

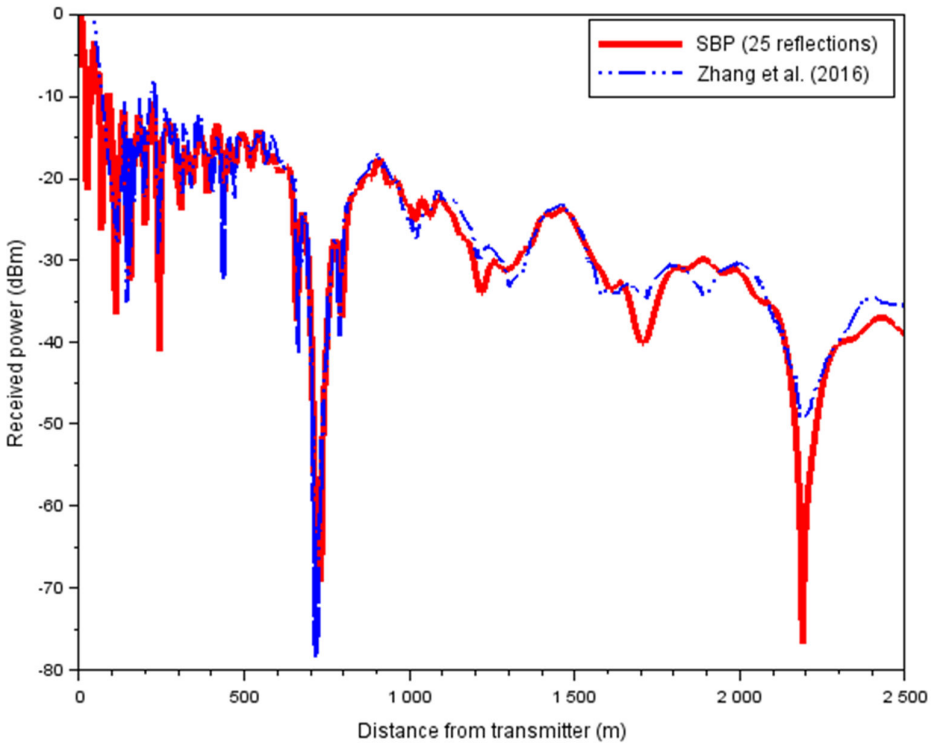
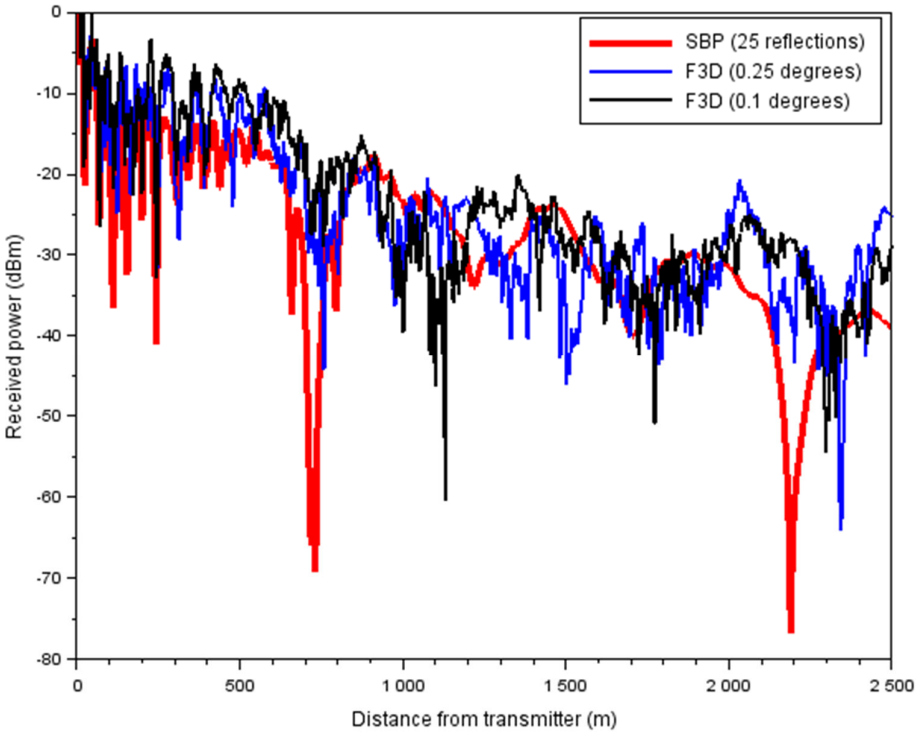


Figure 6.4 Zhang et al. (2016) ray tracing results for Massif Central Tunnel at 900 MHz

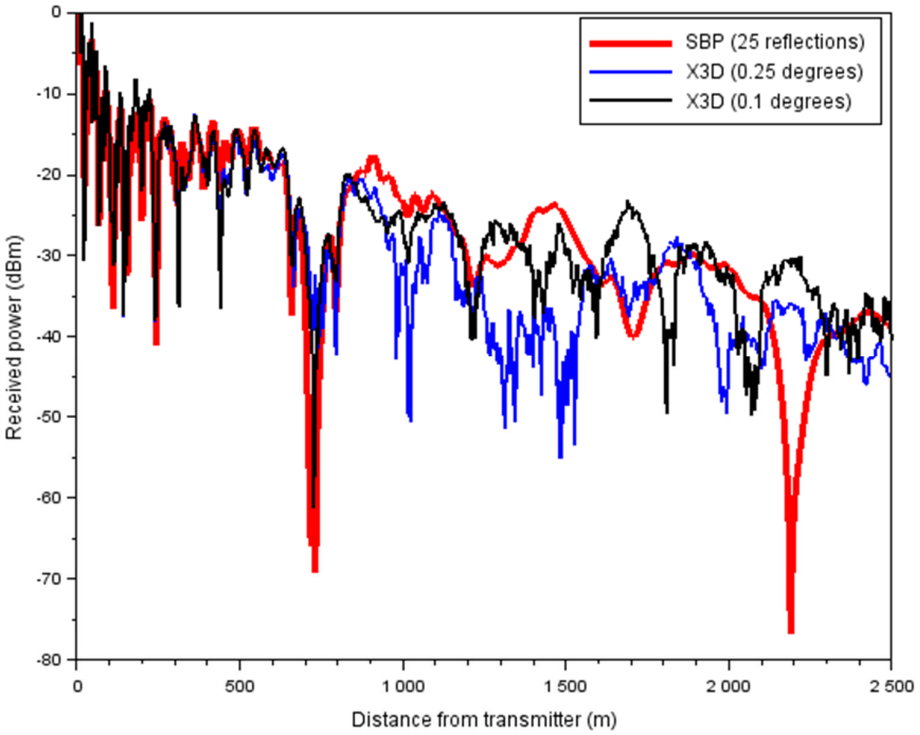
fail to predict the pseudo-periodic fading pattern and the results vary with the choice of ray separation. Figure 6.5 shows the comparison of their simulation results and ours. F3D results differ substantially from the ideal GO solutions, likely due to inexact ray paths. They also show no sign of convergence at 0.25 degrees ray separation (the default setting). X3D makes the extra effort to compute exact ray paths and hence its results better match the ideal GO solutions over the range where its results converge, i.e. up to about 800 m from the transmitter. The convergence range should increase with smaller ray separation but we did not try further due to the long simulation time.

How long does it take for the SBP ray tracers to compute the ideal GO solutions? At 25 reflections, the CCP version takes 14 s and the BSP version takes 15 s. That is about 3500 times faster than that reported by Zhang et al. (2016) and 20 times faster than Wireless InSite's ray tracers. The 14 s are mainly for computing ray paths and fields. The SBP process (image generation) takes less than a second. The SBP ray tracers are not only more accurate and more time efficient, they are also more memory efficient. They have a small memory footprint because they do not require images or ray paths to be stored for field computation. Field contributions are computed on the fly as individual image and ray path are generated and discarded. A summary of the comparisons is given in Table 6.1.

So far, the comparisons are between general purpose ray tracers. A better, and probably the best, ray tracer for the rectangular tunnel is a special purpose image-based ray tracer whereby images are determined analytically, exploiting



(a) F3D



(b) X3D

Figure 6.5 Wireless Insite's ray tracing results for Massif Central Tunnel at 900 MHz

Table 6.1 Ray tracers' accuracy, computation time, and memory consumption for a rectangular tunnel environment with 25 reflections

Ray tracer	Accuracy [1]	Time (s)	Memory (MB)
Zhang et al. (2016)	Good	49700 [2]	Not reported
F3D	Poor	273	143
X3D	Poor	342 [3]	2600
SBP-CCP	Ideal	14	1.2
SBP-BSP	Ideal	15	1.2
Analytical	Ideal	11	1.0

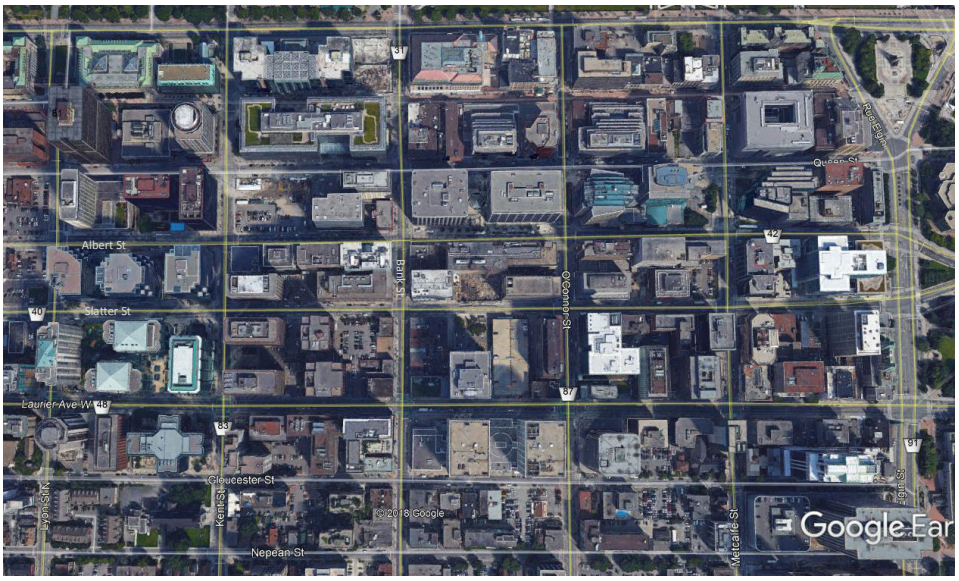
[1] relative to ideal GO solution
[2] run on a different workstation
[3] run with GPU parallel computing

the fact that the tunnel is rectangular. Compared to the special purpose analytical ray tracer, the main overhead of the SBP ray tracers is the reception test because the SBP process is found to be negligible for rectangular tunnel. On the other hand, the analytical ray tracer has ambiguity about whether the vertical or horizontal wall is intersected depending on the receiver location, i.e. every segment of a ray path requires computation of 2 line-plane intersections instead of one. We have implemented the analytical ray tracer. The fact that the tunnel is rectangular is exploited. The analytical ray tracer takes 11 s for 25 reflections, just a few seconds faster than the SBP ray tracers. The simulation results are identical to those of the SBP ray tracers. Hence, unlike other general purpose ray tracers, the SBP ray tracers are as good as the special purpose analytical ray tracer both in terms accuracy and efficiency.

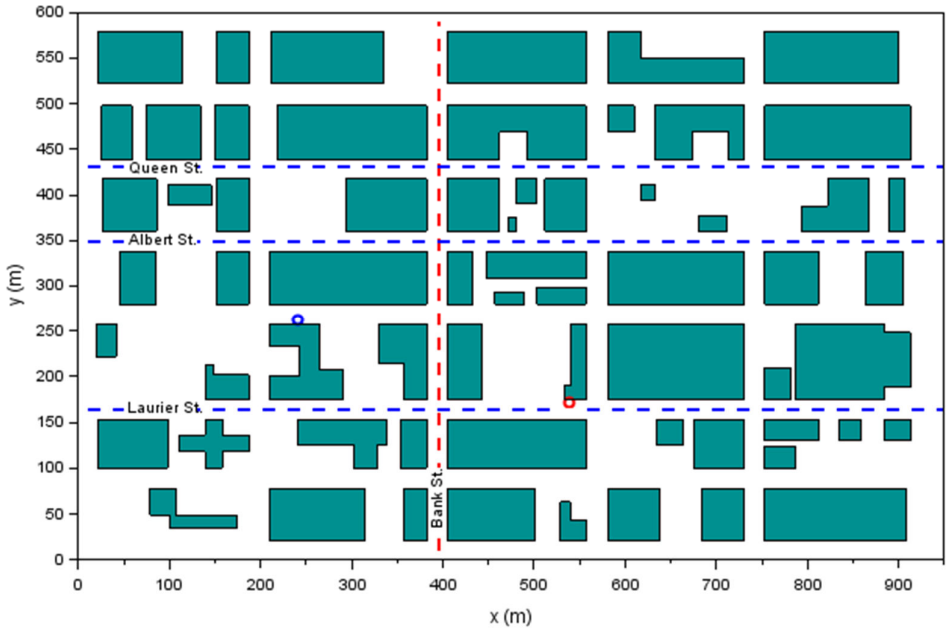
In this Section, both versions of the SBP ray tracers are equally good. The improvements due to CCP and improved diffraction calculation are yet to be seen. The comparisons, however, have highlighted the strengths of SBP and justified the significance of our work to improve an SBP ray tracer.

6.3 Ottawa City Streets

We present here the application of the ray tracers to Ottawa city streets. The measurements are reported by Whitteker (1988). Figure 6.6(a) shows an aerial view of the simulated Ottawa city streets obtained from Google Earth. The buildings may have changed but the street layout and the fact that the city streets mainly consist of high rise buildings remain the same. Figure 6.6(b) shows the model layout (Tan and Tan, 1995) used in the ray tracing simulations. Four receiver routes exhibiting urban canyon characteristics, where over-roof-top diffraction is not significant, are simulated. They are routes along Bank Street, Laurier Street, Albert Street, and Queen Street, shown as dashed lines in Figure 6.6(b). The transmitter for the Bank Street route is located at (538, 171, 8.5), marked as red circle in Figure 6.6(b). The transmitter for the other routes is located at (240, 263, 8.5), marked as blue circle in Figure 6.6(b). The transmitting and receiving antennas are modelled as vertical half-wave dipoles. The net measurement system gain, i.e. antenna gains plus system losses, is assumed to be 0 dB. The operating frequency is 910 MHz. As suggested by Tan and Tan (1995), the dielectric constant and conductivity of the walls are 7 and



