UCLouvain
Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Thesis submitted in the partial fulfillment of the requirements for the degree of
Docteur en sciences de l'ingénieur et technologie

Differentiable Ray Tracing for Radio Propagation Modeling

Jérome Eertmans

July 2026

Thesis committee:

Prof. Claude Oestges, *advisor*	Université catholique de Louvain
Prof. Laurent Jacques, *advisor*	Université catholique de Louvain
Prof. Christophe Craeye, *chairman*	Université catholique de Louvain
Prof. Christophe De Vleeschouwer, *secretary*	Université catholique de Louvain
Prof. Philippe De Doncker	Université libre de Bruxelles
Prof. Enrico Maria Vitucci	Università di Bologna, Italy
Dr. Jakob Hoydis	NVIDIA Research, France

"Any sufficiently advanced technology is
indistinguishable from magic."

– Arthur C. Clarke

Fortunately for us, ray tracing is far simpler than magic.

Acknowledgments

This thesis would not have been possible without the exceptional support of my advisors, Prof. Claude Oestges and Prof. Laurent Jacques. Since we began writing the proposal for this subject in 2020, you have been there for me throughout this journey. While I am aware that the academic world sometimes prioritizes quantity over quality, I never felt this pressure. Instead, I was fortunate to work in a remarkably healthy environment where scientific rigor came first. For this, and for your mentorship, I am truly grateful.

I would like to thank the members of my thesis committee: my two advisors; Prof. Christophe Craeye, chairman; Prof. Christophe De Vleeschouwer, secretary; Prof. Philippe De Doncker, from Université Libre de Bruxelles; Prof. Enrico Maria Vitucci, from Università di Bologna; and Dr. Jakob Hoydis, from NVIDIA Research. Thank you all for your time and insightful feedback on this work.

Special thanks go to Prof. Vittorio Degli Esposti and Prof. Enrico Maria Vitucci. During my visits in April and from September to December 2024, your hospitality made me feel like a member of the team from the very first day.

To my colleagues: thank you for being the best part of this journey. While I learned a lot from the conferences and workshops we attended, my favorite memories will always be the offline moments—our coffee breaks, our ping-pong tournaments, and our GeoGuessr sessions.

I must also acknowledge the invisible help of the open-source community. To the developers who create free tools and the forum members who help solve code issues: your collective effort is the backbone of my research, and my work has benefited greatly from it.

Finally, to my family, friends, and loved ones: thank you for your support.

Preface

Today, we are connected more than ever. Whether we are indoors or outdoors, our devices constantly communicate with each other through wireless signals. But have you ever wondered how these signals travel from one device to another? How do they navigate through buildings, trees, and other obstacles in their path? How do we model the complex behavior of radio wave propagation? How do we simulate it on a computer?

In this book, we will explore the fascinating world of radio wave propagation modeling using a technique called ray tracing. Ray tracing is a powerful method that allows us to simulate how radio waves interact with their environment. By tracing the paths of individual rays, we can gain insight into how signals propagate, reflect, diffract, and scatter in different scenarios.

To understand the title of this book, "Differentiable Ray Tracing for Radio Propagation Modeling", we will break it down into its three main components: radio wave propagation, ray tracing, and differentiability.

The first component, radio wave propagation, is discussed in Part 'Understanding' to provide the necessary background for constructing a ray tracing model. Part 'Building' explains, as its name suggests, how to build a modern ray tracer from scratch, covering the mathematical foundations and algorithms required to accurately and efficiently simulate ray propagation, and explaining what differentiability means in this context. To conclude, Part 'Using' explores practical applications of differentiable ray tracing in radio propagation modeling.

Open-source commitment: The source code for building this book, along with complementary material, can be found at

`https://github.com/jeertmans/DiffeRT-thesis`.

Any issues should be reported on the GitHub repository, and errata will be published there as well.

Contributions

This chapter provides a comprehensive overview of the original contributions made during my thesis research.

1 Core Methodological Contributions

The primary focus of this thesis was the modernization of radio propagation modeling through differentiable programming and machine learning. The core methodological contributions can be summarized as follows:

- I developed the *min-path-tracing* method for exact multipath computation. Paths are formulated as a constrained minimization problem that uses per-interaction constraints, enabling flexible handling of complex geometries such as spheres and metasurfaces. While computationally more expensive than Fermat-based approaches, this method is well-suited for small-scale simulations requiring geometric flexibility.
- I introduced a physics-inspired formulation for smoothing hard visibility discontinuities in differentiable ray tracing. This technique eliminates zero-gradient plateaus caused by visibility transitions and enables stable optimization in heavily shadowed propagation scenarios.
- In collaboration with Profs. Vittorio Degli-Esposti and Enrico Maria Vitucci, I introduced multipath lifetime maps, a novel representation of the spatial distribution of multipath interactions. Based on the concept of *multipath lifetime*, this representation quantifies the stability of interactions with respect to small scene perturbations.
- With Dr. Nicola Di Cicco and Profs. Vittorio Degli-Esposti and Enrico Maria Vitucci, I developed a machine-learning-assisted ray tracing method using generative flow networks to predict valid interaction sequences. This approach reduces reliance on exhaustive enumeration while learning geometrical rather than electromagnetic properties, operating alongside (not replacing) the ray tracer. Early results demonstrate promising potential for accelerating large-scale simulations and motivate its applicability to variable scene sizes.
- With Sophie Lequeu and Prof. Benoît Legat, I developed a GPU-efficient differentiable path tracing framework using fixed-iteration optimization

on planar diffraction–reflection sequences. Implicit differentiation is used to shorten the automatic differentiation path, enabling efficient gradient computation while maintaining parallelism and bounded memory.

2 Scientific Publications

In addition to the publications listed below, I have also contributed to the not-yet-published book of the European Cooperation in Science and Technology (COST) Intelligence-Enabling Radio Communications for Seamless Inclusive Interactions (INTERACT) action. More specifically, I co-wrote the section on machine learning techniques applied to the modeling of radio propagation. Several publications below were first shared as reports within the aforementioned COST action.

Preprints

[1] Jérome Eertmans, Enrico M. Vitucci, Vittorio Degli-Esposti, Nicola Di Cicco, Laurent Jacques, and Claude Oestges. "Transform-Invariant Generative Ray Path Sampling for Efficient Radio Propagation Modeling". Submitted to npj Wireless Technology. 2026. DOI: 10.48550/arXiv.2603.01655.

[2] Jérome Eertmans, Sophie Lequeu, Benoît Legat, Laurent Jacques, and Claude Oestges. "Fast, Differentiable, GPU-Accelerated Ray Tracing for Multiple Diffraction and Reflection Paths". Accepted at EuCAP 2026 - 20th European Conference on Antennas and Propagation. 2025. DOI: 10.48550/arXiv.2510.16172.

Conference Proceedings

[3] Jérome Eertmans, Nicola Di Cicco, Claude Oestges, Laurent Jacques, Enrico M. Vitucci, and Vittorio Degli-Esposti. "Towards Generative Ray Path Sampling for Faster Point-to-Point Ray Tracing". In: *2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*. 2025, pp. 1–6. DOI: 10.1109/ICMLCN64995.2025.11140249.

[4] Jérome Eertmans, Claude Oestges, and Laurent Jacques. "Demonstrating DiffeRT: An Open-Source Library for Optimizing Radio Networks with Differentiable Ray Tracing". In: *2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*. 2025, pp. 1–2. DOI: 10.1109/ICMLCN64995.2025.11139997.

[5] Jérome Eertmans, Enrico M. Vitucci, Vittorio Degli-Esposti, Laurent Jacques, and Claude Oestges. "Comparing Differentiable and Dynamic Ray Tracing: Introducing the Multipath Lifetime Map". In: *2025 19th European Conference on Antennas and Propagation (EuCAP)*. 2025, pp. 1–5. DOI: 10.23919/EuCAP63536.2025.10999736.

[6] Jérome Eertmans, Laurent Jacques, and Claude Oestges. "Fully Differentiable Ray Tracing via Discontinuity Smoothing for Radio Network Optimization". In: *2024 18th European Conference on Antennas and Propagation (EuCAP)*. 2024, pp. 1–5. DOI: 10.23919/EuCAP60739.2024.10501570.

[7] Jérome Eertmans, Claude Oestges, and Laurent Jacques. "Min-Path-Tracing: A Diffraction Aware Alternative to Image Method in Ray Tracing". In: *2023 17th European Conference on Antennas and Propagation (EuCAP)*. 2023, pp. 1–5. DOI: 10.23919/EuCAP57121.2023.10132934.

Journal Articles

[8] Jérome Eertmans. "Manim Slides: A Python package for presenting Manim content anywhere". In: *Journal of Open Source Education* 6.66 (2023). The Open Journal publishes brief articles that undergo an open peer-review process on GitHub issues., p. 206. DOI: 10.21105/jose.00206.

[9] Jérome Eertmans, Claude Oestges, and Laurent Jacques. "DiffeRT2d: A Differentiable Ray Tracing Python Framework for Radio Propagation". In: *Journal of Open Source Software* 9.98 (2024). The Open Journal publishes brief articles that undergo an open peer-review process on GitHub issues., p. 6915. DOI: 10.21105/joss.06915.

Each of the publications listed above was accompanied by open-source code and tutorials, enabling readers to reproduce the results and figures presented in the papers.

Finally, although it did not result in a publication, I developed a fast graph traversal algorithm for generating path candidates in DiffeRT, as discussed in Appendix B. For performance reasons, I implemented this algorithm in Rust and integrated it into the Python codebase using PyO3. The Rust and Python implementations are both open source, and the latter is packaged as `differt-core` and contains other performance-critical components of DiffeRT, such as scene-file parsers, that cannot be efficiently implemented in JAX.

3 Conferences, Scientific Workshops, Seminars, and Visits

Thanks to the support of my two supervisors and funding instruments like the Fonds de la Recherche Scientifique - FNRS (FNRS) and the COST, I had the opportunity to:

- Participate in four international meetings and one doctoral school of the COST INTERACT action.
- Attend and present at the following conferences: European Conference on Antennas and Propagation (EuCAP) 2023 in Florence (Italy), EuCAP 2024 in Glasgow (Scotland), EuCAP 2025 in Stockholm (Sweden), International Conference on Machine Learning for Communications and Networking (ICMLCN) 2025 in Barcelona (Spain), and EuCAP 2026 in Dublin (Ireland) (**Best Propagation Paper Award**).
- Attend the following workshops and seminars: the FNRS-Contact Group on "Wavelets and Applications" (CGWA) online in 2021, the 42nd WIC Symposium on Information Theory and Signal Processing in the Benelux in Louvain-la-Neuve (Belgium) in 2022, the Technical Seminar on satcom from the IEEE ComVT Benelux Chapter in Sint-Niklaas (Belgium) in 2022, the Silicon Austria Labs (SAL) Symposium on 6G in Linz (Austria) in 2023.
- Co-organize the "Perspectives on Wireless Networks" workshop in Louvain-la-Neuve (Belgium), with international speakers and attendance from 50+ researchers.
- Visit the University of Siegen (Germany) for two days in July 2023, and the University of Bologna (Italy) for two weeks in April 2024 and for four months in late 2024.

All the presentations I have given during these events are available on my personal website*. These include slides, papers, and recordings when available.

* https://eertmans.be

4 Open Source Software

As a strong advocate for open-source software in research, I have contributed to and maintained several projects supporting my work.

My most significant projects are:

- **DiffeRT** (`jeertmans/DiffeRT`): I developed and released the DiffeRT library, a differentiable ray tracer designed for my research. It is built on top of JAX to enable high-performance, differentiable code. It is fully open source, along with its 2D variant, DiffeRT2d (`jeertmans/DiffeRT2d`).
- **Manim Slides** (`jeertmans/manim-slides`): I developed this open-source presentation tool to produce higher-quality presentations at conferences, which you can find on my personal website. I am proud to say that it is used by other researchers and teachers around the world for teaching, conference presentations, or thesis defenses; see the example gallery.

The "Projects" page on my personal website provides a broader overview of my open-source contributions. The rest of my projects are available on my GitHub profile (`jeertmans`), some of which have received significant attention from the community in terms of the number of stars and forks.

For all my projects, I have dedicated a significant amount of time to ensuring that the code is well-documented, tested, and maintained to facilitate its use by other researchers and developers.

I also believe that contributing to existing open-source projects is essential for advancing the field. During my thesis, I have contributed code, bug fixes, and documentation to several well-known libraries.

Here is a list of the most important contributions:

- Implementing the missing `jax.scipy.special.fresnel` function (`#jax/22843`).
- Adding support for all modes in `numpy.correlate` for Numba's just-in-time (JIT) compilation (`#numba/7543`).
- Working on removing the dependency on FFmpeg in Manim[†] (`#manim/3501`).

I have made many more contributions to various open-source projects, which can be found at this URL (for printed versions, refer to my GitHub profile).

[†] While not directly related to my research, this contribution helped me develop Manim Slides, the tool I use to present at conferences and workshops.

5 Peer Reviews

I have performed peer reviews for the following journals and conferences:

- European Conference on Antennas and Propagation: EuCAP 2026 (5 reviews).
- IEEE: Access (5 reviews), Transactions on Vehicular Technology (7 reviews), Antennas and Wireless Propagation Letters (1 review), and Open Journal of the Communications Society (1 review).
- Nature Partner Journals: Wireless Technology (1 review).
- The Open Journal: Journal of Open Source Software (1 review, see `#joss-reviews/9384`).

Evidence of my reviews can be found on my ORCID profile: `https://orcid.org/0000-0002-5579-5360`.

6 In This Book

All the figures in this book are original, except for the figures in Section 2.1 (History of Ray Tracing). These figures are new or modified versions of the original figures in the cited papers, and they were generated from data created with DiffeRT2d, DiffeRT, or Sionna RT. Due to the nature of this writing, my contributions are spread across several chapters. The following is a non-exhaustive overview of how my publications relate to the content of this book:

- The min-path-tracing (MPT) [7] method is described in Section 3.1.2.
- The smoothing technique for a *fully differentiable ray tracing framework* [6] and its applications are described in Section 4.4.
- The multipath lifetime map (MLM) [5] method is described in Section 7.2.
- The machine-learning-assisted generative path sampling method [1, 3] is described in Section 7.1.
- The GPU-accelerated differentiable path tracing framework [2] is described in Sections 3.1.3, 4.3.4 and 5.6.3.

This manuscript is written as a textbook and mainly focuses on the qualitative and conceptual aspects of the methods, with a strong emphasis on physical intuition and practical implications. Readers interested in mathematical details, algorithmic implementations, and quantitative results are encouraged to refer to the original publications cited throughout the book. My publications are open access and freely available online through their preprint versions.

Contents

Understanding

1 The Propagation of Radio Waves

The study of radio wave propagation establishes a connection between the theoretical foundation of electromagnetic theory and the practical implementation of wireless communication systems. In order to construct sophisticated algorithms for ray tracing and channel modeling (topics that will be addressed in the subsequent chapters of this book), it is essential to first understand the fundamental physics governing the transmission of energy from a source to a destination. This chapter serves as a short introduction to the fundamental concepts required to understand the propagation of radio waves, rigorously deriving their behavior as they traverse free space, interact with material boundaries, and navigate complex environments. In the next sections, we assume that the reader is familiar with vector analysis and basic electromagnetic theory. For a more in-depth introduction to electromagnetic theory, including the necessary background in vector analysis, we recommend reading Griffiths's *Introduction to Electrodynamics* [1], as well as Saunders and Aragón-Zavala's *Antennas and Propagation for Wireless Communication Systems* [2] for more applied content on radio propagation. Moreover, the present chapter takes great inspiration from both of these textbooks, and we will often refer to them for further reading.

[1]: Griffiths (2024), *Introduction to Electrodynamics*

[2]: Saunders et al. (2024), *Antennas and Propagation for Wireless Communication Systems*

The foundation of ray tracing is based on two pillars: the concept of a radio channel and classical electrodynamics, mathematically described by Maxwell's equations. We first define what we mean by a radio channel. Then, starting from Maxwell's equations, we derive the wave equation to elucidate the oscillatory nature of electromagnetic fields. From there, we examine the behavior of plane waves, including their polarization and propagation through lossy media. We then explore how these waves interact with material boundaries through reflection and refraction, governed by Snell's law and Fresnel's equations, as well as

other effects such as diffraction and scattering. Finally, we introduce the fundamentals of antennas, which serve as the sources and sinks of these radio waves, and establish the basic link budget equation for line-of-sight communication.

1.1 Fundamentals of the Wireless Channel

The wireless channel is the physical medium that connects the transmitter and receiver, and understanding its characteristics is crucial for the design and analysis of any wireless communication system. Whether for cellular networks, satellite links, or local area networks, the channel imposes fundamental limits on performance.

A generic communication system can be described by the architecture proposed by Claude Shannon [3], usually depicted as consisting of five functional blocks: the information source, the transmitter, the channel, the receiver, and the destination (see Figure 1.1). The transmitter processes the information into a signal suitable for the channel, while the receiver's task is to recover this information from the received signal, which has been degraded by the channel.

[3]: Shannon (1948), *A mathematical theory of communication*

The degradation caused by the channel can be mathematically modeled as a linear time-varying filter. The received signal $y(t)$ is generally expressed as the convolution of

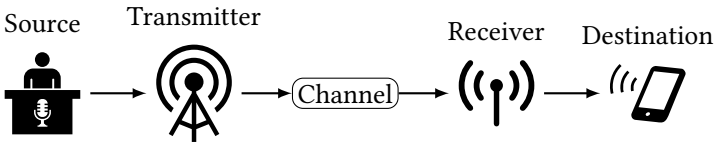


Figure 1.1: Simplified schematic of a generic communication system where a podcaster broadcasts a show to a listener's smartphone through a wireless channel, inspired by [3, Fig. 1] and [2, Fig. 1.2].

the transmitted signal $x(t)$ with the channel's impulse response $h(t)$, corrupted by additive noise $n(t)$,

$$y(t) = h(t) * x(t) + n(t). \tag{1.1}$$

This formulation clearly distinguishes between two main types of impairments: additive noise and multiplicative fading effects.

1.1.1 Additive Noise

The term $n(t)$ represents additive noise, which generally arises from thermal fluctuations within the receiver electronics or from external interference sources such as cosmic radiation and man-made signals. In many theoretical analyses, this noise is modeled as a Gaussian random process. While noise sets a strict lower bound on the detectable signal level, in many practical wireless scenarios—particularly in dense urban settings—system performance is constrained primarily by the complex multiplicative effects of the channel rather than simple additive noise.

1.1.2 Multiplicative Effects

The impulse response $h(t)$ encapsulates the multiplicative effects resulting from the complex propagation environment. As the radio wave travels from the transmitter to the receiver, it interacts with physical objects through several mechanisms:

- *Reflection* from large, smooth surfaces such as building facades or the ground.
- *Transmission and absorption* as the wave passes through materials like glass, brick, or foliage, losing energy in the process.
- *Diffraction* around sharp edges like building corners or rooflines, allowing signals to reach shadowed areas.

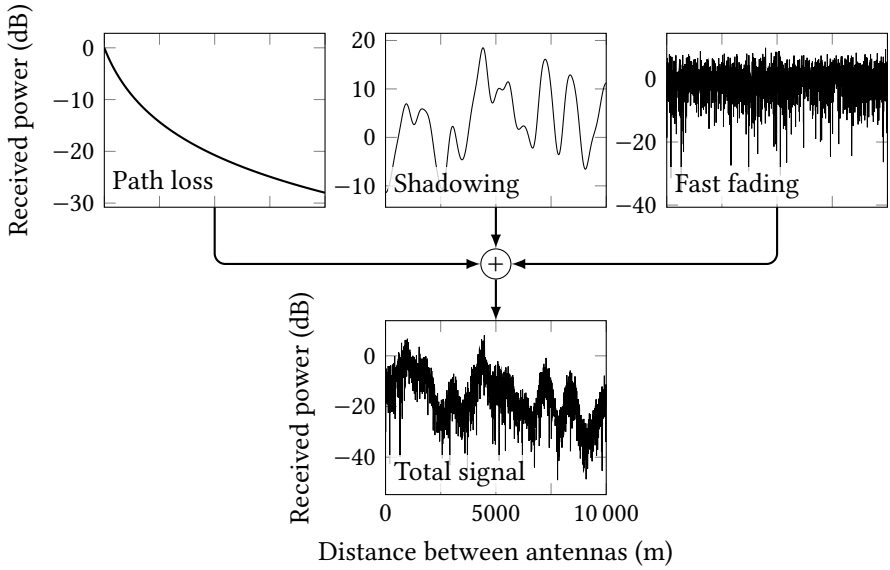


Figure 1.2: The three scales of signal variation, adapted from [2, Fig. 1.5].

- *Scattering* from rough surfaces or objects that are small relative to the wavelength.

These interactions cause the signal to arrive at the receiver via multiple paths, leading to constructive and destructive interference known as fading. These variations occur over different spatial scales, as illustrated in Figure 1.2.

- *Path loss* is the deterministic decay of average signal power as the distance between transmitter and receiver increases. It represents the large-scale trend influenced by the geometry of the environment and free-space spreading.
- *Shadowing* represents medium-scale variations caused by specific obstacles blocking the direct path, such as buildings or hills. Shadowing is often modeled as a log-normal random process superimposed on the path loss [2, Chap. 9].
- *Fast fading* consists of the small-scale, rapid fluctuations in signal amplitude and phase occurring over distances of just a few wavelengths. This results from

[2]: Saunders et al. (2024), *Antennas and Propagation for Wireless Communication Systems*

the vector addition of multiple signal copies arriving at the receiver with different phases.

The next chapters will present how ray tracing can be used to model these multiplicative effects by simulating the physical interactions of rays with the environment, providing a powerful tool for predicting channel behavior in complex scenarios.

1.1.3 The Electromagnetic Spectrum

The electromagnetic spectrum is a fundamental resource for wireless transmissions. Commercial and military radio systems typically operate within the frequency range of 3 kHz to 300 GHz. The choice of operating frequency involves several trade-offs. For example, higher frequency bands provide larger available bandwidths for high-speed data transmission, but this comes at the cost of increased path loss and weaker penetration capabilities through building materials. As a rule of thumb, the free-space path loss formula predicts that signal power decreases with the square of both frequency and distance. Figure 1.3 and Table 1.1 provide a summary of the standard frequency bands used in wireless communications [4].

[4]: (2025), *ITU-R Recommendation V.431: Nomenclature of the frequency and wavelength bands used in telecommunications*

Throughout this book, exact frequency values are often secondary, as we operate under the assumption that the wave-

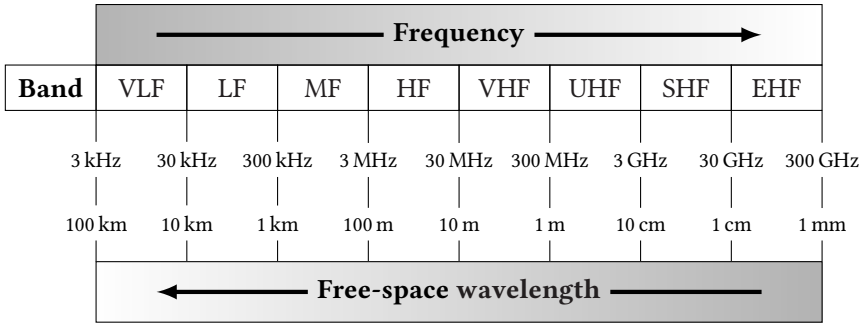


Figure 1.3: Frequency spectrum used in wireless communications, adapted from [2, Fig. 1.6].

Table 1.1: Naming conventions for frequency bands, reproduced from [2, Table 1.1].

Band name	Frequency range	Band name	Frequency range
Very low frequency (VLF)	3 – 30 kHz	L band	1 – 2 GHz
Low frequency (LF)	30 – 300 kHz	S band	2 – 4 GHz
Medium frequency (MF)	0.3 – 3.0 MHz	C band	4 – 8 GHz
High frequency (HF)	3 – 30 MHz	X band	8 – 12 GHz
Very high frequency (VHF)	30 – 300 MHz	Ku band	12 – 18 GHz
Ultra high frequency (UHF)	0.3 – 3.0 GHz	K band	18 – 26 GHz
Super high frequency (SHF)	3 – 30 GHz	Ka band	26 – 40 GHz
Extra high frequency (EHF)	30 – 300 GHz	V band	40 – 75 GHz
		W band	75 – 111 GHz

length is sufficiently small to justify the high-frequency ray-optical approximation. The usage limits of this approximation are detailed in Chapter 2.

For a comprehensive overview of historical developments and wireless communication systems, we refer the reader to [2, Chap. 1].

[2]: Saunders et al. (2024), *Antennas and Propagation for Wireless Communication Systems*

The previous section described the *system-level consequences* of propagation through the channel response $h(t)$ (path loss, shadowing, and fading). We now move to the *field-level mechanism*: Maxwell's equations and constitutive laws, which explain how these effects emerge from electromagnetic wave behavior in space.

1.2 Electromagnetic Foundations

The propagation of radio waves is governed by the laws of classical electrodynamics. These laws, formulated by James Clerk Maxwell in the 19th century [5], unify electric and magnetic phenomena into a single theoretical framework. In this section, we review the fundamental field equations and the constitutive relations that describe how electromagnetic fields interact with matter.

[5]: Maxwell (1865), *VIII. A dynamical theory of the electromagnetic field*

1.2.1 Maxwell's Equations

The behavior of electromagnetic fields is described by Maxwell's equations. In their differential form, which applies at every point in space and time, they are given by

$$\nabla \cdot \mathbf{D} = \rho_v, \tag{1.2}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{1.3}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \tag{1.4}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}, \tag{1.5}$$

where $\mathbf{E}$ is the electric field intensity ($\mathrm{V\,m^{-1}}$), $\mathbf{H}$ is the magnetic field intensity ($\mathrm{A\,m^{-1}}$), $\mathbf{D}$ is the electric flux density ($\mathrm{C\,m^{-2}}$), $\mathbf{B}$ is the magnetic flux density (T), $\mathbf{J}$ is the electric current density ($\mathrm{A\,m^{-2}}$), and ρ_v is the electric volume charge density ($\mathrm{C\,m^{-3}}$).

Gauss's law (1.2) establishes electric charge as the source of the displacement field [1, Chap. 2], while (1.3) asserts the non-existence of magnetic monopoles [1, Chap. 5]. Faraday's law (1.4) describes how a time-varying magnetic flux induces an electric field [1, Chap. 7]. Finally, the Ampère–Maxwell law (1.5) introduces the displacement current, which allows electromagnetic waves to propagate through a vacuum [1, Chap. 5 & 7].

[1]: Griffiths (2024), *Introduction to Electrodynamics*

1.2.2 Constitutive Relations

Maxwell's equations alone are not sufficient to solve for the fields; they must be supplemented by constitutive relations that characterize the macroscopic properties of the medium. For a *linear*, *isotropic*, and *homogeneous* material, these relations are given by

$$\mathbf{D} = \epsilon \mathbf{E}, \quad \mathbf{B} = \mu \mathbf{H}, \quad \mathbf{J} = \sigma \mathbf{E}, \tag{1.6}$$

where ϵ is the *permittivity* ($\mathrm{F\,m^{-1}}$), μ is the *permeability* ($\mathrm{H\,m^{-1}}$), and σ is the *conductivity* ($\mathrm{S\,m^{-1}}$).

In free space, these parameters reduce to $\epsilon_0 \approx 8.854 \times 10^{-12}\,\mathrm{F\,m^{-1}}$ and $\mu_0 \approx 4\pi \times 10^{-7}\,\mathrm{H\,m^{-1}}$. The speed of light in vacuum is then related to these constants by $c = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \approx 3 \times 10^8\,\mathrm{m\,s^{-1}}$.

It is often convenient to express the permittivity and permeability relative to their free-space values

$$\epsilon = \epsilon_r \epsilon_0, \quad \mu = \mu_r \mu_0, \tag{1.7}$$

where ϵ_r and μ_r are the dimensionless *relative permittivity* (or dielectric constant) and *relative permeability*, respectively.

Finally, we note that in *anisotropic* media, the relationships between the fields depend on their direction, meaning that $\mathbf{D}$ is not necessarily parallel to $\mathbf{E}$, and $\mathbf{B}$ is not parallel to $\mathbf{H}$. In such cases, the scalar permittivity and permeability must be replaced by tensors. However, since ray tracing techniques are primarily developed for propagation in large-scale homogeneous environments, we will restrict our discussion to isotropic media in this book.

1.2.3 The Time-Harmonic Case

General time-varying electromagnetic fields can be complex to analyze directly in the time domain. However, due to the linearity of Maxwell's equations, any arbitrary time-dependent field can be expressed as a superposition of time-harmonic (sinusoidal) components via the Fourier transform. Therefore, it is often sufficient and more convenient to solve Maxwell's equations for a single frequency ω at a time.

For a time-harmonic field oscillating at angular frequency $\omega = 2\pi f$, we can represent the instantaneous real-valued field $\mathscr{E}(\mathbf{r}, t)$ using a complex phasor $\mathbf{E}(\mathbf{r})$ as

$$\mathscr{E}(\mathbf{r}, t) = \mathrm{Re}\left[\mathbf{E}(\mathbf{r}) e^{j\omega t}\right], \tag{1.8}$$

where j is the imaginary unit, $\mathbf{r}$ is the position vector, and $\mathrm{Re}[\cdot]$ denotes the real part of the complex expression.

In this phasor domain, the time derivative operator $\frac{\partial}{\partial t}$ is replaced by multiplication with $j\omega$. Consequently, Maxwell's equations in a source-free region ($\rho_v = 0, \mathbf{J} = 0$) simplify to

$$\nabla \cdot \mathbf{E} = 0, \tag{1.9}$$

$$\nabla \cdot \mathbf{H} = 0, \tag{1.10}$$

$$\nabla \times \mathbf{E} = -j\omega\mu\mathbf{H}, \tag{1.11}$$

$$\nabla \times \mathbf{H} = j\omega\epsilon\mathbf{E}. \tag{1.12}$$

Here, we have used the constitutive relations to express $\mathbf{B}$ and $\mathbf{D}$ in terms of $\mathbf{H}$ and $\mathbf{E}$.

For lossy media, where $\sigma \neq 0$, the conductivity can be incorporated into a *complex permittivity*

$$\epsilon_c = \epsilon - j\frac{\sigma}{\omega}. \tag{1.13}$$

By substituting Ohm's law ($\mathbf{J} = \sigma\mathbf{E}$) into Ampère's law, we obtain

$$\nabla \times \mathbf{H} = j\omega\epsilon\mathbf{E} + \sigma\mathbf{E} = j\omega\left(\epsilon - j\frac{\sigma}{\omega}\right)\mathbf{E} = j\omega\epsilon_c\mathbf{E}. \tag{1.14}$$

This formulation allows us to treat lossy materials as formally lossless dielectrics with a complex-valued permittivity.

1.2.4 The Wave Equation

By taking the curl of Faraday's law

$$\nabla \times (\nabla \times \mathbf{E}) = -j\omega\mu(\nabla \times \mathbf{H}), \tag{1.15}$$

and substituting the Ampère–Maxwell law, we can decouple the electric and magnetic fields to obtain a single

equation for the electric field

$$\nabla \times (\nabla \times \mathbf{E}) = -j\omega\mu(j\omega\epsilon\mathbf{E}). \tag{1.16}$$

Using the vector identity $\nabla \times (\nabla \times \mathbf{A}) = \nabla(\nabla \cdot \mathbf{A}) - \nabla^2\mathbf{A}$ and the fact that $\nabla \cdot \mathbf{E} = 0$ in a source-free region, we arrive at the vector *Helmholtz equation*

$$\nabla^2\mathbf{E} + k^2\mathbf{E} = 0, \tag{1.17}$$

where $k = \omega\sqrt{\mu\epsilon}$ is the *wavenumber* ($\mathrm{rad\,m^{-1}}$).

This equation describes the spatial variation of the field intensity for a monochromatic wave and forms the basis for the ray-optical approximations discussed in Chapter 2.

1.2.5 The Poynting Vector

Electromagnetic waves carry energy as they propagate. The flow of electromagnetic power is described by the *Poynting vector* $\mathbf{S}$ ($\mathrm{W\,m^{-2}}$), defined as the cross product of the electric and magnetic fields

$$\mathbf{S} = \mathbf{E} \times \mathbf{H}. \tag{1.18}$$

The vector $\mathbf{S}$ represents the instantaneous directional energy flux density (power per unit area) at a given point in space. It points in the direction of energy propagation.

For theory and applications involving time-harmonic fields, the instantaneous power oscillates at twice the frequency of the wave. In most practical engineering problems, such as calculating antenna radiation patterns or link budgets, we are interested in the *time-average power flow*. For phasor fields $\mathbf{E}$ and $\mathbf{H}$, the time-average Poynting vector is given by

$$\mathbf{S}_{\mathrm{avg}} = \frac{1}{2}\,\mathrm{Re}\left[\mathbf{E} \times \mathbf{H}^*\right], \tag{1.19}$$

where $\mathbf{H}^*$ denotes the complex conjugate of the magnetic field phasor. This quantity is fundamental for determining the power density of plane waves, as we will see in the following section.

1.3 Plane Waves

While Maxwell's equations govern all classical electromagnetic phenomena, solving them for arbitrary boundary conditions can be analytically intractable. A powerful approach is to decompose complex fields into a superposition of simpler components. Among these, the *plane wave* is the most fundamental building block. Through spectral decomposition, arbitrary wave fields can be expressed as a weighted sum of plane waves.

In the context of radio propagation modeling, particularly ray tracing, we frequently operate in the *far-field* region of the transmitting antenna. Here, the spherical wavefronts have expanded sufficiently to be approximated as locally flat surfaces (see Figure 1.4). This *plane wave approximation* allows us to treat radio wave interactions with the environment using simplified plane wave physics, which is central to many methods discussed in this book.

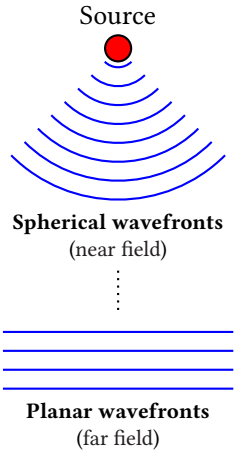


Figure 1.4: Spherical wavefronts from a point source can be approximated as locally planar in the far-field region.

1.3.1 Field Relationships

A uniform plane wave propagating in the $+z$ direction is characterized by having no field variations in the transverse xy-plane. The electric and magnetic fields are perpendicular to the direction of propagation (forming a transverse electromagnetic (TEM) wave) and are mutually orthogonal.

For a wave traveling in free space, if we align the electric field vector with the x-axis, the corresponding phasor fields

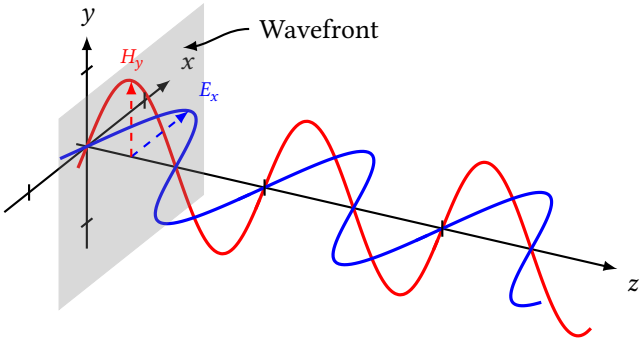


Figure 1.5: Representation of a plane wave propagating in the $+z$ direction.

are

$$\mathbf{E}(z) = E_0 e^{-jkz}\hat{\mathbf{x}}, \tag{1.20}$$

$$\mathbf{H}(z) = H_0 e^{-jkz}\hat{\mathbf{y}}, \tag{1.21}$$

where E_0 and H_0 are the complex amplitudes at $z = 0$. The orthogonality condition is generally expressed as

$$\mathbf{H} = \frac{1}{Z}(\hat{\mathbf{k}} \times \mathbf{E}), \tag{1.22}$$

where $\hat{\mathbf{k}}$ is the direction of propagation and Z is the intrinsic impedance of the medium (see below).

1.3.2 Wave Impedance

The ratio of the electric field magnitude to the magnetic field magnitude is a constant of the medium known as the *intrinsic impedance* Z, where

$$Z = \frac{E_x}{H_y} = \sqrt{\frac{\mu}{\epsilon}}. \tag{1.23}$$

In free space, this value is approximately

$$Z_0 = \sqrt{\frac{\mu_0}{\epsilon_0}} \approx 377\ \Omega. \tag{1.24}$$

1.3.3 Phase Velocity and Wavelength

The speed at which the phase fronts of the wave propagate is called the *phase velocity* v, with

$$v = \frac{\omega}{k} = \frac{1}{\sqrt{\mu\epsilon}}. \tag{1.25}$$

The distance over which the phase changes by 2π radians is the *wavelength* λ

$$\lambda = \frac{v}{f} = \frac{2\pi}{k}. \tag{1.26}$$

As we will see later, it is common practice to compare the dimensions of objects to the wavelength, rather than using absolute values.

1.3.4 Power Density

The power carried by a plane wave can be calculated using the time-average Poynting vector. Substituting the plane wave expressions into the definition derived in the previous section, we obtain

$$\mathbf{S}_{\text{avg}} = \frac{1}{2}\,\text{Re}\,(\mathbf{E} \times \mathbf{H}^*) = \frac{|E_0|^2}{2Z}\hat{\mathbf{z}}. \tag{1.27}$$

This indicates that power flows in the direction of propagation and is proportional to the square of the electric field amplitude.

1.3.5 Polarization Fundamentals

The polarization of a wave describes the locus of the tip of the electric field vector as a function of time at a fixed point in space. Figure 1.6 illustrates the three fundamental polarization states.

- *Linear polarization* is when the electric field oscillates along a single line. This is the most common case for simple dipole antennas.
- *Circular polarization* is when the electric field vector rotates in a circle around the direction of propagation, see Figure 1.7. This is useful for satellite communications as it is robust to ionospheric rotation.
- *Elliptical polarization* is the most general case, where the field vector traces an ellipse.

Mathematically, the polarization state can be completely described with the sum of two orthogonal linearly polarized waves. For boundary interactions, these components are defined relative to the *plane of incidence*, i.e., the plane spanned by the incident propagation direction and the local surface normal. This means that the electric field can be represented by two components, usually denoted s (from *senkrecht*, German for perpendicular) and p (parallel), or by using the subscripts $\perp$ and $\parallel$. To avoid confusion, we only use the latter notation in this book. In Chapter 2, this decomposition is formalized using Jones vectors and explicit basis rotations between global and local interaction frames.

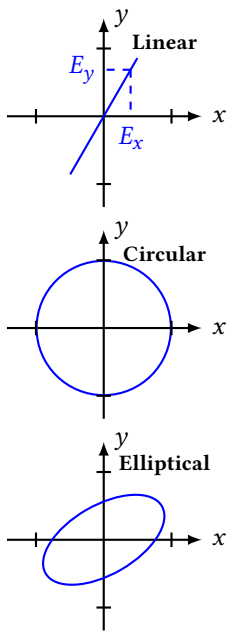


Figure 1.6: Possible polarization states for a plane wave propagating in the $+z$ direction.

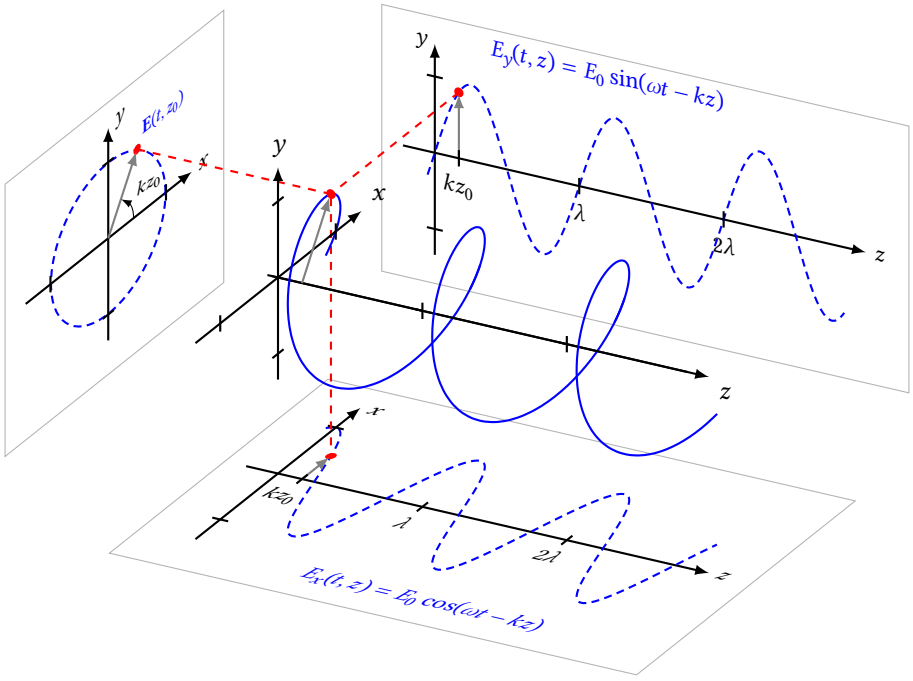


Figure 1.7: Representation of a circularly polarized plane wave propagating in the $+z$ direction.

1.3.6 Propagation in Lossy Media

In a lossy medium with conductivity σ, the permittivity becomes complex, leading to a complex propagation constant $\gamma = \alpha + j\beta$. The real part α is the *attenuation constant* (m^{-1}), given by

$$\alpha = \omega \sqrt{\frac{\mu\epsilon}{2}\left(\sqrt{1 + \left(\frac{\sigma}{\omega\epsilon}\right)^2} - 1\right)}, \tag{1.28}$$

which causes the wave amplitude to decay exponentially as

$$|\mathbf{E}(z)| = |E_0| e^{-\alpha z}. \tag{1.29}$$

The distance $\delta = 1/\alpha$ over which the amplitude decays to $\frac{1}{e}$ (about 37 %) of its initial value is called the *skin depth*.

We generally identify two types of lossy media:

- *Good dielectrics* or *insulators*, where $\sigma \ll \omega\epsilon$. In this case, the attenuation constant is low, approximately equal to
$$\alpha \approx \frac{\sigma}{2}\sqrt{\frac{\mu}{\epsilon}}, \tag{1.30}$$
allowing penetration of the wave into the medium.
- *Good conductors*, where $\sigma \gg \omega\epsilon$. In this case, the attenuation constant is high, approximately equal to
$$\alpha \approx \sqrt{\frac{\omega\mu\sigma}{2}}, \tag{1.31}$$
and the skin depth is thus very small, meaning waves are confined to the surface, which effectively makes metals perfect reflectors.

We conclude this section by noting that, in ray tracing, we typically consider wave propagation to occur in air, which is treated as free space (a lossless medium). Except for a few specific scenarios, such as modeling propagation through dense foliage or thick walls, we generally do not account for extended propagation within lossy media. Furthermore, while the plane wave formulation suggests a constant power density over distance, realistic sources radiate spherical waves. To account for the resulting decrease in power density due to spatial spreading, ray tracing models waves macroscopically as expanding spherical tubes or individual rays. The plane wave approximation is then strictly restricted to modeling local interactions, such as reflections and transmissions at boundaries. This fundamental interplay between macroscopic spherical spreading and local plane wave interactions will be further discussed in the next section and in Chapter 2.

1.4 Fundamentals of Antennas

Before discussing how waves reflect and diffract, we must understand how they are generated. Antennas act as the

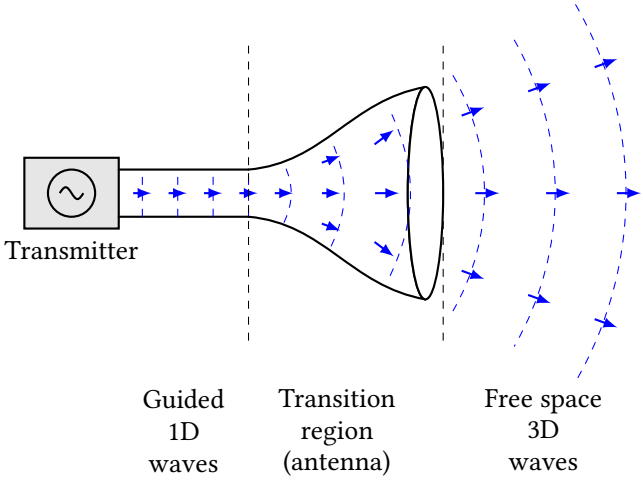


Figure 1.8: Antenna transition regions between guided and free-space waves, reproduced from [2, Fig. 4.1].

interface between the guided waves in a circuit and the free-space electromagnetic waves, converting electrical fields traveling in a transmission line into radiated fields (see Figure 1.8), and vice versa. While antennas are complex devices and ray tracing methods are usually developed independently of their behavior, it is important to understand how they work in order to understand both the physical assumptions underlying ray tracing and how various antenna parameters are used to accurately estimate the received power.

1.4.1 The Radiation Mechanism

Time-varying currents on a conductor generate electromagnetic fields. Fundamentally, the acceleration of charge is the source of radiation. In the region immediately surrounding the antenna, known as the *near field*, the electric and magnetic fields are complex and reactive, storing energy rather than propagating it. However, at a sufficient distance, the radiated fields dominate, detaching from the source and propagating as waves.

1.4.2 Far-Field Approximation

The region where the angular field distribution is essentially independent of the distance from the antenna is called the *far-field* (or Fraunhofer) region. This region begins at the *Fraunhofer distance* d_f (see Figure 1.9), defined as

$$d_f = \frac{2D^2}{\lambda}, \tag{1.32}$$

where D is the maximum dimension of the antenna and λ is the wavelength.

In the far field ($r > d_f$), the spherical wavefronts emitted by the antenna appear locally planar to a receiver. This *plane wave approximation* will allow us to use the plane wave reflection and transmission coefficients derived in the next chapter for ray tracing applications, provided the interactions occur far from the source.

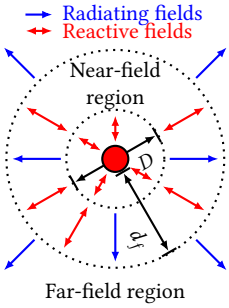


Figure 1.9: Field regions of an antenna, adapted from [2, Fig. 4.3].

1.4.3 Antenna Parameters

To characterize how an antenna converts guided waves into free-space radiation (and vice versa), we define several key parameters. A comprehensive discussion of these concepts can be found in Saunders and Aragón-Zavala's book [2, Chap. 4].

Radiation Pattern and Intensity

As discussed previously, the time-averaged electromagnetic power density is related to the strength of the electric field. The *radiation pattern* of an antenna is a plot of the far-field radiation from the antenna (see Figure 1.10). More specifically, it usually represents the *radiation intensity* U (measured in watts per unit solid angle). The radiation intensity is intimately linked to the electric and magnetic fields radiated by the antenna; it is obtained by multiplying the time-averaged power density S (the magnitude of the

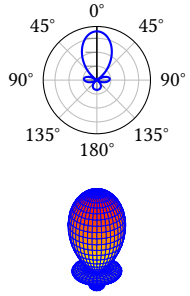


Figure 1.10: Example of a far-field radiation pattern for a directional antenna.

[2]: Saunders et al. (2024), *Antennas and Propagation for Wireless Communication Systems*

time-averaged Poynting vector) at a given distance r by the square of that distance,

$$U(\theta, \varphi) = r^2 S(\theta, \varphi), \tag{1.33}$$

where θ and φ are the polar and azimuthal angles, respectively.

In the far field, because the power depends on the square of the electric field strength, which decreases as r^{-1}, the dependence on the distance disappears, ensuring that U depends only on the direction (θ, φ).

Pattern Cuts

Although the radiation intensity U is fundamentally a 3D function of both θ and φ, it is common practice for manufacturers to specify antenna patterns in terms of *cuts*, i.e., using two separate patterns along orthogonal directions. The full 3D pattern in any other direction can then be approximated by assuming the pattern is separable into the product of these two orthogonal functions, i.e.,

$$U(\theta, \varphi) = U_\theta(\theta) U_\varphi(\varphi). \tag{1.34}$$

Directivity

The *directivity* $D(\theta, \varphi)$ of an antenna is defined as the ratio of its radiation intensity in a specific direction to the mean radiation intensity across all directions, i.e.,

$$D(\theta, \varphi) = \frac{U(\theta, \varphi)}{\frac{1}{4\pi} \int_0^{2\pi} \int_0^{\pi} U(\theta', \varphi') \sin\theta' \, \mathrm{d}\theta' \, \mathrm{d}\varphi'}, \tag{1.35}$$

where θ' and φ' are dummy integration variables.

Equivalently, it is the radiation intensity of the antenna compared to that of an ideal *isotropic* antenna radiating the same total power.

Power Gain

The *power gain* $G(\theta, \varphi)$, or simply the gain, is similar to directivity but also accounts for the electrical efficiency of the antenna. It is defined as the ratio of the radiation intensity to that of an isotropic antenna radiating the same total power *captured* by the real antenna. When the term "gain" is used without specifying a direction, it typically refers to the *maximum power gain*, which is the peak value of $G(\theta, \varphi)$, i.e.,

$$G \stackrel{\text{def}}{=} \max_{\theta,\varphi} G(\theta, \varphi). \tag{1.36}$$

Similarly, the *maximum directivity* is defined as

$$D \stackrel{\text{def}}{=} \max_{\theta,\varphi} D(\theta, \varphi). \tag{1.37}$$

The antenna gain and directivity are related by

$$G(\theta, \varphi) = \eta_{\text{rad}} D(\theta, \varphi), \tag{1.38}$$

where η_{rad} is the antenna radiation efficiency.

Effective Area

For receiving antennas, we assess their performance with the directional *effective area* $A_e(\theta, \varphi)$, which is an equivalent area (m^2) measuring how much power the antenna can capture from an incoming plane wave with a given power density $S(\theta, \varphi)$, such that the received power P_r is

$$P_r(\theta, \varphi) = A_e(\theta, \varphi) S(\theta, \varphi). \tag{1.39}$$

In practical simulators, this relation is evaluated direction-wise through the receive pattern and polarization projection, then integrated into the per-path power accumulation. Under the reciprocity theorem, the directive gain and the

effective area in any given direction are related by

$$G(\theta, \varphi) = \frac{4\pi A_e(\theta, \varphi)}{\lambda^2}. \tag{1.40}$$

Consequently, their maximum values (peak gain G and peak effective area A_e) satisfy the standard relation

$$G = \frac{4\pi A_e}{\lambda^2}. \tag{1.41}$$

1.5 Wave Interactions at Boundaries

In a realistic environment, electromagnetic waves do not propagate solely in free space. They encounter various objects and interfaces between different media, leading to complex interaction mechanisms, allowing the wave to propagate in directions other than the initial one, but also to reach non-line-of-sight locations. The four primary mechanisms are reflection, transmission, diffraction, and scattering. Recently, a fifth mechanism involving programmable metasurfaces has emerged.

1.5.1 Reflection and Transmission

When a wave encounters a smooth interface between two media with different electrical properties, solving Maxwell's equations results in two new waves, each at the same frequency as the incident wave: a reflected wave and a transmitted wave, where the energy of the incident wave is divided between these two components, accounting for any associated material losses. Both waves propagate in the plane containing the incident propagation vector and the surface normal (see Figure 1.11). This plane is called the *plane of incidence*. In the more general case of diffuse interactions where the outgoing wave deviates from the specular direction, the plane containing the incident and scattered wave vectors is distinctly referred to as the *scattering plane*.

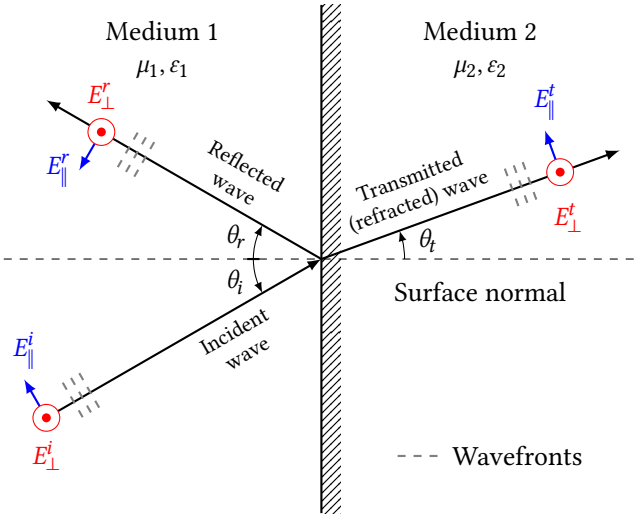


Figure 1.11: Plane wave reflection and transmission at a planar interface between two media, adapted from [2, Fig. 3.1].

In lossless media, *Snell's law of reflection* states that the angle of incidence equals the angle of reflection, i.e.,

$$\theta_i = \theta_r, \tag{1.42}$$

where both angles are measured with respect to the surface normal.

For the transmitted wave, *Snell's law of refraction* states that the angles of incidence and refraction are related by

$$n_1 \sin\theta_i = n_2 \sin\theta_t, \tag{1.43}$$

where n_1 and n_2 are the refractive indices of the first and second media, respectively.

The refractive index is defined as the ratio of the free-space velocity, equal to c for electromagnetic waves, to the phase velocity in the medium, i.e.,

$$n = \frac{c}{v} = \frac{\sqrt{\epsilon\mu}}{\sqrt{\epsilon_0\mu_0}} = \sqrt{\epsilon_r\mu_r}. \tag{1.44}$$

In fact, Snell's laws are a direct consequence of *Fermat's principle*, which we will discuss in Chapter 2.

The split in energy between the reflected and transmitted waves depends on the polarization of the incident wave relative to the plane of incidence, the angles of reflection and refraction, and the impedance of the two media. The ratio of the reflected and transmitted electric fields to the incident electric field is given by the *Fresnel reflection and transmission coefficients*, defined as

$$R_{\parallel} = \frac{E_{\parallel}^{r}}{E_{\parallel}^{i}} = \frac{Z_2 \cos\theta_t - Z_1 \cos\theta_i}{Z_2 \cos\theta_t + Z_1 \cos\theta_i} \qquad R_{\perp} = \frac{E_{\perp}^{r}}{E_{\perp}^{i}} = \frac{Z_2 \cos\theta_i - Z_1 \cos\theta_t}{Z_2 \cos\theta_i + Z_1 \cos\theta_t} \tag{1.45}$$

$$T_{\parallel} = \frac{E_{\parallel}^{t}}{E_{\parallel}^{i}} = \frac{2Z_2 \cos\theta_i}{Z_2 \cos\theta_t + Z_1 \cos\theta_i} \qquad T_{\perp} = \frac{E_{\perp}^{t}}{E_{\perp}^{i}} = \frac{2Z_2 \cos\theta_i}{Z_2 \cos\theta_i + Z_1 \cos\theta_t}, \tag{1.46}$$

where $Z_1 = \sqrt{\mu_1/\epsilon_1}$ and $Z_2 = \sqrt{\mu_2/\epsilon_2}$ are the intrinsic impedances of the first and second media, respectively.

In lossy media, the reflection and transmission coefficients are complex, and the transmitted wave is attenuated as it propagates through the medium. Snell's law of reflection (1.42) still holds, while the law of refraction (1.43) must be modified to account for the change in phase velocity, see Balanis's *Advanced Engineering Electromagnetics* [6] for more details.

[6]: Balanis (2024), *Advanced Engineering Electromagnetics*

To conclude this short introduction to reflection and transmission, two special cases are worth mentioning.

First, if the wave travels from a dense to a less dense medium ($n_1 > n_2$), there exists an incidence angle, called the *critical angle*, θ_c, such that no energy is transmitted. This phenomenon is called *total internal reflection*. This angle is given by

$$\theta_c = \arcsin \frac{n_2}{n_1}. \tag{1.47}$$

For water to air propagation, where $n_1 \approx 1.33$ and $n_2 = 1$, this angle is approximately 48.6°, see Figure 1.12.

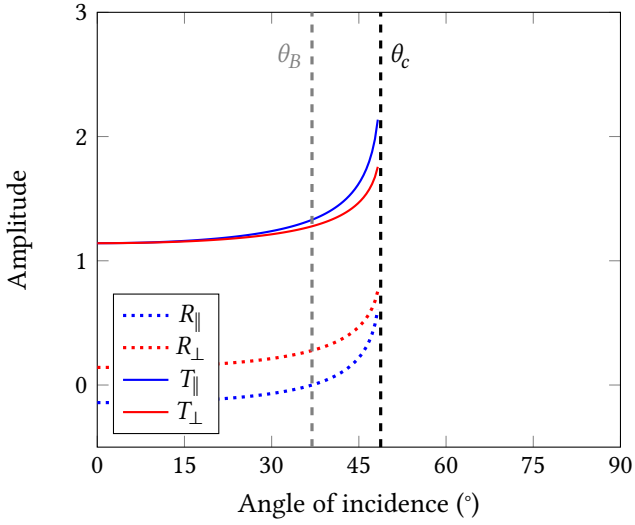


Figure 1.12: Fresnel reflection and transmission coefficients for water to air propagation.

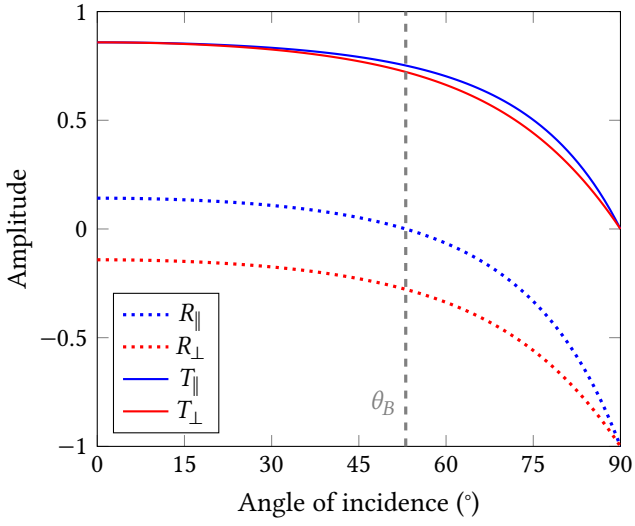


Figure 1.13: Fresnel reflection and transmission coefficients for air to water propagation.

Second, there exists a specific angle called the *Brewster angle*, θ_B, where $R_\parallel = 0$, meaning the reflected wave is purely perpendicularly polarized and the parallel component is fully transmitted. This angle is given by

$$\theta_B = \arctan \frac{n_2}{n_1}. \tag{1.48}$$

For air to water propagation, this angle is approximately 53°, see Figure 1.13.

1.5.2 Scattering

Reflection assumes the surface is smooth relative to the wavelength. However, objects are never perfectly smooth, and the roughness of a surface can significantly affect the reflection properties of electromagnetic waves. Specifically, when a wave impinges on a rough surface or an object with height variations comparable in size to the wavelength, *scattering* occurs. Rather than specular reflection, energy is dispersed in many directions. A height difference of Δh between two points on the surface induces a phase difference on the reflected waves of

$$\Delta\phi = \frac{4\pi\Delta h \cos\theta_i}{\lambda}. \tag{1.49}$$

If this phase shift is below 90°, the surface can reasonably be considered to be smooth. This threshold defines the *Rayleigh roughness criterion*,

$$\Delta h < \frac{\lambda}{8\cos\theta_i}. \tag{1.50}$$

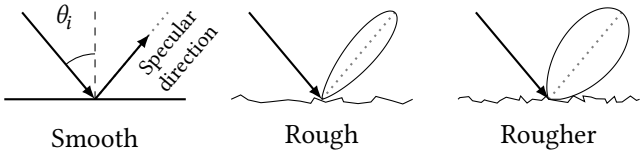


Figure 1.14: The effect of surface roughness on reflection, reproduced from [2, Fig. 3.6].

However, for accuracy, it is recommended to use a lower threshold, such as 22.5°, which is a quarter of the Rayleigh criterion, below which the surface is considered smooth.

When the surface is rough, the amplitude of the specular reflection component is significantly reduced. This reduction can be quantified by multiplying the ideal smooth-surface reflection coefficient by a roughness factor, which will be discussed in more detail in Section 2.7.

1.5.3 Diffraction

Diffraction is the mechanism that allows waves to bend around obstacles and propagate into shadow regions where no line-of-sight or reflected path exists. It is fundamentally explained by the *Huygens–Fresnel principle*, which states that every point on a wavefront acts as a source of secondary spherical wavelets (see Figure 1.15).

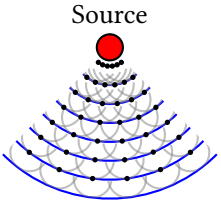
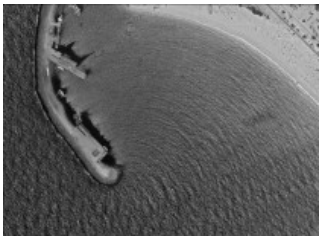


Figure 1.15: The Huygens–Fresnel principle.

One of the most common examples of diffraction is called *single slit diffraction*, where a wave passes through a narrow opening (i.e., small with respect to the wavelength) and spreads out on the other side. This can be observed in Figure 1.16a. However, in the context of radio, especially outdoor propagation, a more common example of diffraction is *knife-edge diffraction*, where a wave passes around a sharp obstacle, such as a mountain or a building. This can be observed in Figure 1.16b.

(a) Single slit diffraction.



(b) Knife-edge diffraction.

Figure 1.16: Diffraction phenomena found in sea environments, Google Earth images.

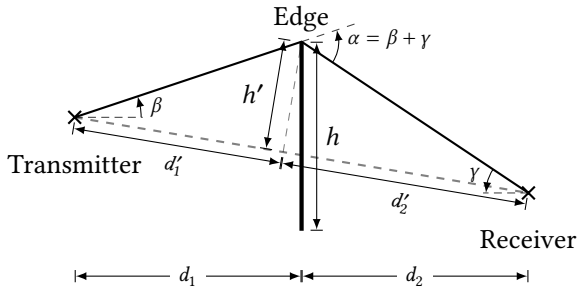


Figure 1.17: Knife-edge diffraction geometric parameters, reproduced from [2, Fig. 3.16].

Single Knife-Edge Diffraction

We first model diffraction of a plane wave over an absorbing plane or *knife-edge*. The simple knife-edge diffraction model mathematically formulates Huygens' principle by combining contributions from an infinite number of secondary sources in the region above the edge.

The final result expresses the propagation loss, or reduction in field strength in decibels, in terms of a diffraction parameter v, which is defined as

$$v = h' \sqrt{\frac{2\left(d_1' + d_2'\right)}{\lambda d_1' d_2'}}, \tag{1.51}$$

where h' is the *excess height* of the edge above the line from the transmitter to the receiver, d_1' and d_2' are the distances from the projection of the edge on this line to the transmitter and receiver, respectively, and λ is the wavelength, as illustrated in Figure 1.17.

The propagation loss is then given by

$$L_{ke}(v) = -20 \log_{10} \left| \frac{\mathbf{E}^d}{\mathbf{E}^i} \right| = -20 \log_{10} |F(v)|, \tag{1.52}$$

where $\mathbf{E}^d$ is the diffracted field, $\mathbf{E}^i$ is the incident field, and $F(v)$ is defined as

$$F(v) = \frac{1+j}{2} \int_v^\infty \exp\left(-j\frac{\pi u^2}{2}\right) \mathrm{d}u, \tag{1.53}$$

Alternatively, its magnitude can be expressed in terms of the Fresnel cosine and sine integrals, $C(\nu)$ and $S(\nu)$, which integrate the real and imaginary parts of the integrand from 0 to ν. To avoid complex values, we can write the squared magnitude of (1.53) as

$$|F(\nu)|^2 = \frac{1}{2}\left[\left(\frac{1}{2} - C(\nu)\right)^2 + \left(\frac{1}{2} - S(\nu)\right)^2\right], \tag{1.54}$$

where

$$C(\nu) = \int_0^\nu \cos\left(\frac{\pi u^2}{2}\right) \mathrm{d}u, \tag{1.55}$$

$$S(\nu) = \int_0^\nu \sin\left(\frac{\pi u^2}{2}\right) \mathrm{d}u. \tag{1.56}$$

While the Fresnel integrals lack closed-form analytical solutions in terms of elementary functions, they can be computed numerically or approximated using asymptotic expansions, which are available in most scientific computing libraries, such as the Cephes C library, on which many other libraries are built, such as SciPy's `scipy.special.fresnel` function.

The knife-edge diffraction loss L_{ke} is illustrated in Figure 1.18. Notably, $L_{ke}(0) \approx 6\,\mathrm{dB}$, which means that the power is reduced by a factor of $10^{0.6} \approx 4$ when the edge is situated exactly on the direct path between the transmitter and the receiver. For values $\nu > 1$, meaning for receivers well within the shadow region, the attenuation increases significantly and can be approximated as

$$L_{ke}(\nu) \approx -20 \log_{10} \frac{1}{\pi\nu\sqrt{2}} \approx -20 \log_{10} \frac{0.225}{\nu}. \tag{1.57}$$

The parameter ν itself is tied to the geometry of the direct

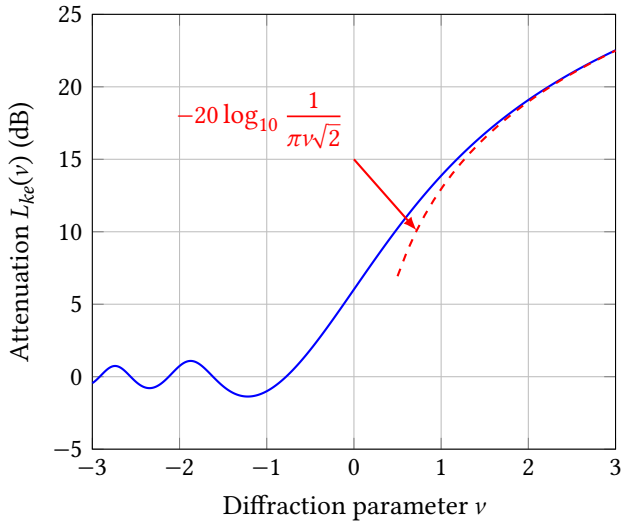


Figure 1.18: Knife-edge diffraction loss $L_{ke}(v)$ as a function of the diffraction parameter v, reproduced from [2, Fig. 3.15].

path relative to the edge, defined in Figure 1.17 as

$$v = h'\sqrt{\frac{2(d_1' + d_2')}{\lambda d_1' d_2'}} = \alpha\sqrt{\frac{2d_1' d_2'}{\lambda(d_1' + d_2')}}, \tag{1.58}$$

where h' is the excess height, and d_1', d_2' the distances from the source and receiver to the point of diffraction. For many practical cases where $d_1, d_2 \gg h$, this expression can be simplified in terms of distances measured along the ground

$$v \approx h\sqrt{\frac{2(d_1 + d_2)}{\lambda d_1 d_2}} = \alpha\sqrt{\frac{2d_1 d_2}{\lambda(d_1 + d_2)}}, \tag{1.59}$$

where α is the sum of the angles of elevation of the transmitter and receiver with respect to the knife-edge, β and γ in Figure 1.17, i.e.,

$$\alpha = \beta + \gamma. \tag{1.60}$$

An equivalent geometric interpretation uses Fresnel zones. Around the direct path, the first Fresnel-zone radius at a point located at distances d_1 and d_2 from transmitter and

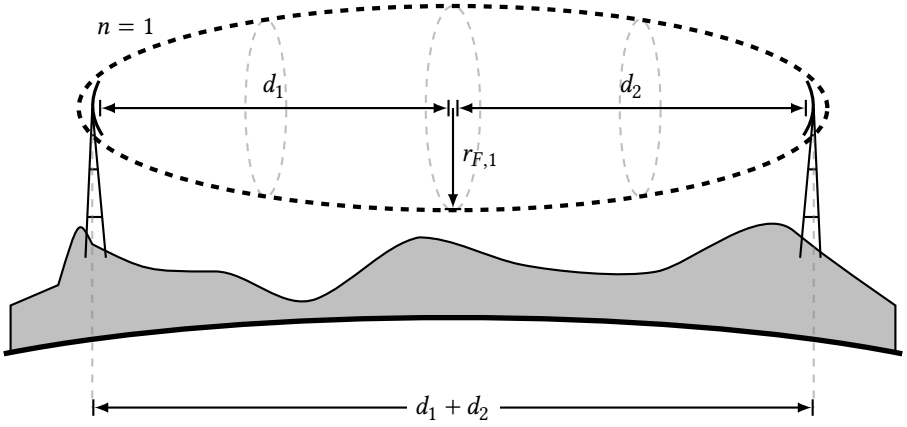


Figure 1.19: First Fresnel zone around the direct path between two antennas.

receiver is

$$r_{F,1} = \sqrt{\frac{\lambda d_1 d_2}{d_1 + d_2}}. \tag{1.61}$$

See Figure 1.19 for an illustration of the first Fresnel zone.

When an obstacle intrudes deeply into this first zone, diffraction loss rises rapidly. As a practical planning rule, preserving most of the first-zone clearance strongly reduces excess diffraction attenuation.

Beyond Knife-Edge Diffraction

While knife-edge diffraction provides valuable physical intuition and a tractable analytical solution based on Huygens' principle, it is fundamentally a simplified scalar model. It assumes the diffracting edge is part of a perfectly absorbing, semi-infinite screen and does not account for the actual structure, material properties, or finite extent of real obstacles. Moreover, it cannot handle oblique incidence, multiple edges, or wedges with arbitrary exterior angles.

In practical propagation scenarios, especially those encountered in urban environments, we frequently deal with complex geometries: building corners, rooftops, and terrain features that diffract electromagnetic waves in ways not captured by the knife-edge approximation. These situations require a more general, rigorous framework that preserves the full vector nature of electromagnetic fields, correctly handles polarization, and remains valid even in transition regions near shadow boundaries.

Such a framework is provided by the *geometrical theory of diffraction (GTD)* and its mathematically uniform extension, the *uniform theory of diffraction (UTD)*. These high-frequency asymptotic theories extend the ray concept to include diffracted rays emanating from edges, vertices, and other singular features. They provide dyadic diffraction coefficients analogous to Fresnel reflection coefficients, enabling precise prediction of field strength and polarization for arbitrary wedge geometries.

Because GTD and UTD form the foundation of modern deterministic ray tracing for radio propagation, their detailed formulation, applicability conditions, and integration with geometrical optics are thoroughly presented in Chapter 2, where we develop the complete mathematical machinery for ray-based channel modeling.

1.5.4 Interaction with Metasurfaces

Traditionally, the propagation of radio waves in wireless environments is dictated by the standard electromagnetic interactions discussed above: reflection, scattering, and diffraction. Standard specular reflection follows the conventional Snell's law (where the angle of incidence equals the angle of reflection), scattering disperses incident energy diffusely across multiple directions, and diffraction allows waves to bend around physical obstacles. In all of these standard interactions, the environment acts as a completely passive, unalterable medium that the communication system must adapt to.

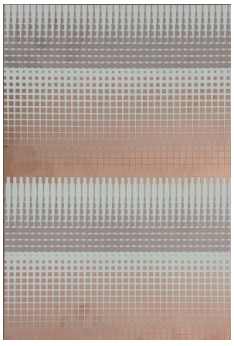


Figure 1.20: Passive metasurface, kindly provided by Adam Abazi from the UCLouvain antenna group.

A paradigm shift in this domain is the introduction of *metasurfaces*. Metasurfaces are the two-dimensional, planar equivalents of metamaterials, composed of dense arrays of sub-wavelength scattering elements (or unit cells) designed to engineer the electromagnetic wavefront, see Figure 1.20. Unlike conventional surfaces, metasurfaces can generate a controllable, spatially varying phase, amplitude, or polarization shift across their aperture. By discretizing a continuous phase profile across its elements, a metasurface can manipulate waves in ways that natural materials cannot, such as by redirecting an incident wave toward an arbitrary, anomalous direction. This interaction is governed by the *generalized Snell's law* [7] for reflection

$$\sin\theta_r - \sin\theta_i = \frac{\lambda_0}{2\pi}\frac{\mathrm{d}\Phi}{\mathrm{d}x}, \tag{1.62}$$

and for refraction

$$n_2 \sin\theta_t - n_1 \sin\theta_i = \frac{\lambda_0}{2\pi}\frac{\mathrm{d}\Phi}{\mathrm{d}x}, \tag{1.63}$$

where n_1 and n_2 are the refractive indices of the incident and transmitted media, θ_i is the angle of incidence, θ_r is the anomalous reflection angle, θ_t is the anomalous transmission angle, λ_0 is the free-space wavelength, and $\frac{\mathrm{d}\Phi}{\mathrm{d}x}$ represents the phase gradient generated along the surface.

[7]: Yu et al. (2011), *Light Propagation with Phase Discontinuities: Generalized Laws of Reflection and Refraction*

A prominent and specific implementation of this technology in wireless communications is the reconfigurable intelligent surface (RIS). A RIS is a dynamically tunable metasurface whose electromagnetic response can be electronically controlled (see Figure 1.21). Built using printed conductive elements on a dielectric substrate and backed by a ground plane, a RIS integrates tunable components into each unit cell. This capability enables the creation of "smart radio environments" where the propagation channel itself becomes a programmable entity. Instead of merely adapting to random fading, systems can actively shape radio paths to bypass blockages, suppress interference, and enhance the overall link budget (see Figure 1.22).

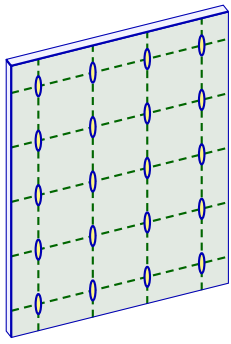


Figure 1.21: A RIS, inspired by [8, Fig. 1].

From a computational perspective, a RIS is usually the only kind of metasurface that we can easily and practically simulate using ray tracing. Because simulating the full-wave electromagnetic interactions of thousands of sub-wavelength unit cells is computationally prohibitive, ray tracing engines generally avoid microscopic, element-level evaluations. Instead, we must rely on simplified macroscopic models that abstract the RIS into a continuous anomalous reflector or a grid of manageable tiles. These fundamental abstractions are crucial for efficient simulation, and we will present some of these simplified analytical models in Chapter 2.

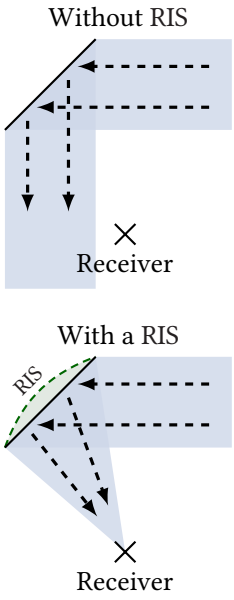


Figure 1.22: Example scenario where a RIS is used to redirect the signal towards the receiver, adapted from [9, Fig. 3].

1.6 Path Loss Models

To evaluate the performance of a wireless communication system, one must predict the *path loss*, which is the ratio of transmitted power to received power, usually expressed in decibels. While numerical methods offer extreme precision, several analytical models provide crucial insights into how propagation mechanisms dictate system range.