(a) Aerial view



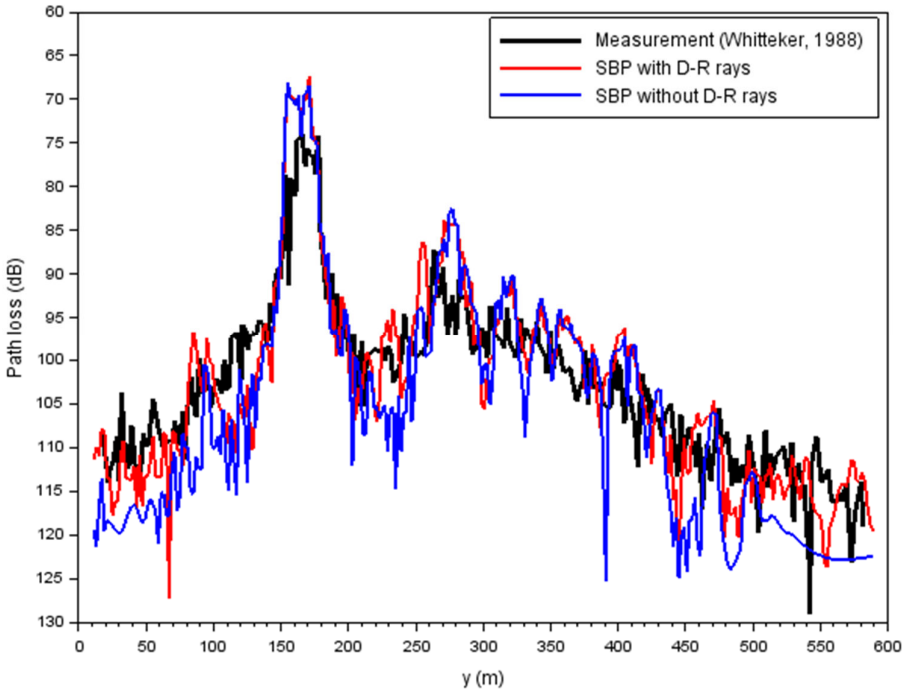
(b) Model layout

Figure 6.6 Ottawa city streets

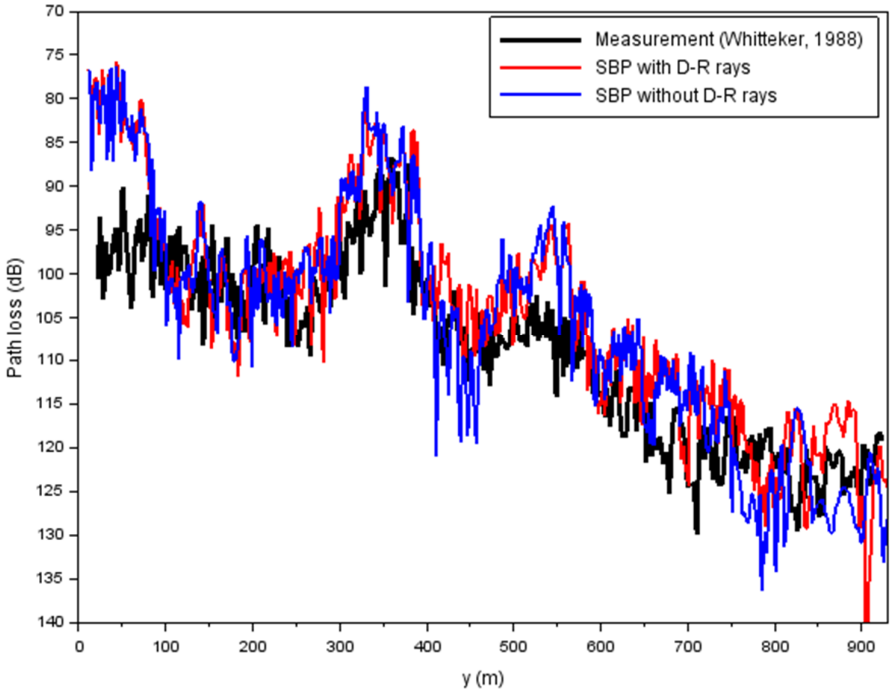
0.2 S/m, respectively. The simulations include up to 10 reflections and 1 diffraction. Over-roof-top diffraction is not included.

Figure 6.7 shows the comparisons of SBP simulation results, with and without diffracted-reflected (D-R) rays, and Whitteker's measurements. The significance of diffracted-reflected rays is seen in receiver routes along Bank Street, Albert Street, and Queen Street where diffracted-reflected rays have improved the accuracy of the ray tracing results in deeply shadowed regions by about 10 dB. Unlike the rectangular tunnel, the SBP simulation time for the urban canyon is dominated by the SBP process (about 90%). SBP-CCP takes about 108 s for one receiver route whereas SBP-BSP takes about 940 s. In other words, the CCP version is more than 8 times faster than the BSP version when applied to an urban canyon with up to 10 reflections. The computation gain factor is higher with a smaller number of reflections, e.g. at 4 reflections, the computation gain factor is 20 (2 s versus 40 s).

The ray tracing results do not do well in some areas, e.g. $y < 60$ m in Laurier Street and Queen Street. Also, the measurements have shown strong signals injected into the parallel streets, i.e. Albert Street and Queen Street, from Metcalfe Street ($y \approx 750$ m) but this is not seen in the ray tracing results. Although such rays from Metcalfe Street exists in the ray tracing results, they are not significantly stronger than rays propagated along the parallel streets. These errors are also seen in Wireless InSite's ray tracing results (UC, F3D and X3D) shown in Figure 6.8. They are likely due to inadequacy of the building models.

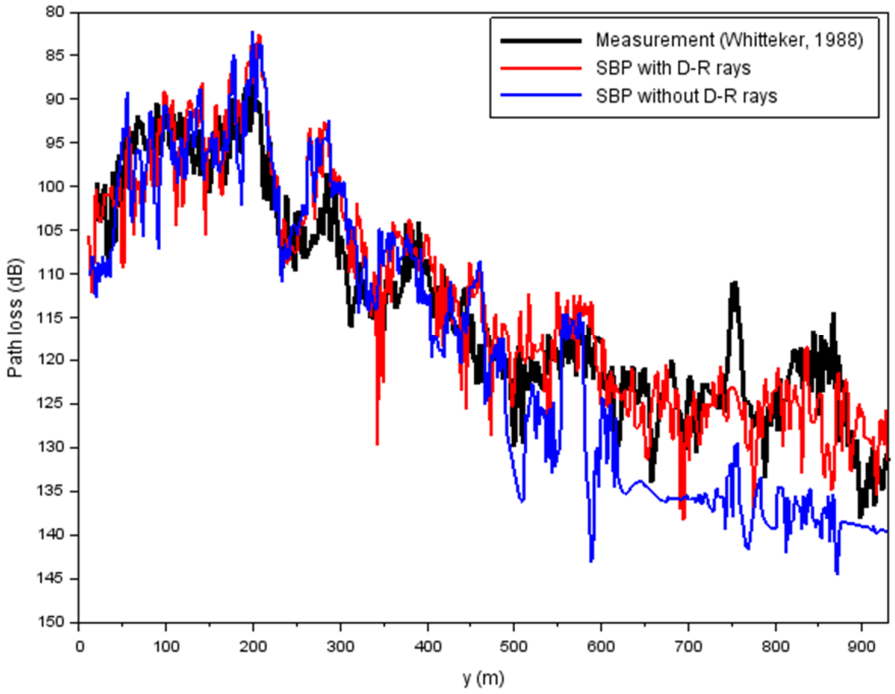


(a) Bank Street

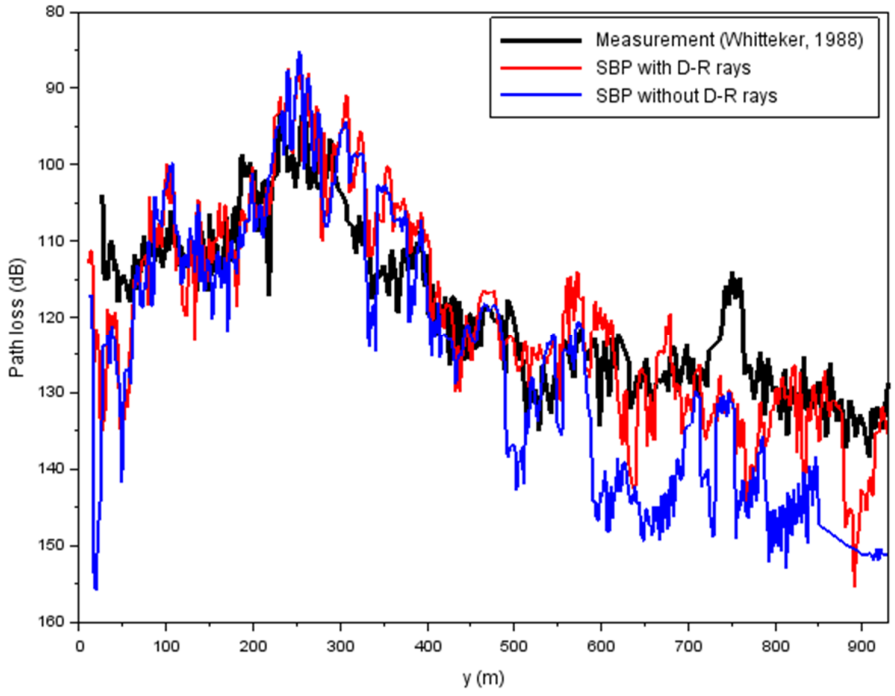


(b) Laurier Street

Figure 6.7 SBP simulation results for Ottawa city streets

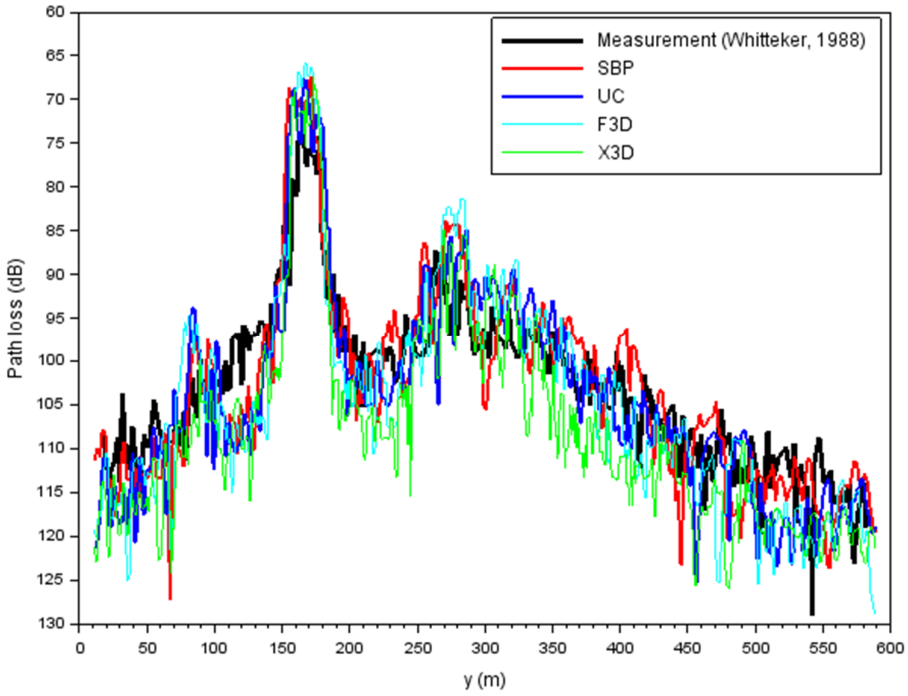


(c) Albert Street

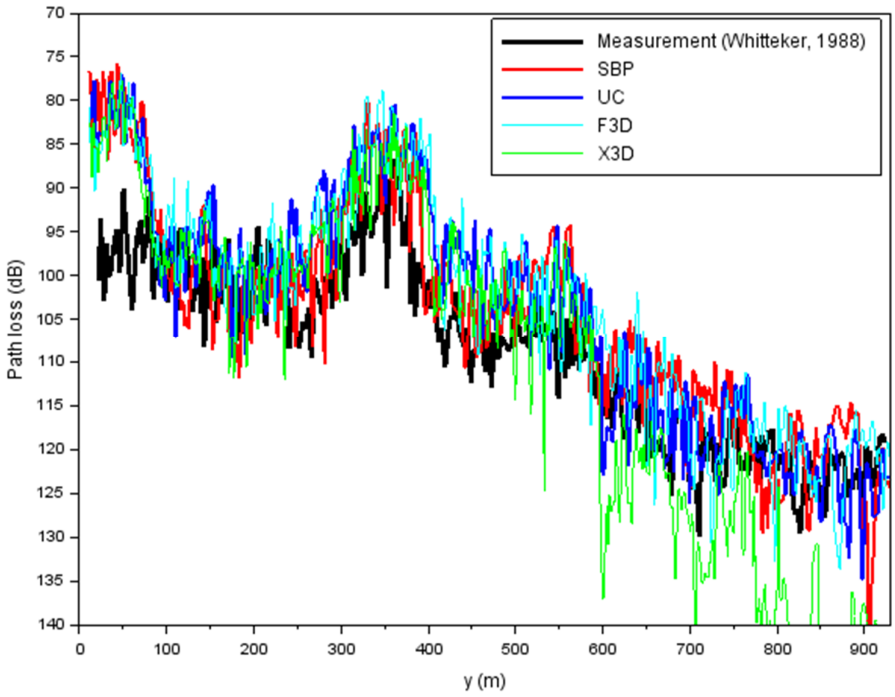


(d) Queen Street

Figure 6.7 SBP simulation results for Ottawa city streets

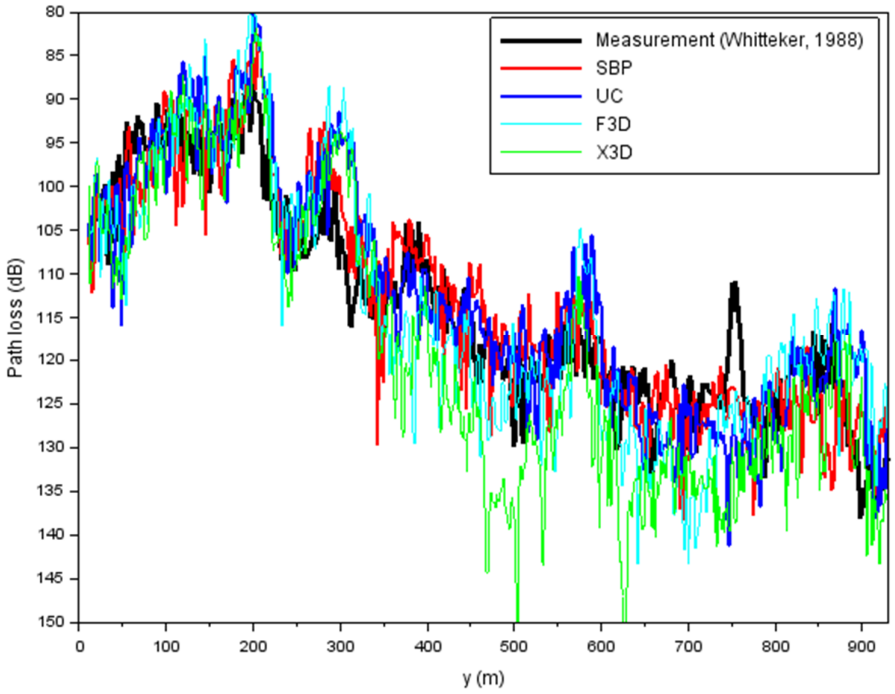


(a) Bank Street

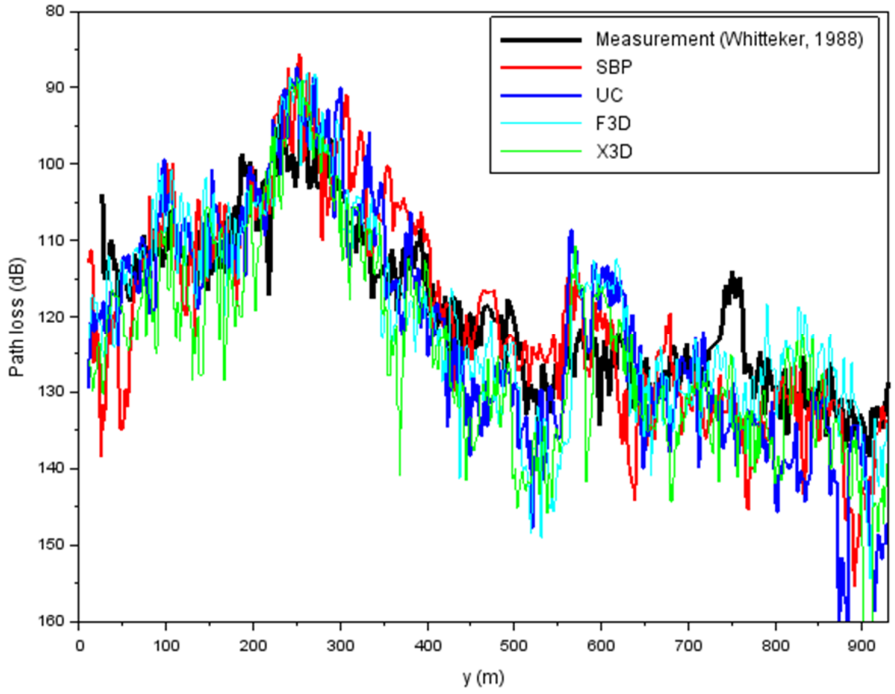


(b) Laurier Street

Figure 6.8 Wireless InSite simulation results for Ottawa city streets



(c) Albert Street



(d) Queen Street

Figure 6.8 Wireless InSite simulation results for Ottawa city streets

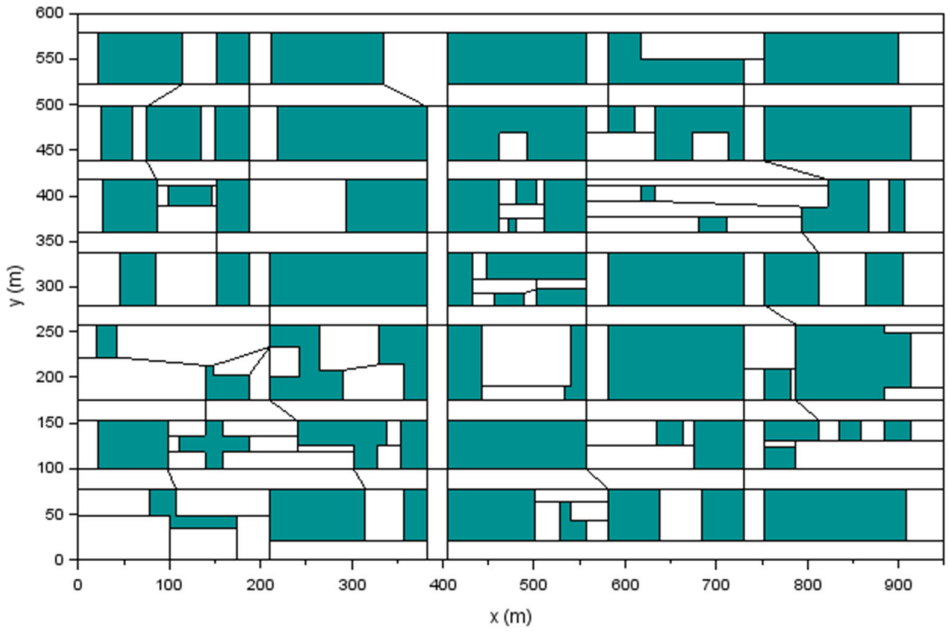
Figure 6.8 shows that Wireless InSite's Urban Canyon (UC) and F3D results are similar to SBP results. The X3D results tend to underestimate the signal levels in shadowed regions. Table 6.2 shows the comparisons in terms of rms error, computation time, and memory consumption. Accuracy wise, X3D does not do well. It may have omitted too many ray paths in compensating the extra effort on computing exact ray paths. SBP-CCP, UC, and F3D are comparable considering the large degree of uncertainties in propagation modelling, e.g. position and dimension errors, idealized building model, and omission of small details. Speed wise, SBP-CCP is not as efficient as the 2D ray tracer (UC) but it is better than the 3D ray tracers (F3D and X3D). Memory wise, again, SBP-CCP has shown very small memory footprint, orders of magnitude smaller than Wireless InSite's ray tracers'.

In the above comparisons, the convex cells for CCP have been defined manually as part of the model description. The convex cells can be generated automatically from a set of predefined walls using the convex cell generator described in Section 5.5 (p. 77). The convex cell generation process takes about 1 s. The SBP-CCP simulation time is about 175 s for one receiver route using the auto-generated convex cells, compared to 108 s using the manually defined convex cells. Although slower, the run time is still significantly faster than SBP-BSP, F3D, and X3D (if run without GPU). Figure 6.9 shows the manually defined and auto-generated convex cells in 2D. The slower run time is due to these reasons (areas for future improvements):

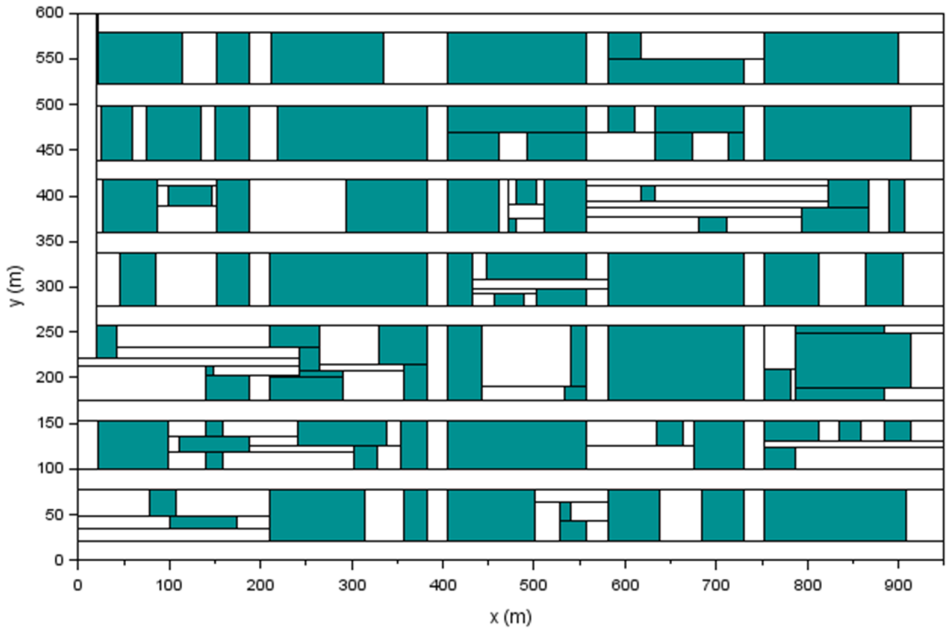
Table 6.2 Ray tracers' rms error, computation time, and memory consumption for Ottawa city streets

Route	Rms error (dB)				Time (s) [1]				Memory (MB)			
	UC	F3D	X3D	SBP-CCP	UC	F3D	X3D	SBP-CCP	UC	F3D	X3D	SBP-CCP
Bank St.	5.8	6.3	7.1	5.4	19	565	101	108	55	193	727	1.2
Laurier St.	7.7	7.9	11.5	7.5	20	759	101	107	71	215	1085	1.4
Albert St.	6.6	7.7	9.4	5.6	21	767	103	103	71	215	883	1.4
Queen St.	8.7	7.2	9.8	7.5	17	739	80	98	71	215	863	1.4

[1] X3D is run with GPU parallel computing



(a) Manual mode



(b) Auto mode

Figure 6.9 Convex cell partitioning for Ottawa city streets

1) In the auto mode, the buildings are split into convex cells which is not necessary for urban canyon simulations. The splits create more walls.
2) There are more portals (virtual walls) in the auto mode partly because the portal connecting two cells is always aligned to the coordinate axes.
3) In the auto mode, there are big convex cells consisting of many walls. Numerical experiments show that efficiency improves if the big convex cells are split into smaller cells with fewer number of walls, as it is done in the manual mode. In the current implementation, this can be achieved in the auto mode by adding proper portals to the set of predefined walls. The merging process will not merge across the predefined portals avoiding the formation of big cells. We are able to attain a run time of 135 s with this technique.

In this Section, we have shown the significance of diffracted-reflected rays in deeply shadowed regions. Diffracted-reflected rays are important additions to an SBP ray tracer, made by diffraction ray-polygon proposed in this thesis (CHAPTER 4). We have shown the considerable improvement made by CCP proposed in this thesis (CHAPTER 5). We have also shown that the resultant SBP-CCP ray tracer is a good candidate for simulating urban canyons when a 3D ray tracer is preferred, e.g. some building walls are sloped, or when a very small memory footprint is desired.

6.4 The Printing House at Trinity College Dublin

In this Section, we show the application of SBP-CCP to an indoor environment. Figure 6.10 shows the simulated indoor scene, provided by Kenny and Nuallain (2017). Blue dots are the receiver points. Narrow rectangles shaded in blue are wooden doors. The other rectangles shaded in blue are glass windows. The transmitter is at (5.468, 6.51, 1.45). The ceiling height is about 3 m. The measurements are reported by Kenny and Nuallain (2017). At each of the receiver points, 5 power measurements, from 5 random points within one-wavelength radius from the receiver point, are taken and averaged. The operating frequency is 900 MHz.

In the ray tracing simulation, we have used dielectric constants and conductivities suggested by Kenny and Nuallain (2017). The exterior walls with windows, the floor, and the ceiling are modelled as half-spaces with dielectric

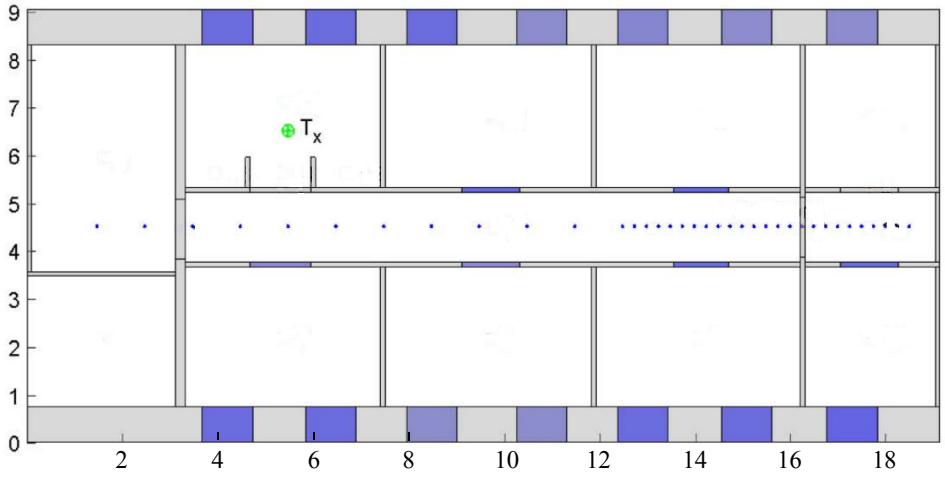


Figure 6.10 Top floor of the printing house at Trinity College Dublin

constant 4.44 and conductivity 0.08 S/m. Thin interior walls are modelled as lossy slabs with dielectric constant 4, conductivity 0.04 S/m, and thickness 15 cm. Thick interior walls are modelled as lossy slabs with dielectric constant 4, conductivity 0.04 S/m, and thickness 30 cm. Wooden doors are modelled as lossless slab with dielectric constant 2.3 and thickness 4 cm. We have included up to 6 reflections / transmissions and 1 diffraction. No significant change in the simulation results is observed with a higher number of interactions.

At each of the receiver points, one receiver is placed on the receiver point and four additional receivers are placed at the corners of a square centered at the receiver point. The sides of the square are parallel to the axes. The four receivers are one-wavelength from the center. The simulated power at the five receivers are averaged. Kenny and Nuallain (2017) have not stated the receiver height, the transmit power, and the measurement system gain or loss. The receiver height is assumed to be about the same as the transmitter height, i.e. 1.45 m above the floor. We found a transmit power of 0 dBm and a system gain of 0 dB produce simulation results that match reasonably well with the measurements, as shown in Figure 6.11. Our simulation results seem to be better than Kenny and Nuallain (2017) simulation results near the line-of-sight region. However, the overall rms error is about the same, 5.6 dB and 5.8 dB, respectively. Our simulation takes about 17 s for 180 receivers (36 × 5). Kenny and Nuallin (2017) have not reported the run time.

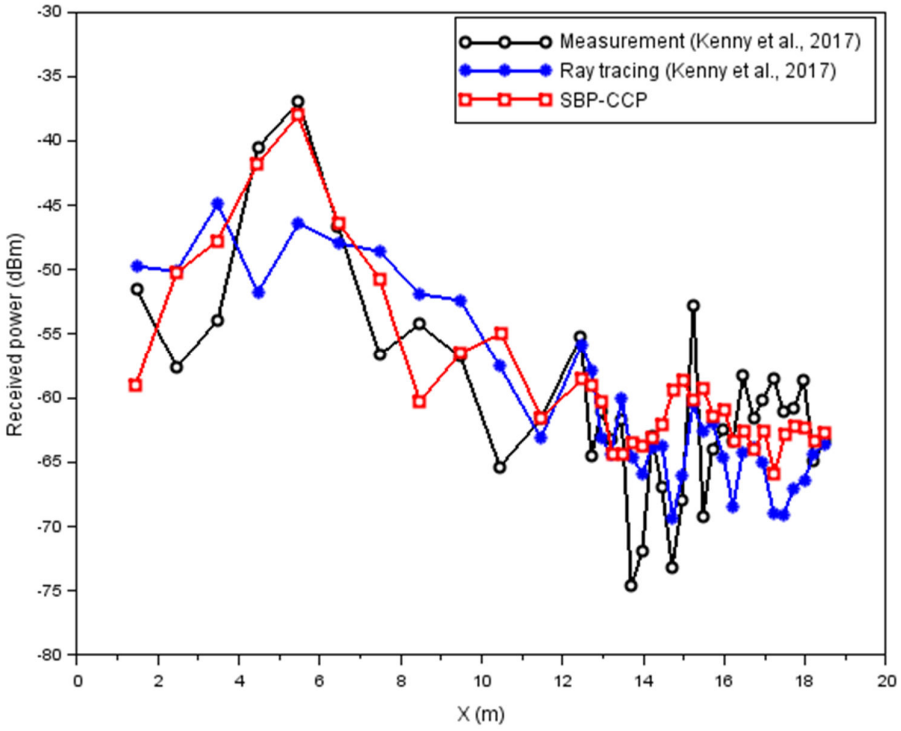


Figure 6.11 Received power at the top floor of the printing house at Trinity College Dublin

6.5 Comparison with Full-Wave Solutions

It is difficult to assess or compare the accuracy of ray tracers using propagation measurements as the benchmark due to the large degree of uncertainties in propagation modelling to the extent that the model parameters need to be calibrated with measurements to achieve reasonable match. A good alternative is to compare against full-wave solutions. It greatly reduces the uncertainties and avoids the need for parameter calibration. Without the uncertainties and curve fitting, the comparison is more demanding. It does a better job in revealing formulation and implementation flaws. Its limitation is

the simulated scenes cannot be too big (due to the limitation of full-wave simulation software) and thus the simulated scenes must be relatively simple. However, this is not an issue because few ray tracers are able to match full-wave solutions even for simple scenes.

We simulate the indoor scene in Figure 6.12. Its complexity is limited by the contradicting requirements of full-wave and GO/UTD solvers. Full-wave solvers require that the overall dimensions should not be too big electrically. On the other hand, GO/UTD solvers require that objects should be electrically large. Although simple, the indoor scene does involve complex 3D interactions of reflection, transmission, and diffraction. It requires accurate treatment of walls with thickness and transmissive hollow wedge. Ray tracers based on rectangular geometry (Kenny and Nuallain, 2017) may have problem with the non-axis-aligned wall. We believe the indoor scene is the most complex of its kind to date. Simpler scenes for comparisons between ray tracing and full-wave solutions are found in Chen and Jeng (1997); Yang, Wu and Ko (1998); Teh, Kung and Chuah (2006).