1.6.1 Free-Space Path Loss

In a completely empty void with only a line-of-sight link, the received power P_r is related to the transmitted power P_t by the (contemporary) *Friis transmission equation* [10]

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi d}\right)^2, \tag{1.64}$$

[10]: Friis (1946), *A note on a simple transmission formula*

where G_t and G_r are the peak gains of the transmitting and receiving antennas, respectively, and d is the distance between the two antennas.

Because the term $\left(\frac{\lambda}{4\pi d}\right)^2$ dictates the power decay, the free-space path loss increases quadratically with distance. Consequently, the signal drops by 20 dB per decade (i.e.,

every time the distance increases tenfold). This represents the theoretical minimum loss for point-to-point propagation when multipath effects are neglected.

1.6.2 The Two-Ray Ground Reflection Model

Environments on Earth are rarely true line-of-sight free space scenarios; most of the time, the ground at least plays a role in reflecting waves. The interaction between a direct path and a ground-reflected path (see Figure 1.23) illustrates the concept of multipath interference.

In the two-ray ground reflection model, the received signal consists of two components: the line-of-sight ray and the ray reflected by the ground. The total received electric field is the sum of these two contributions. Using the parameters defined in Figure 1.23, where d' is the length of the direct path and $d_1 + d_2$ is the length of the reflected path, the received power P_r can be expressed as an extension of the Friis transmission equation

$$P_r = P_t G_t G_r \left(\frac{\lambda}{4\pi}\right)^2 \left| \frac{e^{-jkd'}}{d'} + R\frac{e^{-jk(d_1+d_2)}}{d_1+d_2} \right|^2, \qquad (1.65)$$

where R is the ground reflection coefficient, which depends, as we have seen in Section 1.5.1, on the electromagnetic properties of the ground, the polarization, and the angle of incidence $\theta = \frac{\pi}{2} - \alpha$.

For most practical scenarios, especially in macrocellular communications, the horizontal distance d separating the transmitter and receiver is much larger than their respective heights h_t and h_r ($d \gg h_t, h_r$). Under this condition, the lengths of the two paths can be approximated using a

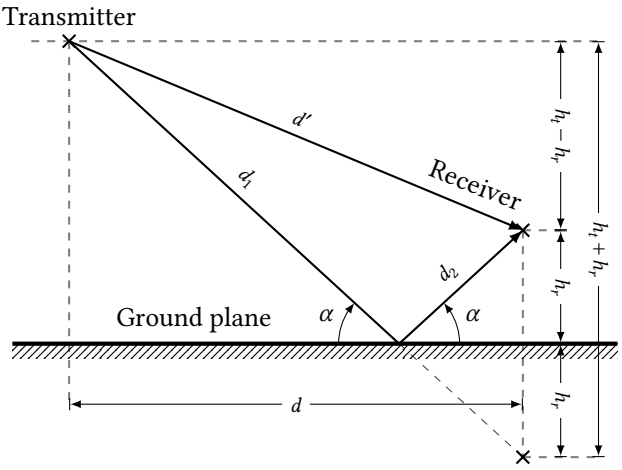


Figure 1.23: The two-ray ground reflection model parameters.

Taylor series expansion

$$d' = \sqrt{d^2 + (h_t - h_r)^2} \approx d + \frac{(h_t - h_r)^2}{2d}, \tag{1.66}$$

$$d_1 + d_2 = \sqrt{d^2 + (h_t + h_r)^2} \approx d + \frac{(h_t + h_r)^2}{2d}. \tag{1.67}$$

The path difference $\Delta d \stackrel{\text{def}}{=} d_1 + d_2 - d'$ then simplifies to

$$\Delta d \approx \frac{(h_t + h_r)^2 - (h_t - h_r)^2}{2d} = \frac{2h_t h_r}{d}. \tag{1.68}$$

This path difference translates to a phase difference $\Delta\phi$ between the two rays given by

$$\Delta\phi = k\Delta d = \frac{2\pi}{\lambda}\frac{2h_t h_r}{d} = \frac{4\pi h_t h_r}{\lambda d}. \tag{1.69}$$

At large distances, the incidence angle approaches a grazing angle ($\alpha \to 0, \theta \to \frac{\pi}{2}$). In this grazing incidence case, the reflection coefficient for both polarizations approaches -1 (i.e., $R \approx -1$). Furthermore, since $d \gg h_t, h_r$, we can approximate the denominators in the received power equation as

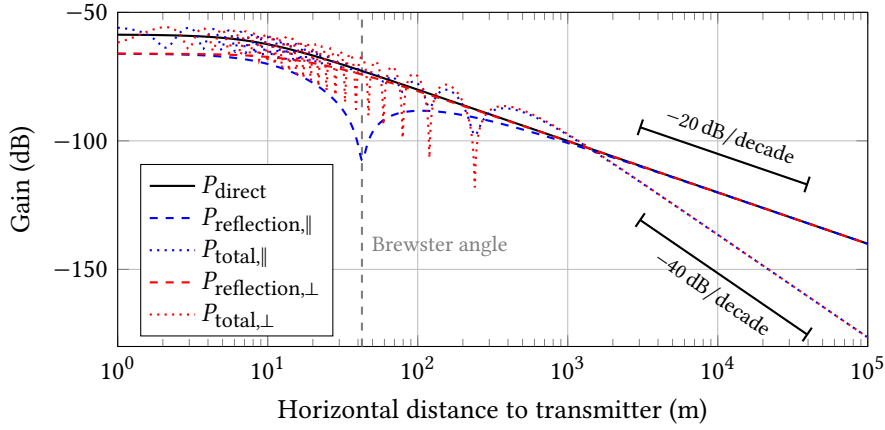


Figure 1.24: Received power gain for the two-ray ground reflection model as a function of the distance ($h_t = 10\,\text{m}$, $h_r = 1.5\,\text{m}$, $f = 2.4\,\text{GHz}$), showing the direct path, reflected path, and total received power for both parallel (∥) and perpendicular (⊥) polarizations. The electromagnetic properties of the ground are approximated as "medium dry ground" from ITU-R Recommendation P.2040 [11, Tab. 3].

$d' \approx d_1 + d_2 \approx d$. The power equation then becomes

$$P_r \approx P_t G_t G_r \left(\frac{\lambda}{4\pi d}\right)^2 \left|1 + Re^{-j\Delta\phi}\right|^2. \tag{1.70}$$

Substituting $R \approx -1$ into this expression yields $1 + Re^{-j\Delta\phi} \approx 1 - e^{-j\Delta\phi}$, which indicates that the reflected wave arrives with an inverted phase relative to the direct wave, causing destructive interference. For very small phase differences $\Delta\phi \ll 1$ (which holds true at large distances), we can use the small-angle approximation $e^{-j\Delta\phi} \approx 1 - j\Delta\phi$. The magnitude squared of the interference term simplifies to

$$|1 - (1 - j\Delta\phi)|^2 = |j\Delta\phi|^2 = (\Delta\phi)^2 = \left(\frac{4\pi h_t h_r}{\lambda d}\right)^2. \tag{1.71}$$

Substituting this back into the power equation yields the simplified two-ray path loss model for large distances

$$P_r \approx P_t G_t G_r \left(\frac{\lambda}{4\pi d}\right)^2 \left(\frac{4\pi h_t h_r}{\lambda d}\right)^2 = P_t G_t G_r \frac{h_t^2 h_r^2}{d^4}. \tag{1.72}$$

This fundamental result demonstrates that at large distances, destructive multipath interference changes the path loss distance exponent from 2 (in free space) to 4 (see Figure 1.24). This means the power falls off at a rapid 40 dB per decade, and is notably independent of the frequency. The two-ray model serves as a baseline demonstrating that environmental interactions severely and predictably impact signal decay in terrestrial environments, particularly highlighting the effect of the ground.

In practice, the transition toward this asymptotic d^{-4} regime can be approximated to start at a breakpoint distance, d_{bp}, defined as

$$d_{\text{bp}} \approx \frac{4h_t h_r}{\lambda}, \tag{1.73}$$

which corresponds to the range where the first Fresnel zone starts to be significantly clipped by the ground-reflected geometry. For $d \ll d_{\text{bp}}$, the behavior is closer to free-space scaling; for $d \gg d_{\text{bp}}$, the asymptotic two-ray law becomes a good approximation.

1.6.3 The Link Budget

The complete assessment of all gains and losses from transmitter to receiver forms the *link budget*. For this, we identify two types of losses: antenna losses and propagation—or path—losses.

For the antennas, we express their gain with respect to an isotropic antenna. The effective isotropic radiated power (EIRP) is given by

$$\text{EIRP} = \frac{P_t G_t}{L_t} = P_{t,\text{iso}}, \tag{1.74}$$

where L_t is the loss in the transmitting antenna system and $P_{t,\text{iso}}$ is the effective isotropic transmitted power.

A similar value can be obtained for the receiver

$$P_{r,\text{iso}} = \frac{P_r G_r}{L_r}, \tag{1.75}$$

where L_r is the loss in the receiving antenna system and $P_{r,\text{iso}}$ is the effective isotropic received power.

Knowing the effective isotropic transmitted and received power, it is possible to determine the path loss

$$L = \frac{P_{t,\text{iso}}}{P_{r,\text{iso}}} = \frac{P_t G_t G_r}{P_r L_t L_r}. \tag{1.76}$$

However, it is rarely desirable to determine the path loss through actual measurements, as it requires—usually expensive—equipment, can be time-consuming, and provides values that are not easily generalizable to other environments or system configurations. As a result, path loss models are crucial for predicting how the signal strength varies in a given environment.

1.7 Conclusion

This chapter has established the physical "ground truth" of radio propagation. We have seen how Maxwell's equations give rise to waves that propagate, reflect, and transmit. We also introduced the antennas that launch these waves. Note that throughout these analytical models, the propagation medium (air) is almost universally approximated as an ideal free space ($\epsilon_r \approx 1, \sigma \approx 0$), with all significant interactions isolated to the boundaries of solid objects.

To determine the signal strength at a receiver in a complex environment like a city, one might consider solving Maxwell's equations numerically for every point in space and time. Full-wave electromagnetic solvers based on methods like finite-difference time-domain (FDTD), finite element method (FEM), or method of moments (MoM) offer extreme precision. However, because they require meshing the entire volume or surfaces at sub-wavelength resolutions, they become computationally intractable for environments larger than a few cubic meters, especially at high frequencies.

Conversely, the simple analytical path loss models discussed in Section 1.6 scale effortlessly but are far too simplistic to capture the severe multipath fading induced by hundreds of building facades and terrain variations.

To simulate radio propagation efficiently and accurately at the urban scale, we need a method that simplifies wave interactions into geometric paths, avoiding the need to discretize empty space. Geometrical optics (GO) and UTD provide this perfect middle ground. As briefly introduced both by the knife-edge diffraction and the two-ray models, we will rely on ray tracing to simulate radio propagation in complex environments. In the next chapter, we will introduce the *high-frequency approximation*, which allows us to transition from the wave domain to the ray domain, forming the mathematical basis of the ray tracing algorithms we aim to construct in Part 'Building'.

Ray Tracing Fundamentals

2

The acoustic echoes of one's voice bouncing off the walls of a tunnel provide an intuitive analogy for radio wave propagation. When a transmitter (TX) emits radio signals, they interact with the environment—reflecting off buildings, the ground, and other obstacles—creating multiple distinct paths to the receiver (RX). This phenomenon is known as multipath propagation. Whether applied to sound, light, or radio waves, ray tracing serves as a powerful technique to model and analyze wave behavior as it propagates through complex environments.

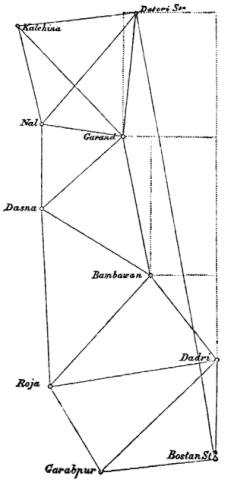


Figure 2.1: In geodesy, ray tracing was historically employed to determine the precise telescope angle required to observe a signal from a distant location. In this illustration from [12, p. xix], any city serves as either the observation point or the light source, and the various ray paths connecting the two locations demonstrate the correct telescope angle.

2.1 History of Ray Tracing

Before addressing the technical details of ray tracing and the theory required for its application to radio waves, it is interesting to understand the method's historical context. While the first explicit mention of "ray tracing" is likely attributable to Sir George Everest in the context of geodesy during the mid-19th century [12], the concept of using rays as a high-frequency approximation of waves was developed independently across several fields over more than a century.

In optical design, Alexander Eugen Conrady demonstrated in the late 1920s that the aberration of a lens system could be quantified by tracing a finite number of representative rays through successive interfaces [13]. This concept—sampling a continuous wavefront with discrete trajectories—enables the solution of optics problems without requiring an exact solution to the wave equation. This computational simplification underpins all subsequent developments in ray tracing.

Beyond glass optics, the notion of a "ray" was rapidly generalized to other physical media. In electron optics,

[12]: Everest et al. (1847), *An account of the measurement of two sections of the meridional arc of India: bounded by the parallels of 18° 3' 15"; 24° 7' 11"; & 29° 30' 48"*

[13]: Conrady (1929), *Applied optics and optical design, pt. I*

Klemperer applied similar principles to charged particles moving in smoothly varying electromagnetic fields, replacing piecewise-linear segments with continuously curved trajectories within an inhomogeneous "refractive index" defined by the potential [14]. In underwater acoustics, Lichte modeled the ocean as a stratified medium with a depth-dependent speed of sound. He utilized ray theory to predict refraction, *shadow zones*[*], and deep sound channels, foreshadowing later work on long-range sonar and the sound fixing and ranging (SOFAR) channel [15]. This body of work was systematized during World War II by the National Defense Research Committee (NDRC) report on the physics of sound in the sea, which popularized ray diagrams as the standard language for explaining propagation, ducting, and shadowing to sonar operators [16].

[14]: Klemperer (1939), *Electron Optics*

[15]: Lichte (1919), *Über den Einfluss horizontaler Temperaturschichtung des Seewassers auf die Reichweite von Unterwasserschallsignalen*

[16]: National Defense Research Committee (NDRC) (1946), *Physics of Sound in the Sea*

Radio physicists developed an equally rich ray theory for the ionosphere. Booker's analysis of wave packets in a doubly refracting plasma clarified that a radio "ray" should be identified with the trajectory of energy flow (group velocity), which differs from the phase-normal direction in anisotropic media [17]. Millington provided graphical constructions for such rays in a curved ionosphere, while Haselgrove recast the problem into a Hamiltonian system of differential equations amenable to numerical integration on early digital computers [18, 19]. Her formulation marked a pivotal transition: the shift from hand-drawn ray sketches to fully algorithmic ray tracing in generic, inhomogeneous media—a pattern that would later reemerge in seismology and computer graphics.

[17]: Booker (1938), *Propagation of wave-packets incident obliquely upon a stratified doubly refracting ionosphere*

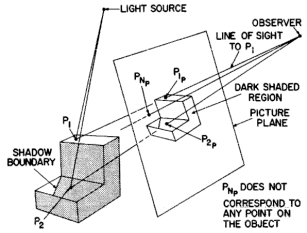
[18]: Millington (1954), *Ray-path characteristics in the ionosphere*

[19]: Haselgrove (1955), *Ray Theory and a New Method for Ray Tracing*

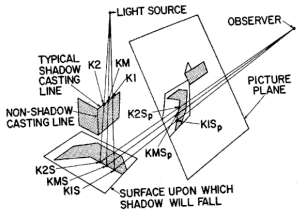
In computer graphics, ray tracing emerged in the late 1960s as a mechanism for solving visibility and shading problems. Appel's algorithm cast rays from the viewpoint through each pixel, intersected them with scene geometry, and spawned secondary rays toward light sources to determine if a point was illuminated or shadowed [20]. Whitted generalized this approach by introducing recursive rays for reflection and refraction, providing a unified framework

[20]: Appel (1968), *Some techniques for shading machine renderings of solids*

[*] Referred to elsewhere in the text as shadow regions.



(a) Point-by-point shading.



(b) Segment-by-segment outlining of shadows.

Figure 2.2: Appel's ray tracing methods for computing the shading and shadowing of an object, from [20, Fig. 6 & 7].

for specular transport capable of rendering mirror-like and transparent objects with unprecedented realism [21]. This lineage culminated in path tracing and the rendering equation, where rays are interpreted as Monte Carlo samples of an underlying transport integral, conceptually echoing Conrady's finite set of representative rays, albeit in a stochastic setting.

Around the same time, acoustics researchers began applying similar techniques to develop an accurate theory of sound reverberation. In 1967, Atal and Schroeder presented what was likely the first application of ray tracing to model room acoustics [22]. Independently, Krokstad, Strøm, and Sørsdal published work on calculating the acoustical room response using ray tracing [23].

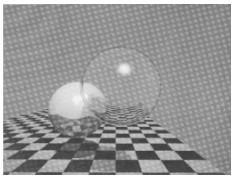


Figure 2.3: Example image rendered using Whitted's model, from [21, Fig. 7].

Simultaneously, seismology adopted ray tracing to study the propagation of seismic waves through the Earth's interior. Julian and Gubbins formulated practical "shooting" and "bending" schemes to solve the two-point boundary-value problem: finding a ray that connects an earthquake source to a receiver through a heterogeneous velocity field [24]. Červený and Hron extended this kinematic ray theory to *dynamic* ray tracing, augmenting the path equations with additional differential equations for geometrical spreading and wavefront curvature. This allowed for the prediction of amplitudes in addition to travel times [25]. These developments established ray tracing as the workhorse of seismic tomography and earthquake location.

From the 1980s onward, ray-based methods in electro-

[22]: Atal et al. (1967), *Study of Sound Decay Using Ray-Tracing Techniques on a Digital Computer*

[23]: Krokstad et al. (1968), *Calculating the acoustical room response by the use of a ray tracing technique*

[24]: Julian et al. (1977), *Three-dimensional seismic ray tracing*

[25]: Červený et al. (1980), *The ray series method and dynamic ray tracing system for three-dimensional inhomogeneous media*

magnetism began to connect more directly to the propagation problems considered in this book. The shooting and bouncing rays (SBR) method combined geometric optics inside complex cavities with a final physical optics (PO) integration on apertures to estimate the radar cross-sections of electrically large structures [26]. In terrestrial wireless communications, deterministic ray tracing in urban environments was used to predict path loss and delay spread by modeling buildings as reflective and diffracting obstacles [27], an approach later surveyed and generalized in comprehensive reviews of radio propagation modeling [28].

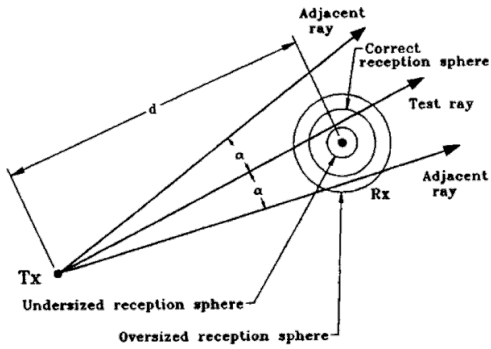


Figure 2.4: Concept of an interception sphere around the receiver to capture incoming rays from all directions, from [27, Fig. 2].

In parallel, Keller's geometrical theory of diffraction (GTD) and the subsequent uniform theory of diffraction (UTD) by Kouyoumjian and Pathak extended classical geometric optics by introducing diffracted rays. This provided a consistent high-frequency treatment of shadow boundaries and edge diffraction, which is now standard in high-frequency radio models [29, 30].

The turn of the 21st century marked a paradigm shift with the utilization of graphics processing units (GPUs) for general-purpose scientific computing, instead of traditional central processing units (CPUs). In 2002, Purcell *et al.* demonstrated the first full ray tracing pipeline running on programmable graphics hardware, effectively treating the GPU as a stream processor for global illumination rather than just rasterization [31]. This proof-of-concept evolved rapidly with the advent of compute-unified device

[26]: Ling et al. (1989), *Shooting and bouncing rays: calculating the RCS of an arbitrarily shaped cavity*

[27]: Shaubackh et al. (1992), *A ray tracing method for predicting path loss and delay spread in microcellular environments*

[28]: Yun et al. (2015), *Ray Tracing for Radio Propagation Modeling: Principles and Applications*

[29]: Keller (1962), *Geometrical Theory of Diffraction*

[30]: Kouyoumjian et al. (1974), *A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface*

[31]: Purcell et al. (2002), *Ray tracing on programmable graphics hardware*

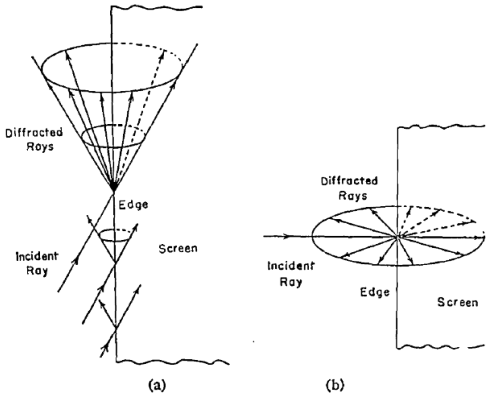


Figure 2.5: Keller's cone (a) and plane (b) of diffraction, from [29, Fig. 1].

architectures (such as NVIDIA CUDA) and dedicated ray tracing engines like NVIDIA OptiX [32], which abstracted the complexities of parallel intersection tests and bounding volume hierarchy management.

[32]: Parker et al. (2010), *OptiX: a general purpose ray tracing engine*

These advances in computer graphics were rapidly adopted by the radio propagation community to address the heavy computational burden of deterministic channel modeling. By porting algorithms like SBR to GPUs, researchers achieved orders-of-magnitude speedups compared to traditional CPU-based solvers. This leap in performance enabled the simulation of massive numbers of interactions in complex urban environments in near real-time, moving ray tracing from offline analysis to potential online applications. The subsequent integration of dedicated hardware acceleration—such as ray tracing cores in modern consumer GPUs—has further cemented this approach, providing the computational throughput necessary for the deep learning integration described below.

Since then, the most recent evolutions of ray tracing have been tightly coupled with machine learning and inverse problems. Differentiable ray tracing, as introduced by Li *et al.*, enables the fast and convenient computation of gradients for rendered images with respect to geometry, materials, and lighting—even in the presence of discontinuities. This capability effectively transforms ray tracing into a differentiable layer within deep learning pipelines [33].

[33]: Li et al. (2018), *Differentiable Monte Carlo ray tracing through edge sampling*

Furthermore, the emergence of GPU-oriented differentiable frameworks such as TensorFlow, PyTorch, or JAX has made it possible to adopt these techniques in more user-friendly and popular programming languages like Python while maintaining high performance through JIT compilation [34–36]. Building on these ideas, Sionna RT and DiffeRT, two open-source Python programs, apply differentiable ray tracing to radio propagation, allowing transmit configurations, antenna orientations, and material parameters to be optimized via gradient-based methods in an end-to-end communication system model [37, 38].

In this sense, the historical trajectories of optics, acoustics, seismology, computer graphics, and electromagnetics converge: the "ray" becomes a universal computational primitive for high-frequency wave propagation, and its differentiable variant is merely the latest iteration.

[34]: Abadi et al. (2015), *TensorFlow, Large-scale machine learning on heterogeneous systems*

[35]: Ansel et al. (2024), *PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation*

[36]: Frostig et al. (2018), *Compiling machine learning programs via high-level tracing*

[37]: Hoydis et al. (2023), *Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling*

[38]: Eertmans et al. (2025), *Demonstrating DiffeRT: An Open-Source Library for Optimizing Radio Networks with Differentiable Ray Tracing*

2.2 The High-Frequency Approximation

In Chapter 1, we discussed the full-wave nature of electromagnetic fields. However, solving the wave equation directly becomes computationally intractable when the environment is large compared to the wavelength. Then, through the knife-edge and two-ray models, we introduced solutions to the wave propagation problem that only consider a discrete number of trajectories that we refer to as *ray paths*. In this section, we will provide the theoretical framework that allows us to approximate the wave as a bundle of rays when the frequency is high enough.

2.2.1 From Waves to Rays: The Lüneburg-Kline Expansion

Using the Lüneburg[†]-Kline ansatz [39, 40], we can express the high-frequency electromagnetic fields in a source-free region occupied by isotropic media as an asymptotic power series in inverse powers of frequency ω as

[39]: Lüneburg (1944), *Mathematical Theory of Optics*

[40]: Kline (1951), *An asymptotic solution of Maxwell's equations*

$$\mathbf{E}(\mathbf{r},\omega) \sim e^{-jk\Psi(\mathbf{r})} \sum_{m=0}^{\infty} \frac{\mathbf{E}_m(\mathbf{r})}{(j\omega)^m}, \tag{2.1}$$

$$\mathbf{H}(\mathbf{r},\omega) \sim e^{-jk\Psi(\mathbf{r})} \sum_{m=0}^{\infty} \frac{\mathbf{H}_m(\mathbf{r})}{(j\omega)^m}, \tag{2.2}$$

where $\sim$ indicates an asymptotic equality, $\Psi(\mathbf{r})$ is the phase function, and $k = \omega v^{-1}$ is the wavenumber.

To derive the fundamental equations of ray optics, we substitute the asymptotic expansions (2.1) and (2.2) into the source-free Maxwell's equations [41, Secs. 2.2.4 and 2.2.5]. For the remainder of this section, we will drop the explicit dependency on $\mathbf{r}$ for brevity. For instance, considering Faraday's law in the time-harmonic case (1.11), and applying the vector identity

[41]: McNamara et al. (1990), *Introduction to the Uniform Geometrical Theory of Diffraction*

$$\nabla \times (f\mathbf{F}) = \nabla f \times \mathbf{F} + f \nabla \times \mathbf{F}, \tag{2.3}$$

with

$$f = e^{-jk\Psi}, \tag{2.4}$$

and

$$\mathbf{F} = \sum_{m=0}^{\infty} \frac{\mathbf{E}_m}{(j\omega)^m}, \tag{2.5}$$

we obtain

$$e^{-jk\Psi} \sum_{m=0}^{\infty} \left[\frac{\nabla \times \mathbf{E}_m}{(j\omega)^m} - \frac{1}{v} \frac{\nabla\Psi \times \mathbf{E}_m}{(j\omega)^{m-1}} \right] = -\mu e^{-jk\Psi} \sum_{m=0}^{\infty} \frac{\mathbf{H}_m}{(j\omega)^{m-1}}, \tag{2.6}$$

where we used the relation $k = \omega v^{-1}$.

[†] While many papers, including Kline's, cite him as Luneberg, the correct spelling is Lüneburg.

At high frequencies, the denominator ω^m becomes extremely large, so that the terms in the series decrease very rapidly. We can therefore reasonably truncate the series when the exponent is greater than 0, and simplify the expression, which leads to

$$\left[\frac{\nu^{-1}(\nabla\Psi\times\mathbf{E}_0)}{(j\omega)^{-1}}+\frac{\nu^{-1}(\nabla\Psi\times\mathbf{E}_1)-\nabla\times\mathbf{E}_0}{(j\omega)^0}\right]=\mu\left[\frac{\mathbf{H}_0}{(j\omega)^{-1}}+\frac{\mathbf{H}_1}{(j\omega)^0}\right]. \tag{2.7}$$

Because (2.7) must hold for different frequencies, we can equate the left- and right-hand side coefficients of the different powers of $j\omega$. For the leading-order terms, we find

$$\frac{1}{\nu}(\nabla\Psi\times\mathbf{E}_0)=\mu\mathbf{H}_0. \tag{2.8}$$

A similar procedure applied to the other Maxwell's equations in the time-harmonic case (1.12), (1.9), and (1.10), yields three additional relations for the leading-order terms

$$\frac{1}{\nu}(\nabla\Psi\times\mathbf{H}_0)=-\epsilon\mathbf{E}_0, \tag{2.9}$$

$$\nabla\Psi\cdot\mathbf{E}_0=0, \tag{2.10}$$

$$\nabla\Psi\cdot\mathbf{H}_0=0. \tag{2.11}$$

Equations (2.10) and (2.11) reveal that the electric and magnetic field vectors are orthogonal to each other and to the direction of propagation $\nabla\Psi$, confirming that high-frequency waves behave locally as uniform TEM plane waves.

2.2.2 The Eikonal Equation

We can rewrite the first relation (2.8) using the intrinsic impedance $Z=\sqrt{\frac{\mu}{\epsilon}}=\mu\nu$ to obtain

$$\mathbf{H}_0=\frac{1}{Z}(\nabla\Psi\times\mathbf{E}_0). \tag{2.12}$$

Substituting this into (2.9) and applying the vector triple product identity gives

$$(\nabla\Psi \cdot \mathbf{E}_0)\nabla\Psi - |\nabla\Psi|^2\mathbf{E}_0 = -\mathbf{E}_0. \tag{2.13}$$

Since $\nabla\Psi \cdot \mathbf{E}_0 = 0$ from (2.10), the previous equation simplifies directly to what is called the *eikonal equation*

$$|\nabla\Psi|^2 = 1. \tag{2.14}$$

As a consequence, the phase function $\Psi(\mathbf{r})$ is sometimes referred to as the *eikonal function*. In this normalization, Ψ has the dimension of length and can be interpreted as an optical path-length coordinate. In homogeneous media, it grows linearly along each ray trajectory.

2.2.3 The Transport Equations

While the eikonal equation determines the phase variation along the propagation path of high-frequency fields, the amplitude variation along this path is governed by the *transport equations*. These are obtained by substituting the Lüneburg-Kline expansion for the electric field (2.1) into the vector Helmholtz equation (1.17). After differentiating and grouping the terms by powers of $j\omega$, we obtain

$$e^{-jk\Psi}\sum_{m=0}^{\infty}\left[\frac{(1-|\nabla\Psi|^2)\mathbf{E}_m}{\nu(j\omega)^{m-2}} - \frac{\mathbf{E}_m\nabla^2\Psi + 2(\nabla\Psi\cdot\nabla)\mathbf{E}_m}{\nu(j\omega)^{m-1}} + \frac{\nabla^2\mathbf{E}_m}{(j\omega)^m}\right] = 0. \tag{2.15}$$

Again, we can equate the coefficients of the different powers of $j\omega$ by grouping terms of equal magnitude in inverse powers of frequency. For the $(j\omega)^2$ coefficients, we re-obtain the eikonal equation. In the case of the coefficients of $(j\omega)^1$, we obtain the *zeroth-order transport equation*

$$2(\nabla\Psi\cdot\nabla)\mathbf{E}_0 + (\nabla^2\Psi)\mathbf{E}_0 = 0. \tag{2.16}$$

This equation enforces the conservation of energy flux. As a wave propagates, the term $\nabla^2\Psi$ acts as a measure of the wavefront's curvature: positive values correspond to local divergence (amplitude decay), while negative values correspond to convergence (amplitude growth). In the locally planar limit, $\nabla^2\Psi \to 0$, so amplitude changes are driven only by higher-order effects.

Equating the $(j\omega)^0$ terms leaves only one remaining relation

$$\nabla^2\mathbf{E}_0 = 0. \tag{2.17}$$

Finally, the *higher-order transport equations* are obtained by equating the coefficients of the remaining powers of $j\omega$, i.e.,

$$2(\nabla\Psi\cdot\nabla)\mathbf{E}_m + (\nabla^2\Psi)\mathbf{E}_m = -\nu\nabla^2\mathbf{E}_{m-1}, \tag{2.18}$$

with $m \geq 1$, but they are not used in the high-frequency approximation.

2.2.4 GO Ray Trajectories Follow Straight Lines

As motivated earlier, the higher-order terms in the asymptotic expansions (2.1) and (2.2) can be neglected in the high-frequency approximation. The only remaining terms are the lowest ones: $\mathbf{E}_0$ and $\mathbf{H}_0$. Those are called the geometrical optics (GO) field terms

$$\mathbf{E}(\mathbf{r},\omega) \underset{\omega\to\infty}{\sim} \mathbf{E}_0(\mathbf{r})e^{-jk\Psi(\mathbf{r})}, \tag{2.19}$$

$$\mathbf{H}(\mathbf{r},\omega) \underset{\omega\to\infty}{\sim} \mathbf{H}_0(\mathbf{r})e^{-jk\Psi(\mathbf{r})}. \tag{2.20}$$

By reusing what we have derived in the previous sections,

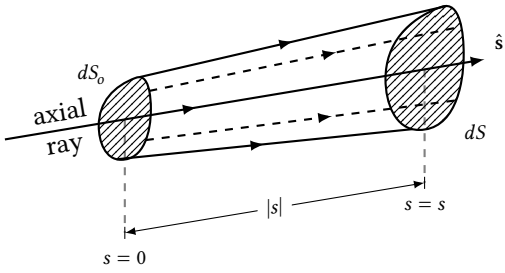


Figure 2.6: Narrow ray tube, reproduced from [41, Fig. 2.2].

we can compute the Poynting vector

$$\mathbf{S} = \mathbf{E} \times \mathbf{H}^*, \tag{2.21}$$

$$= \frac{1}{Z}\mathbf{E}_0 \times \mathbf{H}_0^*, \tag{2.22}$$

$$= \frac{1}{Z}(\mathbf{E}_0 \cdot \mathbf{E}_0^*)\nabla\Psi, \tag{2.23}$$

$$= \frac{1}{Z}|\mathbf{E}_0|^2\nabla\Psi, \tag{2.24}$$

which indicates that the power flows along the direction of propagation $\nabla\Psi$.

Moreover, the eikonal equation (2.14) imposes that $|\nabla\Psi| = 1$, which means that the direction of the Poynting vector $\hat{\mathbf{s}}$ is exactly equal to $\nabla\Psi$. We call this the *ray direction*. Since $\hat{\mathbf{s}}$ is a unit vector, GO rays can be understood as curves everywhere tangent to this direction, representing the paths along which electromagnetic energy propagates.

A fundamental property of GO is that energy transport occurs exclusively along ray trajectories—there is no transverse energy flow perpendicular to a ray. This leads naturally to the concept of a *ray tube*: a bundle of neighboring rays surrounding a central reference ray, as illustrated in Figure 2.6. Because the boundary of such a tube is constructed from the trajectories of adjacent rays, and because energy cannot cross these boundaries, the electromagnetic power passing through any cross-section of a given ray tube remains constant in the absence of absorption or scattering. This power conservation principle is central to understanding how field amplitudes evolve as wavefronts

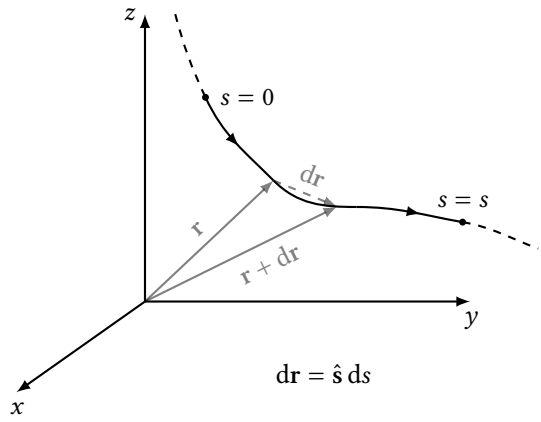


Figure 2.7: Ray coordinate system, reproduced from [41, Fig. 2.3].

expand or contract.

From the orthogonality of $\mathbf{E}_0$ and $\mathbf{H}_0$ with respect to $\nabla\Psi$, we can deduce that the GO fields act locally as plane waves.

Now that we have established this, we still are missing information about the ray trajectory, polarization, amplitude and phase of the wave as it propagates.

Ray Trajectories

For the trajectory, we express the position vector $\mathbf{r}$ as a function of a single coordinate s measured along the propagation direction $\hat{\mathbf{s}}$, as shown in Figure 2.7. An infinitesimal increment in the position vector $\mathrm{d}\mathbf{r}$ is related to the infinitesimal increment in s through the relation

$$\mathrm{d}\mathbf{r} = \hat{\mathbf{s}}\mathrm{d}s. \tag{2.25}$$

Then, we can rewrite the eikonal equation (2.14) as

$$\frac{\mathrm{d}\Psi}{\mathrm{d}s} = 1, \tag{2.26}$$

and the zeroth-order transport equation (2.16) as

$$2\frac{\mathrm{d}\mathbf{E}_0}{\mathrm{d}s} + (\nabla^2\Psi)\mathbf{E}_0 = 0. \tag{2.27}$$

For the ray trajectory, differentiating the relation $\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}s} = \nabla\Psi$ with respect to s yields

$$\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}s^2} = \frac{\mathrm{d}}{\mathrm{d}s}(\nabla\Psi) = (\hat{\mathbf{s}}\cdot\nabla)\nabla\Psi = (\nabla\Psi\cdot\nabla)\nabla\Psi. \tag{2.28}$$

Using the vector identity $(\nabla\Psi\cdot\nabla)\nabla\Psi = \frac{1}{2}\nabla(|\nabla\Psi|^2)$ along with the eikonal equation (2.14), we obtain the second-order differential equation,

$$\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}s^2} = 0, \tag{2.29}$$

which has a solution in the form of a straight line

$$\mathbf{r}(s) = \mathbf{A} + s\mathbf{B}, \tag{2.30}$$

where $\mathbf{A}$ and $\mathbf{B}$ are constant vectors.

Although we have demonstrated that the trajectories of GO fields follow straight lines in homogeneous media, it is important to note that **this is not true in inhomogeneous media**, where the ray trajectories are curved. While it is not trivial to solve for the exact ray trajectories in inhomogeneous media, it is possible to do so in piecewise homogeneous media, which is the case in most practical scenarios. In these media, rays travel in straight lines within homogeneous regions but bend at interfaces according to Snell's law.

Lastly, it is essential to highlight that ray tracing algorithms almost always assume that ray trajectories are linear. Consequently, for the remainder of this book, when we refer to ray trajectories, we are referring to straight line segments.

Polarization

As discussed in Chapter 1, we can express the electric and magnetic fields of a locally plane wave as combinations of two orthogonal polarizations. In the non-restrictive case of a single polarization direction for both fields, which we denote as the unit vectors $\hat{\mathbf{e}}$ and $\hat{\mathbf{h}}$ for the electric and magnetic fields, respectively, we can write

$$\hat{\mathbf{e}} = \frac{\mathbf{E}_0}{|\mathbf{E}_0|}, \quad \text{and} \quad \hat{\mathbf{h}} = \frac{\mathbf{H}_0}{|\mathbf{H}_0|}. \tag{2.31}$$

From the properties established in the previous sections, these unit vectors are orthogonal to the ray direction $\hat{\mathbf{s}}$, yielding

$$\hat{\mathbf{e}} \cdot \hat{\mathbf{s}} = 0, \quad \text{and} \quad \hat{\mathbf{h}} = \hat{\mathbf{s}} \times \hat{\mathbf{e}}. \tag{2.32}$$

To determine how the polarization behaves as the wave propagates, we evaluate its change along the ray trajectory by examining the derivative $\frac{\mathrm{d}\hat{\mathbf{e}}}{\mathrm{d}s}$. Using the definition of $\hat{\mathbf{e}}$ along with the zeroth-order transport equation (2.16), we can algebraically deduce that this derivative vanishes

$$\frac{\mathrm{d}\hat{\mathbf{e}}}{\mathrm{d}s} = 0. \tag{2.33}$$

This result demonstrates that the polarization vector $\hat{\mathbf{e}}$ remains constant as the ray propagates through homogeneous regions. Consequently, once the polarization is initialized at a reference point, it is perfectly defined along the entirety of the ray path until it interacts with an object. This property simplifies ray tracing, allowing us to only consider polarization changes during discrete interactions (like reflections, transmissions, or diffractions).

For more complex fields exhibiting circular or elliptical polarizations, linearity allows us to decompose the field into two orthogonal linear components, track their constant polarization directions independently along the ray paths, and recombine them to find the total state.

Phase

The phase variation of the GO field along a ray trajectory is governed by the eikonal equation (2.14), taking the form $|\nabla\Psi| = 1$. In a homogeneous medium, where rays are straight lines and orthogonal to the equiphase surfaces, the incremental phase change $\mathrm{d}\Psi$ over a small distance $\mathrm{d}s$ is simply

$$\mathrm{d}\Psi = |\nabla\Psi|\mathrm{d}s = \mathrm{d}s. \tag{2.34}$$

Integrating this relation along the ray path from a reference origin s_0 to an arbitrary distance s yields

$$\Psi(s) - \Psi(s_0) = s - s_0. \tag{2.35}$$

By adopting the convention that the reference point is placed at $s_0 = 0$, the phase function evolution simplifies to

$$\Psi(s) = \Psi(0) + s. \tag{2.36}$$

Consequently, the exponential phase term describing the spatial oscillation of the field along the ray reduces to

$$e^{-jk\Psi(s)} = e^{-jk\Psi(0)}e^{-jks}. \tag{2.37}$$

Amplitude

The final element required to fully define the GO field behavior is the amplitude continuation function. To obtain this, we must solve the zeroth-order transport equation (2.16). This differential equation can be integrated with respect to s from the reference point $s = 0$ to yield

$$\mathbf{E}_0(s) = \mathbf{E}_0(0)\exp\left[-\frac{1}{2}\int_0^s \nabla^2\Psi\,\mathrm{d}s'\right]. \tag{2.38}$$

To make (2.38) practically useful, we must evaluate the Laplacian of the phase, $\nabla^2\Psi$, which relates to the curvature

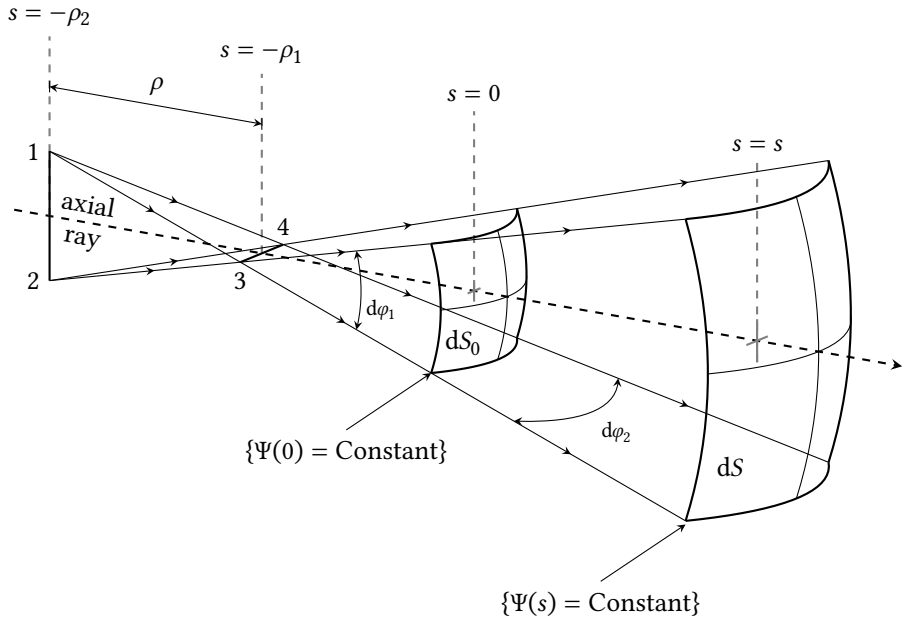


Figure 2.8: Diverging astigmatic ray tube, reproduced from [41, Fig. 2.4].

of the wavefront. First, we define an *astigmatic ray tube* to be a narrow bundle of rays centered around a central ray, as illustrated in Figure 2.6. At the reference wavefront $\Psi(0)$, the tube has two principal radii of curvature, ρ_1 and ρ_2. As the wave propagates a distance s along the central ray to reach surface $\Psi(s)$, these principal radii expand to $\rho_1 + s$ and $\rho_2 + s$. This is illustrated in Figure 2.8.

The ratio of the physical cross-sectional area of the ray tube at distance s to its area at $s = 0$ can be connected to the divergence of the power density vector. Through the divergence theorem, this leads to the intensity law (or power conservation along the ray tube), allowing us to rewrite the exponential term analytically as the square root of the ratio of the ray tube areas. Thus, the field amplitude

evolves as

$$\mathbf{E}_0(s) = \mathbf{E}_0(0)\sqrt{\frac{\rho_1\rho_2}{(\rho_1+s)(\rho_2+s)}}. \tag{2.39}$$

Combining the amplitude, phase, and polarization components, the complete and final expression for the GO electric field at a distance s from a reference point is

$$\mathbf{E}(s) = \mathbf{E}(0)\sqrt{\frac{\rho_1\rho_2}{(\rho_1+s)(\rho_2+s)}}e^{-jks}, \tag{2.40}$$

where we have incorporated the initial phase $e^{-jk\Psi(0)}$ into the complex reference field $\mathbf{E}(0)$.

Equation (2.40) represents the *ray optical continuation*. Given the initial complex vector field $\mathbf{E}(0)$ and the wavefront's principal radii of curvature ρ_1 and ρ_2 at a known point, we can compute the precise characteristics of the GO field at any subsequent point along the ray trajectory. The square-root term,

$$A(s) = \sqrt{\frac{\rho_1\rho_2}{(\rho_1+s)(\rho_2+s)}}, \tag{2.41}$$

is called the *spreading factor*, and it accounts for the divergence or convergence of the ray tube as the wavefront propagates. For plane waves ($\rho_1, \rho_2 \to \infty$), the spreading factor approaches unity, while for curved wavefronts it modulates the amplitude decay (or growth, if the wave converges).

This analytical simplicity is a key strength of ray tracing, although the formulation does break down when the ray tube converges precisely to a line or point ($\rho_1 + s = 0$ or $\rho_2 + s = 0$). Such locations are known as *caustics*, and at these points, equation (2.40) incorrectly predicts an infinite field magnitude. Modeling the true bounded fields near caustics requires more advanced techniques that fall beyond the scope of simple GO.

Three special cases of wavefront curvature are worth not-

ing: planar wavefronts, cylindrical wavefronts, and spherical wavefronts. All of them are solutions to the eikonal equation (2.14).

Plane waves For a plane wave, both principal radii of curvature approach infinity ($\rho_1 \to \infty$ and $\rho_2 \to \infty$), indicating that the wavefront surfaces are perfectly flat and parallel. Substituting these limits into (2.40), the spreading factor reduces to unity, yielding

$$\mathbf{E}(s) = \mathbf{E}(0)e^{-jks}. \tag{2.42}$$

This expression shows that plane waves propagate without amplitude decay due to spreading—only the phase advances linearly with distance. This idealization is often used to approximate fields in the far-field region of a source, where the wavefront curvature becomes negligible over the observation region.

Cylindrical waves A cylindrical wave arises when one principal radius of curvature is finite while the other approaches infinity. Without loss of generality, let $\rho_1 \to \infty$ and $\rho_2 = \rho$. This corresponds to a wavefront that is flat in one transverse direction but curved in the other, such as the field radiated by an infinite line source. Applying these conditions to (2.40) gives

$$\mathbf{E}(s) = \mathbf{E}(0)\sqrt{\frac{\rho}{\rho + s}}e^{-jks}. \tag{2.43}$$

The amplitude decays proportionally to $\frac{1}{\sqrt{s}}$ as s becomes large compared to ρ, reflecting the one-dimensional spreading of energy in the curved direction.

Spherical waves When both principal radii of curvature are equal and finite ($\rho_1 = \rho_2 = \rho$), the wavefront is

spherical, characteristic of radiation from a point source. Equation (2.40) simplifies to

$$\mathbf{E}(s) = \mathbf{E}(0)\frac{\rho}{\rho + s}e^{-jks}. \tag{2.44}$$

For large distances where $s \gg \rho$, this reduces to the familiar $\frac{1}{s}$ amplitude decay, corresponding to the inverse-distance law for spherical wave propagation. This is the most common wavefront geometry encountered in practical antenna radiation and point-source scattering problems.

2.2.5 Fermat's Principle

An alternative and powerful formulation for determining ray trajectories is provided by *Fermat's principle*, which states that the path taken by a ray between two points is such that the *optical path length* is stationary (typically a minimum, but more generally an extremum). The optical path length L along a trajectory from point **A** to point **B** is defined as

$$L = \int_{\mathbf{A}}^{\mathbf{B}} n(s)\,\mathrm{d}s, \tag{2.45}$$

where $n(s)$ is the refractive index of the medium and the integral is taken along the ray path.

In a homogeneous medium with constant n, this reduces to simply the geometric distance, and Fermat's principle confirms that rays travel in straight lines. However, the true utility of Fermat's principle emerges when rays interact with boundaries or propagate through inhomogeneous media. At interfaces, the principle directly leads to laws of reflection and refraction, as derived by Snell, without requiring explicit solution of the transport equations.

Fermat's principle provides a practical advantage to ray tracing algorithms. Rather than solving differential equations for ray trajectories, the problem can be redefined

as an optimization problem: finding the path that extremizes optical length. This variational approach is extremely valuable in scenarios involving reflections, refractions, and diffractions at surfaces because the interaction points and ray directions can be determined by enforcing the stationarity condition. This geometric interpretation is fundamental to many efficient ray tracing techniques, the most important of which will be introduced in Chapter 3, and it provides intuitive insight into the physical behavior of high-frequency wave propagation.

2.2.6 Geometrical Optics versus Physical Optics

At this stage, we should clearly distinguish two asymptotic frameworks that apply to high frequencies: geometrical optics (GO) and physical optics (PO). Both operate in the short-wavelength regime, but they model wave–object interactions differently.

GO is the framework developed in the previous sections. It represents propagation through ray trajectories governed by the eikonal equation, with amplitude and phase continued by the transport equations and the spreading factor. Interactions are treated locally through geometric laws (e.g., reflection or refraction), which makes the method computationally efficient and well suited to large deterministic channel simulations. As we will see later in this chapter, its main limitation is also geometric: classical GO predicts discontinuous fields at shadow boundaries and no field in shadow regions.

PO, in contrast, is a surface-current formulation. The incident field (often obtained from a GO construction) is first used to approximate equivalent electric and magnetic currents on illuminated parts of a scatterer, and the scattered field is then evaluated through surface integration of those currents. This description better captures finite-surface scattering and naturally yields continuous transitions near shadow boundaries, but at a much higher computational

cost because each interaction requires evaluating surface integrals.

For the remainder of this book, we adopt a GO-based ray tracing model and account for its shadow-boundary deficiencies with UTD. The purpose of this subsection is therefore not to develop PO, but to clarify terminology and scope before introducing reflection and diffraction models in the next sections.

2.3 Antenna Modeling in Ray Tracing

In Chapter 1, we introduced the fundamental properties of antennas, including their radiation patterns, directivity, and gain. However, a complete electromagnetic model of an antenna involves solving Maxwell's equations in the near-field region, which is computationally prohibitive for large-scale ray tracing simulations. Fortunately, for typical wireless communication scenarios, the interactions between transmitters and receivers occur in the far-field region, where a significantly simplified model can be employed. This section describes how antennas are modeled in ray tracing frameworks through far-field approximations.

2.3.1 Transmitting Antenna

In the far-field region of an antenna radiating in free space, the electric field can be accurately represented as a *spherical wave* emanating from the antenna's phase center. At a point in space defined by spherical coordinates (r, θ, φ), where r is the radial distance, θ is the polar angle, and φ is the azimuth angle, the electric field phasor of the radiated wave takes the form

$$\mathbf{E}(r, \theta, \varphi) = \mathbf{E}_0(\theta, \varphi) \frac{e^{-jkr}}{r}, \tag{2.46}$$

where $\mathbf{E}_0(\theta, \varphi)$ is the complex vector amplitude that depends on the direction of observation but not on the distance. This $\frac{1}{r}$ decay reflects the spherical spreading of energy as discussed in the previous section.

To make this expression practical for ray tracing, we introduce the *normalized complex antenna field pattern* $\mathbf{F}(\theta, \varphi)$, which characterizes the directional radiation properties of the antenna. According to IEEE Std 145, a radiation pattern describes the spatial distribution of a physical quantity (such as field strength or power density) that characterizes the electromagnetic field generated by an antenna. In particular, a field pattern is defined for a specific component of the electric field (e.g., the θ or φ polarization component) and is normalized to its maximum magnitude. To capture the full polarimetric vector nature, we define the normalized complex field pattern vector $\mathbf{F}(\theta, \varphi)$ through its components in the spherical basis as

$$\mathbf{F}(\theta, \varphi) = F_\theta(\theta, \varphi)\hat{\boldsymbol{\theta}} + F_\varphi(\theta, \varphi)\hat{\boldsymbol{\varphi}}, \tag{2.47}$$

where each component pattern is normalized by the maximum magnitude of the total electric field in the far field, i.e.,

$$F_\theta(\theta, \varphi) = \frac{E_{0,\theta}(\theta, \varphi)}{\max_{\theta', \varphi'} \|\mathbf{E}_0(\theta', \varphi')\|}, \quad F_\varphi(\theta, \varphi) = \frac{E_{0,\varphi}(\theta, \varphi)}{\max_{\theta', \varphi'} \|\mathbf{E}_0(\theta', \varphi')\|}. \tag{2.48}$$

By this construction, the maximum magnitude of the normalized complex vector field pattern satisfies

$$\max_{\theta, \varphi} \|\mathbf{F}(\theta, \varphi)\| = 1, \tag{2.49}$$

where F_θ and F_φ are complex-valued functions capturing both the amplitude and phase variation of the respective polarization components. Because the field pattern is defined using the field envelope $\mathbf{E}_0$ rather than the full spherical wave $\mathbf{E}(r, \theta, \varphi)$, the radial factor $\frac{1}{r}$ is factored out.

The time-averaged Poynting vector for such a spherical wave, which represents the power density per unit area, is

given by

$$\mathbf{S}_{\text{avg}}(r,\theta,\varphi) = \frac{1}{2Z_0}\|\mathbf{E}(r,\theta,\varphi)\|^2\hat{\mathbf{r}}, \tag{2.50}$$

where $\hat{\mathbf{r}}$ is the radial unit vector.

To describe the angular distribution of radiated power independently of distance, we define the *radiation intensity* $U(\theta,\varphi)$ as the power radiated per unit solid angle. It is obtained by multiplying the radial power density by the square of the distance, i.e.,

$$U(\theta,\varphi) = r^2\|\mathbf{S}_{\text{avg}}(r,\theta,\varphi)\| = \frac{1}{2Z_0}\|\mathbf{E}_0(\theta,\varphi)\|^2. \tag{2.51}$$

Since the time-averaged power density $\|\mathbf{S}_{\text{avg}}\|$ decays as $\frac{1}{r^2}$ in the far field due to the spherical spreading of the electric field magnitude (proportional to $\frac{1}{r}$), the multiplication by r^2 cancels this distance dependence, leaving $U(\theta,\varphi)$ as a purely angular quantity.

Using the same notation as in the previous chapter, we can find the expression for the directivity $D(\theta,\varphi)$ of a transmitting antenna, which compares its radiation intensity to that of an isotropic radiator, given by

$$D(\theta,\varphi) = \frac{U(\theta,\varphi)}{\frac{1}{4\pi}\int_0^{2\pi}\int_0^{\pi} U(\theta',\varphi')\sin\theta'\,\mathrm{d}\theta'\,\mathrm{d}\varphi'}. \tag{2.52}$$

The gain $G(\theta,\varphi)$ of the transmitting antenna is related to directivity by its radiation efficiency η_{rad}. Directivity is purely angular and lossless by construction, whereas gain additionally accounts for ohmic and mismatch losses; the two quantities coincide only when $\eta_{\text{rad}} = 1$. Assuming an ideal, lossless antenna, the total radiated power equals the net power captured by the antenna from the connected transmitter, denoted as P_t, and the gain is given by

$$G(\theta,\varphi) = \frac{U(\theta,\varphi)}{\frac{P_t}{4\pi}} = \frac{2\pi}{Z_0 P_t}\|\mathbf{E}_0(\theta,\varphi)\|^2. \tag{2.53}$$

The maximum gain of the transmitting antenna is denoted as

$$G_t \triangleq \max_{\theta,\varphi} G(\theta, \varphi). \tag{2.54}$$

Because $G(\theta, \varphi)$ is proportional to $\|\mathbf{E}_0(\theta, \varphi)\|^2$, the normalized field pattern is related to the gain by $G(\theta, \varphi) = G_t\|\mathbf{F}(\theta, \varphi)\|^2$. Therefore, we can express the complex amplitude as

$$\mathbf{E}_0(\theta, \varphi) = \sqrt{\frac{P_t G_t Z_0}{2\pi}}\mathbf{F}(\theta, \varphi). \tag{2.55}$$

By combining these expressions, we obtain the electric far field of a transmitting antenna as a function of the transmit power, maximum gain, and normalized field pattern, given by

$$\mathbf{E}_t(r, \theta_t, \varphi_t) = \sqrt{\frac{P_t G_t Z_0}{2\pi}}\frac{e^{-jkr}}{r}\mathbf{F}_t(\theta_t, \varphi_t), \tag{2.56}$$

where θ_t and φ_t are the angles defining the direction of the ray emanating from the transmitting antenna.

2.3.2 Receiving Antenna

While the transmitting antenna radiates a spherical wave, the receiving antenna typically observes an *incoming plane wave* $\mathbf{E}_r$ arriving from a specific direction (θ_r, φ_r), defined in the local spherical coordinate system of the receiver. This is because, by reciprocity, the field that has propagated over large distances and potentially undergone multiple interactions appears locally planar at the receiving antenna location. The Poynting vector of this incoming wave is

$$\mathbf{S}_r = -\frac{1}{2Z_0}\|\mathbf{E}_r\|^2\hat{\mathbf{r}}(\theta_r, \varphi_r), \tag{2.57}$$

where the negative sign indicates power flow toward the receiving antenna.

The antenna aperture (or effective area) $A_{e,r}$ of the receiving antenna characterizes its ability to extract power from

an incident plane wave. For an antenna with gain G_r, this aperture is related to the wavelength by

$$A_{e,r}(\theta_r, \varphi_r) = G_r(\theta_r, \varphi_r)\frac{\lambda^2}{4\pi}, \tag{2.58}$$

where $\lambda = \frac{2\pi}{k}$ is the wavelength. For perfect polarization matching and optimal antenna orientation toward the incoming wave direction, the maximum available received power P_r is given by

$$P_r = A_{e,r}\|\mathbf{S}_r\| = G_r\frac{\lambda^2}{4\pi}\frac{\|\mathbf{E}_r\|^2}{2Z_0}. \tag{2.59}$$

However, in general, the polarization of the incident wave may not align with the receiving antenna's polarization preference, and the wave may arrive from a direction where the antenna gain is not maximal. To account for these effects, we express the received power in terms of the receiving antenna's normalized complex field pattern $\mathbf{F}_r(\theta_r, \varphi_r)$ and its maximum gain G_r. The polarization mismatch is quantified by the polarization efficiency (or polarization mismatch factor) η_{pol}, which is the squared magnitude of the inner product between the unit polarization vector representing the antenna's receiving polarization state

$$\hat{\boldsymbol{\rho}}_r = \frac{\mathbf{F}_r^*(\theta_r, \varphi_r)}{\|\mathbf{F}_r(\theta_r, \varphi_r)\|}, \tag{2.60}$$

and the unit polarization vector of the incident wave

$$\hat{\boldsymbol{\rho}}_i = \frac{\mathbf{E}_r}{\|\mathbf{E}_r\|}, \tag{2.61}$$

given by

$$\eta_{\text{pol}} = |\hat{\boldsymbol{\rho}}_r \cdot \hat{\boldsymbol{\rho}}_i|^2 = \left|\frac{\mathbf{F}_r^*(\theta_r, \varphi_r)\mathbf{E}_r}{\|\mathbf{F}_r(\theta_r, \varphi_r)\|\|\mathbf{E}_r\|}\right|^2, \tag{2.62}$$

where the superscript $*$ denotes the Hermitian transpose (conjugate transpose). Combining this mismatch factor

with the directional effective area

$$A_{e,r}(\theta_r, \varphi_r) = G_r \|\mathbf{F}_r(\theta_r, \varphi_r)\|^2 \frac{\lambda^2}{4\pi}, \tag{2.63}$$

and the incident power density

$$\|\mathbf{S}_r\| = \frac{\|\mathbf{E}_r\|^2}{2Z_0}, \tag{2.64}$$

the received power becomes

$$P_r = A_{e,r}(\theta_r, \varphi_r)\|\mathbf{S}_r\|\eta_{\text{pol}} = G_r \|\mathbf{F}_r(\theta_r, \varphi_r)\|^2 \frac{\lambda^2}{4\pi} \frac{\|\mathbf{E}_r\|^2}{2Z_0} \left| \frac{\mathbf{F}_r^*(\theta_r, \varphi_r)\mathbf{E}_r}{\|\mathbf{F}_r(\theta_r, \varphi_r)\|\|\mathbf{E}_r\|} \right|^2. \tag{2.65}$$

Since the norms $\|\mathbf{F}_r(\theta_r, \varphi_r)\|^2$ and $\|\mathbf{E}_r\|^2$ in the numerator and denominator cancel out, this simplifies to the compact received power formula

$$P_r = G_r \frac{\lambda^2}{8\pi Z_0} \left| \mathbf{F}_r^*(\theta_r, \varphi_r)\mathbf{E}_r \right|^2. \tag{2.66}$$

The inner product term $\mathbf{F}_r^*\mathbf{E}_r$ naturally accounts for polarization mismatch—if the incident field is orthogonal to the antenna pattern (e.g., a horizontally polarized wave incident on a vertically polarized antenna), the received power vanishes.

The transmitting and receiving antenna models thus define the boundary conditions of a ray-based channel: they determine how fields are launched into the environment and how arriving fields are projected at the receiver. However, as discussed in Chapter 1, propagation through realistic environments involves boundary interactions (reflection, transmission, diffraction, etc.) that modify polarization in addition to amplitude and phase. The next ingredient is therefore a compact formalism to propagate polarization through these interactions without repeatedly switching to full vector-field algebra.

2.4 Introduction to Jones Calculus

The propagation and scattering of electromagnetic waves in ray tracing require careful tracking of the polarization state as a field interacts with various objects in the environment. In Chapter 1, polarization was introduced as the superposition of two orthogonal transverse components, using the $(\perp, \parallel)$ notation relative to the plane of incidence. We now formalize that idea algebraically. A compact tool for this purpose is *Jones calculus*, originally introduced by R. Clark Jones for polarized light [42]. Although developed in optics, the same complex two-component formalism applies directly to monochromatic electromagnetic waves in radio propagation: the field is represented in a local transverse basis, and each interaction is modeled as a 2×2 complex matrix acting on that vector.

[42]: Jones (1941), *A New Calculus for the Treatment of Optical Systems. I. Description and Discussion of the Calculus*

2.4.1 Jones Vectors

Consider a monochromatic plane wave oscillating in a plane perpendicular to its direction of propagation. The complex electric field of this wave can be expressed as a superposition of two orthogonal polarization components. At any point in space, we can decompose this field into components in a local basis; for interactions with surfaces, it is conventionally chosen as the plane of incidence and its perpendicular direction.

Let $\hat{\mathbf{e}}_\perp$ and $\hat{\mathbf{e}}_\parallel$ denote orthonormal basis vectors perpendicular and parallel to the plane of incidence, respectively. The complex electric field $\mathbf{E}$ can then be written as

$$\mathbf{E} = E_\perp \hat{\mathbf{e}}_\perp + E_\parallel \hat{\mathbf{e}}_\parallel, \tag{2.67}$$

where $E_\perp$ and $E_\parallel$ are complex amplitudes representing the field components. The *Jones vector* is the compact representation of this field in the local basis

$$\mathbf{e} = \begin{bmatrix} E_\perp \\ E_\parallel \end{bmatrix}. \tag{2.68}$$

With this representation, the magnitude of the Jones vector gives the overall field amplitude, and the relative phase between the two components encodes the polarization state (linear, circular, or elliptical).

2.4.2 Jones Matrices

When an electromagnetic wave interacts with an object—whether it reflects off a surface, transmits through a dielectric, or diffracts around an edge—the polarization state changes. In Jones calculus, this interaction is modeled by a 2×2 complex matrix, the *Jones matrix* $\underline{\mathbf{J}}$, that transforms the incident Jones vector into the scattered (or transmitted) Jones vector

$$\mathbf{e}^{\text{out}} = \underline{\mathbf{J}}\mathbf{e}^{\text{in}}. \tag{2.69}$$

For an isotropic interaction at a smooth interface (such as a planar surface separating two homogeneous media), the Jones matrix is diagonal in the plane-of-incidence basis

$$\underline{\mathbf{J}} = \begin{bmatrix} J_\perp & 0 \\ 0 & J_\parallel \end{bmatrix}, \tag{2.70}$$

where $J_\perp$ and $J_\parallel$ are complex coefficients (e.g., Fresnel reflection or transmission coefficients) that depend on the material properties and angle of incidence.

In the ray optics literature, Jones matrices are often referred to as *dyadic coefficients* (or simply *dyadics*) when they represent the field transformation at an interaction. For example, the reflection and diffraction interactions introduced later in this chapter are characterized by dyadic reflection and diffraction coefficients, which are 2×2 complex matrices that act on the transverse field components in exactly the same manner as Jones matrices. This terminology emphasizes the tensor nature of the polarization transformation and is standard in GTD and UTD formulations.

2.4.3 Basis Rotations

In three-dimensional ray tracing, the ray propagates through a global coordinate frame (e.g., a Cartesian system fixed to the scene). However, each scattering interaction is naturally described in its own local frame, where the principal directions of polarization are perpendicular and parallel to the plane of incidence.

To apply the Jones matrix formalism, one must:

1. *Rotate from global to local basis* to transform the incident field vector from the global polarization frame to the local frame of the interaction using an appropriate rotation matrix.
2. *Apply the Jones matrix* to compute the scattered field by multiplying the (rotated) incident Jones vector by the interaction's Jones matrix.
3. *Rotate back to global basis* to transform the scattered field vector from the local frame back to the global polarization frame for subsequent propagation and interactions.

Mathematically, if $\underline{\mathbf{R}}$ is the matrix that projects global polarization components onto the local interaction basis (and $\underline{\mathbf{R}}^{-1}$ maps them back), the complete transformation is

$$\mathbf{e}^{\text{out}}_{\text{global}} = \underline{\mathbf{R}}^{-1}\underline{\mathbf{J}}\underline{\mathbf{R}}\mathbf{e}^{\text{in}}_{\text{global}}. \tag{2.71}$$

In practice, $\underline{\mathbf{R}}$ is built directly from basis dot products. Let $\{\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2\}$ denote the global polarization basis used to carry the field, and let $\{\hat{\ell}_1, \hat{\ell}_2\}$ denote the local interaction basis (typically $\hat{\mathbf{e}}_\perp, \hat{\mathbf{e}}_\parallel$). Then

$$R_{ij} = \hat{\mathbf{g}}_i \cdot \hat{\ell}_j, \tag{2.72}$$

that is,

$$\underline{\mathbf{R}} = \begin{bmatrix} \hat{\mathbf{g}}_1 \cdot \hat{\ell}_1 & \hat{\mathbf{g}}_1 \cdot \hat{\ell}_2 \\ \hat{\mathbf{g}}_2 \cdot \hat{\ell}_1 & \hat{\mathbf{g}}_2 \cdot \hat{\ell}_2 \end{bmatrix}. \tag{2.73}$$

This gives an implementation-ready recipe: normalize both bases and fill a 2×2 matrix with projection coefficients.

For intuition, if the local basis is obtained by rotating the global basis by an angle γ in the transverse plane, one can write

$$\underline{\mathbf{R}}(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma \\ -\sin\gamma & \cos\gamma \end{bmatrix}, \tag{2.74}$$

so the local Jones components are $\mathbf{e}_{\text{local}} = \underline{\mathbf{R}}(\gamma)\,\mathbf{e}_{\text{global}}$.

This basis-agnostic approach ensures that polarization effects are correctly accounted for regardless of the global frame orientation, making it robust for implementation in ray tracing software that must handle arbitrary scene geometries.

2.4.4 Cascade of Interactions

A key advantage of the Jones matrix formalism is that a sequence of interactions can be represented as a product of Jones matrices. If a ray undergoes n interactions in order, the overall transformation from the initial field to the final field is simply the matrix product

$$\mathbf{e}^{\text{final}} = \underline{\mathbf{J}}_n \underline{\mathbf{J}}_{n-1} \cdots \underline{\mathbf{J}}_2 \underline{\mathbf{J}}_1 \mathbf{e}^{\text{initial}}, \tag{2.75}$$

provided that all matrices are expressed in a consistent (either global or appropriately rotated local) basis. This cascade structure mirrors the transfer matrix formalism we will introduce in Section 2.10, enabling systematic computation of the end-to-end polarization transformation for any ray path.

2.5 Geometrical Optics Reflected Fields

Having established the theoretical underpinnings of ray tracing (Section 2.2) and introduced the polarization calculus (Section 2.4), we now detail how high-frequency fields behave when they interact with a smooth reflecting surface. This section is the GO-level continuation of

the reflection and transmission physics introduced in Section 1.5.1: Snell's law determines the reflected direction, Fresnel coefficients determine polarization-dependent field transformation, and ray-optical continuation determines spreading and phase after reflection.

2.5.1 The Specular Point and Law of Reflection

In GO, scattering from a smooth, extended surface is dominated locally by a small region where the phase of the scattered field is stationary with respect to the source and observation point coordinates. This region is called the *specular point* $\mathbf{x}_r$, and its location is determined by the *law of reflection* (1.42).

The reflected ray direction $\hat{\mathbf{s}}^r$ is related to the incident ray direction $\hat{\mathbf{s}}^i$ and the surface normal $\hat{\mathbf{n}}$ (pointing outward from the surface) by

$$\hat{\mathbf{s}}^r = \hat{\mathbf{s}}^i - 2(\hat{\mathbf{n}} \cdot \hat{\mathbf{s}}^i)\hat{\mathbf{n}}. \tag{2.76}$$

Equation (2.76) is illustrated in Figure 2.9.

This relation is equivalent to stating that the angle of incidence equals the angle of reflection, both measured from the surface normal. The *plane of incidence* is defined by the vectors $\hat{\mathbf{s}}^i$ and $\hat{\mathbf{n}}$; it contains both the incident ray and the surface normal, and hence also the reflected ray.

For smooth surfaces, the high-frequency scattering integral can be evaluated using the stationary-phase method,

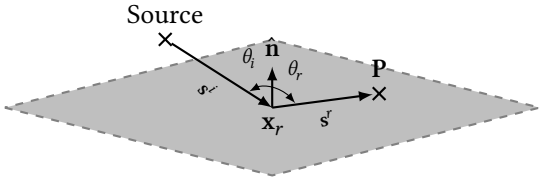


Figure 2.9: Specular reflection on a locally smooth, planar surface.

which concentrates the contribution to a small neighborhood of $\mathbf{x}_r$. In this immediate vicinity, the surface curvature determines how the scattered field amplitude evolves as it propagates away from the scattering point.

2.5.2 Polarization: The Dyadic Reflection Coefficient

At the specular point, the incident field $\mathbf{E}^i(\mathbf{x}_r)$ is a locally plane wave, consistent with the local plane-wave assumptions discussed in Chapter 1. Its polarization is decomposed into perpendicular and parallel components $(E_\perp, E_\parallel)$ with respect to the plane of incidence, using the same convention as in Section 1.5.1. The reflected field at the specular point $\mathbf{E}^r(\mathbf{x}_r)$ is related to the incident field through the *dyadic reflection coefficient* $\underline{\mathbf{R}}$,

$$\mathbf{E}^r(\mathbf{x}_r) = \underline{\mathbf{R}}\mathbf{E}^i(\mathbf{x}_r), \tag{2.77}$$

which acts on the field in the local plane-of-incidence basis.

For a homogeneous, isotropic interface separating two lossless media, the dyadic reflection coefficient is diagonal and given by

$$\underline{\mathbf{R}} = \begin{bmatrix} R_\perp & 0 \\ 0 & R_\parallel \end{bmatrix}, \tag{2.78}$$

where $R_\perp$ and $R_\parallel$ are the Fresnel reflection coefficients for perpendicular and parallel polarization, respectively, as previously defined in Section 1.5.1.

For code implementation, the local ray-fixed basis is obtained from geometry at the specular point. With incident direction $\hat{\mathbf{s}}^i$, reflected direction $\hat{\mathbf{s}}^r$, and surface normal $\hat{\mathbf{n}}$,

$$\hat{\mathbf{e}}_\perp = \frac{\hat{\mathbf{s}}^i \times \hat{\mathbf{n}}}{\|\hat{\mathbf{s}}^i \times \hat{\mathbf{n}}\|}, \tag{2.79}$$

$$\hat{\mathbf{e}}_\parallel^{\,i} = \hat{\mathbf{e}}_\perp \times \hat{\mathbf{s}}^i, \qquad \hat{\mathbf{e}}_\parallel^{\,r} = \hat{\mathbf{e}}_\perp \times \hat{\mathbf{s}}^r. \tag{2.80}$$

Therefore, the perpendicular direction is shared,

$$\hat{\mathbf{e}}_{\perp}^{i} = \hat{\mathbf{e}}_{\perp}^{r} = \hat{\mathbf{e}}_{\perp}, \tag{2.81}$$

whereas the parallel directions are generally different,

$$\hat{\mathbf{e}}_{\parallel}^{i} \neq \hat{\mathbf{e}}_{\parallel}^{r}. \tag{2.82}$$

This geometric distinction is critical; failing to account for it is a frequent source of polarization mismatch errors in simulation software.

For a perfect electrical conductor (PEC), i.e., a material with infinite conductivity, these coefficients are

$$R_{\perp} = -1, \tag{2.83}$$

$$R_{\parallel} = 1. \tag{2.84}$$

More generally, for a dielectric interface at angle of incidence θ_i, the Fresnel coefficients are determined by the permittivity and permeability of the two media and vary with frequency (wavelength) and incidence angle in a well-defined manner.

The use of Jones vectors and the dyadic matrix representation ensures that the method naturally handles arbitrary polarization states, including linear, circular, and elliptical polarizations, and correctly predicts depolarization and cross-polarization isolation effects that arise in practical scenarios.

2.5.3 Amplitude and Phase Continuation

Once the reflected field at the specular point $\mathbf{E}^r(\mathbf{x}_r)$ is computed, it must be propagated to the observation point (e.g., the receiver antenna) along the reflected ray. The reflected ray travels as an astigmatic ray tube; refer to Figure 2.10 for the definition of its parameters.