The 3D full-wave simulation is done using CST Time Domain solver. The indoor scene has an interior height of 3 m. The exterior wall, floor, and ceiling have a thickness of 40 cm. Their exterior surfaces are bordered by open boundaries (perfectly matched layers) of the full-wave solver. The interior walls have a thickness of 10 cm. All walls have dielectric constant $\varepsilon_r = 5$ and conductivity $\sigma = 0.01$ S/m, typical of concrete walls. The transmitting antenna is a *z*-polarized half-wave dipole antenna at (4.4, 1.4, 2.4), marked as a red circle

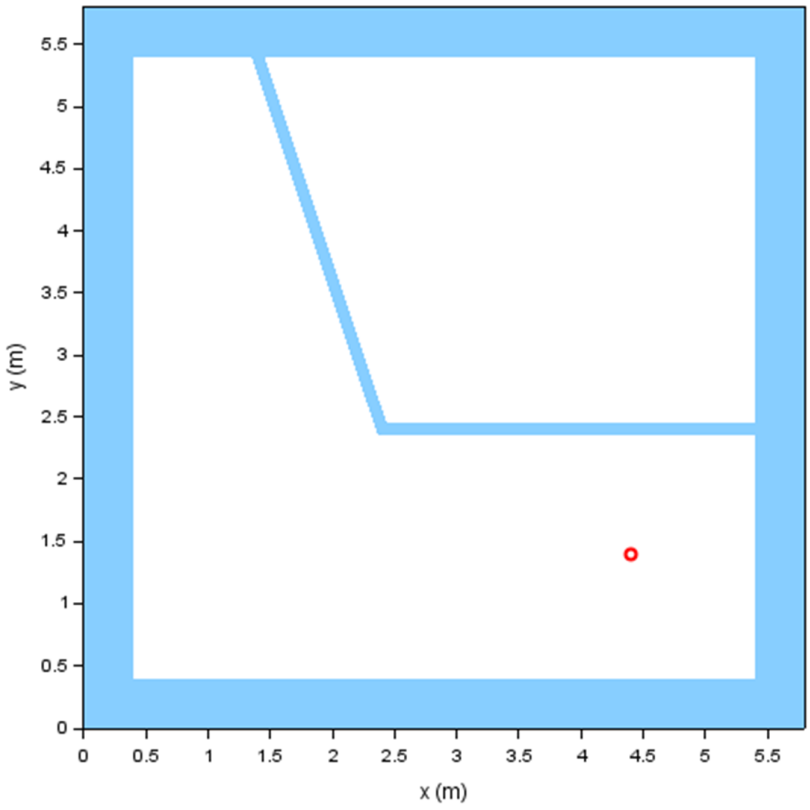


Figure 6.12 An indoor scene with tapered corridor and transmissive hollow wedge

in Figure 6.12. A field monitor is placed at z = 1.4 m (1 m above the floor). The operating frequency is 1.5 GHz. Maximum mesh cell size is 1/20 wavelength.

The ray tracing simulation includes up to 6 reflections / transmissions, one diffraction, and any combination of them. The exterior walls, floor, and ceiling are modelled as half spaces. The interior walls are modelled as lossy dielectric slabs. The plane at z = 1.4 m is covered by 99 × 99 = 9801 receivers at a resolution of 5 cm, e.g. the first receiver is at (0.45, 0.45, 1.4). For ideal comparison with full-wave solutions, the receivers should be merely field monitors that are not associated with antennas. In cases where a receiver must

be associated with an antenna, a dipole antenna may be used; the difference in relative electric field intensity is very small.

Figure 6.13 and Figure 6.14 shows the full-wave and SBP-CCP results for z-polarized relative electric field intensity at z = 1.4 m, respectively. There is a very good match between the results. The rms error is 2.1 dB. The full-wave solver takes more than 5 hours on a high-end workstation. The SBP-CCP ray tracer takes 62 seconds on the laptop computer, mainly to compute ray paths and fields for the large number of receivers, about 6 ms per receiver. The SBP process (image generation) takes less than a second. The inclusion of diffraction into the hollow wedge increases the run time to 90 s, but the rms error only improves slightly to 2.0 dB.

Figure 6.15 and Figure 6.16 shows the F3D and X3D results, respectively. Visually, the X3D results are more accurate than the F3D results due to the extra effort to compute exact ray paths. X3D is able to show fine variations in the field pattern. However, it does not do well in point-to-point comparison. X3D rms error is greater than F3D rms error. Table 6.3 shows the comparisons in terms of rms error, computation time, and memory consumption. It is evident that our SBP-CCP ray tracer is faster, more accurate, and more memory efficient than Wireless InSite's ray tracers in this indoor application. It is also shown that the delay corrections proposed in CHAPTER 3 have significantly improved the accuracy (from 4.1 dB to 2.1 dB in rms error) without compromising computation complexity (only a few seconds increase in run time).

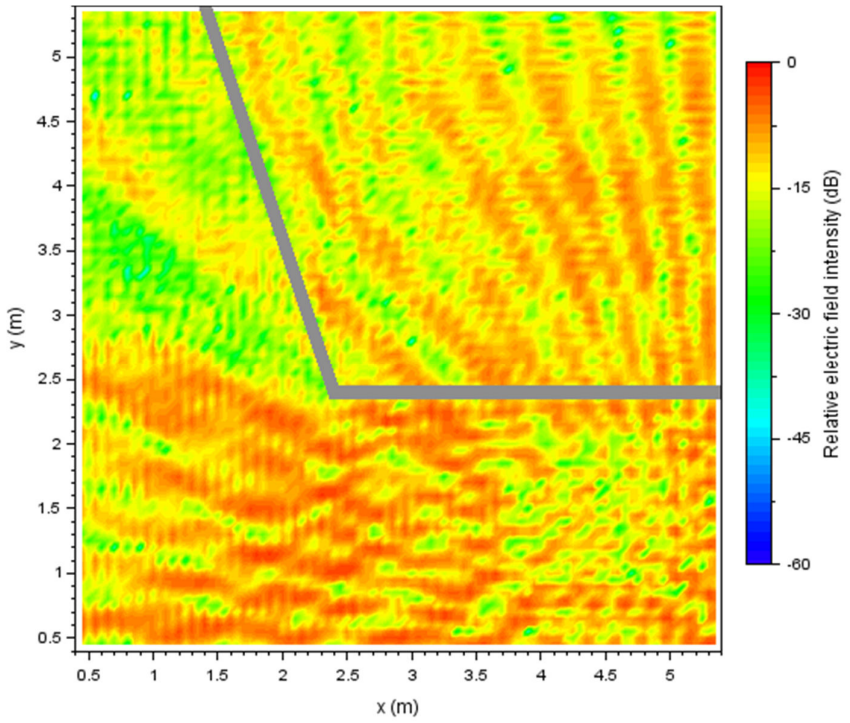


Figure 6.13 CST full-wave results for Figure 6.12

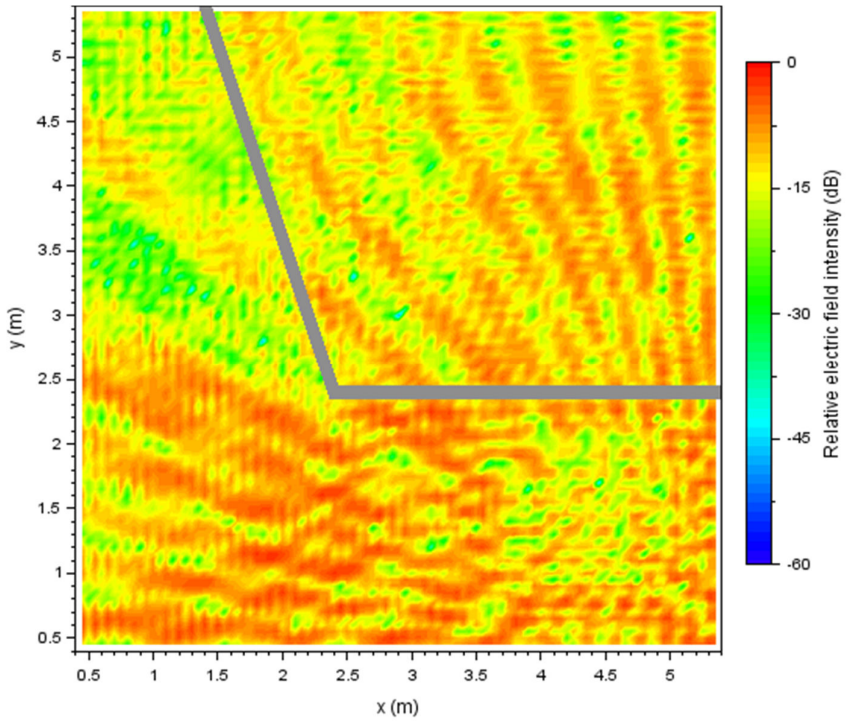


Figure 6.14 SBP-CCP simulation results for Figure 6.12

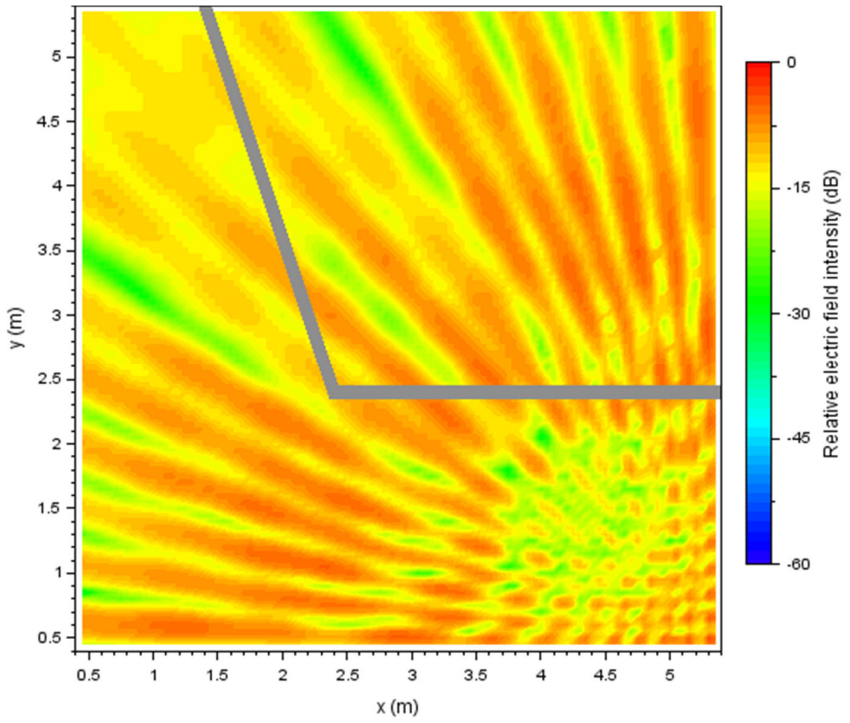


Figure 6.15 F3D simulation results for Figure 6.12

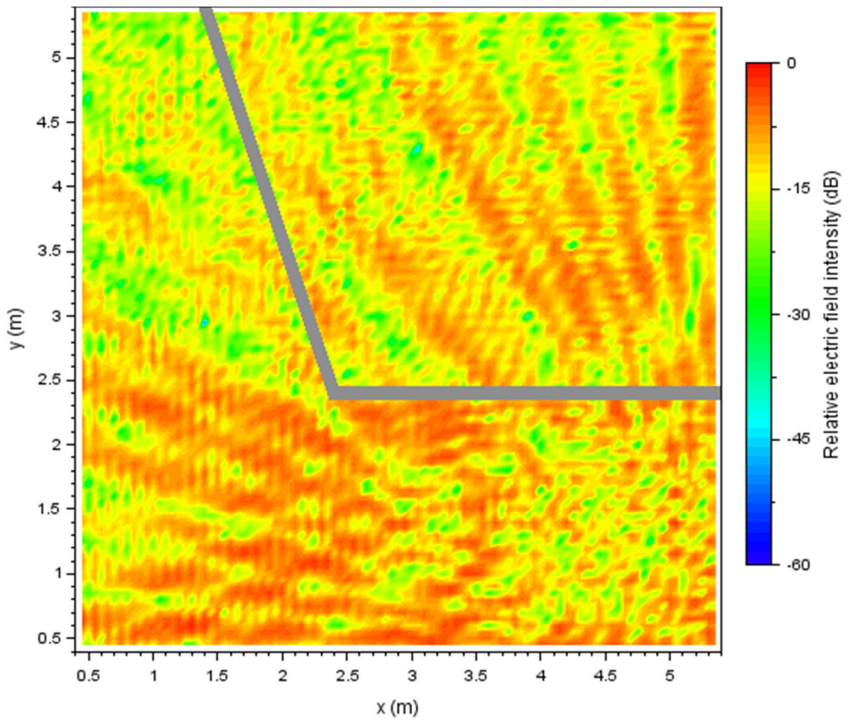


Figure 6.16 X3D simulation results for Figure 6.12

Table 6.3 Ray tracers' rms error, computation time, and memory consumption for the indoor scene in Figure 6.12

Ray tracer	Rms error (dB) [1]	Time (s)	Memory (MB)
F3D	4.6	8773	614
X3D	5.5	1244 [2]	6686
SBP-CCP	2.1	62	5.0
SBP-CCP (with diffraction into hollow wedge)	2.0	90	5.0
SBP-CCP (without delay correction)	4.1	58	5.0

[1] relative to full-wave solutions
[2] run with GPU parallel computing

Next, we simulate a concrete rectangular room having a 1 m × 1 m perfectly conducting column in the middle of the room, as shown in Figure 6.17. The transmitter is at (4.4, 1.4, 1.9). Other simulation settings are similar to those of Figure 6.12. This time we show the field distribution on a vertical plane at $x = 1.4$ m, see Figure 6.18 and Figure 6.19. Again, there is a very good match between the SBP results and full-wave solutions. The rms error is 2.0 dB.

One characteristic of the ray tracing simulations presented in this Section is there is a large number of receivers. Shooting-and-bouncing-ray (SBR) ray tracers are generally regarded as more time efficient than image-based ray tracers when the number of receivers is large. This is because the expensive SBR process is independent of receiver. The same set of rays are shared by all receivers. On the other hand, the image method computes unique rays for each receiver. Furthermore, in many image-based ray tracers, many of the computed rays may turn out to be invalid. Our SBP-CCP ray tracer also uses the image

method to backward trace unique exact rays for each receiver, which is necessary for accurate ray tracing simulation. However, the backward tracing process is very efficient. Firstly, there is no ambiguity in the sequence of ray-object interactions. This greatly reduces the number of ray-object intersection computation. Secondly, the number of invalid ray paths is greatly reduced by SBP. Despite the large number of receivers, Table 6.3 shows that our SBP-CCP ray tracer is still significantly faster than commercial SBR ray tracers.