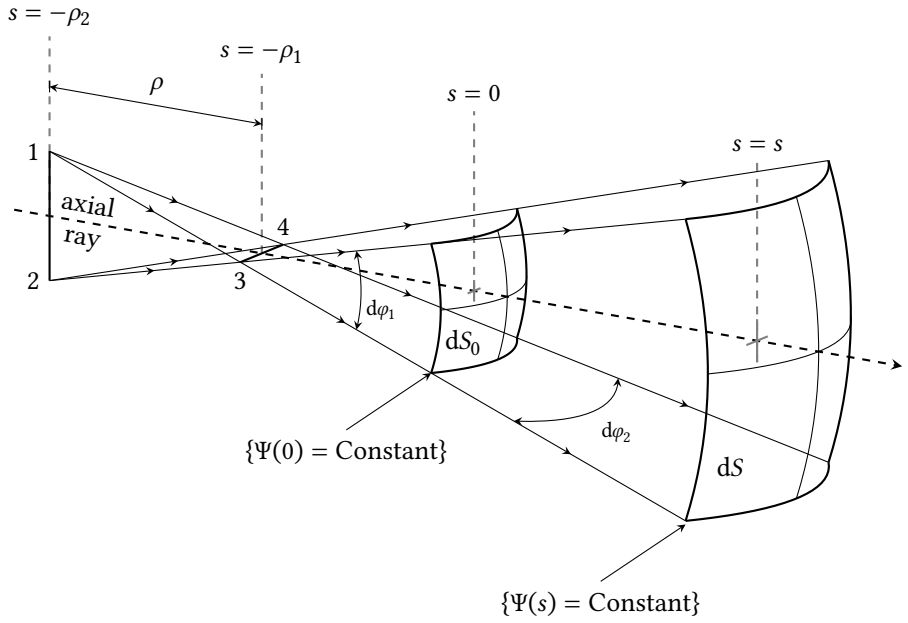


Figure 2.10: Diverging astigmatic ray tube, repeated here for convenience from Figure 2.8.

For general three-dimensional surfaces, the reflected radii ρ_1^r and ρ_2^r are obtained by mapping the incident wavefront radii (ρ_1^i, ρ_2^i) through the local surface curvature, represented by a curvature matrix built from the principal surface radii (a_1, a_2) at $\mathbf{x}_r$, see Figure 2.11. The complete generalized 3D formulas (e.g., the Kouyoumjian–Pathak expression [30]) are intricate and are not reproduced here; we refer the reader to [41, Sec. 3.5] for full expressions and implementation details.

[30]: Kouyoumjian et al. (1974), *A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface*

[41]: McNamara et al. (1990), *Introduction to the Uniform Geometrical Theory of Diffraction*

As the reflected ray propagates a distance s^r from $\mathbf{x}_r$ to the observation point $\mathbf{P}$, the field evolves according to the ray optical continuation law

$$\mathbf{E}^r(\mathbf{P}) = \mathbf{E}^r(\mathbf{x}_r)\, A(s^r; \rho_1^r, \rho_2^r)\, e^{-jks^r}, \tag{2.85}$$

where ρ_1^r and ρ_2^r are the principal radii of curvature of the reflected wavefront at $\mathbf{x}_r$, s^r is the distance from the

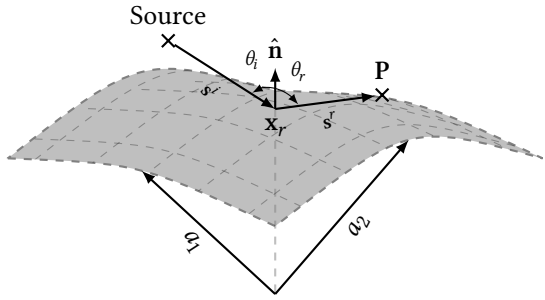


Figure 2.11: Specular reflection on a locally smooth, curved surface.

specular point to the observation point $\mathbf{P}$, and $A(s^r; \rho_1^r, \rho_2^r)$ is the spreading factor defined in (2.41) with parameters $s = s^r$, $\rho_1 = \rho_1^r$, and $\rho_2 = \rho_2^r$.

The three factors in this expression encode distinct physics:

- The *phase term* e^{-jks^r} advances linearly with distance traveled, as expected for a wave of wavenumber k.
- The *spreading factor* $A(s^r; \rho_1^r, \rho_2^r)$, introduced in (2.41), accounts for the transverse spreading (or focusing) of the energy in the ray tube as the wavefront propagates. When both radii are large compared to s^r, the spreading is slow (approaching unity for plane waves). When the radii become comparable to or smaller than s^r, the wave spreads (defocuses) more rapidly. Conversely, if the radii are negative (indicating convergent wavefronts), the wave focuses, and the amplitude initially grows; at the focal distance $|\rho|$, the first-order GO approximation breaks down (caustic). As noted by McNamara *et al.* [41, Sec. 2.2.8], the inability of GO to resolve fields near caustics is generally acceptable in practical outdoor ray tracing, as receivers are highly unlikely to be situated precisely at focal points. For a detailed discussion about how to handle caustics, we therefore refer the reader to [41].
- The *amplitude at specular point* $\mathbf{E}^r(\mathbf{x}_r)$ encodes both the magnitude and phase of the reflected field, including polarization-dependent effects through the Fresnel coefficients.

[41]: McNamara et al. (1990), *Introduction to the Uniform Geometrical Theory of Diffraction*

2.5.4 Special Cases: Spherical and Plane Reflectors

Two important special cases simplify the application of the GO reflected field formula.

Plane Reflector

For a planar (flat) reflecting surface, both principal radii of curvature are infinite: $\rho_1^r \to \infty$ and $\rho_2^r \to \infty$. Equation (2.85) simplifies to

$$\mathbf{E}^r(\mathbf{P}) = \mathbf{E}^r(\mathbf{x}_r)e^{-jks^r}, \tag{2.86}$$

showing that the amplitude is unchanged (apart from the phase and the reflection coefficient itself) as it propagates from the specular point to the observation point. This is the classical result for specular reflection: the reflected field propagates as a plane wave with no spreading loss.

Spherical Reflector

For a spherically curved surface with both principal radii equal ($\rho_1^r = \rho_2^r = R^r$), the spreading factor becomes

$$\sqrt{\frac{(R^r)^2}{(R^r + s^r)^2}} = \frac{R^r}{R^r + s^r}, \tag{2.87}$$

so that

$$\mathbf{E}^r(\mathbf{P}) = \mathbf{E}^r(\mathbf{x}_r)\frac{R^r}{R^r + s^r}e^{-jks^r}. \tag{2.88}$$

This form shows that when $s^r \ll R^r$ (observation point close to the specular point), the field weakens as

$$\frac{R^r}{R^r + s^r} \approx 1 - \frac{s^r}{R^r}. \tag{2.89}$$

For large $s^r \gg R^r$, the spreading factor behaves as $\frac{R^r}{s^r}$, which is characteristic of diverging spherical waves. The surface

curvature thus modulates the rate of amplitude decay, enabling GO to model focusing and defocusing effects with sufficient accuracy at high frequencies.

2.5.5 Shadow Boundaries and GO Breakdown

The reflected GO contribution is bounded by two shadow boundaries (see Figure 2.12): the incident shadow boundary (ISB), where incident illumination vanishes, and the reflected shadow boundary (RSB), where the specular reflected contribution vanishes.

In classical GO, the field is forced to drop abruptly to zero beyond these boundaries, which creates unphysical discontinuities and fully dark shadow zones. This limitation motivates the next section: the uniform theory of diffraction (UTD) introduces diffracted fields and transition behavior that restore a physically consistent field in and around shadow regions.

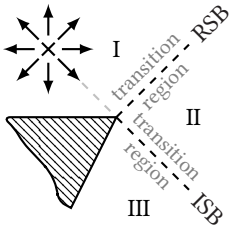


Figure 2.12: Regions and their boundaries in GO.

2.6 The Uniform Theory of Diffraction

As discussed in the previous section, GO fails in shadow regions (region III in Figure 2.12) because direct and specular reflected rays are absent there. Diffraction restores a physically meaningful field in those regions and is therefore essential for deterministic radio channel modeling.

2.6.1 Diffraction Mechanisms and the Keller Cone

Diffraction from edges is inherently three-dimensional: an incident ray striking an edge generates a cone of diffracted rays (see Figure 2.13), referred to as the *Keller cone*. The

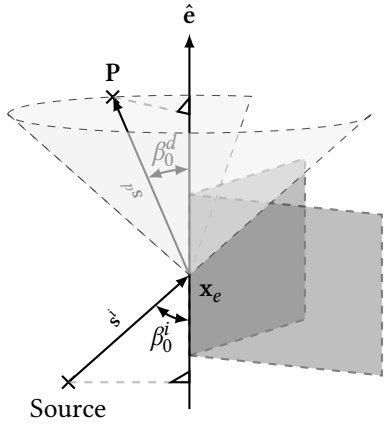


Figure 2.13: The cone of diffraction generated by a ray striking an edge.

diffracted ray direction $\hat{\mathbf{s}}^d$ is related to the incident ray direction $\hat{\mathbf{s}}^i$ and the edge direction $\hat{\mathbf{e}}$ by the law of diffraction, given by

$$\sin \beta_0 = \|\hat{\mathbf{s}}^i \times \hat{\mathbf{e}}\| = \|\hat{\mathbf{s}}^d \times \hat{\mathbf{e}}\|, \tag{2.90}$$

where $0 \leq \beta_0 \leq \pi$ is the angle between the rays and the edge direction. This implies that the diffracted rays lie on a cone with half-angle β_0 centered on the edge.

Similarly to the specular reflection case, we have that the incident and diffraction angles are equal, i.e.,

$$\beta_0 \stackrel{\text{def}}{=} \beta_0^i = \beta_0^d. \tag{2.91}$$

The two-dimensional wedge (see Figure 2.14) corresponds to perpendicular incidence ($\beta_0 = \frac{\pi}{2}$), where the Keller cone degenerates into a flat disc. However, practical radio propagation engines require the full three-dimensional treatment to algorithmically handle arbitrary incidence angles, curved surfaces, and complete vector polarization transformations.

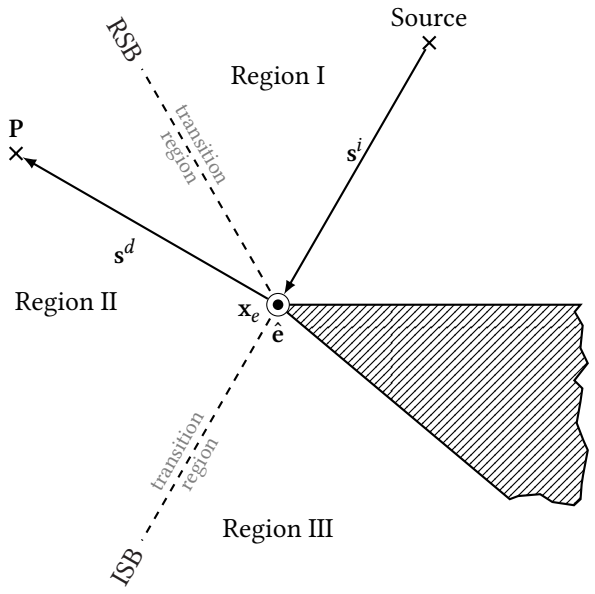


Figure 2.14: Geometry of a wedge illuminated by an incident plane wave.

2.6.2 Edge-Fixed Coordinate System

To treat three-dimensional diffraction rigorously, we construct an *edge-fixed coordinate system*, illustrated in Figure 2.16. This choice exploits the cylindrical symmetry of the diffraction cone. Consider a point $\mathbf{x}_e$ on the edge with unit tangent vector $\hat{\mathbf{e}}$. Let $\hat{\mathbf{s}}^i$ denote the incident ray direction and $\hat{\mathbf{s}}^d$ the diffracted ray direction. We define orthonormal basis vectors for the incident field as

$$\hat{\boldsymbol{\phi}}^i = \frac{-\hat{\mathbf{e}} \times \hat{\mathbf{s}}^i}{\|\hat{\mathbf{e}} \times \hat{\mathbf{s}}^i\|}, \tag{2.92}$$

$$\hat{\boldsymbol{\beta}}_0^i = \hat{\boldsymbol{\phi}}^i \times \hat{\mathbf{s}}^i, \tag{2.93}$$

and similarly for the diffracted field,

$$\hat{\boldsymbol{\phi}}^d = \frac{\hat{\mathbf{e}} \times \hat{\mathbf{s}}^d}{\|\hat{\mathbf{e}} \times \hat{\mathbf{s}}^d\|}, \tag{2.94}$$

$$\hat{\boldsymbol{\beta}}_0^d = \hat{\boldsymbol{\phi}}^d \times \hat{\mathbf{s}}^d. \tag{2.95}$$

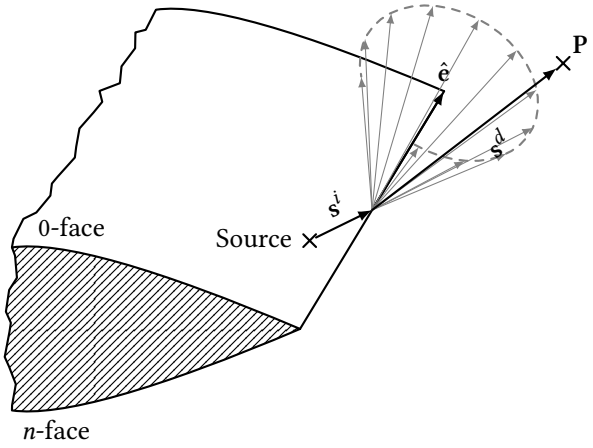


Figure 2.15: Another illustration of the three-dimensional Keller cone geometry, where an incident ray strikes an edge at point $\mathbf{x}_e$ and generates a cone of diffracted rays.

The vectors $(\hat{\boldsymbol{\beta}}_0^i, \hat{\boldsymbol{\phi}}^i, \hat{\mathbf{s}}^i)$ form a right-handed orthonormal triplet for the incident ray, with $\hat{\boldsymbol{\beta}}_0^i$ perpendicular to the edge and lying in the plane containing $\hat{\mathbf{s}}^i$ and $\hat{\mathbf{e}}$, while $\hat{\boldsymbol{\phi}}^i$ is perpendicular to this plane. The diffracted field triplet $(\hat{\boldsymbol{\beta}}_0^d, \hat{\boldsymbol{\phi}}^d, \hat{\mathbf{s}}^d)$ is constructed analogously. This edge-fixed coordinate system is illustrated in Figure 2.16.

Polarization in the Edge-Fixed System

The edge-fixed coordinate system is essential because it reduces the general diffraction dyadic $\underline{\mathbf{D}}$ to a computationally efficient diagonal form. In a generic ray-fixed frame, the coupling between polarization states is extremely complicated.

However, in the edge-fixed system, the diffraction process naturally decouples into independent parallel and perpendicular polarizations, allowing the transformation to be modeled precisely as a Jones matrix. The $\parallel$ polarization corresponds to the transverse magnetic (TM) mode satisfying the Dirichlet boundary condition ($R_\parallel = -1$ for a PEC), while the $\perp$ polarization corresponds to the transverse electric (TE) mode satisfying the Neumann boundary condition ($R_\perp = +1$). Because perfectly conducting wedge boundaries preserve the independence of these two polarization states,

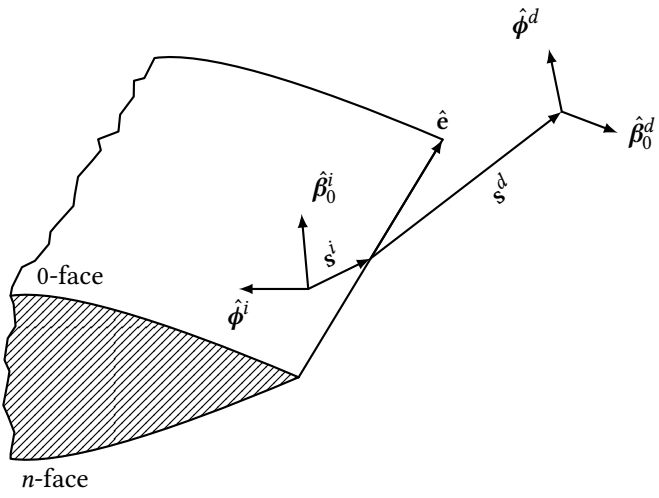


Figure 2.16: Edge-fixed coordinate system for three-dimensional wedge diffraction as illustrated in Figure 2.15, reproduced from [41, Fig. 6.2].

the component $E^i_{\phi^i}$ parallel to the edge couples only to $E^d_{\phi^d}$, and the component $E^i_{\beta_0^i}$ perpendicular to the edge couples only to $E^d_{\beta_0^d}$. For finite-conductivity materials—which are ubiquitous in urban environments—cross-polarization terms (off-diagonal elements) can emerge due to surface wave coupling; however, we focus on the idealized PEC case here for clarity and because it captures the dominant physics of diffraction in many practical scenarios.

Consequently, within the local edge-fixed coordinate frame, the diffraction interaction operates strictly as a diagonal Jones matrix transformation, mirroring the behavior of the Fresnel reflection matrix and preventing complex cross-polarization terms during pure PEC diffraction. Throughout this text, we use the simple notation $\parallel$ and $\perp$ for both edge-based (diffraction) and plane-of-incidence-based (reflection or transmission) decompositions, with the reference frame determined by context.

An arbitrarily polarized incident field is decomposed into

edge-fixed components

$$E^i_{\beta_0^i} = \mathbf{E}^i \cdot \hat{\boldsymbol{\beta}}_0^i, \tag{2.96}$$

$$E^i_{\phi^i} = \mathbf{E}^i \cdot \hat{\boldsymbol{\phi}}^i, \tag{2.97}$$

and the total incident field is reconstructed as

$$\mathbf{E}^i = E^i_{\beta_0^i}\,\hat{\boldsymbol{\beta}}_0^i + E^i_{\phi^i}\,\hat{\boldsymbol{\phi}}^i. \tag{2.98}$$

Similarly, the diffracted field components are projected onto the diffracted edge-fixed basis and reconstructed as

$$\mathbf{E}^d = E^d_{\beta_0^d}\,\hat{\boldsymbol{\beta}}_0^d + E^d_{\phi^d}\,\hat{\boldsymbol{\phi}}^d. \tag{2.99}$$

2.6.3 From GO to GTD to UTD: Motivation and Applicability

The first major extension to GO is Keller's geometrical theory of diffraction (GTD) [29]. GTD augments the GO ray family with *diffracted rays* launched from edges and other geometric singularities. Its key postulate is locality: the diffracted field is governed mainly by the geometry in a neighborhood of the diffraction point. Once the diffraction point $\mathbf{x}_e$ is identified, the diffracted field is written in a ray-optical form similar to reflected fields

[29]: Keller (1962), *Geometrical Theory of Diffraction*

$$\mathbf{E}^d(\mathbf{P}) = \mathbf{E}^i(\mathbf{x}_e)\,\underline{\mathbf{D}}\,A_d(s)\,e^{-jks}, \tag{2.100}$$

where $\underline{\mathbf{D}}$ is the dyadic diffraction coefficient, $A_d(s)$ is a spreading factor, and s is the distance from $\mathbf{x}_e$ to the observation point $\mathbf{P}$.

We label the wedge faces as the *0-face* and *n-face*, and measure both ϕ^i (incidence) and ϕ^d (observation) angles from the 0-face. The notation "0-face" utilizes the numeral zero—reflecting its position at the angular origin ($\phi = 0$)—contrasting with some literature that employs the letter

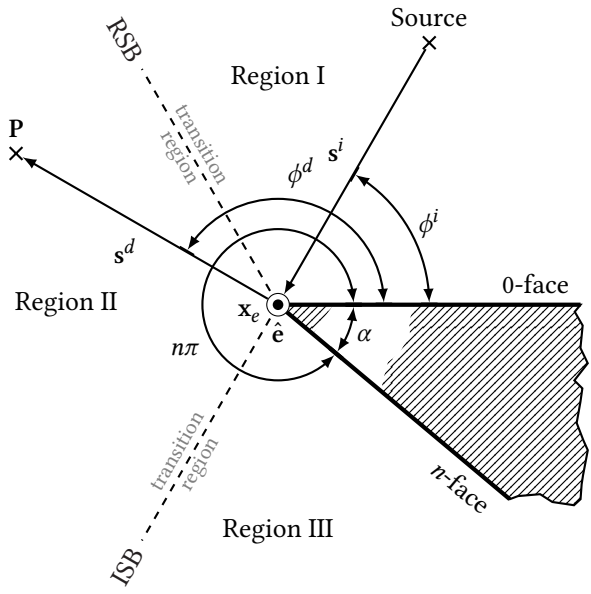


Figure 2.17: Geometry of a wedge illuminated by an incident plane wave, annotated.

"o". With wedge interior angle α, we define

$$1 \leq n \leq 2, \tag{2.101}$$

through

$$n = \frac{2\pi - \alpha}{\pi}. \tag{2.102}$$

The 0-face is therefore at $\phi^d = 0$ and the n-face at $\phi^d = n\pi$.

In the two-dimensional case as illustrated in Figure 2.17, corresponding to a perpendicular incidence on the edge (i.e., $\beta_0 = \frac{\pi}{2}$) where the cone degenerates into a plane, the Keller diffraction coefficients are given by

$$D^k_{\parallel,\perp} = \frac{-e^{-j\frac{\pi}{4}} \sin\frac{\pi}{n}}{2n\sqrt{2\pi k}} \left[\left(\frac{1}{\cos\frac{\pi}{n} - \cos\frac{\phi^d - \phi^i}{n}} \right) \mp \left(\frac{1}{\cos\frac{\pi}{n} - \cos\frac{\phi^d + \phi^i}{n}} \right) \right]. \tag{2.103}$$

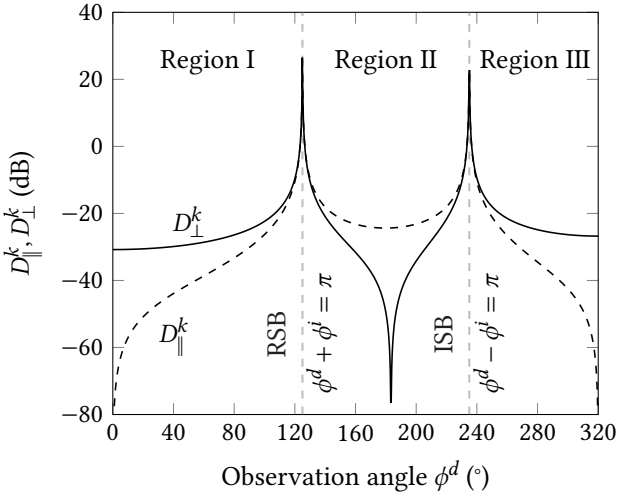


Figure 2.18: Keller's diffraction coefficients evaluated at 10 GHz for a wedge with plane wave incidence ($\phi^i = 55°$ and $\alpha = 40°$), reproduced from [41, Fig. 4.10].

GTD Singularity at Shadow Boundaries

The main limitation of classical GTD is that its diffraction coefficients contain terms that become infinite exactly at the shadow boundaries ($\phi^d \pm \phi^i = \pi$). In the canonical two-dimensional wedge example (Figure 2.17), the field is therefore non-uniform across the boundaries between fully lit, partially shadowed, and deep shadow regions. This behavior is visible both in the coefficients (Figure 2.18) and in the corresponding scattered field (Figure 2.19), which becomes unbounded at the boundaries.

A useful way to read the geometry is as follows. Depending on ϕ^i, either the n-face or the 0-face blocks the direct field, which sets the ISB. When the n-face is shadowed ($0 \le \phi^i \le (n-1)\pi$), the ISB is determined by

$$\phi^d_{\mathrm{ISB}} - \phi^i = \pi, \tag{2.104}$$

$$\pi \le \phi^d_{\mathrm{ISB}} \le n\pi. \tag{2.105}$$

When the 0-face is shadowed ($\pi \le \phi^i \le n\pi$), the ISB shifts

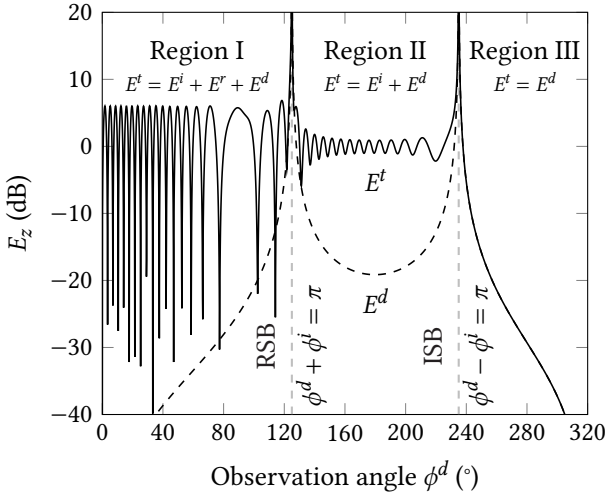


Figure 2.19: Scattered field from a wedge with plane wave incidence using Keller's diffraction coefficients. The geometry is as in Figure 2.14, with $f = 3\,\text{GHz}$, $s = 1\,\text{m}$, and parallel polarization (electric field parallel to edge). The level of the incident field is 0 dB. Note that the diffracted fields are unbounded at the shadow boundaries. Reproduced from [41, Fig. 4.11].

to

$$\phi^d_{\text{ISB}} - \phi^i = -\pi, \tag{2.106}$$

$$0 \leq \phi^d_{\text{ISB}} \leq (n-1)\pi, \tag{2.107}$$

which is compactly summarized by

$$|\phi^d_{\text{ISB}} - \phi^i| = \pi. \tag{2.108}$$

For reflected contributions, the active reflecting face determines the RSB. Reflection from the 0-face occurs for $0 \leq \phi^i \leq \pi$ and yields

$$\phi^d_{\text{RSB}} + \phi^i = \pi, \tag{2.109}$$

$$0 \leq \phi^d_{\text{RSB}} \leq \pi. \tag{2.110}$$

Reflection from the n-face occurs for $(n-1)\pi \leq \phi^i \leq n\pi$ and yields

$$\phi^d_{\text{RSB}} + \phi^i = (2n-1)\pi, \tag{2.111}$$

$$(n-1)\pi \leq \phi^d_{\text{RSB}} \leq n\pi. \tag{2.112}$$

Together, these relations describe how the two wedge faces

partition space into illuminated, reflected, and shadowed regions. In some configurations both faces generate valid reflected contributions, so two RSBs coexist. The UTD transition function smooths this partition and removes the non-physical discontinuities produced by pure GO–GTD switching.

2.6.4 The UTD Solution

In 1974, Kouyoumjian and Pathak published a landmark paper [30] introducing the *uniform theory of diffraction* (UTD). To resolve the GTD singularities, they multiplied the diffraction coefficients by a *transition function* that vanishes at the same rate as the GTD terms diverge near shadow boundaries. The resulting product remains finite, so the diffracted field is bounded and continuous across lit, transition, and shadow region.

[30]: Kouyoumjian et al. (1974), *A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface*

Since its introduction, UTD has become a cornerstone of high-frequency electromagnetic modeling. However, it is important to stress that it still relies on asymptotic assumptions: the incident field should be ray-optical and locally planar at the diffraction point. When these assumptions break down (for example near caustics or for rapidly varying incidence), slope diffraction or other specialized extensions are required.

2.6.5 Three-Dimensional UTD Diffraction Coefficients

The general three-dimensional UTD diffracted field is expressed in compact dyadic form as

$$\mathbf{E}^d(\mathbf{P}) = \mathbf{E}^i(\mathbf{x}_e) \cdot \underline{\mathbf{D}} \sqrt{\frac{\rho^d}{s^d(s^d + \rho^d)}}\, e^{-jks^d}, \tag{2.113}$$

where $\mathbf{E}^i(\mathbf{x}_e)$ is the incident field evaluated at the edge point $\mathbf{x}_e$, $\underline{\mathbf{D}}$ contains the UTD diffraction coefficients, s^d is the

distance from $\mathbf{x}_e$ to the observation point $\mathbf{P}$, and ρ^d is the caustic distance of the diffracted ray tube. The spreading factor, defined by

$$\sqrt{\frac{\rho^d}{s^d(s^d+\rho^d)}}, \tag{2.114}$$

describes the divergence of the diffracted ray tube as it propagates from the edge caustic at distance ρ^d behind the edge.

In the edge-fixed coordinate system introduced previously, the dyadic $\underline{\mathbf{D}}$ reduces to a diagonal matrix, representing a Jones matrix transformation occurring within the edge-fixed local coordinate frame. The diffracted field components can be written as

$$\begin{bmatrix} E^d_{\beta_0^d} \\ E^d_{\phi^d} \end{bmatrix} = \begin{bmatrix} -D_\parallel & 0 \\ 0 & -D_\perp \end{bmatrix} \begin{bmatrix} E^i_{\beta_0^i}(\mathbf{x}_e) \\ E^i_{\phi^i}(\mathbf{x}_e) \end{bmatrix} \sqrt{\frac{\rho^d}{s^d(s^d+\rho^d)}}\, e^{-jks^d}. \tag{2.115}$$

The parallel and perpendicular diffraction coefficients share a generic structure built upon fundamental physical effects: spatial attenuation, phase matching, and singularity cancellation. They are expressed as

$$D_{\parallel,\perp}(L^i, L^{r0}, L^{rn}, \phi^i, \phi^d, \beta_0, n) = D_1 + D_2 \mp (D_3 + D_4), \tag{2.116}$$

where the minus sign applies to parallel polarization ($R_\parallel = -1$) and the plus sign to perpendicular polarization ($R_\perp = +1$) for the PEC reflection coefficients of the wedge surfaces.

Each component D_1 through D_4 follows a similar structure, given by

$$D_1 = \frac{-e^{-j\frac{\pi}{4}}}{2n\sqrt{2\pi k}\sin\beta_0}\cot\left[\frac{\pi+(\phi^d-\phi^i)}{2n}\right]F[kL^i a^+(\phi^d-\phi^i)], \tag{2.117}$$

$$D_2 = \frac{-e^{-j\frac{\pi}{4}}}{2n\sqrt{2\pi k}\sin\beta_0}\cot\left[\frac{\pi-(\phi^d-\phi^i)}{2n}\right]F[kL^i a^-(\phi^d-\phi^i)], \tag{2.118}$$

$$D_3 = \frac{-e^{-j\frac{\pi}{4}}}{2n\sqrt{2\pi k}\sin\beta_0}\cot\left[\frac{\pi+(\phi^i+\phi^d)}{2n}\right]F[kL^{r0} a^+(\phi^i+\phi^d)], \tag{2.119}$$

$$D_4 = \frac{-e^{-j\frac{\pi}{4}}}{2n\sqrt{2\pi k}\sin\beta_0}\cot\left[\frac{\pi-(\phi^i+\phi^d)}{2n}\right]F[kL^{rn} a^-(\phi^i+\phi^d)]. \tag{2.120}$$

The presence of $\sin\beta_0$ explicitly demonstrates the generalization from 2D perpendicular incidence ($\beta_0 = \frac{\pi}{2}$) to 3D.

To compute these coefficients, we first define the angular helper functions $a^{\pm}(\beta^{\pm})$, which govern phase matching near the shadow boundaries. They are written as

$$a^{\pm}(\beta^{\pm}) = 2\cos^2\left(\frac{2\pi nN^{\pm}-\beta^{\pm}}{2}\right), \tag{2.121}$$

with $\beta^{\pm} = \phi^i \pm \phi^d$, and integers $N^{\pm}$ chosen to most nearly satisfy the corresponding phase singularity equations

$$2\pi nN^{+} - \beta^{\pm} = \pi, \tag{2.122}$$

$$2\pi nN^{-} - \beta^{\pm} = -\pi. \tag{2.123}$$

In a practical implementation, one selects the integers nearest to the corresponding boundary via

$$N^{\pm} = \text{round}\left[\frac{\beta^{\pm}\pm\pi}{2\pi n}\right], \tag{2.124}$$

with tie-breaking chosen consistently across rays to avoid branch-jump artifacts.

Next, the singularity cancellation is handled by the transi-

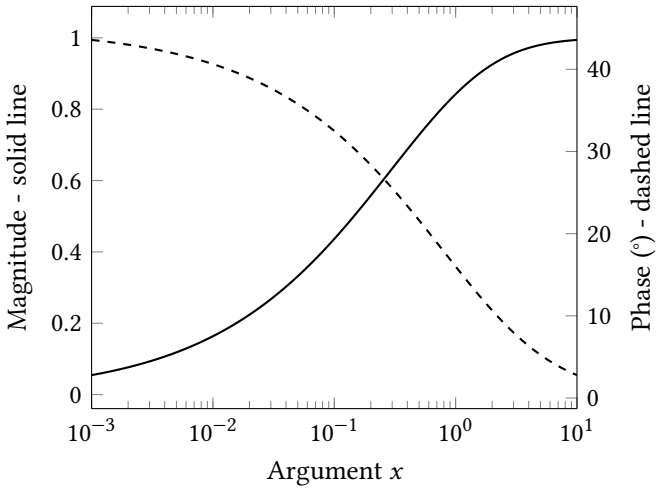


Figure 2.20: Transition function $F(x)$ (2.125), reproduced from [41, Fig. 4.16].

tion function $F(x)$, which is defined as

$$F(x) = 2j\sqrt{x}e^{jx}\int_{\sqrt{x}}^{\infty} e^{-ju^2}\,\mathrm{d}u. \tag{2.125}$$

This regularization term enforces continuity in UTD and acts similarly to the knife-edge smoothing (1.53) introduced in Chapter 1. By suppressing the non-physical GTD cotangent singularity, it produces a bounded, continuous total field at the ISB and RSB. The function is plotted in Figure 2.20.

As for the knife-edge transition function, the UTD transition can be rewritten as

$$F(x) = 2j\sqrt{x}e^{jx}\left[\sqrt{\frac{\pi}{2}}\frac{1-j}{2} - C(\sqrt{x}) - j\big(S(\sqrt{x})\big)\right], \tag{2.126}$$

where $C(x)$ and $S(x)$ are the Fresnel cosine and sine integrals, respectively.

Beyond physical continuity, the numerical stability of the transition function is crucial for optimization problems. Its continuous, bounded nature prevents the infinite gradient spikes (Dirac delta functions in the derivative) that

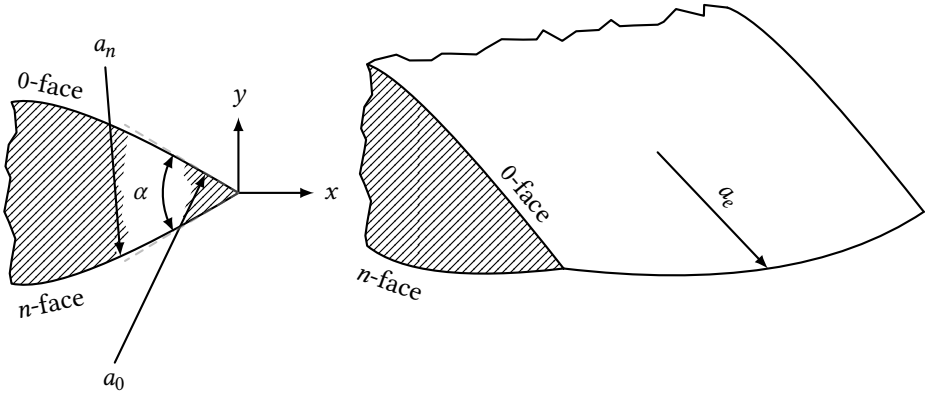


Figure 2.21: Two- and three-dimensional view of a curved wedge geometry. The 0-face and n-face have radii of curvature a_0 and a_n at the edge, respectively, and the edge has radius of curvature a_e.

would occur if classical GTD coefficients were subjected to differentiation at shadow boundaries. These details will be discussed in more detail in Chapter 4.

Finally, the scale of spatial attenuation is governed by the distance parameters L^i, L^{r0}, and L^{rn}, which rely on the wavefront and edge curvature.

The distance parameter L^i associated with the incident shadow boundaries is derived from the principal radii of curvature ρ_1^i and ρ_2^i of the incident astigmatic wavefront, such that

$$L^i = \frac{s^d(\rho_e^i + s^d)\rho_1^i\rho_2^i}{\rho_e^i(\rho_1^i + s^d)(\rho_2^i + s^d)} \sin^2\beta_0, \tag{2.127}$$

where ρ_e^i is the radius of curvature of the incident wavefront measured in the plane containing the incident ray and the edge at $\mathbf{x}_e$. The factor $\sin^2\beta_0$ reflects the projection of the wavefront curvature onto the plane perpendicular to the edge; it ensures that grazing incidence ($\beta_0 \to 0$) produces a vanishing distance parameter.

For a spherical wavefront, we have $\rho_1^i = \rho_2^i = \rho_e^i = s^i$,

yielding

$$L^i = \frac{s^d s^i}{s^d + s^i} \sin^2 \beta_0. \tag{2.128}$$

For plane-wave incidence, where the wavefront is flat ($\rho_1^i, \rho_2^i, \rho_e^i \to \infty$), L^i simplifies to $s^d \sin^2 \beta_0$.

Similarly, the distance parameters for the reflection shadow boundaries are determined by mapping the reflection radii and are given by

$$L^{r0,n} = \frac{s^d(\rho_e^{r0,n} + s^d)\rho_1^{r0,n}\rho_2^{r0,n}}{\rho_e^{r0,n}(\rho_1^{r0,n} + s^d)(\rho_2^{r0,n} + s^d)} \sin^2 \beta_0, \tag{2.129}$$

where the reflected radii $\rho_1^{r0,n}$ and $\rho_2^{r0,n}$ (from the 0- and n-faces) are obtained from the reflection plane radii $\rho_e^{r0,n}$, which satisfy

$$\frac{1}{\rho_e^{r0,n}} = \frac{1}{\rho_e^i} - \frac{2(\hat{\mathbf{n}}_e \cdot \hat{\mathbf{n}}_{0,n})(\hat{\mathbf{s}}^i \cdot \hat{\mathbf{n}}_{0,n})}{|a_e| \sin^2 \beta_0}, \tag{2.130}$$

with $\hat{\mathbf{n}}_{0,n}$ normal to the respective faces, a_e describing the edge radius of curvature, and $\hat{\mathbf{n}}_e$ normal to the edge (see Figure 2.21).

The formulas above apply to general three-dimensional wedges with curved surfaces. Because diffracted rays propagate in straight lines, the edge-diffracted field may not illuminate certain shadow regions between the reference planes tangential to the curved wedge faces and the actual curved surfaces. This scenario introduces the need for creeping-wave mechanisms in certain grazing-incidence configurations, which extend the validity of UTD to curved-surface diffraction.

For the ray tracer developed in this book, UTD provides a robust single-edge baseline. For cascaded multiple diffractions, the usual ray-field assumption must still be checked: each subsequent edge should lie outside the transition region of the previous one. Otherwise, higher-order diffraction models or calibrated heuristics are required.

2.6.6 Transition Regions and Asymptotic Validity

The transition function $F(x)$ in the UTD coefficients enforces continuity across shadow boundaries. Because $F(x)$ approaches zero at the same rate as the cotangent singularities in $D_{1,2,3,4}$ diverge, the product remains finite at the incident and reflection shadow boundaries (ISB and RSB). This cancellation is the essence of the *uniform* theory.

Transition Function Asymptotic Behavior

As the argument of the transition function increases, the Fresnel integral asymptotically approaches unity as

$$\lim_{x\to\infty} F(x) = 1. \tag{2.131}$$

Because $F(x)$ is complex-valued, this limit means both magnitude and phase approach one and zero, respectively.

In practice, we often adopt the criterion [41, Eq. 4.129]

$$F(x) \approx 1 \quad \text{if} \quad x \geq 2\pi. \tag{2.132}$$

[41]: McNamara et al. (1990), *Introduction to the Uniform Geometrical Theory of Diffraction*

Largeness Parameter and Boundary Condition

The argument of the transition function defines a dimensionless *largeness parameter*,

$$\kappa = kL\,a^{\pm}, \tag{2.133}$$

where L is the appropriate distance parameter (either L^i or $L^{r0,n}$), and $a^{\pm}$ are the angular helper functions. The UTD diffraction coefficients are derived via asymptotic evaluation of exact integral representations; their validity requires that κ be sufficiently large.

The *transition region* is the spatial domain where the transition function deviates significantly from unity, i.e., defined by $\kappa < 2\pi$. Outside the transition region ($\kappa \geq 2\pi$), the

transition function saturates at $F \approx 1$, and the diffracted field becomes purely ray-optical and reduces to Keller's GTD expressions. By defining an approximate value for the boundary of this transition region as $\kappa = 2\pi$, we obtain the condition

$$kL\,a^{\pm} = 2\pi \quad \Rightarrow \quad L\,a^{\pm} = \lambda. \tag{2.134}$$

When rewritten in this form, this boundary condition highlights three primary reasons why the UTD asymptotic assumptions may fail: (i) the frequency is too low (resulting in a large wavelength λ), (ii) the observation distance is too short (resulting in a small distance parameter L), or (iii) the angular separation from the shadow boundary is too small (resulting in a small $a^{\pm}$).

Limitations on Cascaded Interactions

As emphasized by Lee *et al.* [43], and documented in [41, Sec. 4.4.4], the fundamental limitation of applying GTD or its uniform extensions (UTD) is that the incident field must strictly constitute a "ray field"—a field whose asymptotic expansion assumes a specific Lüneburg-Kline form. Diffracted fields inside transition regions are not ray-optical fields, as they intrinsically behave as functions of the parameters upon which the transition functions themselves depend. Consequently, they exhibit spatial variations that violate standard ray-optical assumptions.

[43]: Lee et al. (1978), *GTD, ray field, and comments on two papers*

[41]: McNamara et al. (1990), *Introduction to the Uniform Geometrical Theory of Diffraction*

The existence of these transition regions therefore questions the theoretical validity of cascading multiple diffraction coefficients. If a second edge lies inside the transition region of the first, mechanically cascading single-edge UTD using the primary diffracted field as an incident ray yields incorrect results. While these transition regions are very small (especially at high frequencies), to address these limitations, some ray tracing tools prefer to implement special double or even triple-order diffraction coefficients that account for the possibility of being within a transition region. Furthermore, the fact that diffracted fields are not

necessarily ray-optical also means that applying the GO reflected fields for a subsequent reflection is not always valid. As a result, many ray tracing tools limit themselves to only simulating a single diffraction interaction at maximum, typically at the last interaction before reaching the receiver.

2.7 Scattering from Rough Surfaces

While conventional ray tracing efficiently handles specular reflections, transmissions, and diffractions as discussed in the preceding sections, it struggles to accurately represent real-world environments where surfaces are rarely perfectly flat. Surface roughness and architectural irregularities on building walls induce diffuse scattering. This phenomenon scatters radio signals in multiple directions other than the purely specular one, heavily influencing both the angular and temporal dispersion of the multipath channel.

The physical threshold where a surface transitions from "smooth" to "rough" was discussed in Chapter 1 and led to the Rayleigh criterion (1.50). When surface irregularities exceed this threshold, macroscopic modeling becomes necessary. To overcome the limitations of purely specular ray tracing, simulators commonly implement the effective roughness (ER) approach [45], see Figure 2.22. In this macroscopic abstraction, a scattering parameter $S \in [0, 1]$ dictates the fraction of reflected energy that scatters diffusely.

[45]: Degli-Esposti (2001), *A diffuse scattering model for urban propagation prediction*

At the level of a local illuminated surface element, this downgrading is obtained from a reflected-field power balance in which coherent reflection and diffuse scattering are jointly scaled by the same roughness factor [46]. Denoting by P_i the incident power on the element and by P_t the transmitted power, the balance can be written as

[46]: Degli-Esposti et al. (2007), *Measurement and Modelling of Scattering From Buildings*

$$1 = \Gamma^2 \left(R^2 + S^2\right) + \frac{P_t}{P_i}, \tag{2.135}$$

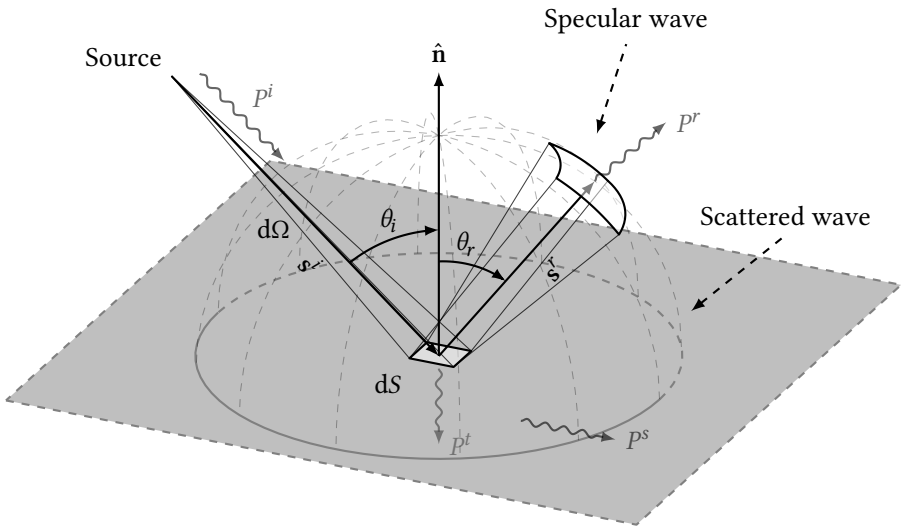


Figure 2.22: Illustration of specular reflection, diffuse scattering, and transmission on a rough surface, adapted from [44, Fig. 2].

where R represents the reflection reduction factor, S is the diffuse scattering parameter, and Γ represents the smooth-surface Fresnel reflection coefficient. Under the standard assumption that penetration losses are dominated by bulk material properties and remain approximately constant with respect to the roughness perturbation, the dependence of $\frac{P_t}{P_i}$ on S can be neglected to first order. The relation then reduces to the normalized constraint $R^2 + S^2 \approx 1$, yielding

$$R \approx \sqrt{1 - S^2}. \tag{2.136}$$

This result provides the theoretical justification for scaling the smooth-surface Fresnel dyadic by $\sqrt{1 - S^2}$, which preserves power consistency and avoids the non-physical growth of coherent reflected contributions in near-grazing configurations.

Consequently, the coherent specular dyadic reflection ma-

trix is downgraded to

$$\underline{\mathbf{R}}_{\text{rough}} = \sqrt{1 - S^2}\,\underline{\mathbf{R}}_{\text{smooth}}, \tag{2.137}$$

where $\underline{\mathbf{R}}_{\text{smooth}}$ is the classical Fresnel Jones matrix. Measurement campaigns have validated this model structure and its associated cross-polarization behavior, indicating that S values between 0.2 and 0.4 are realistic for standard building materials [46, 47].

[46]: Degli-Esposti et al. (2007), *Measurement and Modelling of Scattering From Buildings*

[47]: Degli-Esposti et al. (2011), *Analysis and Modeling on co- and Cross-Polarized Urban Radio Propagation for Dual-Polarized MIMO Wireless Systems*

The spatial distribution of the diffusely scattered field is defined by a scattering pattern, which determines the magnitude of scattered rays launched in various directions. Common angular models include [46, Sec. II]:

- The *Lambertian pattern*, which assumes the maximum scattered power is directed strictly parallel to the surface normal $\hat{\mathbf{n}}$. Its power scales proportionally to
$$\cos\theta_s, \tag{2.138}$$
where θ_s is the scattered polar angle relative to the normal.
- The *directive pattern* (single-lobe model), which is often more physically accurate for moderately rough surfaces. It centers the primary scattering lobe around the specular reflection direction $\hat{\mathbf{s}}^r$, with the scattered power scaling as
$$\left(\frac{1 + \cos\psi_R}{2}\right)^{\alpha_R}, \tag{2.139}$$
where ψ_R is the angle between the scattering direction and the specular direction, and α_R is the tunable directivity exponent.
- The *backscattering pattern*, which adds a secondary lobe proportional to $\left(\frac{1 + \cos\psi_i}{2}\right)^{\alpha_i}$ pointing back toward the incident source. This is crucial for very irregular obstacles, yielding a scattered power pro-

portional to

$$\left[\Lambda\left(\frac{1+\cos\psi_R}{2}\right)^{\alpha_R}+(1-\Lambda)\left(\frac{1+\cos\psi_i}{2}\right)^{\alpha_i}\right], \tag{2.140}$$

where ψ_i is the angle between the scattering direction and the incidence direction, $\Lambda \in [0,1]$ is the tunable mixing factor, and α_i is the tunable backscattering exponent.

These mathematical pattern formulations allow advanced ray tracing engines to employ importance sampling by launching more scattered rays into regions of higher expected power. Moreover, all the cosine terms in the above equations can be obtained through dot products between the corresponding ray directions, making the implementation of these scattering models straightforward in ray tracing engines.

Beyond spatial dispersion, diffuse scattering represents a primary source of signal depolarization [47]. While pure specular reflections on vertical planes geometrically preserve TM–TE polarization states (diagonal Jones matrices), diffusely scattered rays depart over random, highly varying elevation angles. This 3D spatial variation induces severe cross-polarization coupling.

[47]: Degli-Esposti et al. (2011), *Analysis and Modeling on co- and Cross-Polarized Urban Radio Propagation for Dual-Polarized MIMO Wireless Systems*

To integrate this rigorously within our Jones calculus framework, we define a diffuse scattering dyadic matrix $\underline{\mathbf{S}}$. Using the depolarization factor $K_{\text{xpol}} \in [0,1]$, which specifies the cross-polarized power coupling fraction, the Jones matrix is given by

$$\underline{\mathbf{S}} = S\begin{bmatrix}\sqrt{1-K_{\text{xpol}}}\,e^{j\chi_1} & -\sqrt{K_{\text{xpol}}}\,e^{j\chi_1}\\ \sqrt{K_{\text{xpol}}}\,e^{j\chi_2} & \sqrt{1-K_{\text{xpol}}}\,e^{j\chi_2}\end{bmatrix}, \tag{2.141}$$

where $e^{j\chi}$ are random, uniformly distributed phase shifts that highlight the incoherent nature of diffuse scattering. The off-diagonal terms computationally guarantee that a purely co-polarized incident field will scatter with a cross-polarized component. Research has demonstrated that an

intrinsic depolarization rating as low as $K_{\text{xpol}} = 0.01$ still drastically degrades the overall cross-polarization discrimination (XPD) of the urban channel [47]. The XPD is defined as the ratio of the co-polarized to the cross-polarized received power, and for the scattered field it is given by

$$\text{XPD} = 10 \log_{10} \left(\frac{1 - K_{\text{xpol}}}{K_{\text{xpol}}} \right). \tag{2.142}$$

[47]: Degli-Esposti et al. (2011), *Analysis and Modeling on co- and Cross-Polarized Urban Radio Propagation for Dual-Polarized MIMO Wireless Systems*

In a ray-launching simulator, energy consistency requires that the statistical expectation of the scattered-field norm, integrated over the scattering hemisphere and weighted by the launch probability density, matches the prescribed diffuse-power fraction. In compact form,

$$\int_{\Omega^+} \left\| \underline{\mathbf{S}}(\Omega)\, \hat{\mathbf{s}}^i \right\|^2 p(\Omega)\, \mathrm{d}\Omega = S^2, \tag{2.143}$$

where $p(\Omega)$ is the scattering-direction probability density over the upper hemisphere Ω^+. This condition serves as the operational criterion used to normalize the sampled scattered rays, preventing artificial energy creation or depletion during Monte Carlo sampling.

As analyzed in the referenced study, this important cross-polarization coupling is not primarily driven by the scattering interaction itself (which largely preserves the original polarization state), but rather by the wide spread of elevation angles exhibited by the diffusely scattered rays. When these non-horizontal rays propagate to the receiver, their 3D trajectories cause a geometrical rotation of the polarization vectors, leading to strong polarization coupling at the receiving antenna.

While diffuse scattering from rough surfaces represents an uncontrollable feature of natural and urban environments, recent technological advances now allow deliberate control of scattering properties through engineered metasurfaces, which are examined in the next section.

2.8 Reconfigurable Intelligent Surfaces

As introduced in Chapter 1, the advent of engineered metasurfaces represents a paradigm shift for modern wireless networks (6G and beyond). Reconfigurable intelligent surfaces (RISs)—also called intelligent reflecting surfaces—consist of arrays of sub-wavelength elements whose electromagnetic properties can be dynamically controlled to impose desired amplitude and phase modulations on incident fields, enabling the creation of programmable "smart radio environments." Unlike the passive scatterers modeled in the preceding sections, RISs enable active and programmable wavefront shaping, allowing anomalous reflection, beam steering, and focusing capabilities that extend far beyond classical specular reflection.

However, as previously discussed, simulating the full-wave electromagnetic interactions of thousands of microscopic unit cells is computationally prohibitive. Integrating RIS technology into deterministic ray tracing frameworks thus relies on simplified macroscopic models (anticipated in Chapter 1) that carefully balance physical rigor with computational efficiency. To this end, the following subsections present the macroscopic modeling approach, the power balance constraints ensuring energy conservation, the ray-based formulation using phase-gradient reflection laws, and the antenna-array representation that enables efficient field evaluation.

2.8.1 Macroscopic Modeling Approach

The integration of RIS into ray tracing demands a macroscopic modeling approach [44, 48]. Microscopic full-wave simulations of individual unit cells—while accurate—are computationally unfeasible for large-scale, system-level propagation scenarios. Instead, modern ray tracing frameworks treat the RIS as a homogenized surface that applies specific wave transformations through a spatial mod-

[44]: Degli-Esposti et al. (2022), *Reradiation and Scattering From a Reconfigurable Intelligent Surface: A General Macroscopic Model*

[48]: Vitucci et al. (2024), *An Efficient Ray-Based Modeling Approach for Scattering From Reconfigurable Intelligent Surfaces*

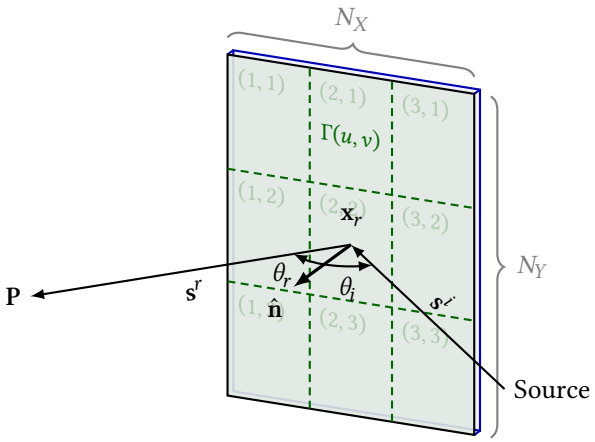


Figure 2.23: Example of a RIS characterized by a complex spatial modulation coefficient $\Gamma(u, v)$ at each surface element indexed by (u, v).

ulation function. This macroscopic abstraction captures the essential electromagnetic behavior (anomalous reflection, beam steering, focusing) while remaining compatible with the ray-optical framework established in previous sections.

The RIS is characterized by a complex spatial modulation coefficient $\Gamma(u, v)$ at each surface element indexed by (u, v), see Figure 2.23. This coefficient encodes both the amplitude and phase modulation imposed by the local configuration of the metasurface elements. This macroscopic representation enables seamless integration with existing ray tracing mechanisms (specular reflection, diffraction, scattering) while preserving the computational efficiency essential for large-scale wireless channel modeling.

2.8.2 Power Balance and Physical Consistency

To ensure physical consistency, the scattering from a RIS must adhere to a strict power balance constraint, as detailed in [44, Sec. II]. The incident power is partitioned into several components: coherent specular reflection from the passive substrate, diffuse scattering due to structural non-

[44]: Degli-Esposti et al. (2022), *Reradiation and Scattering From a Reconfigurable Intelligent Surface: A General Macroscopic Model*

idealities, anomalous reradiated modes dictated by the smart surface, and internally dissipated power.

To fully capture polarization-altering metasurface architectures within our established tracking structure, we expand the macroscopic spatial modulation into a Jones tensor $\underline{\mathbf{\Gamma}}(u, v)$ at each generic surface element (u, v), such that

$$\underline{\mathbf{\Gamma}}(u, v) = R \cdot \sqrt{m} \cdot K_m(u, v)\, e^{j\Phi_m(u,v)} \underline{\mathbf{P}}, \tag{2.144}$$

where R is the reduction factor accounting for macroscopic RIS non-idealities (often related to structural effective roughness), m is the power amplitude coefficient of the considered reradiation mode, $K_m(u, v)$ represents the amplitude modulation pattern, $\Phi_m(u, v)$ is the phase modulation profile imposed by the RIS control, and $\underline{\mathbf{P}}$ acts as the local polarization transfer matrix [44, Eq. 11].

[44]: Degli-Esposti et al. (2022), *Reradiation and Scattering From a Reconfigurable Intelligent Surface: A General Macroscopic Model*

The global power balance constraint guarantees energy conservation and strictly prevents unphysical field amplification by ensuring the fractional power summation equals unity [44, Eq. 10], i.e.,

$$1 = R^2\rho + S^2 + R^2 \sum_n m_n + \tau, \tag{2.145}$$

where ρ is the specular reflectance of the surface, m_n is the power amplitude coefficient of the n-th reradiation mode, S^2 is the fraction of power diverted to diffuse scattering, and τ is the dissipation factor.

2.8.3 Ray-Based Anomalous Reradiation (GO-Style Framework)

In a *forward ray launching*‡ approach, anomalous reflection is governed by a *generalized law of reflection* (1.62) that incorporates the phase gradient $\nabla\Phi_m$ introduced by the RIS [48, Eq. 13]. Unlike global superposition models, this

[48]: Vitucci et al. (2024), *An Efficient Ray-Based Modeling Approach for Scattering From Reconfigurable Intelligent Surfaces*

‡ These terms will be further elaborated upon in Chapter 3.

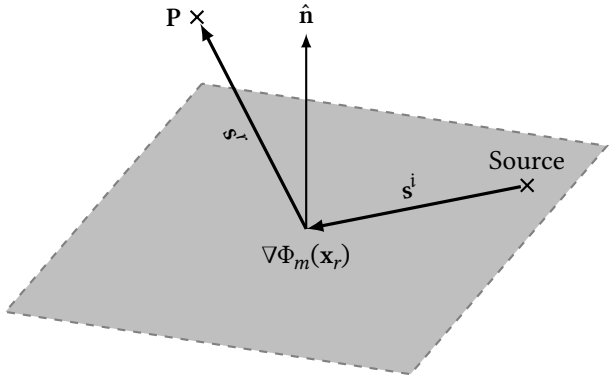


Figure 2.24: Geometry of anomalous reflection over a RIS, including local phase gradient and reflected-ray steering.

method treats the interaction as a purely localized trajectory redirection, aligning perfectly with the GO framework for efficient computation.

Building upon the 1D phase gradient $\frac{d\Phi}{dx}$ intuition developed in Section 1.5.4, the full 3D direction of the anomalously reflected ray $\hat{\mathbf{s}}^r$ is dictated by the vector gradient. Denoting the tangential component of the incident ray direction as

$$\hat{\mathbf{s}}_t^i = \hat{\mathbf{s}}^i - (\hat{\mathbf{s}}^i \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}, \tag{2.146}$$

the reflected direction is given by [48, Eq. 13]

$$\hat{\mathbf{s}}^r = \left(\hat{\mathbf{s}}_t^i - \frac{\nabla\Phi_m(\mathbf{x}_r)}{k_0}\right) + \sqrt{1 - \left|\hat{\mathbf{s}}_t^i - \frac{\nabla\Phi_m(\mathbf{x}_r)}{k_0}\right|^2}\,\hat{\mathbf{n}}, \tag{2.147}$$

[48]: Vitucci et al. (2024), *An Efficient Ray-Based Modeling Approach for Scattering From Reconfigurable Intelligent Surfaces*

where $\nabla\Phi_m(\mathbf{x}_r)$ is the 2D surface phase gradient at the interaction point $\mathbf{x}_r$, k_0 is the free-space wavenumber, and $\hat{\mathbf{n}}$ is the outward surface normal.

This expression correctly reduces to the classical specular reflection law

$$\hat{\mathbf{s}}^r = \hat{\mathbf{s}}_t^i + \sqrt{1 - |\hat{\mathbf{s}}_t^i|^2}\,\hat{\mathbf{n}}, \tag{2.148}$$

when $\nabla\Phi_m = 0$ (passive surface), while accommodating arbitrary steering when $\nabla\Phi_m \neq 0$, as exemplified in Figure 2.24.

To rigorously model the spatial divergence of the ray tube

as it propagates away from the RIS, the local wavefront curvature matrix of the reflected wave $\mathbf{Q}^r$ is derived via localized phase matching [48, Eq. 21]

$$\mathbf{Q}^r = \mathbf{L}^\top \left[\mathbf{Q}^i - \frac{1}{k_0} \nabla\nabla\Phi_m \right] \mathbf{L}, \tag{2.149}$$

[48]: Vitucci et al. (2024), *An Efficient Ray-Based Modeling Approach for Scattering From Reconfigurable Intelligent Surfaces*

where $\mathbf{Q}^i$ is the incident wave's curvature matrix and $\mathbf{L}$ is a coordinate projection operator. The term $\nabla\nabla\Phi_m$ represents the Hessian matrix of the phase profile. Physically, this Hessian acts as a lens operator: its eigenvalues dictate the focusing or defocusing power of the RIS, directly contracting or expanding the astigmatic ray tube radii defined in Section 2.2.4.

Additionally, the finite size of the RIS introduces edge diffraction effects [48, Sec. II-C]. In this case, the edge-launched field is no longer described by the classical Keller cone alone, but by an *anomalous Keller cone* whose aperture is shifted by the tangential phase gradient along the edge direction. Let $\hat{\mathbf{e}}$ denote the unit vector along the edge, β^i the classical incidence angle with respect to the edge, and β^d the aperture angle of the anomalous cone. The generalized diffraction law [48, Eq. 24] is

$$\cos\beta^d = \hat{\mathbf{s}}^d \cdot \hat{\mathbf{e}} = \cos\beta^i - \frac{1}{k_0} \frac{\partial \Phi_m}{\partial \hat{\mathbf{e}}}, \tag{2.150}$$

which explicitly shows that the imposed phase gradient modifies the cone aperture. The corresponding edge-caustic distance controlling spatial attenuation is also modified as [48, Eq. 26]

$$\frac{1}{\rho^d} = \frac{1}{\rho_e^i} \frac{\sin^2\beta^i}{\sin^2\beta^d} - \frac{1}{k_0 \sin^2\beta^d} \frac{\partial^2 \Phi_m}{\partial \hat{\mathbf{e}}^2}, \tag{2.151}$$

where ρ_e^i is the incident wavefront curvature radius in the edge-fixed diffraction plane. The second derivative of the modulation phase therefore perturbs the effective curvature seen by the diffracted field, directly entering the attenuation law through ρ^d.

2.8.4 Discrete Antenna-Array Formulation (PO-Style Superposition)

Alternatively, the reradiated field can be evaluated by treating each RIS tile as a coherently emitting equivalent antenna element [44, Sec. III-B]. This constitutes a physical optics (PO)-style superposition approximation. Just as PO integrates equivalent surface currents to evaluate scattered fields, this approach sums the discrete spherical wavelets radiated by each tile. It is particularly useful when analyzing near-pattern envelope fidelity or edge-tapering effects that localized GO rays might under-resolve.

[44]: Degli-Esposti et al. (2022), *Reradiation and Scattering From a Reconfigurable Intelligent Surface: A General Macroscopic Model*

Following this discrete antenna-array model, the field $\Delta\mathbf{E}^m$ reradiated by the (u,v)-th tile to an observation point $\mathbf{P}$ is concisely modeled as [44, Eq. 30]

$$\begin{aligned}\Delta\mathbf{E}^m(\mathbf{P}|_{(u,v)}) = &\frac{\lambda}{\sqrt{8\pi}} E_1^i \underline{\mathbf{\Gamma}}(u,v) \\ &\frac{\sqrt{D_m}\sqrt{f_m(\theta_i(u,v))}\sqrt{f_m(\theta_m(\mathbf{P}|_{(u,v)}))}}{\sqrt{\int_0^\pi f_m(\theta_m)\sin(\theta_m)\,\mathrm{d}\theta_m}} \\ &\frac{e^{-jk[r_i(u,v)+r_m(\mathbf{P}|_{(u,v)})]}}{r_i(u,v)\, r_m(\mathbf{P}|_{(u,v)})},\end{aligned} \tag{2.152}$$

where E_1^i is the normalized incident field amplitude, D_m is the directivity of the RIS element in the m-th reradiation mode, $f_m(\theta)$ represents its radiation pattern, and r_i and r_m are the respective distances between the transmitter, the tile, and the observation point. Notice that $\underline{\mathbf{\Gamma}}(u,v)$ now acts as the polarization transformation tensor. The normalizing integral in the denominator ensures that the sum of all radiated power matches the macroscopic power balance constraint derived earlier.

For electromagnetic consistency, this discrete superposition is admissible only if each hypothetical reradiating tile satisfies an aperture feasibility constraint. Denoting by $A_{e,m}$ the effective area associated with mode m and by ΔS the physical tile area, the physically admissible regime

requires [44, Eq. 18]

$$A_{e,m} \leq \Delta S. \tag{2.153}$$

For square tiles with side length Δl, this condition implies the practical bound [44, Eq. 20]

$$0.28\,\lambda \leq \Delta l \leq 0.5\,\lambda. \tag{2.154}$$

Violating the lower bound leads to a non-physical passive element that reradiates more power than it intercepts, while violating the upper bound introduces grating lobes in the reradiated field. When specific element patterns are prescribed, the lower bound tightens: approximately $\Delta l \gtrsim 0.4\lambda$ for an exponential-Lambertian pattern and $\Delta l \gtrsim 0.49\lambda$ for a Huygens-source pattern [44, Eqs. 23 and 25]. These constraints show that a macroscopic tile in the array model cannot be naively identified with the microscopic sub-wavelength unit cell.

This antenna-array formulation inherently evaluates field intensity via a dense summation of all visible tiles. While it bypasses the geometric complexity of calculating ray-tube spreading factors (the Hessian) characteristic of the GO approach, it incurs a steeper computational cost due to the sheer number of required paths per surface. In summary, the ray-based GO model offers computationally robust, localized, and easily differentiable trajectory redirection, whereas the PO-style array model provides detailed near-pattern field superposition at the cost of intense combinatorial integration.

[44]: Degli-Esposti et al. (2022), *Reradiation and Scattering From a Reconfigurable Intelligent Surface: A General Macroscopic Model*

2.9 Modeling Transmitted Waves

Until now, we have predominantly considered wave propagation within a single medium or interacting at boundaries where the primary interest lies in the reflected or

diffracted components. However, as introduced in the previous chapter, waves do not only reflect and diffract, but can also be transmitted through materials. In this section, we will discuss how to model transmitted fields in ray tracing. While the models for free-space propagation are well-established, models for air-to-medium, or air-to-medium-to-air propagation, are usually simplified because tracking rays through multiple internal reflections within a material is computationally expensive and complex. Therefore, many outdoor ray tracing tools neglect transmitted rays, as their contribution is often negligible compared to direct or diffracted paths. In contrast, in indoor scenarios or mixed indoor-outdoor scenarios, transmitted rays play a crucial role, allowing radio waves to propagate through walls, floors, and other obstacles. An important example is propagation through glass windows.

In the following subsections, we will discuss how to model transmitted waves through single or multiple slabs, employing the model from ITU-R Recommendation P.2040 [11, Sec. 2.2.2]. This formulation is essentially a combination of propagation in lossy media (Section 1.3.6) and Fresnel's transmission coefficients (Section 1.5.1). We adopt the $\perp$ and $\parallel$ polarization notation as established in the previous sections, corresponding to the TE and TM incident polarizations.

[11]: International Telecommunication Union (2025), *ITU-R Recommendation P.2040-3: Effects of building materials and structures on radiowave propagation above about 100 MHz*

2.9.1 General Method for a Multi-Layer Slab

When a plane wave strikes a slab comprising N individual layers—where each layer has flat, parallel, and smooth boundaries—a recursive impedance-based method can be used to model the interaction (as illustrated in Figure 2.25). Let us define the relative permittivity and thickness of the n-th layer as $\epsilon_{r,n}$ and d_n, respectively. We assume the slab is embedded in air, represented by layers 0 and $N+1$, both having a relative permittivity of 1 and a thickness of 0. The incoming wave arrives at an incidence angle θ_0 and exits

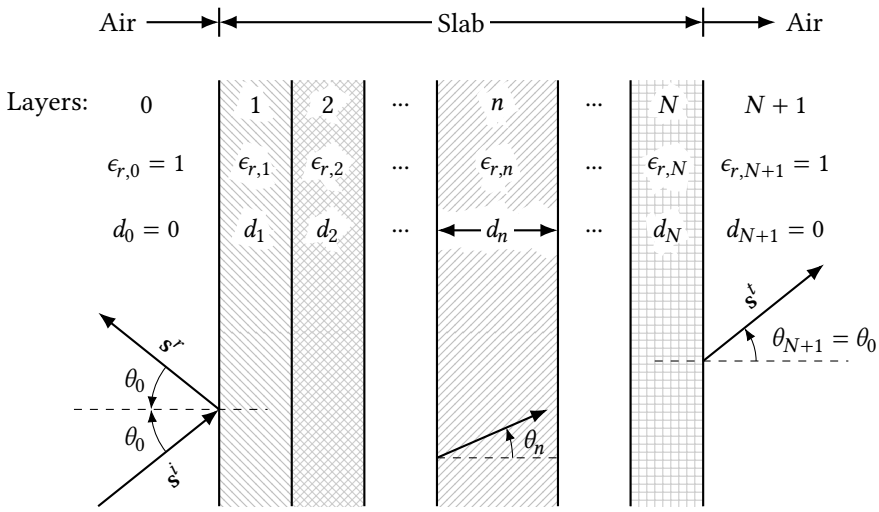


Figure 2.25: Plane wave incident on a multi-layer slab, adapted from [11, Fig. 3].

the final layer N at the same angle $\theta_{N+1} = \theta_0$, while the propagation angle inside layer n is denoted as θ_n. Instead of tracking each individual ray through a complex series of internal reflections, this technique simplifies the process by calculating equivalent, global reflection and transmission coefficients for the complete multi-layer structure.

To compute the equivalent reflection coefficient for the entire multi-layer assembly, we evaluate a recursive equation that defines the reflection at the interface between layer n and layer $n + 1$. This evaluation proceeds backwards from $n = N$ down to $n = 0$, starting with the initial condition $R_{\parallel,\perp}(N + 1) = 0$, and is given by

$$R_{\parallel,\perp}(n) = \frac{R^n_{\parallel,\perp} + R_{\parallel,\perp}(n+1)\, e^{-2j\gamma_{n+1}d_{n+1}}}{1 + R^n_{\parallel,\perp} R_{\parallel,\perp}(n+1)\, e^{-2j\gamma_{n+1}d_{n+1}}}, \tag{2.155}$$

where $R^n_{\parallel,\perp}$ denotes the Fresnel reflection coefficient (1.45) at the n-th interface. The propagation constant γ_n and the angle of refraction θ_n are related through Snell's law and

are given by

$$\gamma_n = k_0\sqrt{\epsilon_{r,c,n} - \sin^2\theta_0}, \tag{2.156}$$

$$\sin\theta_n = \frac{\sin\theta_0}{\sqrt{\epsilon_{r,c,n}}}, \tag{2.157}$$

$$k_n = \frac{2\pi}{\lambda_0}\sqrt{\epsilon_{r,c,n}}, \tag{2.158}$$

with k_0 and λ_0 denoting the free-space wavenumber and wavelength, respectively, and $\epsilon_{r,c,n}$ representing the complex relative permittivity of layer n to correctly account for material losses and conductivity.

After recursively computing (2.155) down to $n = 0$, the overall reflection and transmission coefficients for the complete slab can be written as

$$R_{\parallel,\perp} = R_{\parallel,\perp}(0), \tag{2.159}$$

$$T_{\parallel,\perp} = \prod_{n=0}^{N} \frac{e^{-j\gamma_n d_n}\left(1 + R^n_{\parallel,\perp}\right)}{1 + R^n_{\parallel,\perp} R_{\parallel,\perp}(n+1) e^{-2j\gamma_{n+1} d_{n+1}}}. \tag{2.160}$$

2.9.2 Simplified Method for a Single-Layer Slab

In the specific case of a single-layer slab ($N = 1$, depicted in Figure 2.26), the generic recursive procedure simplifies to direct, closed-form equations. Assuming the material has a thickness d and a complex relative permittivity $\epsilon_{r,c}$, let $R'_{\parallel,\perp}$ denote the Fresnel reflection coefficients evaluated at the interface between the air and the material. The equivalent reflection and transmission coefficients of the slab simplify to

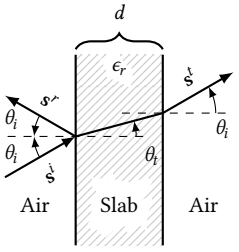


Figure 2.26: Plane wave incident on a single-layer slab.

$$R_{\parallel,\perp} = \frac{R'_{\parallel,\perp}(1 - e^{-j2q})}{1 - R'^2_{\parallel,\perp} e^{-j2q}}, \tag{2.161}$$

$$T_{\parallel,\perp} = \frac{(1 - R'^2_{\parallel,\perp})\, e^{-jq}}{1 - R'^2_{\parallel,\perp} e^{-j2q}} e^{jk_0 d\cos\theta_0}, \tag{2.162}$$

where q, a term capturing both the phase variation and attenuation scaling, is given by

$$q = \frac{2\pi d}{\lambda_0}\sqrt{\epsilon_{r,c} - \sin^2\theta_0}, \tag{2.163}$$

with θ_0 being the angle of incidence. The additional phase term $e^{jk_0 d\cos\theta_0}$ in the transmission coefficient is necessary in ray tracing engines to compensate for the phase shift already accounted for by the straight-line path length calculation, which assumes the ray travels the distance d in free space. This reduced model offers a computationally inexpensive yet highly precise approximation for common architectural elements—such as isolated glass windows, concrete walls, or brick barriers—provided they can be reasonably treated as a standalone, homogeneous medium suspended in air.

2.10 Computing the Total Channel

Having explored the fundamental mechanisms of ray-based propagation—from the high-frequency approximation and polarization calculus to specific interactions like reflection, diffraction, and RISs—we can now combine these models to fully characterize the wireless channel.

2.10.1 Cascading Interactions for a Single Path

In a realistic propagation scenario, an electromagnetic wave emitted by the transmitter undergoes a sequence of interactions before reaching the receiver. By combining the models for the antennas (Section 2.3) and the models for wave-surface interactions, we can fully describe the field propagating along any given path.

For a single path p experiencing K interactions, we can resort to cascading the individual effects. Using the Jones

calculus introduced in Section 2.4, the received electric field $\mathbf{E}_p^{\mathrm{RX}}$ arriving at the receiver is a product of interaction dyadics

$$\mathbf{E}_p^{\mathrm{RX}} = \left(\prod_{k=K}^{1} \underline{\mathbf{T}}_p^k \right) \mathbf{E}_p^{\mathrm{TX}}, \tag{2.164}$$

where $\mathbf{E}_p^{\mathrm{TX}}$ is the initial electric field radiated by the transmitting antenna in the direction of departure of the path, and $\underline{\mathbf{T}}_p^k$ is the dyadic operator representing the k-th interaction (which encapsulates reflection/diffraction coefficients, spatial spreading factors, and phase shifts along propagation segments). The product is ordered right-to-left (hence the index running from K to 1).