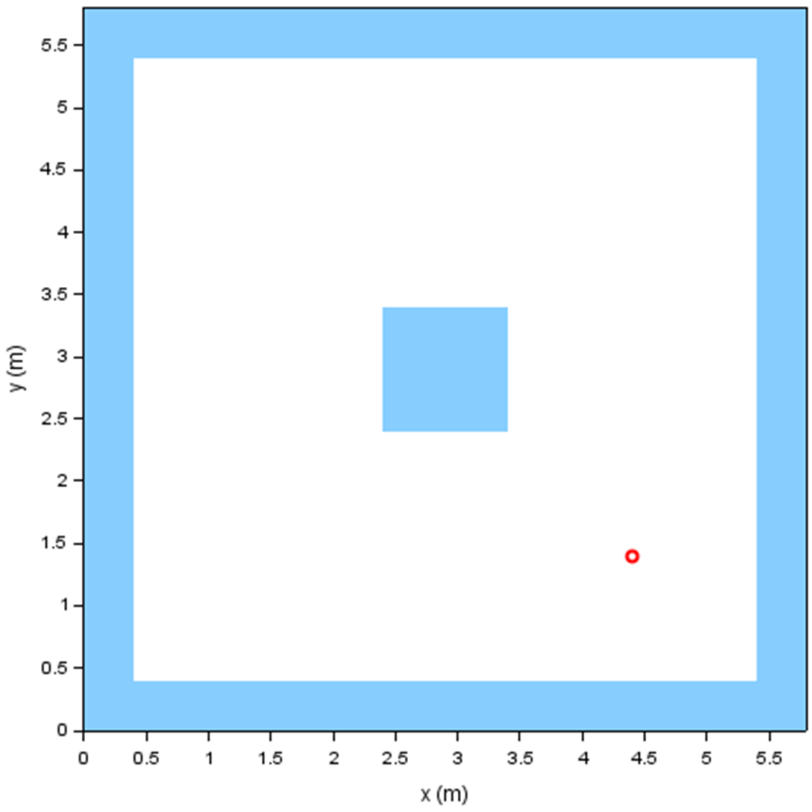


Figure 6.17 A rectangular room with perfectly conducting column

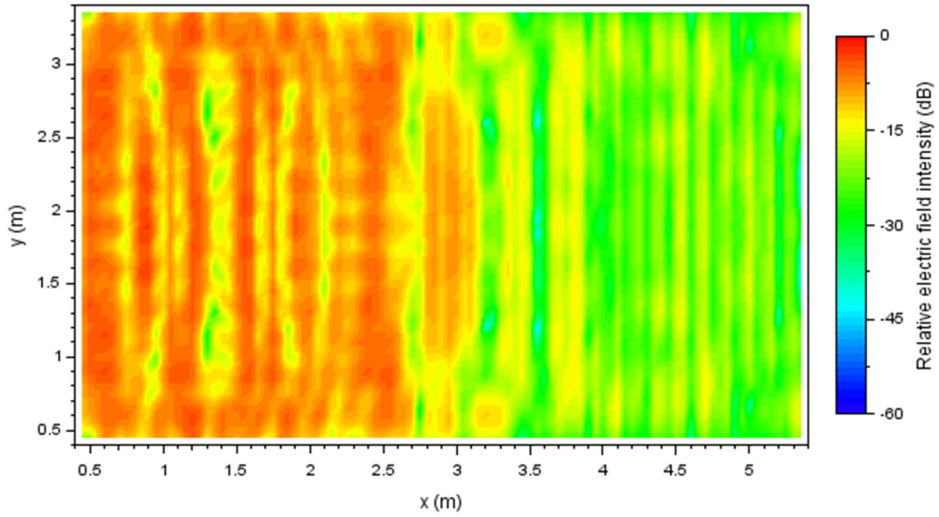


Figure 6.18 CST full-wave results for Figure 6.17

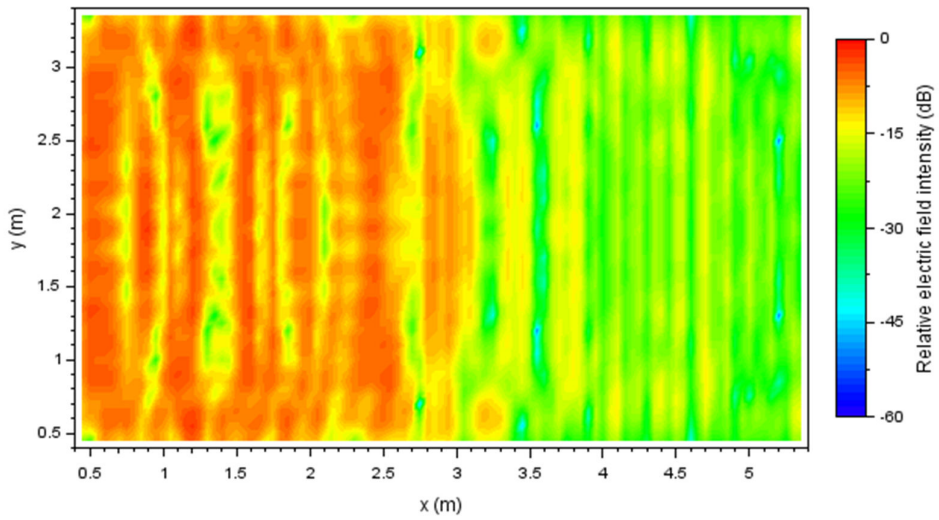
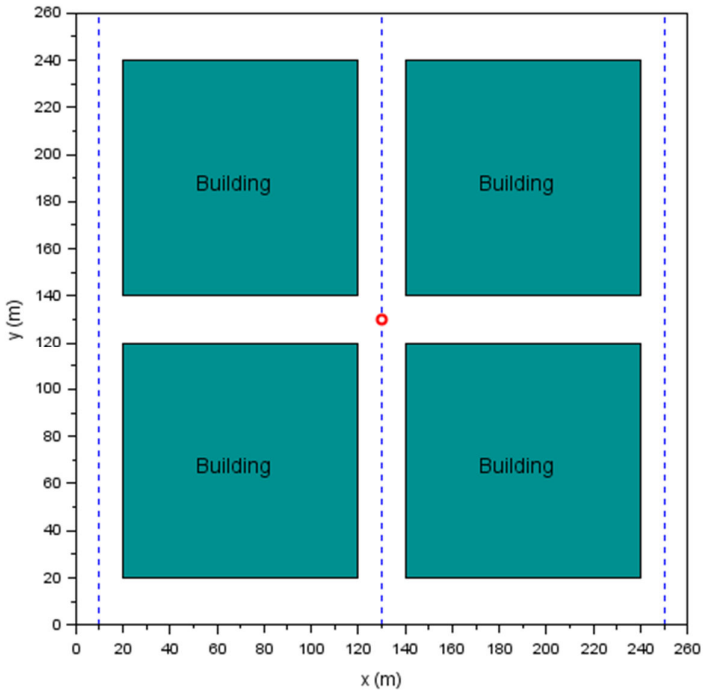


Figure 6.19 SBP-CCP simulation results for Figure 6.17

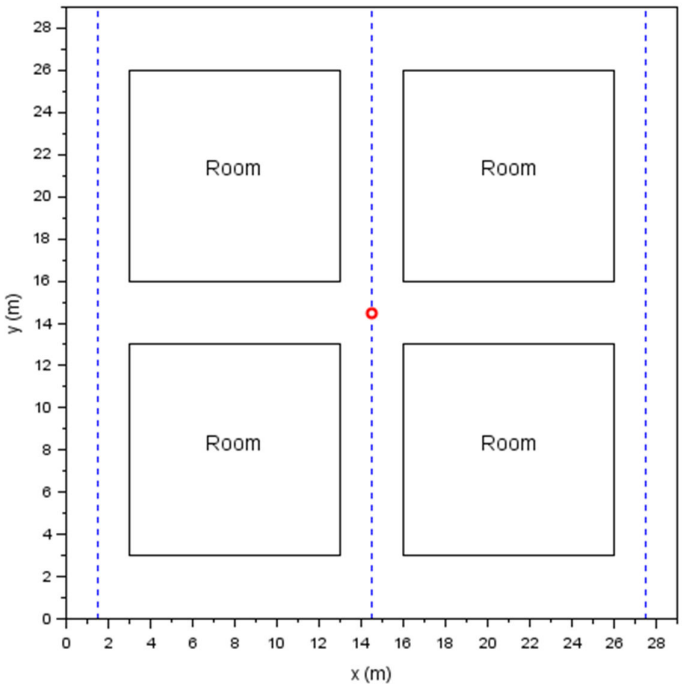
6.6 Big-O Time and Memory Complexity

Big-O complexity of an algorithm is the asymptotic complexity of the algorithm when the size of one of its inputs grows large. It is of interest to know the Big-O complexity of a ray tracer when the number of walls and edges (E), the number of interactions (N), and the number of receivers (R) grow large. Big-O complexity analysis is typically done analytically. However, a decent ray tracer is made of many algorithms interacting with each other. Assessing its overall complexity analytically is difficult and prone to errors. Hence, in this Section, we conduct numerical experiments to estimate the Big-O time and memory complexity of our SBP-CCP ray tracer.

Apparently, the ray tracer complexity is highly dependent on the simulated scene. A standard average scene will need to be defined to enable systematic evaluation of the ray tracer complexity. Noting the difference between urban canyon and indoor propagation, e.g. urban canyon has no reflection from ceiling nor transmission through wall, we have defined two standard scenes, one for urban canyon and one for indoor. Figure 6.20 shows the two standard scenes used in the complexity study. They have similar layouts which are scalable. The number of walls can be increased by adding more building blocks (for urban canyon) or rooms (for indoor). The number of diffracting edges is about the same as the number of walls. The indoor scene has ceiling and exterior walls and its walls are transmissive. A transmitter is placed in the middle of the scenes, marked by circles in Figure 6.20. Receivers are placed along the streets or corridors, marked by dashed lines in Figure 6.20.



(a) Urban canyon

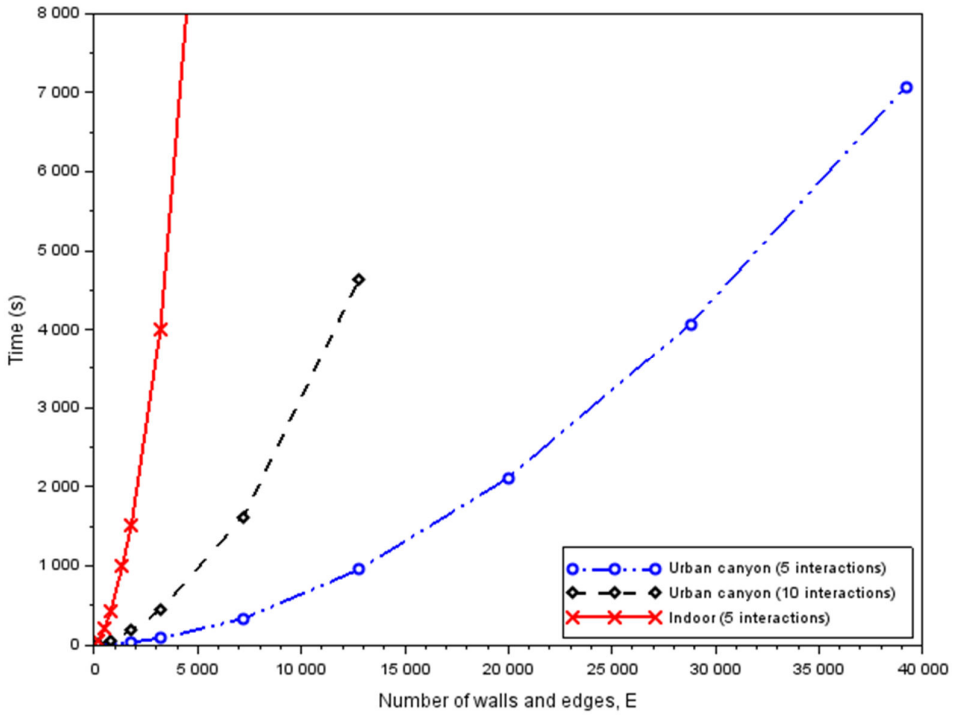


(b) Indoor

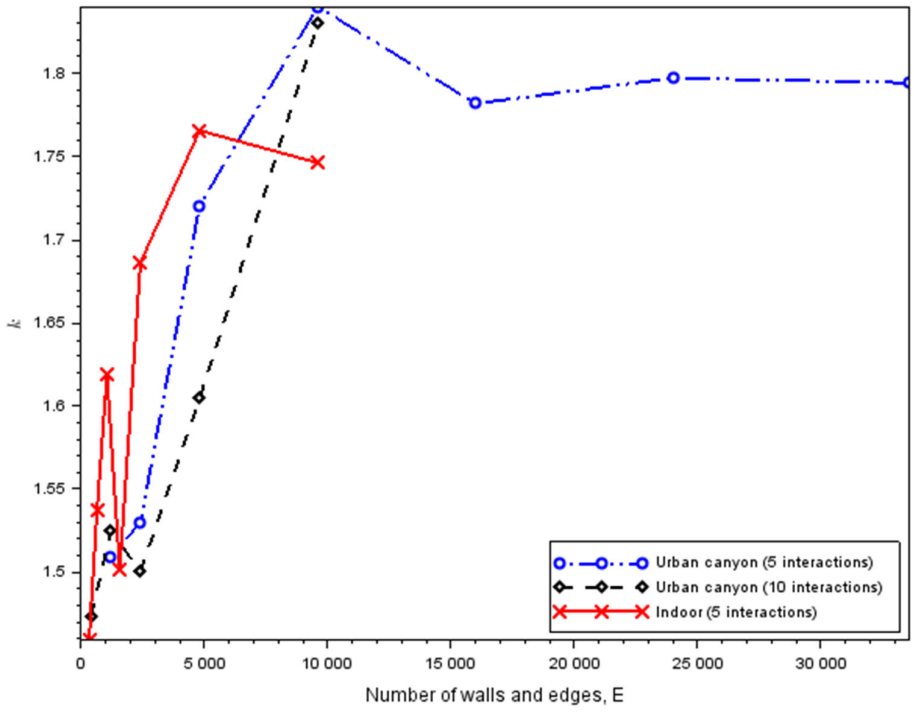
Figure 6.20 2D layouts of standard scenes used in the complexity study

The SBP process and the ray-paths-and-fields computation are different in nature, e.g. the SBP process is independent of receiver whereas the ray-paths-and-fields computation is receiver dependent. Hence, we have divided the run time into two parts, one part due to the SBP process and the other due to the ray-paths-and-fields computation. The part due to the SBP process can be measured directly by running the simulation without receiver. The part due to the ray-paths-and-fields computation is the difference between the run time with and without receivers. For example, if the measured run time with and without 10000 receivers are 1000 s and 100 s, respectively, the SBP process run time is 100 s and the ray-paths-and-fields computation run time is (1000 – 100)/10000 = 0.09 s per receiver.

The Ottawa city scene in Figure 6.6 has about 600 walls and edges. We have conducted the numerical experiments up to about 40,000 walls and edges. The quoted numbers are raw numbers before the walls are split or the insertion of virtual walls to form convex cells. Figure 6.21(a) shows the relationship between the SBP process run time and the number of walls and edges. Due to the long simulation time, simulations with 10 interactions or the indoor scene are run up to about 12,000 walls and edges only. Long simulation time is less accurate and less reproducible because it is more likely to be affected by other background computer activities. The curves in Figure 6.21(a) can be approximated by polynomials. For large number of walls and edges (E), the run time (T) is dominated by the highest order term E^k. In other words, $T \propto E^k$ for large E where k can be estimated from the slope of $log(T)$ versus $log(E)$ curve, i.e. $k = \frac{d(log(T))}{d(log(E))}$. Figure 6.21(b) shows the values of k for different E. The



(a) Run time

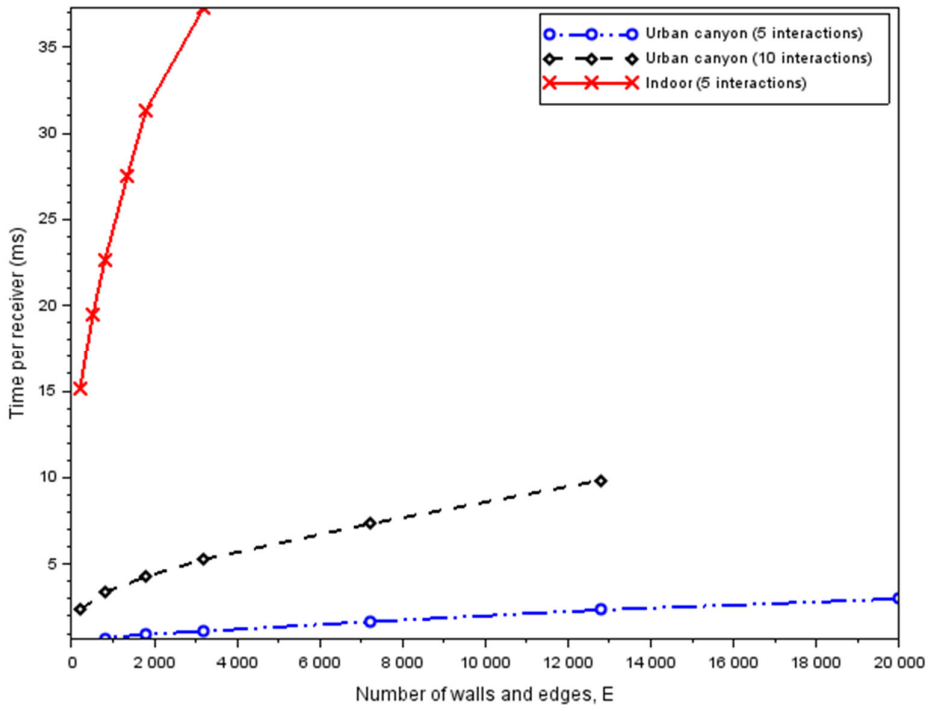


(b) *k*

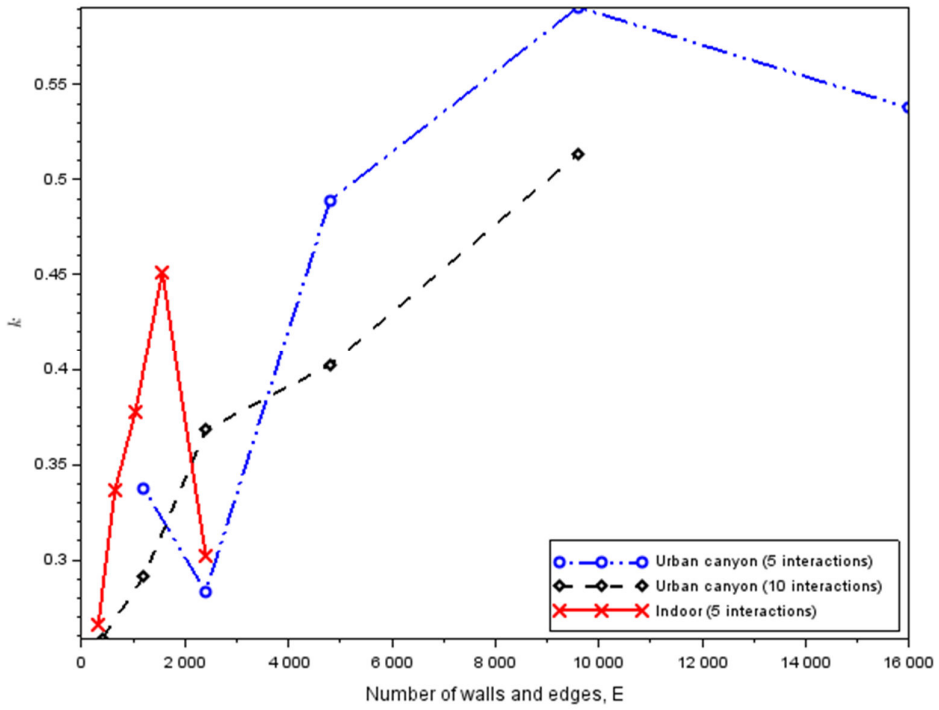
Figure 6.21 SBP process complexity and the number of walls and edges

values of k are about the same for all the three scenarios presented in Figure 6.21. It is reasonable to conclude that k is not sensitive to the number of interactions nor the inclusion of reflection from ceiling and transmission through wall. For large E, a good estimate of k is 1.8. This suggest that the SBP process has $O(E^{1.8})$ time complexity for urban canyon and indoor environments.

Figure 6.22(a) shows the relationship between the ray-paths-and-fields computation run time and the number of walls and edges. The per-receiver run time is computed as explained above (p. 116). The number of receivers used in the urban canyon and indoor experiments are about 100,000 and 10,000, respectively. The actual numbers are integral multiples of the number of streets or corridors. Figure 6.22 shows the urban canyon (5 interactions) data up to about 20,000 walls and edges only because beyond that the ray-paths-and-fields computation constitutes less than 10% of the total run time. A 5% error in the total run time, which is not uncommon, can cause huge error in the per-receiver run time. For the same reason, the indoor data is shown up to about 3000 walls and edges only. The values of k are about the same for all the three scenarios presented in Figure 6.22, i.e. k is not sensitive to the number of interactions or the inclusion of reflection from ceiling and transmission through wall. For large E, a good estimate of k is 0.55. This suggests that the ray-paths-and-fields computation has $O(E^{0.55})$ time complexity for urban canyon and indoor environments. As a comparison, Wireless InSite's SBR ray tracers are said to have $O(E^2)$ complexity (REMCOM, 2018b) which is about the same as the SBP process complexity. It is likely that the SBP ray tracer will maintain its time efficiency advantage shown in the above Sections, for much larger scenes with



(a) Run time



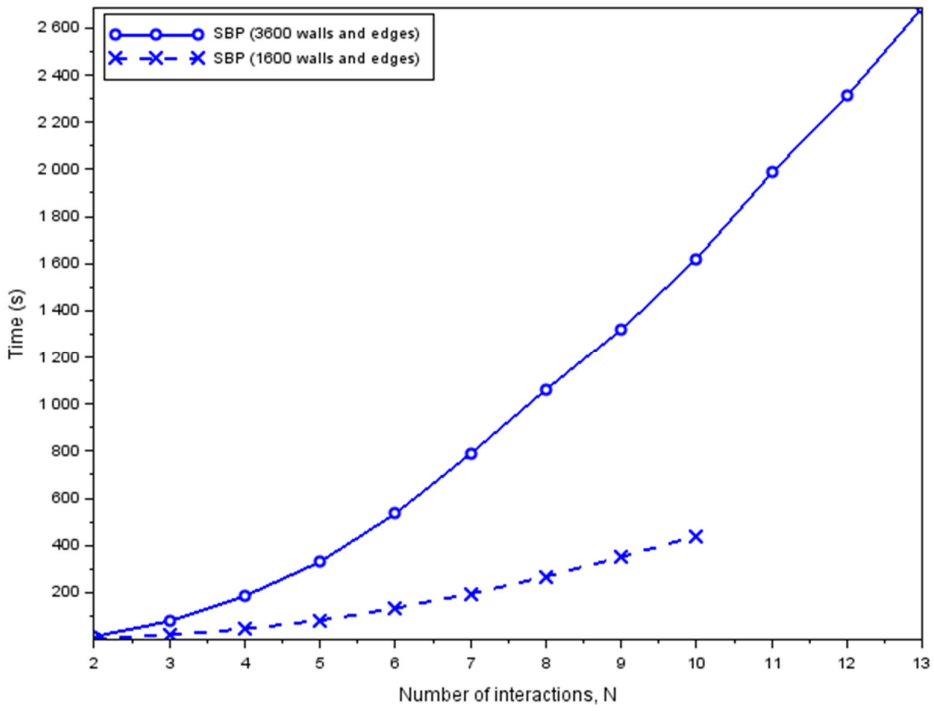
(b) k

Figure 6.22 Ray-paths-and-fields computation complexity and the number of walls and edges

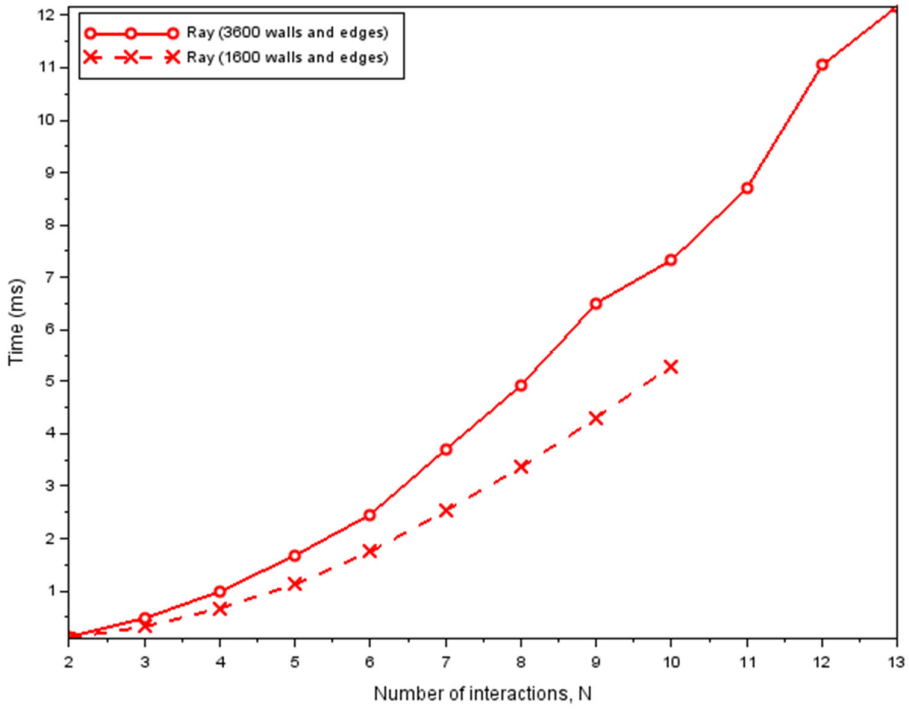
more walls and edges, especially if the ray separation of the SBR ray tracers is reduced to cope with the increasing distance.

Figure 6.23 shows the relationship between the run time (T) and the number of interactions (N), for urban canyon simulations. Because N is relatively small (compared to E), using the same method above will result in relatively large k. Considering that a ray beam may be split to multiple ray beams after each interaction, the time complexity is better described by $O(c^N)$ where c is a constant which can be estimated from $c = \frac{T(N+1)}{T(N)}$. Figure 6.24 shows the values of c for different N. It is clear that, for big N, c is not sensitive to the computation types nor the number of walls and edges. For big N, a good estimate of c is 1.16. This suggests a time complexity of $O(1.16^N)$ for urban canyon simulations.

Similar plots for indoor simulations are given in Figure 6.25 and Figure 6.26. For big N, a good estimate of c is 2.4. This suggests a time complexity of $O(2.4^N)$ for indoor simulations. The increase in complexity (compared to urban canyon simulations) is because transmission through walls cause more ray beams to be spawned after each interaction. Our complexity study includes reflected-diffracted-reflected ray paths. Without diffraction, the time complexities with respect to N can be deduced analytically. For sufficiently big N, a ray beam will be small enough to spawn only one reflected ray beam and one transmitted ray beam (for indoor) after each interaction. Hence, the time complexities with respect to N will be the same as those of SBR ray tracers, i.e.



(a) SBP process



(b) Ray-paths-and-fields computation

Figure 6.23 SBP-CCP run time and the number of interactions, for urban canyon simulations

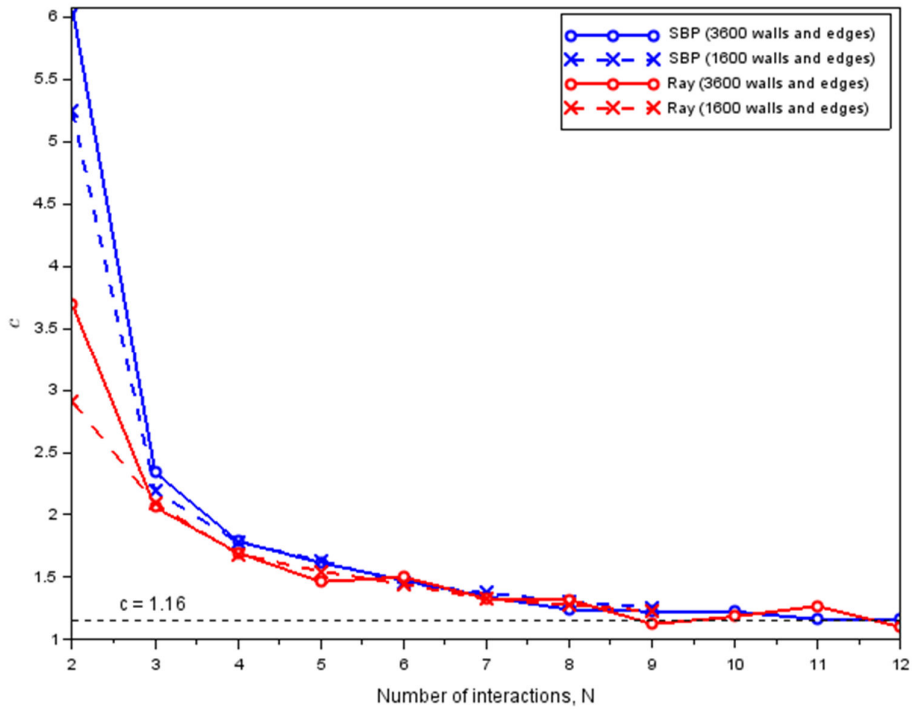
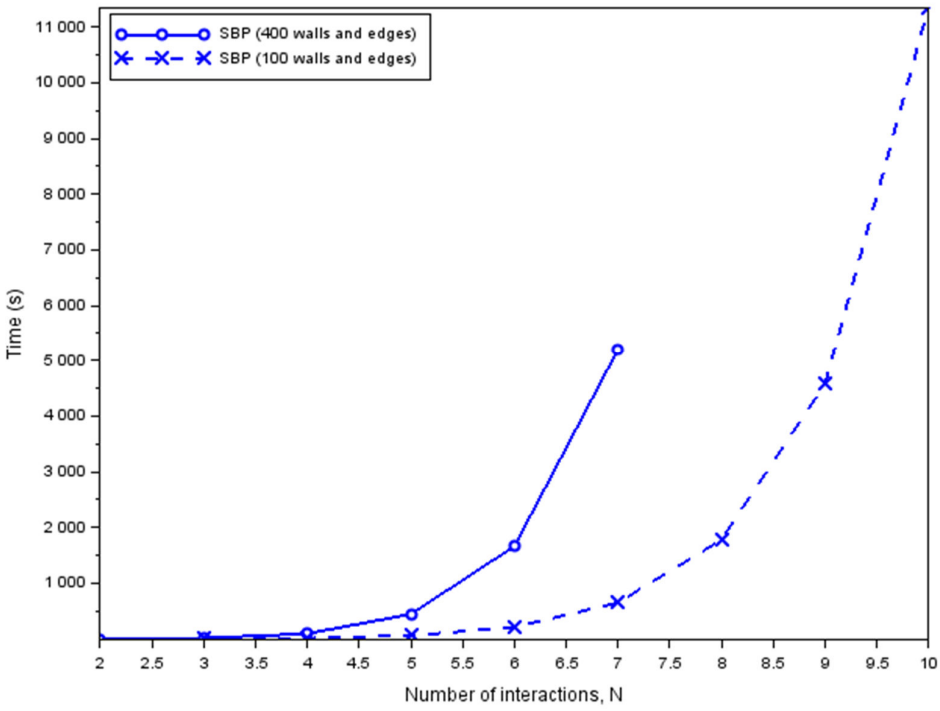


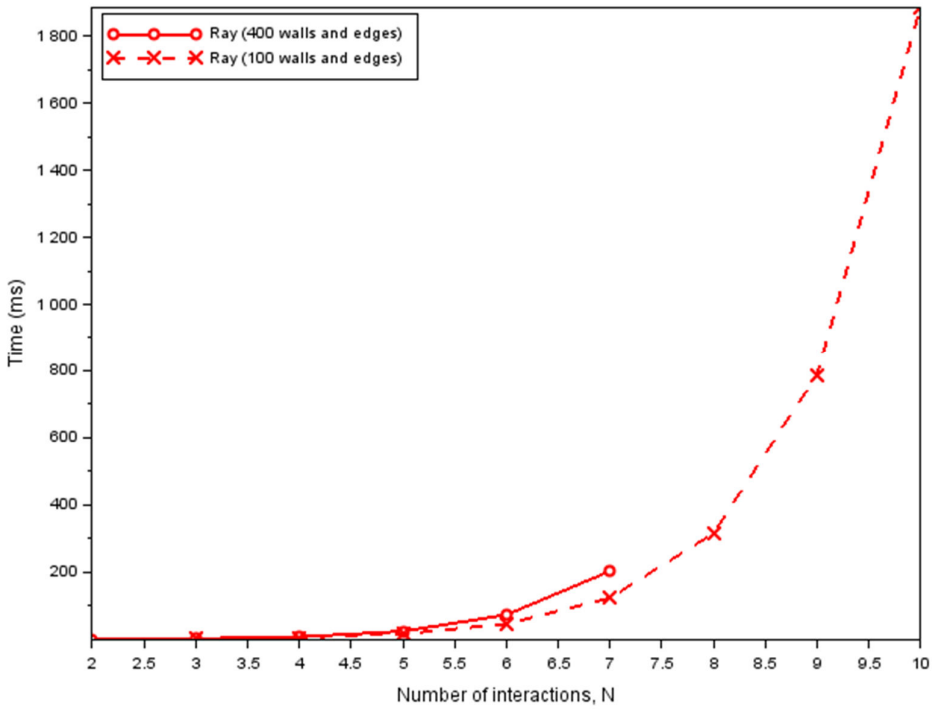
Figure 6.24 The values of *c* for urban canyon simulations

$O(N)$ and $O(2^N)$ for urban canyon and indoor simulations without diffraction, respectively.