While this cascading approach is highly effective, care must be taken to ensure that the ray-optical assumptions remain valid. For instance, consecutive interacting objects must be sufficiently far apart relative to the wavelength for the local plane-wave approximation to hold.

To translate the electromagnetic field $\mathbf{E}_p^{\mathrm{RX}}$ into a channel coefficient, we project it onto the receiving antenna pattern. We define the complex scalar path coefficient $a_p \in \mathbb{C}$ for path p as

$$a_p = \frac{\lambda}{4\pi} \mathbf{C}_r^*(\theta_p^{\mathrm{RX}}, \varphi_p^{\mathrm{RX}}) \left(\prod_{k=K}^{1} \underline{\mathbf{T}}_p^k \right) \mathbf{C}_t(\theta_p^{\mathrm{TX}}, \varphi_p^{\mathrm{TX}}), \tag{2.165}$$

where

$$\mathbf{C}_t(\theta, \varphi) = \sqrt{G_t} \mathbf{F}_t(\theta, \varphi), \tag{2.166}$$

and

$$\mathbf{C}_r(\theta, \varphi) = \sqrt{G_r} \mathbf{F}_r(\theta, \varphi). \tag{2.167}$$

These vectors are the directive complex field pattern vectors (often referred to as the *embedded antenna patterns*) of the transmit and receive antennas. The factor $\frac{\lambda}{4\pi}$ accounts for the reference free-space path loss scale. Under this definition, the received signal amplitude at the output of the antenna is directly proportional to a_p, and the received power for a single path is given by $P_{r,p} = P_t |a_p|^2$.

2.10.2 Multipath Summation and Channel Response

In general, the wireless channel is composed of a multitude of such paths, which we treat individually and then aggregate. Because each path p travels a different geometric distance d_p, it arrives at the receiver with a specific propagation delay $\tau_p = \frac{d_p}{c}$.

This multipath nature is captured by the channel impulse response (CIR) $h(\tau)$, which is essential for characterizing wideband systems where not all paths arrive at the same time, and is given by

$$h(\tau) = \sum_{p=1}^{P} a_p \, \delta(\tau - \tau_p), \tag{2.168}$$

where P is the total number of paths. In wideband systems, paths with sufficiently different delays can be resolved by the receiver, leading to frequency-selective fading.

For multi-antenna links with N_t transmit and N_r receive elements, the channel cannot be summarized by a single scalar impulse response. These systems, commonly referred to as multiple-input multiple-output (MIMO), require keeping the path contributions before the final array projection, leading to a matrix-valued impulse response $\boldsymbol{H}(\tau) \in \mathbb{C}^{N_r \times N_t}$. If the individual array elements are located at relative positions $\mathbf{r}_{t,n}$ (for $n = 1, \ldots, N_t$) and $\mathbf{r}_{r,m}$ (for $m = 1, \ldots, N_r$) with respect to the array phase centers, the channel coefficient $H_{m,n}(\tau)$ between transmit element n and receive element m is given by

$$H_{m,n}(\tau) = \sum_{p=1}^{P} a_{p,m,n} \, \delta(\tau - \tau_{p,m,n}), \tag{2.169}$$

where $a_{p,m,n}$ and $\tau_{p,m,n}$ are the element-to-element path coefficient and delay. Assuming the array aperture is small relative to the total path lengths, the delay is approximately uniform across the elements ($\tau_{p,m,n} \approx \tau_p$). If the elements of

the arrays are identical and copy the reference patterns $\mathbf{C}_t$ and $\mathbf{C}_r$, the spatial response of the arrays can be factored out, yielding

$$\boldsymbol{H}(\tau) = \sum_{p=1}^{P} a_p \, \mathbf{a}_{\mathrm{RX}}(\theta_p^{\mathrm{RX}}, \varphi_p^{\mathrm{RX}}) \, \mathbf{a}_{\mathrm{TX}}^{*}(\theta_p^{\mathrm{TX}}, \varphi_p^{\mathrm{TX}}) \, \delta(\tau - \tau_p), \tag{2.170}$$

where:

- The scalar path coefficient a_p between the reference phase centers is given by

$$a_p = \frac{\lambda}{4\pi} \mathbf{C}_r^{*}(\theta_p^{\mathrm{RX}}, \varphi_p^{\mathrm{RX}}) \left(\prod_{k=K}^{1} \underline{\mathbf{T}}_p^{k} \right) \mathbf{C}_t(\theta_p^{\mathrm{TX}}, \varphi_p^{\mathrm{TX}}). \tag{2.171}$$

- The transmit and receive *steering vectors*, $\mathbf{a}_{\mathrm{TX}} \in \mathbb{C}^{N_t}$ and $\mathbf{a}_{\mathrm{RX}} \in \mathbb{C}^{N_r}$, capture the relative phase delays across the array elements and are given by

$$\mathbf{a}_{\mathrm{TX}}(\theta, \varphi) = \left[e^{-j\mathbf{k}^{\mathrm{TX}}(\theta,\varphi)\cdot\mathbf{r}_{t,1}} \quad \dots \quad e^{-j\mathbf{k}^{\mathrm{TX}}(\theta,\varphi)\cdot\mathbf{r}_{t,N_t}} \right]^{\top}, \tag{2.172}$$

$$\mathbf{a}_{\mathrm{RX}}(\theta, \varphi) = \left[e^{-j\mathbf{k}^{\mathrm{RX}}(\theta,\varphi)\cdot\mathbf{r}_{r,1}} \quad \dots \quad e^{-j\mathbf{k}^{\mathrm{RX}}(\theta,\varphi)\cdot\mathbf{r}_{r,N_r}} \right]^{\top}, \tag{2.173}$$

  with $\mathbf{k}^{\mathrm{TX}}$ and $\mathbf{k}^{\mathrm{RX}}$ being the wave vectors of departure and arrival, respectively. The superscript $*$ denotes the conjugate transpose (Hermitian) operator.

Conversely, in narrowband systems, the differences in delay are small compared to the inverse bandwidth of the signal. The paths temporally overlap, and the total received channel coefficient is simply the *coherent sum* of all individual path contributions

$$a_{\mathrm{total}} = \sum_{p=1}^{P} a_p. \tag{2.174}$$

Coherent summation preserves the complex phase relationships between paths, capturing the constructive and destructive interference that characterizes small-scale fading. The total coherent received power is proportional to

$|a_{\text{total}}|^2$.

Alternatively, if we are only interested in the macroscopic average power (e.g., for general coverage predictions) and want to smooth out rapid fading fluctuations, we can perform a *non-coherent (or incoherent) summation*. Here, we sum the powers of the individual paths rather than their complex amplitudes, which yields

$$P_{\text{incoherent}} \propto \sum_{p=1}^{P} |a_p|^2. \tag{2.175}$$

This approach disregards phase interference entirely, providing a stable estimate of the local mean power. In fact, this approach is often preferred to the coherent summation in large-scale coverage predictions, as it is less sensitive to small positioning errors of various objects in the environment, which are inevitable in real-world scenarios. Unless explicitly stated otherwise, all coverage maps presented in this book are generated using this non-coherent summation approach.

2.10.3 The Role of Ray Tracing

The mathematical formulation of the channel relies entirely on knowing the properties of every valid path p. Thus, the primary role of a ray tracing engine in this pipeline is twofold:

1. *Finding the geometrical paths* that connect the transmitter to the receiver through valid sequences of interactions in the environment.
2. *Aggregating the various physical models* (antennas, materials, RISs) along these trajectories to compute the electromagnetic fields and, ultimately, the total channel response.

While finding paths for purely specular reflections is relatively straightforward, a major challenge arises with more complex interactions. Specular reflection and transmission

map an incident ray to a discrete number of outgoing rays. However, phenomena such as edge diffraction (Section 2.6) or diffuse scattering (Section 2.7) split a single incident ray into a continuous infinity of outgoing rays (e.g., along a Keller cone or over a scattering hemisphere).

This ray splitting raises significant questions about how to proceed in a practical simulator. A naive approach would lead to a combinatorial explosion, attempting to track an infinite number of child rays. Efficiently discovering these paths and sampling these continuous distributions without overwhelming computational resources is a core algorithmic challenge, setting the stage for the methodologies discussed in the next chapter.

2.11 Conclusion

This chapter has built the physical toolkit for ray-based radio propagation modeling—the first of the three main components introduced in the preface. Starting from the high-frequency asymptotic expansion of Maxwell's equations (Section 2.2), we showed that waves can be replaced by rays carrying amplitude, phase, and polarization, each governed by the eikonal equation and the ray optical continuation formula. We then described how transmit and receive antennas modulate the fields at the endpoints of a path (Section 2.3), and how polarization is tracked coherently through cascaded interactions using Jones calculus (Section 2.4). The physics of each individual interaction was then treated in turn: specular reflection from smooth surfaces with Fresnel coefficients (Section 2.5), edge diffraction via the UTD (Section 2.6), rough-surface scattering (Section 2.7), engineered RISs (Section 2.8), and slab transmission (Section 2.9). Finally, the previous section showed how the contributions of all ray paths are coherently summed to form the received field and the wireless channel impulse response.

Two fundamental questions have been deliberately left

open, however. First, *how do we find the geometrical ray paths* that undergo an arbitrary sequence of reflections, diffractions, or other interactions in complex 3D scenes? The answer—covering exact and approximate methods, such as the image method and shooting-and-bouncing rays, and their algorithmic trade-offs—is the subject of Chapter 3. Second, *what does differentiability mean* in the context of ray tracing, and why does it matter for radio propagation? Chapter 4 addresses that question, showing how gradients of the channel with respect to scene parameters can be computed and exploited for gradient-based optimization and integration with machine learning pipelines.

Building

How To Trace Paths | 3

Building on the theoretical foundations established in the previous chapters, we now turn our attention to the practical implementation of ray tracing algorithms. While Chapter 2 provided the mathematical framework based on geometric optics, this chapter focuses on computational techniques for tracing ray paths through complex environments.

In this book, we categorize path tracing algorithms into two main approaches: exact and inexact methods. Exact methods, such as the image method, determine the precise coordinates of a ray path connecting two communicating nodes (e.g., a source and a receiver) for a prescribed sequence of interactions, such as specular reflections. In contrast, inexact methods seek approximations and often trade geometric precision for computational speed.

Both exact and inexact methods rely on the geometrical optics principles discussed in Chapter 2. They typically assume that the wavelength of radio waves is much smaller than the objects in the environment. Under this assumption, propagation is modeled as rays traveling in straight lines and interacting with surfaces through reflection, refraction, and diffraction. Fermat's principle underlies all methods presented in this chapter and provides the governing physical law for ray trajectories.

3.1 Exact Methods

All exact methods address the same geometric task: given a transmitter, a receiver, and an ordered interaction sequence

$$\boldsymbol{I} = [I_1, I_2, \ldots, I_K], \tag{3.1}$$

where I_k represents the k-th interaction (e.g., a reflection on a specific wall), find the corresponding ray path(s), if

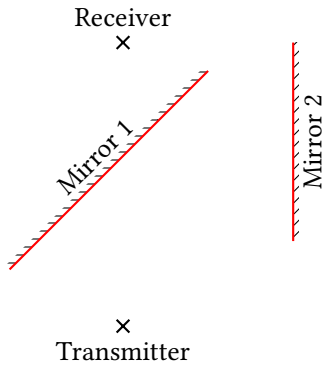


Figure 3.1: Illustration of the exact path problem for a double-reflection case: how to find the exact path coordinates of a ray joining a transmitter and a receiver while undergoing a specular reflection on the first wall (mirror) and then on the second wall?

any, that satisfy the physical interaction laws in that exact order. We call such an ordered sequence a *path candidate*. Depending on the geometry and interaction types, one path candidate may produce no valid path, one valid path, or multiple valid paths.

In this chapter, we focus on solving the *exact-coordinate problem* for a given path candidate. Methods for generating path candidates efficiently are discussed later in Chapter 5, and a practical walkthrough is provided in Appendix A. For example, if the path candidate is a double reflection (wall then floor), an exact method computes the precise interaction coordinates that realize this sequence (see Figure 3.1).

3.1.1 The Image Method

Although distinct from the "method of images" used to solve boundary-value problems in differential equations, the image method in ray tracing shares a conceptual and historical lineage with this earlier technique. In the method of images, the goal is to replace boundary conditions with image sources. In path tracing, boundaries are planar reflectors, such as walls. The reflected ray path is obtained by first computing the image of the source (transmitter) with respect to the reflector, and then determining the intersection between the reflector and the line segment

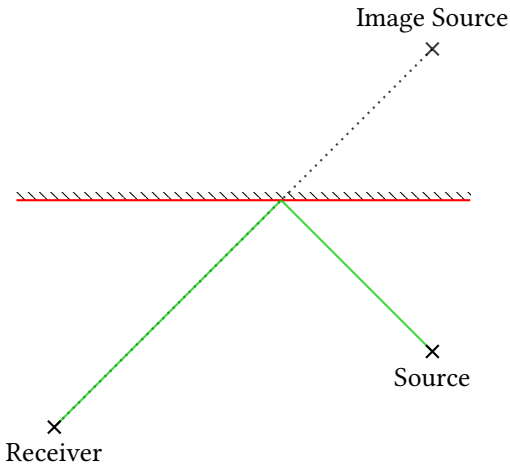


Figure 3.2: Illustration of the image-source method for a single reflection. The virtual source is the mirror image of the actual source across the reflecting surface. The path from the source to the receiver via the reflection point is equivalent to the direct path from the image source to the receiver.

connecting the image and the receiver. This intersection point corresponds exactly to the specular reflection point on the surface, and the ray path can thus be traced (see Figure 3.2). In its modern form, the method was introduced for rectangular room-acoustics simulation by Allen and Berkley [49] and later extended to arbitrary polyhedra by Borish [50]. The same geometric principle was subsequently adopted for deterministic radio-network design in site-specific propagation studies.

[49]: Allen et al. (1979), *Image method for efficiently simulating small-room acoustics*

[50]: Borish (1984), *Extension of the image model to arbitrary polyhedra*

Mathematical Formulation

The image method determines the exact paths between a transmitter (TX) and a receiver (RX), with a certain number K of specular reflections, by computing the successive images of TX using orthogonal symmetry across the surfaces. As illustrated in Figure 3.3, all images are first computed successively across each surface: the image of TX across the first surface is computed, then the image of this image across the second surface, and so on until the last surface is reached.

This *forward* procedure is defined by

$$\mathbf{i}_k = \mathbf{i}_{k-1} - 2\left[(\mathbf{i}_{k-1} - \mathbf{p}_k) \cdot \hat{\mathbf{n}}_k\right] \hat{\mathbf{n}}_k, \tag{3.2}$$

where $\mathbf{i}_k$ and $\mathbf{p}_k$ are, respectively, the k-th image and any point on the k-th surface, and $\mathbf{i}_0$ is the location of TX.

Subsequently, the exact interaction coordinates are derived via a recursive backward pass, from RX to TX, by intersecting each surface with the line joining the subsequent interaction point (or RX) and the corresponding image. This *backward* step is defined by

$$\mathbf{x}_k = \mathbf{x}_{k+1} + \frac{(\mathbf{p}_k - \mathbf{x}_{k+1}) \cdot \hat{\mathbf{n}}_k}{(\mathbf{x}_{k+1} - \mathbf{i}_k) \cdot \hat{\mathbf{n}}_k} (\mathbf{x}_{k+1} - \mathbf{i}_k), \tag{3.3}$$

where $\mathbf{x}_k$ is the interaction point on the k-th surface, $\mathbf{x}_0$ is the location of TX, and $\mathbf{x}_{K+1}$ is the location of RX.

This expression follows from a standard line–plane intersection by parameterizing the line from $\mathbf{x}_{k+1}$ toward $\mathbf{i}_k$ as

$$\mathbf{x}(t) = \mathbf{x}_{k+1} + t(\mathbf{x}_{k+1} - \mathbf{i}_k), \tag{3.4}$$

and enforcing the plane constraint $(\mathbf{x}(t) - \mathbf{p}_k) \cdot \hat{\mathbf{n}}_k = 0$.

From these formulas, the connection to Fermat's principle is not immediate. However, one can verify *a posteriori* that the traced ray path satisfies the law of reflection.

For specular reflections on planar surfaces, the image method is among the most computationally efficient approaches, as it requires a number of operations that is linear in the number of interactions.

Extension to Arbitrary Geometries and Interactions

Although the above algorithm is strictly formulated for specular reflections on planar surfaces, the guiding principle of the image method—constructing a sequence of virtual sources that satisfies boundary conditions—can be

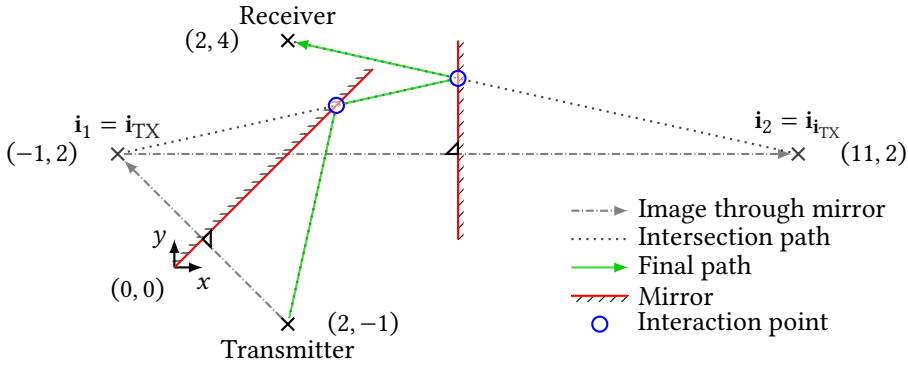


Figure 3.3: Example application of the image method to the double-reflection scenario illustrated in Figure 3.1. The method determines the only valid path joining TX and RX through two successive specular reflections (the interaction order matters). First, the successive images of TX are computed across each mirror using orthogonal symmetry. Second, the intersection points with the mirrors are computed backward—from the last mirror to the first—by connecting RX, and then each successive intersection point, to the corresponding image of TX. Finally, the valid path is assembled by joining TX, the intermediate intersection points, and RX.

generalized to reflections, diffractions, or other interactions on non-planar objects. In practice, dedicated formulations are required for these cases, and they generally support only one diffraction, located either at the first or at the last interaction [51]. As geometries become curved or interactions such as edge diffraction are introduced, the analytical construction of these virtual sources becomes significantly more complex because the virtual loci are no longer points, but curves or surfaces [52]. This limitation motivates the alternative formulations discussed next.

[51]: Quatresooz et al. (2021), *Tracking of Interaction Points for Improved Dynamic Ray Tracing*

[52]: Aguado Agelet et al. (2000), *Efficient ray-tracing acceleration techniques for radio propagation modeling*

3.1.2 Min-Path-Tracing

The min-path-tracing (MPT) method is a deterministic, minimization-based path-finding approach developed to address the limitations of the image method for complex interaction chains and non-planar geometries [53]. While the image method is highly efficient for specular reflections on planar surfaces, it does not handle diffraction in its usual form and becomes mathematically cumbersome for

[53]: Eertmans et al. (2023), *Min-Path-Tracing: A Diffraction Aware Alternative to Image Method in Ray Tracing*

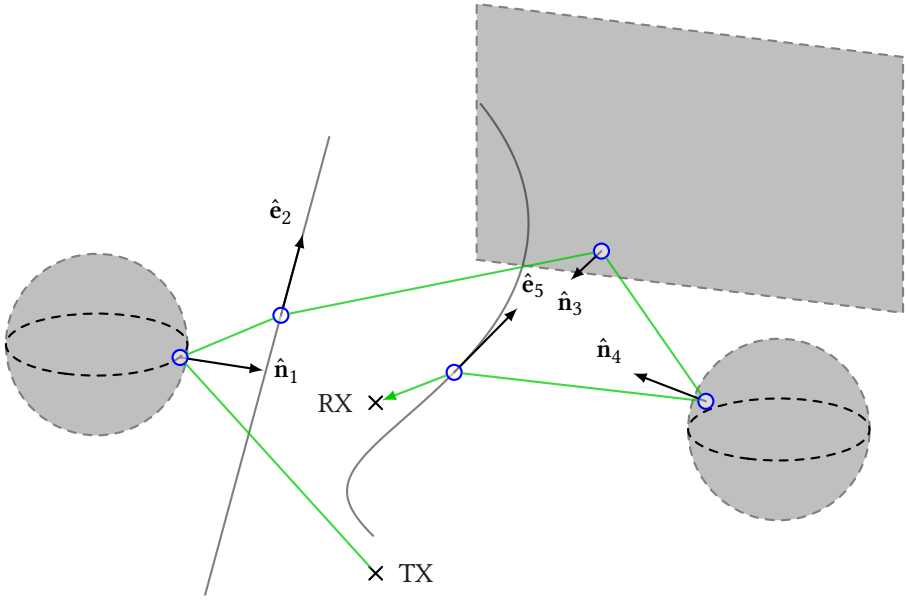


Figure 3.4: Example application of the MPT method to a complex scenario comprising, in order: a reflection on a sphere, a diffraction on a straight edge, a double reflection (first on a plane, then on a sphere), and finally a diffraction on a helicoidal edge. A 2D top-down view of the same scenario is provided in Figure 3.5.

curved surfaces. MPT formulates exact coordinate recovery for a given path candidate as a non-linear optimization problem. Importantly, this optimization view is not limited to reflections and diffractions. It can incorporate refraction constraints as additional interaction laws, and it is also well-suited to modeling programmable interactions such as RIS through interaction terms that encode generalized reflection or refraction behavior.

Problem Formulation

The fundamental principle underlying MPT is to formulate path tracing as the minimization of a cost function C that quantifies the violation of physical interaction laws and object-membership constraints. The method iteratively adjusts the unknown interaction coordinates to minimize

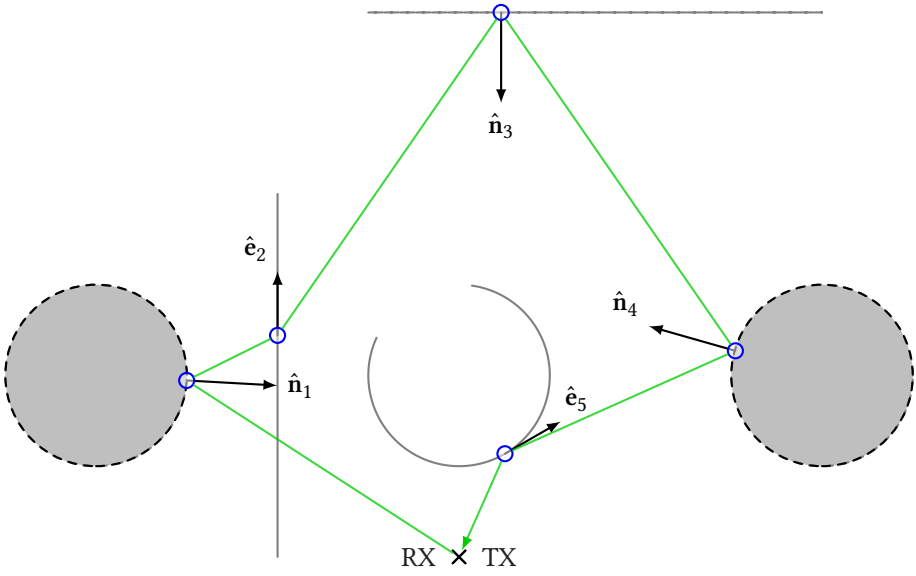


Figure 3.5: 2D top-down view of the 3D scenario illustrated in Figure 3.4. The equality between incidence and reflection angles on the plane and spheres is visually apparent. For edge diffraction, the equality of angles between incoming and outgoing rays and the edge direction vector is less evident because both edge direction vectors have a non-zero z-component.

this cost, effectively searching for a configuration that simultaneously satisfies all physical laws.

In 3D, the unknown interaction coordinates can be collected into a single matrix $\boldsymbol{X}$ of size $K \times 3$, where K is the number of interactions in the path candidate. The goal is to find the optimal $\boldsymbol{X}^\star$ that minimizes the cost function $C(\boldsymbol{X})$, i.e.,

$$\boldsymbol{X}^\star = \underset{\boldsymbol{X} \in \mathbb{R}^{K\times 3}}{\arg\min}\, C(\boldsymbol{X}), \tag{3.5}$$

and to verify that $C(\boldsymbol{X}^\star)$ approaches zero to ensure that the solution is physically valid.

Although one could use a root-finding algorithm to solve $C(\boldsymbol{X}) = 0$ directly, a minimization approach is generally more robust because it handles cases in which constraints cannot be satisfied exactly due to numerical effects or modeling approximations. Minimizing C therefore yields

the best admissible solution even when an exact solution does not exist.

For each interaction index k, MPT defines two complementary cost terms:

1. An *interaction* term C_k^{int}, enforcing the local physical law (specular reflection or edge diffraction) between $(\mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{x}_{k+1})$.
2. An *object-belonging* term C_k^{obj}, enforcing that $\mathbf{x}_k$ lies on the object designated by interaction I_k.

The global objective is then built as the sum of the squared ℓ^2-norms of the vectors of these local penalties over all interactions

$$C(\boldsymbol{X}) = \underbrace{\|\boldsymbol{c}^{\text{int}}(\boldsymbol{X})\|^2}_{C^{\text{int}}} + \underbrace{\|\boldsymbol{c}^{\text{obj}}(\boldsymbol{X})\|^2}_{C^{\text{obj}}}, \tag{3.6}$$

where $\boldsymbol{c}^{\text{int}}(\boldsymbol{X})$ and $\boldsymbol{c}^{\text{obj}}(\boldsymbol{X})$ are the vectors of interaction and object-belonging costs, respectively, and are given by

$$\boldsymbol{c}^{\text{int}}(\boldsymbol{X}) = \left[c_1^{\text{int}}(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2) \quad \dots \quad c_K^{\text{int}}(\mathbf{x}_{K-1}, \mathbf{x}_K, \mathbf{x}_{K+1})\right]^\top, \tag{3.7}$$

$$\boldsymbol{c}^{\text{obj}}(\boldsymbol{X}) = \left[c_1^{\text{obj}}(\mathbf{x}_1) \quad \dots \quad c_K^{\text{obj}}(\mathbf{x}_K)\right]^\top. \tag{3.8}$$

The individual cost terms c_k^{int} and c_k^{obj} are not necessarily scalar: they can be vector-valued to capture multiple constraints simultaneously, hence the use of bold notation. In (3.7) and (3.8), the total cost vectors are formed by concatenating all individual cost vectors. Figures 3.6 and 3.7 illustrate the interaction and object-belonging cost functions for a single reflection, respectively.

Specular Reflection For a specular reflection at a point $\mathbf{x}_k$ on a surface with local normal vector $\hat{\mathbf{n}}_k$, the incoming incident vector $\mathbf{s}^i$ and the outgoing reflected vector $\mathbf{s}^r$ must

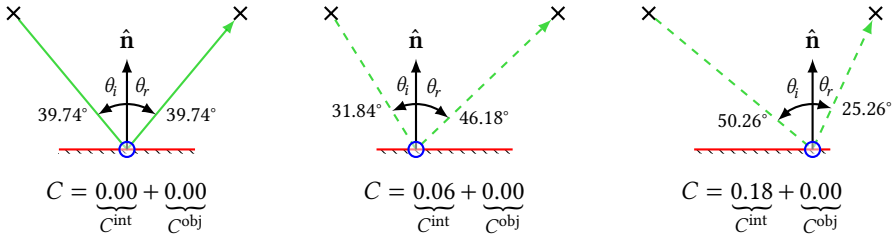


Figure 3.6: Illustration of the residual interaction-based function C^{int} for a single reflection.

satisfy the law of reflection, meaning they make the same angle with the normal,

$$\hat{\mathbf{s}}^r = \hat{\mathbf{s}}^i - 2\left(\hat{\mathbf{s}}^i \cdot \hat{\mathbf{n}}_k\right)\hat{\mathbf{n}}_k. \tag{3.9}$$

If an implicit equation $f_k(\mathbf{x}) = 0$ is known, the normal vector is readily obtained by normalizing the gradient of the surface function, i.e., $\hat{\mathbf{n}}_k = \frac{\nabla f_k}{\|\nabla f_k\|}$. To circumvent some numerical singularities that optimization routines can encounter when normalizing vectors of unknown sizes, MPT relaxes the strict unit-norm requirement on the reflected vector by rescaling the relationship with $\gamma_k = \frac{\|\mathbf{s}^i\|}{\|\mathbf{s}^r\|}$, yielding

$$\gamma_k \cdot \mathbf{s}^r = \mathbf{s}^i - 2\left(\mathbf{s}^i \cdot \hat{\mathbf{n}}_k\right)\hat{\mathbf{n}}_k. \tag{3.10}$$

By equating (3.10) to zero, the residual interaction cost for a reflection is then defined as

$$\begin{aligned} \mathbf{c}_k^{\text{int}}(\mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{x}_{k+1}) = \gamma_k\left(\mathbf{x}_{k+1} - \mathbf{x}_k\right) \\ - \left(\mathbf{x}_k - \mathbf{x}_{k-1}\right) \\ + 2\left(\left(\mathbf{x}_k - \mathbf{x}_{k-1}\right) \cdot \hat{\mathbf{n}}_k\right)\hat{\mathbf{n}}_k, \end{aligned} \tag{3.11}$$

and the residual object-belonging cost is defined as

$$c_k^{\text{obj}}(\mathbf{x}_k) = f_k(\mathbf{x}_k). \tag{3.12}$$

Edge Diffraction Similar to the reflection case, the interaction cost for edge diffraction is derived from the vector

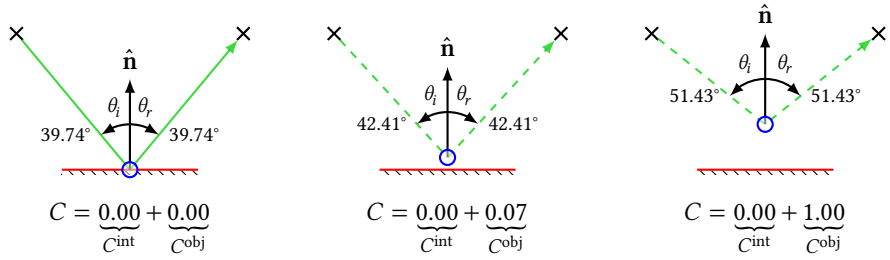


Figure 3.7: Illustration of the residual object-belonging function C^{obj} for a single reflection.

form of the geometrical theory of diffraction. An incident vector $\mathbf{s}^i$ striking an edge with local direction vector $\hat{\mathbf{e}}_k$ diffracts into a cone of rays $\mathbf{s}^d$. The incident and diffracted vectors must form the same angle with the edge and are thus related by

$$\frac{\mathbf{s}^i \cdot \hat{\mathbf{e}}_k}{\|\mathbf{s}^i\|} = \frac{\mathbf{s}^d \cdot \hat{\mathbf{e}}_k}{\|\mathbf{s}^d\|}. \tag{3.13}$$

For a parametrically defined edge $\mathbf{e}_k(s)$, its direction vector is obtained as $\hat{\mathbf{e}}_k = \frac{\mathrm{d}\mathbf{e}_k}{\mathrm{d}s}$.

By defining the departure point from the previous interaction as $\mathbf{x}_{k-1}$ and the arrival point at the next interaction as $\mathbf{x}_{k+1}$ (with $\mathbf{x}_0$ and $\mathbf{x}_{K+1}$ being TX and RX, respectively), MPT forms residuals for each interaction triplet $(\mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{x}_{k+1})$ as

$$\begin{aligned} c_k^{\text{int}}(\mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{x}_{k+1}) = &\frac{(\mathbf{x}_k - \mathbf{x}_{k-1}) \cdot \hat{\mathbf{e}}_k}{\|\mathbf{x}_k - \mathbf{x}_{k-1}\|} \\ &- \frac{(\mathbf{x}_{k+1} - \mathbf{x}_k) \cdot \hat{\mathbf{e}}_k}{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|}, \end{aligned} \tag{3.14}$$

and the residual object-belonging cost is defined as

$$c_k^{\text{obj}}(\mathbf{x}_k) = f_k(\mathbf{x}_k), \tag{3.15}$$

where $f_k(\mathbf{x}_k) = 0$ is an implicit equation of the edge. Note that the interaction cost for edge diffraction is scalar, as it only captures a single constraint, while the interaction cost for reflection is vector-valued, as it captures three

constraints (one per coordinate).

Modeling Other Interactions

The MPT framework is flexible and can be extended to model other types of interactions beyond specular reflection and edge diffraction. For example, refraction can be modeled by defining an interaction cost based on Snell's law, which relates the angles of incidence and refraction to the refractive indices of the media. Similarly, programmable interactions such as those involving RIS can be modeled by defining custom interaction costs that capture the desired reflection or refraction behavior. The key advantage of the MPT approach is that it provides a unified optimization framework that can accommodate a wide variety of interaction types and geometries, making it a powerful tool for path tracing in complex environments.

Parametric Reduction

Optimization in $\mathbb{R}^{K\times 3}$ entails estimating three coordinates per interaction point while simultaneously satisfying the interaction term C^{int} and the object-belonging term C^{obj}. However, when a parametric mapping is available for the interacting objects, the optimization can be significantly simplified. Expressing the problem in the parametric space not only reduces the dimensionality but also eliminates the C^{obj} term, which is then implicitly satisfied. By establishing parametric–Cartesian mapping functions

$$(s_k, t_k) \leftrightarrow (x_k, y_k, z_k) \qquad \text{for surfaces,} \tag{3.16}$$

$$(s_k) \leftrightarrow (x_k, y_k, z_k) \qquad \text{for edges,} \tag{3.17}$$

where the parameters represent 2D coordinates for surfaces and 1D coordinates for edges, the object-belonging constraint is always satisfied by construction, since any parametric coordinate maps to a point on the object. Consequently, the dimensionality drops from $3K$ to $2K_r + K_d$, where K_r is the number of surface parameters and K_d is the

number of edge parameters. The objective then simplifies to minimizing the interaction residual vector

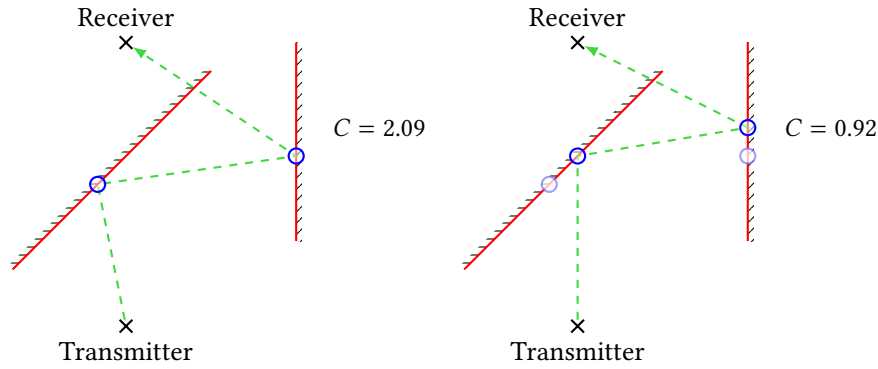
$$\underset{\boldsymbol{t} \in \mathbb{R}^{2K_r+K_d}}{\text{minimize}} \left\| \mathbf{c}^{\text{int}} \left(\mathcal{X}(\boldsymbol{t}) \right) \right\|^2, \tag{3.18}$$

where $\mathcal{X}(\boldsymbol{t})$ is the parametric-to-Cartesian coordinate mapping, and $\boldsymbol{t}$ is the vector of parametric coordinates.

Numerical Considerations

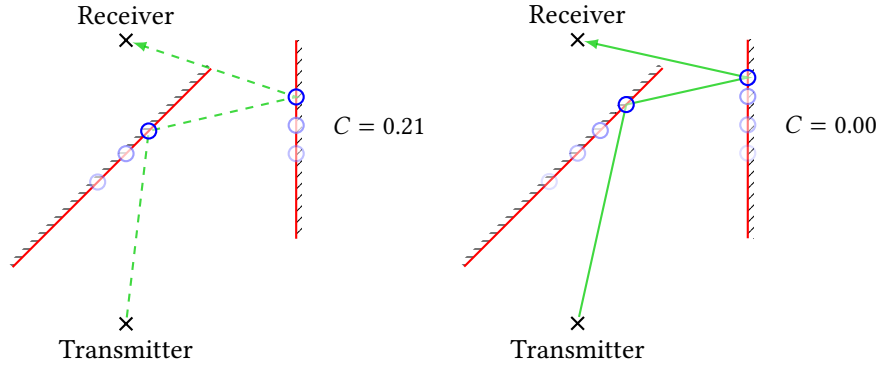
The optimization problem (3.18) is non-convex in general and may admit multiple local minima. Empirical observations indicate that for planar surfaces and straight edges, the minimization landscape is favorable, and standard gradient descent methods converge to valid solutions regardless of initialization (see Figure 3.8 for an example). For more general geometries (e.g., curved surfaces), the optimization should be run multiple times using different random initializations to increase the probability of discovering all distinct physical solutions.

As mentioned earlier, the cost function is designed such that valid paths correspond to global minima with zero cost. However, due to numerical precision limits and the non-convex nature of the problem, it is possible for the optimization to converge to local minima that do not correspond to valid paths (i.e., where $C(\boldsymbol{X}^\star) \neq 0$). Therefore, it is crucial to verify the validity of each converged solution by checking that the cost is sufficiently close to zero and that the interaction points satisfy the physical constraints of their respective interactions.