The time complexity dependence on the number of receivers (R) is obvious and it can be deduced analytically. The SBP process is independent of receiver. Hence, it has $O(1)$ time complexity. The ray-paths-and-fields computation is similar for every receiver. Hence, it has $O(R)$ time complexity. This has been verified numerically, e.g. the ray-paths-and-fields run time per-receiver is about the same when 10,000 or 100,000 receivers are used.



(a) SBP process



(b) Ray-paths-and-fields computation

Figure 6.25 SBP-CCP run time and the number of interactions, for indoor simulations

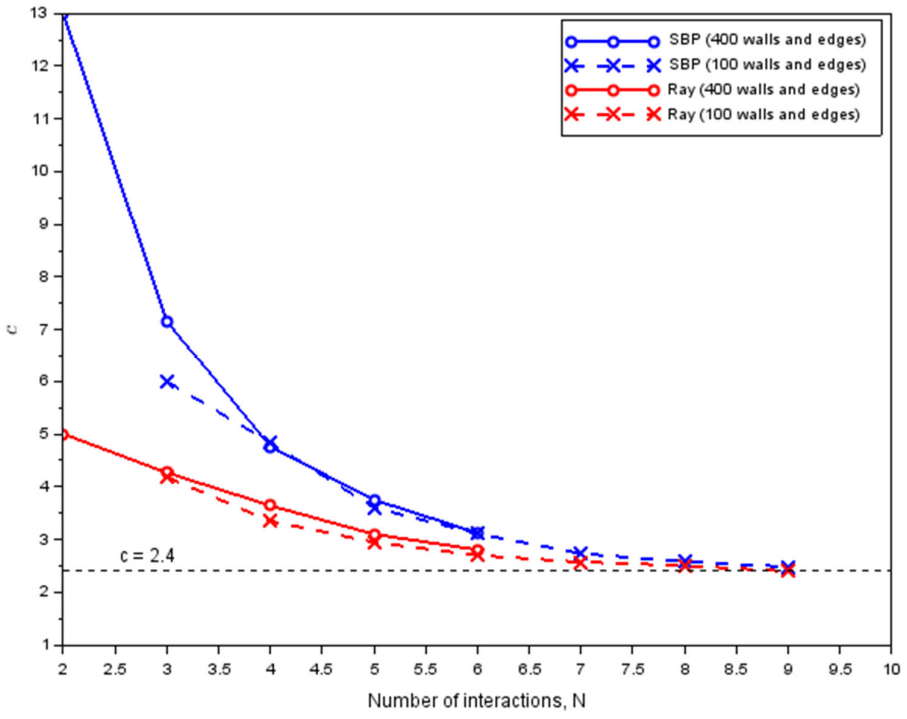


Figure 6.26 The values of *c* for indoor simulations

The memory complexity of the SBP-CCP ray tracer is also obvious and it can be deduced analytically. The number of convex cells is a linear function of the number of walls. The memory required to hold cell, wall and edge information grows linearly with the number of walls and edges. The SBP-CCP ray tracer has no other data structure that will grow with the number of walls and edges (E). Hence, it has $O(E)$ memory complexity. This has been verified numerically, as shown in Figure 6.27. The slope is about 1.3 kBytes per wall-edge pair.

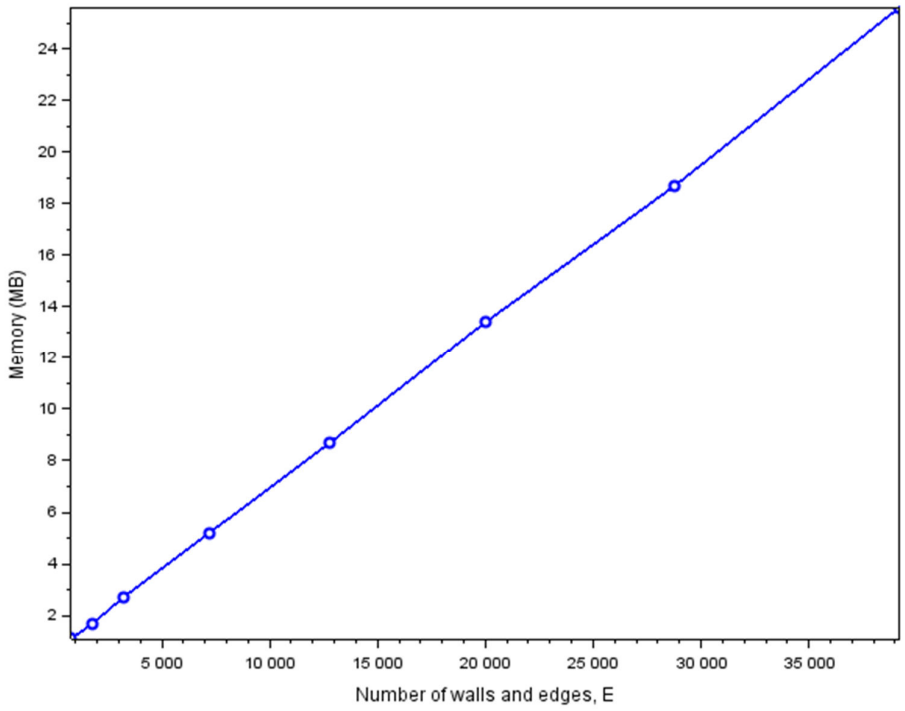


Figure 6.27 SBP-CCP memory consumption and the number of walls and edges

For N interactions, N images ($2N$ with transmission through wall) need to be stored on the trace stack. The size of ray path information is also proportional to N. Hence, the memory complexity should be $O(N)$. However, the increment of memory consumption due to increasing N is very small relative to the total memory consumption. Take Figure 6.3 (p. 87) for example, from N = 0 to 60, memory consumption increases from 1.1 MB to 1.3 MB. Because N is typically much smaller than 60, the increment of memory consumption due to increasing N is negligibly small. For all numerical experiments presented in Figure 6.23 to Figure 6.26, the change in memory consumption due to increasing N is not noticeable. Hence, the memory complexity with respect to N is better described by $O(1)$.

The memory complexity dependence on the number of receivers (R) is obviously linear, i.e. $O(R)$. This has been verified numerically. The amount of memory required per receiver is dependent on the desired output e.g. power, power delay profile, or ray path. If only power is desired, the amount of memory required per receiver is about 445 bytes as obtained from the numerical experiments. Table 6.4 shows a summary of the time and memory complexity of our SBP-CCP ray tracer as obtained from the numerical experiments.

Table 6.4 SBP-CCP time and memory complexity

Parameters	Time complexity				Memory complexity
	SBP process		Ray paths and fields		
	Urban Canyon	Indoor	Urban Canyon	Indoor	
Number of walls and edges, E	$O(E^{1.8})$	$O(E^{1.8})$	$O(E^{0.55})$	$O(E^{0.55})$	$O(E)$
Number of receivers, R	$O(1)$	$O(1)$	$O(R)$	$O(R)$	$O(R)$
Number of interactions, N	$O(1.16^N)$	$O(2.4^N)$	$O(1.16^N)$	$O(2.4^N)$	$O(1)$

CHAPTER 7

CONCLUSIONS

7.1 Achievements

We have described three important improvements to a shooting-and-bouncing-polygon (SBP) ray tracer. Firstly, we have derived delay correction factors for reflection, transmission, and diffraction, based on a multilayer lossy wall model. They allow walls with thickness to be handled accurately and efficiently using one-patch walls. We have pointed out that the correction for transmission is required even if a less efficient two-patch method is used.

Secondly, we have introduced edge-fixed diffraction ray-polygon to trace diffracted and diffracted-reflected ray beams. We have described the edge-fixed projection problems and presented an algorithm to perform the projection correctly. The addition enables the SBP ray tracer to trace and account for diffracted-reflected rays. This improves the SBP ray tracer's accuracy when diffracted-reflected rays play an important role. By comparing ray tracing results with published measurements, we have explicitly shown that diffracted-reflected rays play an important role in shadow regions in urban canyon simulations.

Thirdly, we have proposed a new spatial partitioning scheme, i.e. convex cell partitioning (CCP), for the SBP ray tracer. Compared to binary-space-partitioning, CCP removes the need to perform polygon subtraction, a relatively expensive procedure. Traversal of the CCP graph is also simpler than the binary tree. Numerical results show that the CCP version (SBP-CCP) is about an order of magnitude faster than the binary-space-partitioning version (SBP-BSP) in an urban canyon application. To improve the usability of the SBP-CCP ray tracer for large complex scenes, we have also presented a working algorithm to automatically generate the convex cell data structures from a given set of walls.

During the course of this project, we have implemented three ray tracers in C/C++ programming language, including testing and debugging. Two general purpose 3D ray tracers, i.e. SBP-BSP and SBP-CCP, and one special purpose image-based ray tracer for rectangular tunnels. Both SBP-BSP and SBP-CCP support diffraction ray-polygon. However, only SBP-CCP is enhanced with the new delay correction factors and CCP. We have written a number of preprocessing tools in C/C++, e.g. automatic generation of various input files for SBP-CCP. We have also written many post-processing tools in Scilab, mainly for visualization of simulation (ray tracing and full-wave) and measurement results. Many of the plots presented in this report can be generated with a mouse click.

The presented indoor scene and results can serve as a good benchmarking case to validate ray tracers or to compare their accuracy. It is suitable for indoor ray tracers that are able to handle non-axis-aligned walls.

Because the parameters are well defined, the case does not allow manipulation of simulation results in the name of parameter optimization. It will do a better job than propagation measurements at revealing formulation and implementation flaws in the ray tracing algorithm or electromagnetic computation. We have shown that it is possible to achieve an rms error of 2 dB in point-to-point comparison using a ray tracer. We believe such benchmark, in addition to propagation measurements, is important to the development of ray tracers for radio propagation modelling. There are not many of such benchmarks available in the literature.

We have compared SBP-CCP performance against commercial REMCOM Wireless InSite ray tracers, in terms of run time, accuracy, and memory consumption. The comparisons show that our 3D SBP-CCP ray tracer outperform REMCOM 3D ray tracers in the 3 aspects (run time, accuracy, and memory consumption) in long tunnel, urban canyon, and indoor applications. Our 3D SBP-CCP ray tracer is 1 to 2 orders of magnitude more memory efficient than REMCOM 2D ray tracer in an urban canyon application, despite the common notion that 2D ray tracers are more efficient than 3D ray tracers. Various results have shown that the SBP-CCP ray tracer developed in this project is an accurate, time efficient, and memory efficient 3D ray tracer. It is a good candidate for simulating long tunnel, urban canyon, and indoor propagation environments. The current implementation does not include double diffraction and it is not suitable for general urban environments which require multiple over-roof-top diffractions. Table 7.1 shows a summary of the SBP-CCP simulation parameters and performance data.

Table 7.1 Simulation parameters and performance data

Scenario	Environment	Measurement	Frequency	Number of reflections	Number of diffractions	Number of receivers	Rms error (dB)	Run time (s)	Memory (MB)
Massif Central Tunnel	Long tunnel	Dudley et al. (2007)	450 MHz	60	0	1250	~5	~180	~1
			900 MHz	40	0	1250	~5	~50	~1
Ottawa City Streets	Urban canyon	Whitteker (1988)	910 MHz	10	1	~900	6~8	~100	~1
The Printing House at Trinity College Dublin	Indoor	Kenny and Nuallain (2017)	900 MHz	6	1	180	~6	~20	~1
Trapezoidal room and tapered corridor	Indoor	-	1.5 GHz	6	1	9801	~2*	~60	~5
Room with column	Indoor	-	1.5 GHz	6	1	5841	~2*	~80	~3

* Relative to full-wave solutions

Based on some of the works in this project, we have published a paper entitled “An accurate and efficient 3D shooting-and-bouncing-polygon ray tracer for radio propagation modeling” in IEEE Transactions on Antennas and Propagation, vol. 66, pp. 7244-7254. Another paper entitled “Multilayer wall correction factors for indoor ray tracing radio propagation modeling” has been submitted to IEEE Transactions on Antennas and Propagation. At the time of writing, it is subject to minor revisions.

7.2 Recommendations of Future Work

Although the SBP-CCP ray tracer has shown good performance, it has plenty of room for improvements. We will discuss some of them here. One of them is to extend SBP-CCP to handle double and higher order diffractions which are required for simulations of general urban environments with over-roof-top diffractions. The extension to multiple diffractions from parallel edges should be straight forward because the same diffraction ray-polygon can be used. The extension to multiple diffractions from non-parallel edges would require a lot more work. Using the SBP framework, a new ray-polygon projection scheme is required.

The ability to handle curved-surface is also important, in particular for simulating curved tunnels. For slightly curved surface (large radius of curvature), piecewise linear approximation may be sufficient. Otherwise, a new ray-polygon projection scheme is required. The change in spreading factor also

needs to be taken into account (Didascalou et al., 2000; Wang and Yang, 2006) when computing the reflected or transmitted field.

At millimeter wave frequencies, diffraction becomes less important but rough surface scattering becomes more important. Rough surface scattering is also important for simulation of mine tunnels. A simple and common way to account for rough surface scattering is to apply a multiplicative roughness factor to tone down the specular reflection coefficient (Chamberlin and Luebbers, 1982). The roughness factor, however, only helps at locations determined to have received the specular reflected field. A different technique is described by Degli-Esposti et al. (2004) and Fügen et al. (2006). However, they too do not seem to have traced diffuse rays. The above methods should be applicable to SBP because they are based on specular rays. Another method based on Monte Carlo ray tracing, also known as path tracing, is described by Barowski, Meiners and Rolfes, (2015). However, it is computationally intensive. Barowski et al. have used more than 10^9 rays, about 3 orders of magnitude more than conventional SBR.

SBP, like many other ray tracing algorithms, is suitable for parallel processing. Parallel processing requires the division and scheduling of tasks and the combining of results. Although we are yet to work out the details, multicore processing should be relatively simple, e.g. trace each ray-polygon on a different thread and processor core. GPU implementation should require more work due to GPU constraints. SBP's low memory consumption is a plus to GPU implementation.

Another important aspect that needs improvements is the ability to convert or translate building data from various sources in various formats into the ray tracer's native format (Yun and Iskander, 2015). If acceptable, an easy way out is to rely on such features provided by a commercial suite like REMCOM. In this case, the problem is reduced to translating from the commercial suite's format into the native format.

The current implementation of the SBP-CCP ray tracer is also far from optimized. The ray tracer consists of a bunch of subroutines doing different tasks. Many of the tasks can be done in many ways. The choices have been made by subjective comparisons, by intuition, or for simplicity. Each of the subroutines can be tweaked and have its effects measured and optimized. Because of the relatively low memory consumption, trading memory for speed may be possible in some cases, e.g. use of look-up tables.

With that we end the report. Thank you for reading.

LIST OF PUBLICATIONS

Teh, C. H., Chung, B. K., and Lim, E. H., 2018. An accurate and efficient 3-D shooting-and-bouncing-polygon ray tracer for radio propagation modeling. *IEEE Trans. Antennas Propag.*, 66, pp. 7244-7254.

Teh, C. H., Chung, B. K., and Lim, E. H., 2019. Multilayer wall correction factors for indoor ray tracing radio propagation modeling. *IEEE Trans. Antennas Propag.*, (subject to minor revisions).

REFERENCES

Aïdi, M. and Lavergnat, 2001. Comparison of Luebbers' and Maliuzhinets' wedge diffraction coefficients in urban channel modelling. *Progress In Electromagnetics Research*, 33, pp. 1-28.

Athanasiadou, G. E., Nix, A. R. and McGeehan, J. P., 2000. A microcellular ray-tracing propagation model and evaluation of its narrow-band and wide-band predictions. *IEEE J. Sel. Areas Commun.*, 18, pp. 322-335.

Balanis, C. A., 1989. *Advanced Engineering Electromagnetics*. Hoboken, NJ: Wiley.

Bernardi, P., Cicchetti, R. and Testa, O., 2002. A three-dimensional UTD heuristic diffraction coefficient for complex penetrable wedges. *IEEE Trans. Antennas Propag.*, 50, pp. 217-224.

Bernardi, P., Cicchetti, R. and Testa, O., 2004. An accurate UTD model for the analysis of complex indoor radio environments in microwave WLAN systems. *IEEE Trans. Antennas Propag.*, 52, pp. 1509-1520.

Bertoni, H. L., Torrico, S. A. and Liang, G., 2005. Predicting the radio channel beyond second-generation wireless systems. *IEEE Antennas Propag. Mag.*, 47, pp. 28-40.

Booysen, A. J. and Pistorius, C. W. I., 1992. Electromagnetic scattering by a two-dimensional wedge composed of conductor and lossless dielectric. *IEEE Trans. Antennas Propag.*, 31, pp. 383-390.

Burnside, W. D. and Burgener, K. W., 1983. High frequency scattering by a thin lossless dielectric slab. *IEEE Trans. Antennas Propag.*, 31, pp. 104-110.

Catedra, M. F., Perez, J., de Adana, F. S. and Gutierrez O., 1998. Efficient raytracing technique for three-dimensional analyses of propagation in mobile communications: Application to picocell and microcell scenarios. *IEEE Antennas Propag. Mag.*, 40, pp. 15-27.

Chen, S. H. and Jeng, S. K., 1997. An SBR/image approach for radio wave propagation in indoor environments with metallic furniture. *IEEE Trans. Antennas Propag.*, 45, pp. 98-106.

Chen, Y. S., Lai, F. P. and Yu, J. W., 2019. Analysis of antenna radiation characteristics using a hybrid ray tracing algorithm for indoor WiFi energy-harvesting rectennas. *IEEE Access*, 7, 38833-38846.

Chuang, J. C. I., 1987. The effects of time delay spread on portable radio communications with digital modulation. *IEEE J. Select. Areas Commun.*, 5, pp. 879-889.

Cocheril, Y. and Vauzelle, R., 2007. A new ray-tracing based wave propagation model including rough surfaces scattering. *Progress in Electromagnetics Research*, PIER 75, 357-381.

Degli-Esposti, V., Guiducci, D., de' Marsi, A., Azzi, P. and Fuschini, F., 2004. An advanced field prediction model including diffuse scattering. *IEEE Trans. Antennas Propag.*, 52, pp. 1717-1728.

Di Giampaolo, E. and Bardati, F., 2009. A projective approach to electromagnetic propagation in complex environments. *Progress In Electromagnetics Research B*, 13, pp. 357-383, 2009.

Didascalou, D., Schäfer, T. M., Weinmann, F. and Wiesbeck, W., 2000. Ray-density normalization for ray-optical wave propagation modelling in arbitrarily shaped tunnels. *IEEE Trans. Antennas Propag.*, 48, pp. 1316-1325.

Demetrescu, C., Constantinou, C.C. and Mehler, M.J., 1997. Scattering by a right-angled lossy dielectric wedge. *IEE Proc -Microw Antennas Propag.*, 144, pp. 392-396.

Dudley, D. G., Liénard, M., Mahmoud, S. F. and Degauque, P., 2007. Wireless propagation in tunnels. *IEEE Antennas Propag. Mag.*, 49, pp. 11–26.

Edwards, R. and Durkin, J., 1969. Computer prediction of service area for VHF mobile radio networks. *Proceedings of the IEE*, 116, pp. 1493-1500.

Egea-Lopes, E., Losilla, F., Pascual-Garcia, J. and Molina-Garcia-Pardo, J. M., 2019. Vehicular networks simulation with realistic physics. *IEEE Access*, 7, 44021-44036.

El-Sallabi, H. M., Rekanos, I. T. and Vainikainen, P., 2002. A New Heuristic Diffraction Coefficient for Lossy Dielectric Wedges at Normal Incidence. *IEEE Antennas Wirel. Propag. Lett.,* 1, pp. 165-168.

Foley, J. D., van Dam, A., Feiner, S. K. and Hughes, J. F., 1996. *Computer Graphics: Principles and Practice*, 2nd. ed., New York: Addison-Wesley.

Fortune, S., 1996. A beam-tracing algorithm for prediction of indoor radio propagation. In: Lin, M. C. and Manocha, D. (eds). *Applied Computational Geometry Towards Geometric Engineering*, Berlin: Springer-Verlag, pp. 157-166.

Fügen, T., Maurer, J., Kayser, T. and Wiesbeck, W., 2006. Capability of 3-D ray tracing for defining parameter sets for the specification of future mobile communications systems. *IEEE Trans. Antennas Propag.*, 54, pp. 3125-3137.

Fuschini, F. and Falciasecca, G., 2012. A mixed rays—modes approach to the propagation in real road and railway tunnels. *IEEE Trans. Antennas Propag.*, 60, pp. 1095-1105.

Fuschini, F., Vitucci, E. M., Barbiroli, M., Falciasecca, G. and Degli-Esposti, V., 2015. Ray tracing propagation modeling for future small-cell and indoor applications: A review of current techniques. *Radio Science*, 50, pp. 469-485.

Guan, K., Peng, B., He, D., Eckhardt, J. M., Rey, S., Ai, B., Zhong, Z. and Kuerner, T., 2019. Channel characterization for intra-wagon communication at 60 GHz and 300 GHz bands. *IEEE Trans. Vehic. Technol.*, in press.

Hata, M., 1980. Empirical formula for propagation loss in land mobile radio services. *IEEE Trans. Vehic. Technol.*, 29, pp. 317-325.

Haumont, D., Debeir, O. and Sillion, F., 2003. Volumetric cell-and-portal generation. In: *Eurographics 2003*, 22, pp. 303-312.

Holm, P., 2000. A new heuristic UTD diffraction coefficient for nonperfectly conducting wedges. *IEEE Trans. Antennas Propag.*, 48, pp. 1211–1219.

Hrovat, A., Kandus, G. and Javornik, T., 2014. A survey of radio propagation modeling for tunnels. *Commun. Surveys Tuts.*, 16, pp. 658-669.

Iskander, M. F. and Yun, Z., 2002. Propagation prediction models for wireless communication systems. *IEEE Trans. Microw. Theory Techn.*, 50, pp. 662-673.

Ishimaru, A., 1991. *Electromagnetic Wave Propagation, Radiation, and Scattering*. Englewood Cliffs, NJ: Prentice-Hall.

Keller, J. B., 1962. Geometrical theory of diffraction. *J. Opt. Soc. Of America.*, 52, pp. 116-130.

Kenny, E. M. and Nuallain, E. O., 2017. Convex space building discretization for ray-tracing. *IEEE Trans. Antennas Propag.*, 65, pp. 2578-2590.

Kong, J. A., 1985. *Electromagnetic Wave Theory*. New York: Wiley.

Kouyoumjian, R. G. and Pathak, P., 1974. A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface. *Proc. IEEE*, 62, pp. 1448-1461,

Leonor, N. R., Fernandes, T. R., Sánchez, M. G. and Caldeirinha, R. F. S., 2019. A 3D model for millimeter-wave propagation through vegetation media using ray-tracing. *IEEE Trans. Antennas Propag.*, in press.

Longley, A. G. and Rice, P. L., 1968. Prediction of tropospheric radio transmission loss over irregular terrain: a computer method. *ESSA Technical Report*, ERL 79 - ITS 67.

Luebbers, R. J., 1984. Finite conductivity uniform GTD versus knife edge diffraction in prediction of propagation path loss. *IEEE Trans. Antennas Propag.*, 32, pp. 70-76.

McNamara, D. A., Pictorius, C. W. L. and Malherbe, J. A. G., 1990. *Introduction to the Uniform Geometrical Theory of Diffraction*, Boston: Artech House.

Maurer, J., Drumm, O., Didascalou, D. L. and Wiesbeck, W., 2000. A novel approach in the determination of visible surfaces in 3D vector geometries for ray-optical wave propagation modelling. In: *Proc. 511st IEEE Veh. Technol. Conf. (VTC2000-Spring)*, 24-28 September 2000, Tokyo: IEEE, pp. 1651-1655.

Mckown, J. W. and Hamilton, R. L., 1991. Ray tracing as a design tool for radio networks. *IEEE Network*, 5, pp. 27-30.

Molina-Garcia-Pardo, J. M., Lienard, M. and Degauque, P., 2012. *Wireless Communication in Tunnels, Digital Communication*. Rijeka, Croatia: InTech.

Mohtashami, V. and Shishegar, A., 2012. Modified wavefront decomposition method for fast and accurate ray-tracing simulation. *IET Microw. Antennas Propag.*, 6, pp. 295-304.

Murta, A., 2017. GPC – General Polygon Clipper library [Online]. Available at: http://www.cs.man.ac.uk/~toby/gpc/ [Accessed: 7 July 2017]

Novak, R., 2019. Bloom filter for double-counting avoidance in radio frequency ray tracing. *IEEE Trans. Antennas Propag.*, 67, 2176-2190.

Okumura, Y., Ohmori, E., Kawano, T. and Fukuda, K., 1968. Field strength and its variability in VHF and UHF land-mobile radio service. *Rev. Elec. Commun. Lab.*, 16, pp. 825-873.

Orfanidis, S. J., 2016. *Electromagnetic Waves and Antennas* [Online]. Available at: http://eceweb1.rutgers.edu/~orfanidi/ewa/ [Accessed: 16 March 2018]

Rajkumar, A., Naylor, B. F., Feisullin, F. and Rogers, L., 1996. Predicting RF coverage in large environments using ray-beam tracing and partitioning tree represented geometry. *Wireless Networks*, 2, pp. 143-154.

Remcom, 2018a. Wireless InSite 3D wireless prediction software. *Remcom Website*. Available at: https://www.remcom.com/ [Accessed: 28 August 2018]

Remcom, 2018b. *Wireless InSite 3.0.0 Reference Manual*, Remcom.

Rouviere, J. F., Douchin, N. and Combes, P. F., 1999. Diffraction by lossy dielectric wedges using both heuristic UTD formulations and FDTD. *IEEE Trans. Antennas Propag.*, 47, pp. 1702-1708.

Saeidi, C. and Hodjatkashani, F., 2010. Modified angular z-buffer as an acceleration technique for ray tracing. *IEEE Trans. Antennas Propag.*, 58, pp. 1822-1825.

Sarkar, T. K., Zhong, J., Kim, K., Medouri, A. and Salazar-Palma, M., 2003. A survey of various propagation models for mobile communication. *IEEE Antennas Propag. Mag.*, 45, pp. 51–82.

Schettino, D. N., Moreira, F. J. S., Borges, K. L. and Rego, C. G., 2007. Novel Heuristic UTD Coefficients for the Characterization of Radio Channels. *IEEE Trans. Magn.*, 43, pp. 1301-1304.

Schmitz, A., Rick, T., Karolski, T., Kuhlen, T. and Kobbelt L., 2011. Efficient rasterization for outdoor radio wave propagation. *IEEE Trans. Vis. Comput. Graph.,* 17, pp. 159-170.

Seidel, S. Y. and Rappaport, T. S., 1994. Site-specific propagation prediction for wireless in-building personal communication system design. *IEEE Trans. Veh. Technol.*, 43, pp. 879-891.

Son, H. W. and Myung, N. H., 1999. A deterministic ray tube method for microcellular wave propagation prediction model. *IEEE Trans. Antennas Propag.*, 47, pp. 1344–1350.

Soni, S. and Bhattacharya, A., 2010. Novel three-dimensional dyadic diffraction coefficient for wireless channel. *Microwave and Optical Technology Letters*, 52, pp. 2132-2136.

Suzuki, H. and Mohan, A. S., 1997. Ray tube tracing method for predicting indoor channel characteristic map. *Electronics Letters*, 33, pp. 1495-1496.

Suzuki, H. and Mohan, A. S., 2003. Measurement and prediction of high spatial resolution indoor radio channel characteristic map. *IEEE Trans. Veh. Technol.*, 49, pp. 1321–1333.

Tam, W. K. and Tran, V. N., 1995. Propagation modelling for indoor wireless communication. *Electronics & Communication Engineering Journal*, 7, pp. 221-228.

Tan, J., Su, Z. and Long, Y., 2015. A full 3-D GPU-based beam-tracing method for complex indoor environments propagation modelling. *IEEE Trans. Antennas Propag.*, 63, pp. 2705-2718.

Tan, S. Y. and Tan, H. S., 1995. Propagation model for microcellular communications applied to path loss measurements in Ottawa city streets. *IEEE Trans. Veh. Technol.*, 44, pp. 313–317.

Tan, S. Y. and Tan, H. S., 1996. A microcellular communications propagation model based on the uniform theory of diffraction and multiple image theory. *IEEE Trans. Antennas Propag.*, 44, pp. 1317–1326.

Tao, Y. B., Lin, H. and Bao, H. J., 2011. Adaptive aperture partition in shooting and bouncing ray method. *IEEE Trans. Antennas Propag.*, 59, pp. 3347-3357.

Teh, C. H. and Chuah, H. T., 2003. An improved image-based propagation model for indoor and outdoor communication channels. *Journal of Electromagnetic Waves and Applications*, 17, pp. 31–50.

Teh, C. H., Kung, F. and Chuah, H. T., 2006. A path-corrected wall model for ray-tracing propagation modelling. *Journal of Electromagnetic Waves and Applications*, 20, pp. 207–214.

Vandamme, J., Baranowski, S. and Mariage, P., 1995. High frequency diffraction by a dielectric wedge—Three dimensional study. In: *Proc. IEEE Int. Symp. on Personal, Indoor. and Mobile Radio Commun.*, 1, pp. 125–129.

Vatti, V. R., 1992. A generic solution to polygon clipping. *Communications of the ACM*, 35, pp. 57-63.

Volakis, J. L., 2013. Diffraction by canonical metallic and material-coated structures: a review. *IEEE Antennas Propag. Mag.*, 55, pp. 21-31.

Walfisch, J. and Bertoni, H. L., 1988. A theoretical model of UHF propagation in urban environments. *IEEE Trans. Antennas Propag.*, 36, pp. 1788-1796.

Walsh, P., 2008. *Advanced 3D game programming with DirectX 10.0*, Texas: Wordware.

Wang, T. S. and Yang, C. F., 2006. Simulations and measurements of wave propagations in curved road tunnels for signals from GSM base stations. *IEEE Trans. Antennas Propag.*, 54, pp. 2577-2584.

Whitteker, J. H., 1988. Measurements of path loss at 910 MHz for proposed microcell urban mobile systems. *IEEE Trans. Veh. Technol.*, 37, pp. 125–129.

Wikipedia, 2018. Binary space partitioning [Online]. Available at: https://en.wikipedia.org/wiki/Binary_space_partitioning [Accessed: 25 Aug 2018]

Yang, C. F., Wu, B. C. and Ko, C. J., 1998. A ray-tracing method for modeling indoor wave propagation and penetration. *IEEE Trans. Antennas Propag.*, 46, pp. 907-919.

Yun, Z. and Iskander, M. F., 2015. Ray tracing for radio propagation modeling: principles and applications. *IEEE Access*, 3, pp. 1089-1100.

Yun, Z., Iskander, M. F. and Zhang, Z., 2000. Fast ray tracing procedure using space division with uniform rectangular grid. *Electron. Lett.*, 36, pp. 895-897.

Yun, Z., Zhang, Z. and Iskander, M. F., 2002. A ray-tracing method based on the triangular grid approach and application to propagation prediction in urban environments. *IEEE Trans. Antennas Propag.*, 50, pp. 2577-2584.

Zhang, X., Sood, N., Siu, J. K. and Sarris, C. D., 2016. A hybrid ray-tracing/vector parabolic equation method for propagation modeling in train communication channels. *IEEE Trans. Antennas Propag.*, 64, pp. 1840-1849.

Zhang, Z., Yun, Z. and M. F. Iskander, 2001. 3D tetrahedron ray tracing algorithm. *Electronics Letters*, 37, pp. 334-335.

Zhou, C., Plass, T., Jacksha, R. and Waynert, J. A., 2015. RF propagation in mines and tunnels. *IEEE Antennas Propag. Mag.*, 57, pp. 88–102.