(a) Random initialization.

(b) After the first iteration.



(c) After the second iteration.

(d) Algorithm has converged.

Figure 3.8: Example application of the MPT minimization algorithm to the double-reflection scenario from Figure 3.1, shown at various stages of convergence.

3.1.3 Fermat-Based Methods

As recalled from Chapter 2, Fermat's principle states that a ray between two points follows a path that makes traversal time stationary with respect to infinitesimal path variations. This naturally raises a practical question: why not find the geometrical interaction points directly by optimizing travel time? The raw variational problem is continuous and infinite-dimensional, making direct computation impractical without discretization. More importantly, geometric constraints—the interaction points must lie on their respective objects in the prescribed order—render the problem generally non-convex. Nevertheless, under two simplifying assumptions that are widely satisfied in radio propagation modeling tools and that we will detail below, the path-finding problem becomes a strictly convex, path-length minimization problem [54, 55].

[54]: Carluccio et al. (2008), *An Efficient Ray Tracing Algorithm for Multiple Straight Wedge Diffraction*

[55]: Eertmans et al. (2025), *Fast, Differentiable, GPU-Accelerated Ray Tracing for Multiple Diffraction and Reflection Paths*

From Time to Length

For a ray path $\mathbf{r}(s)$ joining source $\mathbf{x}_0$ and receiver $\mathbf{x}_{K+1}$, parameterized by arc length $s \in [0, s_{\text{total}}]$, the traversal time is proportional to the optical path length

$$L = \int_0^{s_{\text{total}}} n(s)\,\mathrm{d}s, \tag{3.19}$$

where $n(s)$ is the local refractive index.

In the common case of a single homogeneous medium with constant refractive index, which is the case in free-space propagation, optimizing L is equivalent to optimizing the total geometric path length. For a path candidate with K interactions, fixing the source and receiver, the problem reduces to finding unknown interaction points $\mathbf{X} = \left[\mathbf{x}_1, \dots, \mathbf{x}_K\right]$ as

$$(\mathbf{x}_1^\star, \dots, \mathbf{x}_K^\star) = \underset{\mathbf{x}_1,\dots,\mathbf{x}_K}{\arg\operatorname{opt}}\, L(\mathbf{X}) \overset{\text{def}}{=} \sum_{k=0}^{K} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|. \tag{3.20}$$

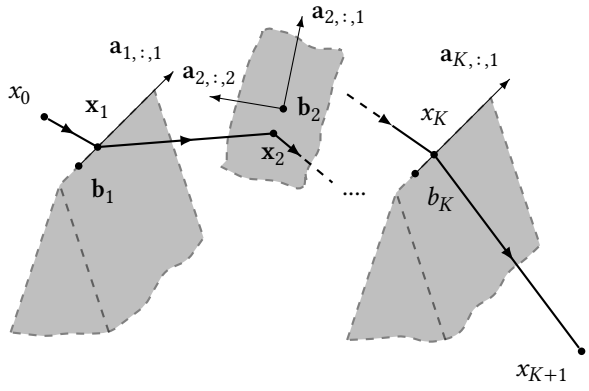


Figure 3.9: Illustration of the path-finding problem under the assumptions of planar surfaces and straight edges, where $\mathbf{a}_{i,:,j}$ is the j-th base vector of the i-th object, and $\mathbf{b}_i$ is a reference point on the i-th object.

One can verify *a posteriori* that, in a homogeneous medium, the optima of (3.20) satisfy the appropriate interaction laws (angle of reflection, Keller cone for diffraction, etc.), confirming the equivalence with Fermat's principle.

Convex Setting Used in Practice

The objective $L(\mathbf{X})$ in (3.20) is not convex in 3D unconstrained space, since the interaction points must lie on their respective objects. Two important assumptions restore strict convexity:

1. Specular reflections occur on *planar* surfaces.
2. Diffractions occur on *straight* edges.

Although these may appear restrictive, most outdoor 3D models are described by polygonal meshes, making both conditions broadly applicable. Under these assumptions, each interaction point $\mathbf{x}_k$ lies in a low-dimensional affine subspace and can be expressed as

$$\mathbf{x}_k = \boldsymbol{A}_k \boldsymbol{t}_k + \mathbf{b}_k, \tag{3.21}$$

where $\mathbf{b}_k \in \mathbb{R}^3$ is a reference point on the k-th object, $\boldsymbol{A}_k \in \mathbb{R}^{3\times d_k}$ is a matrix whose columns are orthonormal basis vectors spanning the object, and $\boldsymbol{t}_k \in \mathbb{R}^{d_k}$ is the parametric

coordinate, with $d_k = 2$ for planar surfaces and $d_k = 1$ for straight edges (see Figure 3.9).

By stacking all parameters into a matrix $\boldsymbol{T}$, the path-length objective becomes

$$L(\boldsymbol{T}) = \sum_{k=0}^{K} \left\| \left(\boldsymbol{A}_{k+1} \boldsymbol{t}_{k+1} + \mathbf{b}_{k+1} \right) - \left(\boldsymbol{A}_k \boldsymbol{t}_k + \mathbf{b}_k \right) \right\|, \tag{3.22}$$

where $\boldsymbol{A}_0 \boldsymbol{t}_0 = \mathbf{0}$ and $\boldsymbol{A}_{K+1} \boldsymbol{t}_{K+1} = \mathbf{0}$, and where $\mathbf{b}_0$ and $\mathbf{b}_{K+1}$ are the initial and final points, typically the transmitter and receiver positions, respectively. Each term is the Euclidean norm of an affine function of $\boldsymbol{T}$, so $L(\boldsymbol{T})$ is strictly convex, guaranteeing a unique global minimum for each candidate path.

Extension to Refraction

By assuming a homogeneous medium, we have so far equated Fermat's principle with path-length minimization. In more general settings, the traversal time depends on the local refractive index, and the objective should be the optical path length. If the medium is piecewise constant (e.g., indoor environments with different materials), refraction can be incorporated by weighting each segment by the refractive index of the traversed medium, yielding the modified objective

$$L(\boldsymbol{T}) = \sum_{k=0}^{K} n_k \left\| \left(\boldsymbol{A}_{k+1} \boldsymbol{t}_{k+1} + \mathbf{b}_{k+1} \right) - \left(\boldsymbol{A}_k \boldsymbol{t}_k + \mathbf{b}_k \right) \right\|, \tag{3.23}$$

where n_k is the refractive index of the medium traversed by the k-th segment. This formulation preserves convexity, enabling efficient optimization and accurate modeling of refraction effects in environments with piecewise-constant media.

Two Families of Fermat-Based Methods

The convex path-length formulation was first exploited by Carluccio and Albani [54] for *diffraction-only* paths, using a modified Newton method with near-quadratic convergence. Puggelli, Carluccio, and Albani [56] then extended this to paths mixing reflections and diffractions via a hybrid strategy: the image method eliminates all reflection unknowns analytically, reducing the problem to pure diffraction minimization. This approach is also referred to as the *image edge model* in the literature [57]. While computationally elegant, this hybrid approach yields problem dimensions that depend on the number and ordering of diffractions, and introduces different code paths for reflections and diffractions—both of which are detrimental to GPU parallelism, where all threads must follow the same execution path.

[54]: Carluccio et al. (2008), *An Efficient Ray Tracing Algorithm for Multiple Straight Wedge Diffraction*

[56]: Puggelli et al. (2014), *A Novel Ray Tracing Algorithm for Scenarios Comprising Pre-Ordered Multiple Planar Reflectors, Straight Wedges, and Vertexes*

[57]: Erraji et al. (2021), *The image edge model*

Another approach is to handle all interaction types within a single optimization program of fixed dimension. In this approach, reflections and diffractions enter the same parametric path-length objective (3.22) with no special cases. This eliminates branching and enables efficient batch processing [55]. For example, reflections ($d_k = 2$) and diffractions ($d_k = 1$) can be handled with a uniform dimension by padding the second column of $\boldsymbol{A}_k$ with zeros for edges. This yields a problem of constant size regardless of interaction type, which is essential for efficient parallel processing without per-interaction branching. However, it comes at the cost of increased computational complexity. This topic will be discussed in further detail in Chapter 5.

[55]: Eertmans et al. (2025), *Fast, Differentiable, GPU-Accelerated Ray Tracing for Multiple Diffraction and Reflection Paths*

3.1.4 The Combinatorial Bottleneck and Path Validation

While exact methods provide precise path coordinates, their practical application is governed by several critical steps that precede or follow the raw coordinate discovery:

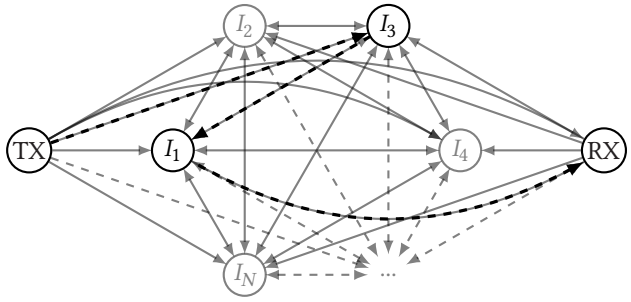


Figure 3.10: Directed graph illustrating all possible path candidates from TX to RX in a scenario with N possible interactions (e.g., N different surfaces) and a maximum interaction order of $K = 3$. Each node represents an interaction, and each directed edge represents a possible transition from one interaction to the next. An example path candidate, emphasized in black, is TX $\rightarrow I_3 \rightarrow I_1 \rightarrow$ RX. The total number of candidates grows exponentially with the number of interactions, since each interaction can lead to multiple subsequent interactions.

managing the combinatorial explosion of candidates, ensuring interactions occur within finite object boundaries, and verifying clear line-of-sight.

Computational Complexity

Each method presented previously has relatively low complexity for a single path candidate. The image method requires a linear number of operations, whereas MPT and Fermat-based methods require several iterations of a gradient-based optimizer. In the worst case, optimization-based methods converge in a number of iterations that grows quadratically with the number of interactions. However, the number of interactions is typically small (often well below ten) because each additional interaction decreases signal power. Consequently, the principal bottleneck of exact path tracing is not coordinate recovery for a single candidate, but the combinatorial explosion of candidates as interaction order increases.

The computational complexity of finding exact paths grows exponentially with the number of interactions. For an environment with N possible interactions (typically proportional to the number of objects) and a maximum interaction

order K, the number of possible interaction sequences, or path candidates, scales as

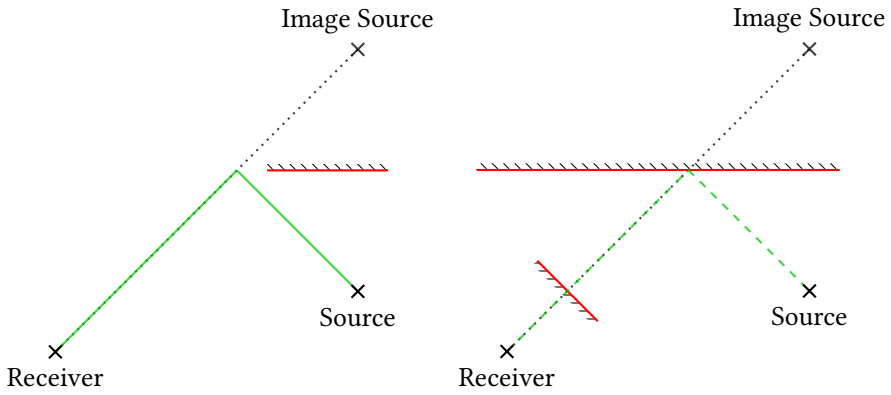
$$\sum_{k=1}^{K} N(N-1)^{k-1} = \frac{N\left((N-1)^K - 1\right)}{N-2}. \tag{3.24}$$

At each interaction order, the number of candidates grows by a factor of N or $N-1$, yielding exponential growth as K increases. This growth is a fundamental challenge in exact path tracing, since exhaustive evaluation becomes computationally infeasible even for moderate values of N and K. Therefore, exact—or exhaustive—methods (mainly used in *point-to-point* ray tracing) are typically limited to scenarios with a small number of interactions or low interaction orders, unless they are combined with sophisticated acceleration or complexity-reduction techniques, some of which are discussed in Chapter 5. The set of all possible path candidates is sometimes referred to as the *image tree* of the scenario, because it can be visualized as a tree where each node represents an interaction and branches represent possible subsequent interactions. This analogy will be particularly useful when discussing strategies to manage the combinatorial explosion of candidates in subsequent chapters.

Assumptions on Infinite Objects and Geometric Validity

As hinted in the introduction to exact methods, the mathematical formulations (e.g., finding an image source across a plane or optimizing coordinates in a parametric space) initially assume that interacting objects are infinitely large and that ray paths are not obstructed by other obstacles in the scene. Under these assumptions, interaction points are determined purely from object geometry.

Consequently, not all discovered paths are physically valid. A separate path-validation phase is mandatory. For a path to be retained:



(a) Mirror is too small.

(b) Ray path is obstructed.

Figure 3.11: Illustration of path validity issues.

1. The intersection points must lie within the finite physical bounds of their corresponding objects (see Figure 3.11a).
2. The sequence of line segments connecting the source, interaction points, and receiver must not be obstructed by other obstacles in the scene (see Figure 3.11b).

These validity and visibility checks are typically performed with ray-scene intersection algorithms. For a single path candidate, they require a linear or quadratic number of ray-object intersection tests and are therefore not the dominant bottleneck compared with exponential candidate growth. However, as the number of candidates increases, the cumulative cost of these checks can become significant, especially in complex scenes with many objects. Efficient implementations are therefore crucial, and techniques to manage this cost are discussed in Chapter 5.

Limitations on Cascaded Diffraction

Finally, it is worth noting that while exact methods can geometrically recover paths undergoing multiple diffractions, calculating their electromagnetic contributions introduces

additional challenges. As discussed in Section 2.6.6, cascading multiple diffraction coefficients is mathematically invalid if subsequent edges fall within preceding transition regions, making the efficient and convenient calculation of these coefficients a complex task. Although these field-computation issues are not discussed in detail here, they must be carefully addressed when developing a ray tracer. In the simulations in this book where electromagnetic fields are actually computed (see Chapter 6), only up to single-order diffraction will be considered.

3.2 Inexact Methods

The exact methods presented in the previous section share a common prerequisite: for each path candidate, the precise interaction coordinates satisfying Fermat's principle must be determined. As discussed in Section 3.1.2, this requires solving a separate problem for every candidate in the exponentially large image tree of the scenario. Inexact methods relax this requirement and trade geometric precision for computational efficiency. This makes them suitable for highly complex environments or very large numbers of path candidates, where exact methods become impractical.

Inexact methods generally follow one of two philosophies: they either discretize the outgoing wavefront into a finite set of rays and trace each forward from the transmitter, or they introduce stochastic sampling over the space of possible paths. In both cases, accuracy is recovered statistically or through spatial aggregation rather than through the deterministic satisfaction of Fermat's principle.

3.2.1 Shooting and Bouncing Rays

The canonical inexact method is shooting and bouncing rays (SBR), also called *ray launching* [26]. Instead of computing path candidates analytically, SBR launches a large

[26]: Ling et al. (1989), *Shooting and bouncing rays: calculating the RCS of an arbitrarily shaped cavity*

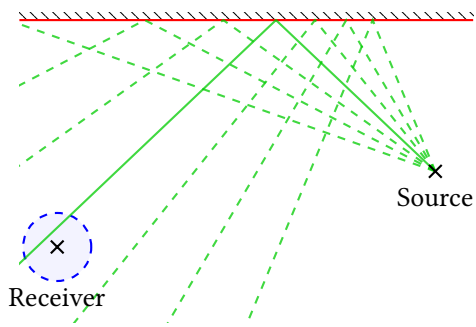


Figure 3.12: Illustration of the SBR method in a simple single-reflection scenario. Rays are launched from the source in various directions, bouncing off walls according to reflection laws. A reception sphere is placed around the receiver, and any ray entering this sphere is counted as received.

but finite number of rays from the source, each in a predetermined direction, and traces them as they propagate and interact with the environment—an approach analogous to *forward ray tracing* in computer graphics, as opposed to the traditional *backward ray tracing* from the camera into the scene. Upon hitting a surface, each ray spawns new rays according to the relevant physical interaction laws (specular reflection, transmission via Snell's law, or diffraction along the Keller cone), and its energy is attenuated accordingly. Rays are terminated when their energy falls below a threshold or when a maximum number of interactions is reached.

Because the wavefront is sampled along a finite set of directions, the angular sampling strategy is a central algorithmic design choice. Common approaches include uniform angular sampling with fixed angular increments [58], Fibonacci sampling [59], icosahedral subdivision for a more isotropic ray distribution [60], and adaptive sampling that concentrates rays in high-gain antenna directions [61]. In addition, hybrid SBR–exact pipelines use SBR for line-of-sight discovery and switch to exact coordinate recovery once a path candidate is identified [58]. The computational cost of SBR grows with the number of launched rays and their maximum interaction order. However, it does not scale with the number of path candidates in the image tree. This makes SBR practical in scenarios where exact methods are computationally prohibitive.

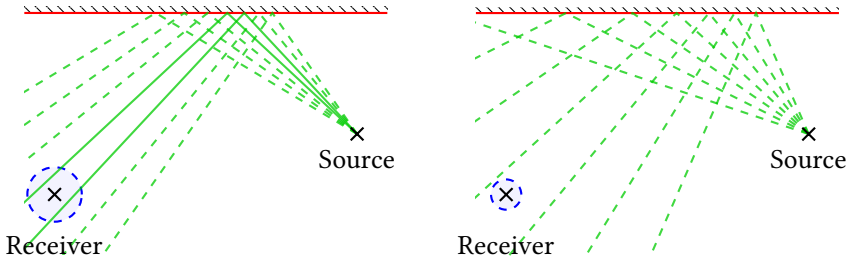
The fundamental challenge of SBR lies in its receiver-

[58]: Chen et al. (1995), *All SBR/image approach to indoor radio propagation modeling*

[59]: Keinert et al. (2015), *Spherical Fibonacci Mapping*

[60]: Kasdorf et al. (2021), *Advancing Accuracy of Shooting and Bouncing Rays Method for Ray-Tracing Propagation Modeling Based on Novel Approaches to Ray Cone Angle Calculation*

[61]: Shi et al. (2015), *Site-specific wave propagation prediction with improved shooting and bouncing ray tracing method*



(a) Double-counting artifact.

(b) Reception sphere is missed.

Figure 3.13: Illustration of possible shortcomings with the SBR method.

detection mechanism. Since an infinitesimally thin ray has a near-zero probability of intersecting a point receiver exactly, a *reception sphere* of finite radius is placed around each receiver, and any ray entering the sphere is counted as received [62]. This approximation introduces systematic artifacts: if the sphere is too large, adjacent rays belonging to the same wavefront may both enter it, artificially inflating received power through double-counting (Figure 3.13a). Conversely, if the sphere is too small, valid contributions are missed (Figure 3.13b). For approximately uniform angular spacing α (in radians) and total unwrapped distance d, a practical guideline is $r \approx \alpha \frac{d}{\sqrt{3}}$, which matches the local ray-tube footprint scale at the receiver distance and balances misses against double-counting. Another artifact, studied in depth by Novak [63], is *inconsistent rays*, i.e., geometrically valid rays belonging to the same wavefront but spatially displaced by the reception-sphere tolerance, which can significantly overestimate signal strength in constrained environments such as tunnels, particularly at distances exceeding 100 m. These artifacts are intrinsic to the reception-sphere mechanism and represent a fundamental limitation of SBR.

[62]: Durgin et al. (1997), *Improved 3D ray launching method for wireless propagation prediction*

[63]: Novak (2022), *Inconsistent Rays in Propagation Prediction by Ray Launching in Rectangular Tunnels*

To mitigate the double-counting issue, some implementations introduce a weighting scheme in which each ray contribution is divided by the number of rays entering the reception sphere, effectively averaging contributions from rays that belong to the same wavefront [62]. Other

implementations use hash-based grouping to identify and merge rays entering the same reception sphere, retaining only the most significant ones (e.g., the closest to the receiver or the shortest path) [64, 65]. In practice, the hash is often computed from the ray's interaction list (i.e., the corresponding path candidate), which enables grouping rays associated with the same wavefront.

[64]: Yun et al. (2001), *Development of a new shooting-and-bouncing ray (SBR) tracing method that avoids ray double counting*

[65]: Novak (2019), *Bloom Filter for Double-Counting Avoidance in Radio Frequency Ray Tracing*

3.2.2 Vertical Plane Launching

In dense macrocellular urban scenarios, full 3D ray launching can be unnecessarily expensive when dominant mechanisms are largely governed by street-canyon guidance and over-rooftop propagation. Vertical plane launching (VPL) addresses this by decoupling propagation into two coupled problems: horizontal exploration across building footprints and vertical-plane evaluation to model rooftop diffraction and height-dependent effects [52, 66]. This 2.5D approximation often captures dominant mechanisms at significantly lower computational cost than exhaustive 3D launching.

[52]: Aguado Agelet et al. (2000), *Efficient ray-tracing acceleration techniques for radio propagation modeling*

[66]: Liang et al. (1998), *A new approach to 3-D ray tracing for propagation prediction in cities*

The method is especially effective when the environment has strong vertical regularity (e.g., blocks of buildings with similar heights) and when out-of-plane scattering is not dominant. Its main limitation is that strongly three-dimensional effects—for example, complex out-of-plane reflections or rich scattering from highly irregular facades—are only approximated. In such cases, full 3D methods remain preferable despite higher runtime.

3.2.3 Volumetric and Extent-Based Tracing

To address the sampling artifacts of SBR, volumetric tracing methods replace infinitesimally thin rays with spatial extents, ensuring that the power carried by each ray tube is coherently assigned to all receivers it encompasses.

Beam Tracing

Beam tracing replaces point rays with thick, polyhedral pyramidal beams, each representing an entire cone of directions from the source [67]. Because a beam covers a continuum of potential ray directions, it eliminates the sampling artifacts of SBR and entirely removes the need for a reception sphere: a receiver is simply illuminated when it falls inside the beam footprint, and the received power is computed analytically from the beam's geometry. However, as beams interact with complex, highly tessellated environments, they fracture into an exponentially growing number of sub-beams, making the method difficult to scale to scenes with many surfaces or curved objects.

[67]: Tan et al. (2015), *A Full 3-D GPU-based Beam-Tracing Method for Complex Indoor Environments Propagation Modeling*

Ray-Tube Tracing

Ray-tube tracing is an intermediate approach between standard SBR and full beam tracing [68]. Each ray is augmented with a surrounding tube footprint, and the algorithm tracks the principal radii of curvature of the associated wavefront as it propagates. This allows macroscopic energy flow to be captured without the combinatorial fragmentation of strict beam clipping. Ray-tube methods are particularly effective in urban cellular scenarios, where the wavefront curvature evolves smoothly over long propagation distances.

[68]: Wang et al. (2003), *A global ray-tube tracing method to determine signal variations in urban areas for mobile communications*

Voxel Cone Tracing

To handle highly detailed curved geometries and diffuse scattering without the geometric fragility of beam clipping, voxel cone tracing discretizes the scene into a hierarchical octree-based voxel grid and traces cones with an expanding footprint via ray marching [69]. This approach is particularly suited to modeling scattering in the super high frequency (SHF) and extra high frequency (EHF) bands, where the detailed surface curvature of objects such as

[69]: Tropkina et al. (2022), *Modeling of SHF/EHF Radio-Wave Scattering for Curved Surfaces With Voxel Cone Tracing*

vehicles significantly influences the scattered field distribution.

Interception Plane Instead of Reception Sphere

Many issues with SBR arise from the reception-sphere mechanism. A sphere is natural when the objective is power at one specific receiver position. In many applications, however, such as network planning, the objective is power over an area, i.e., a *coverage map*. In that setting, it is often more efficient (and more accurate) to replace the reception sphere with a large *interception plane* covering the region where rays may arrive. Each ray is tested for intersection with this plane, and the intersection point is used to accumulate received-power contributions. After the path tracing phase, the interception plane is tiled into cells, from which received power (or other quantities) is estimated around each position. This approach removes artifacts caused by the reception-sphere radius and captures all valid contributions while keeping computational complexity independent of plane size. It does not, however, fully remove double-counting: multiple rays may still intersect the same cell and overestimate power locally. Careful ray-grouping and weighting strategies are therefore still required.

This approach is used, for example, by Sionna RT and is detailed in their technical report [70, Sec. 4].

[70]: Aoudia et al. (2025), *Sionna RT: Technical Report*

3.2.4 Stochastic and Monte Carlo Methods

An alternative to deterministic wavefront discretization is to use controlled randomness in path generation, drawing on the extensive body of Monte Carlo (MC) methods developed in the computer graphics community for light transport simulation [71].

[71]: Veach (1997), *Robust Monte Carlo Methods for Light Transport Simulation*

Monte Carlo Ray Tracing

Instead of launching rays on a predetermined angular grid, MC ray tracing samples ray directions from a probability density function designed to concentrate samples in high-contribution directions [71]. To prevent infinite recursion, rays are terminated probabilistically using *Russian roulette*: a low-energy ray is discarded with some probability, while surviving rays are upweighted to maintain an unbiased estimator. Variance reduction techniques, including importance sampling, stratified sampling, and control variates, can be used to accelerate convergence [71].

[71]: Veach (1997), *Robust Monte Carlo Methods for Light Transport Simulation*

Bidirectional Path Tracing

Bidirectional path tracing (BDPT) traces sub-paths simultaneously from both the transmitter and the receiver and then connects them by evaluating their mutual visibility [72]. By using large, mathematically rigorous interaction surfaces instead of reception spheres, BDPT eliminates the geometric artifacts of standard SBR and reduces errors in multiple-diffraction scenarios [72]. Multiple importance sampling [71] can further be applied to combine the forward and backward sub-paths optimally according to their respective contributions, yielding a statistically efficient, unbiased estimator.

[72]: Taygur (2023), *Wave Propagation Simulations by Bidirectional Ray-Tracing and Investigations on Ray Based Channel Modeling for Massive MIMO*

Metropolis Light Transport

Metropolis light transport (MLT) applies a Markov chain MC technique directly to path space [73]. Starting from a valid path found by a standard MC method, it generates new candidate paths by applying small random perturbations (mutations) to the current path, accepting or rejecting each proposal according to its relative energy contribution. This local exploration of path space allows MLT to efficiently discover and densely sample high-contribution paths in heavily occluded environments, without wasting computations on geometrically unproductive regions.

[73]: Veach et al. (1997), *Metropolis Light Transport*

Photon Mapping

Photon mapping is a two-pass algorithm that treats electromagnetic energy as discrete packets [74]. In the first pass, photons are emitted from the source, traced through the environment, and their interaction points are stored in a spatial data structure (typically a k-d tree [75]). In the second pass, received power at any point is estimated by gathering the nearest stored photons using a kernel density estimator. The method provides local estimates of the delay spread and handles diffuse scattering naturally, without the combinatorial growth of the image tree.

[74]: Schmitz et al. (2006), *Wave propagation using the photon path map*

[75]: Bentley (1975), *Multidimensional binary search trees used for associative searching*

3.2.5 Machine Learning Surrogates

As in other computational physics domains, machine learning surrogates have emerged as a potential alternative to traditional channel modeling methods, including ray tracing. The hope is that, by learning from data, these models can become more efficient than ray tracing while maintaining high accuracy.

Most machine learning applications in radio propagation focus on predicting an electromagnetic quantity directly—for example path loss, received power, or field values—rather than predicting the geometry of propagation paths [76]. A smaller line of work targets richer physical outputs such as the CIR [77], and only a few approaches explicitly model the path geometry itself. One notable example is the SANDWICH model [78], which learns to predict the coordinates of interaction points by treating a ray's path as a step-by-step sequence. To achieve this, SANDWICH uses a neural network to build the trajectory one bounce at a time, similar to the traditional SBR method: at each step, the model determines the ray's next direction and how it interacts with the environment, such as bouncing off a wall or passing through it.

[76]: Vasudevan et al. (2024), *Machine Learning for Radio Propagation Modeling: A Comprehensive Survey*

[77]: Wang et al. (2025), *A Physics-Informed Deep Ray Tracing Network for Regional Channel Impulse Response Estimation*

[78]: Jin et al. (2025), *SANDWICH: Towards an Offline, Differentiable, Fully-Trainable Wireless Neural Ray-Tracing Surrogate*

Learning channel characteristics directly can be highly effective in fixed settings, but it typically ties the model to

the data distribution used for training: geometry, materials, carrier frequency, antenna patterns, and scenario type. As a result, transferring such models to new environments often requires retraining or substantial adaptation [76]. This dependency is one reason why purely data-driven surrogates remain difficult to deploy as general-purpose replacements for geometric ray tracing.

[76]: Vasudevan et al. (2024), *Machine Learning for Radio Propagation Modeling: A Comprehensive Survey*

Conversely, learning ray-path geometry is not always the most effective computational target. In point-to-point ray tracing, the main bottleneck is typically not solving the coordinates of one candidate path, but the combinatorial number of path candidates that must be generated and tested. Predicting path geometry alone therefore does not remove the core complexity driver unless it also reduces candidate exploration. Building on this observation, we present in Chapter 7 a machine-learning-aided ray tracing model designed to reduce the computational complexity of point-to-point ray tracing [79].

[79]: Eertmans et al. (2026), *Transform-Invariant Generative Ray Path Sampling for Efficient Radio Propagation Modeling*

3.3 Conclusion

This chapter has presented the main families of path tracing algorithms used in radio wave propagation modeling, ranging from exact deterministic methods to inexact wavefront-sampling and data-driven surrogate approaches. Exact methods—the image method, MPT, and Fermat-based minimization—recover precise ray paths satisfying Fermat's principle for a given path candidate. However, they are subject to the $\mathcal{O}(N^K)$ combinatorial explosion of the image tree, which limits their scalability to scenarios with few interactions or small numbers of objects. Inexact methods, including SBR, VPL-style 2.5D approximations, volumetric tracing, and MC methods, trade geometric rigor for scalability. They address the combinatorial bottleneck by either sampling the wavefront discretely or exploring path space stochastically, at the cost of systematic artifacts (reception spheres, statistical noise) or reduced accuracy for specular paths. Machine learning surrogates represent an emerging

third paradigm that can bypass geometric path tracing once trained, although they require substantial data and often generalize less readily to new environments.

The methods and trade-offs presented in this chapter provide the necessary background for the rest of this book. In particular, the exponential growth of the image tree motivates the development of more efficient algorithms for generating and evaluating path candidates, which is the focus of Chapter 5. More fundamentally, the use of direct models to determine the channel response raises an open question: can the channel characteristics be computed not only forward, but also *differentiably* with respect to scene parameters? Chapter 4 addresses this question by introducing differentiable ray tracing as a way to compute gradients of the channel response with respect to environmental geometry, enabling inverse design and optimization problems that forward methods alone cannot address.

Differentiable Ray Tracing | 4

In the previous chapters, we established the physical and algorithmic foundations of forward ray tracing for radio propagation. In particular, Chapter 3 formalized exact path recovery through the image method, Fermat-based optimization, and MPT, while emphasizing the combinatorial growth of path candidates. These methods provide physically faithful forward maps from scene parameters to channel responses, but they are still most often used in a one-way "simulate-then-evaluate" workflow.

With a purely forward workflow, the simulator is effectively treated as a "black box." To quantify the effect of a small change in transmitter position, material properties, or reflector orientation, one typically reruns the entire simulation. In high-dimensional design spaces, this repeated evaluation rapidly becomes the main computational bottleneck.

This chapter introduces *differentiable ray tracing*, which augments exact forward solvers with derivative computation. The key point is structural: the same path-finding engines developed in Chapter 3 define the primal computations on top of which reverse-mode derivatives are built. In this way, gradients remain tied to the underlying propagation physics rather than to a surrogate approximation.

We first motivate inverse electromagnetic design and then present the main concepts of automatic differentiation. Next, we discuss the common constraints faced in practical implementation, and finally present methods for smoothing out geometric discontinuities in the optimization landscape.

4.1 The Paradigm Shift to Inverse Electromagnetic Design

The path tracing methods described in Chapter 3 are primarily designed for forward simulation. However, many current challenges in wireless systems—especially in the context of 6G [80, 81]—are inverse problems. In practice, one seeks transmitter placements, antenna settings, or RIS configurations [82, 83] that satisfy coverage, capacity, and latency targets.

This shift from forward simulation to inverse design has motivated recent work in differentiable ray tracing, notably for site-specific electromagnetic digital twins [84]. Frameworks such as Sionna RT [37] and DiffeRT [38, 85] illustrate that differentiability with respect to scene and transceiver parameters enables efficient gradient-based optimization and straightforward integration with machine learning pipelines.

4.1.1 Traditional Optimization Challenges

Classical wireless network optimization commonly relies on one of the following approaches:

- *Brute-force search* uses exhaustive evaluation of parameter combinations over a discretized grid.
- *Heuristic methods* rely on derivative-free algorithms such as genetic algorithms, simulated annealing, or particle swarm optimization.
- *Simplified analytical models* use empirical or statistical approximations that sacrifice site-specific physical accuracy for mathematical tractability.

While useful in many settings, these approaches face important limitations in large modern deployments:

1. *Computational inefficiency* occurs since grid searches scale exponentially with the dimensionality of the parameter space (the curse of dimensionality).

[80]: Wang et al. (2020), *6G Wireless Channel Measurements and Models: Trends and Challenges*

[81]: Chowdhury et al. (2020), *The role of massive MIMO in 6G wireless: advancements, challenges, and opportunities*

[82]: Di Renzo et al. (2019), *Smart radio environments empowered by reconfigurable AI meta-surfaces: an idea whose time has come*

[83]: Wu et al. (2020), *Towards Smart and Reconfigurable Environment: Intelligent Reflecting Surface Aided Wireless Network*

[84]: Testolina et al. (2024), *Boston Twin: the Boston Digital Twin for Ray-Tracing in 6G Networks*

[37]: Hoydis et al. (2023), *Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling*

[38]: Eertmans et al. (2025), *Demonstrating DiffeRT: An Open-Source Library for Optimizing Radio Networks with Differentiable Ray Tracing*

[85]: Eertmans et al. (2024), *DiffeRT2d: A Differentiable Ray Tracing Python Framework for Radio Propagation*

2. *Suboptimal convergence* arises because heuristic methods often require thousands of forward evaluations and frequently become trapped in local optima without gradient guidance to point them toward the true minimum.
3. *Model inaccuracy* remains because simplified analytical models inherently miss multipath effects, diffraction, and specific blockages that are critical for high-frequency (millimeter-wave and sub-terahertz) communications.

To make this scaling issue explicit, let $F(\boldsymbol{\theta})$ be a scalar objective and let $M = \dim(\boldsymbol{\theta})$ denote the number of parameters:

- A *brute-force discretization* tests c grid points per variable, with computational complexity $\mathcal{O}(c^M)$.
- *Finite differences* require $\mathcal{O}(M)$ evaluations of the forward problem for a single gradient step, i.e.,

$$\frac{\partial F}{\partial \theta_i} \approx \frac{F(\boldsymbol{\theta} + \varepsilon \boldsymbol{e}_i) - F(\boldsymbol{\theta})}{\varepsilon}, \quad \boldsymbol{e}_i = \frac{\partial \boldsymbol{\theta}}{\partial \theta_i}, \quad i = 1, \dots, M, \tag{4.1}$$

  with overall cost $\mathcal{O}(M \cdot \mathrm{cost}(F))$ per optimization iteration. Furthermore, this approach is highly sensitive to the choice of step size ε and is inherently prone to numerical instability.
- Other *heuristics* typically require thousands to millions of objective evaluations to reach competitive solutions in non-convex multipath landscapes.

For large RIS designs, where M can reach thousands, finite-difference gradients are therefore often prohibitive even before accounting for truncation and round-off errors.

4.1.2 The Differentiable Advantage

Differentiable ray tracing addresses these limitations through three main properties:

- *Gradient information* provides exact analytical derivatives rather than finite-difference approximations, enabling directed search in parameter space.
- *Physical accuracy* is preserved because optimization remains coupled to full-wave simulation or high-fidelity asymptotic models.
- *Scalability* follows from first-order optimization, which is essential when handling thousands of design variables, for example in RIS optimization [8] or surrogate-model training [86, 87].

The core mathematical idea is straightforward. Once the gradient $\nabla_{\boldsymbol{\theta}} F$ is available, one can apply first-order optimization methods. The simplest case is gradient descent,

$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \alpha \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}^{(i)}), \tag{4.2}$$

where α is the learning rate. In practice, one often uses more advanced variants such as Adam [88] or problem-specific first-order methods.

For realistic ray tracing pipelines, manual derivation is generally impractical because the forward code is nonlinear and includes substantial control flow. A systematic and automated method is therefore required to compute exact derivatives together with the primal evaluation. This is the role of *automatic differentiation*.

4.2 Automatic Differentiation

Automatic differentiation (AD) [89, 90] is the computational technique that makes differentiable ray tracing practical. In contrast to numerical differentiation (finite differences), which suffers from truncation and round-off errors, and symbolic differentiation, which can cause severe expression growth, AD produces machine-precision derivatives with cost proportional to function evaluation. Modern frameworks such as PyTorch [35], TensorFlow [34], and JAX [36] provide optimized AD implementations and

[8]: An et al. (2025), *Emerging Technologies in Intelligent Metasurfaces: Shaping the Future of Wireless Communications*

[86]: Orekondy et al. (2023), *WiNeRT: Towards Neural Ray Tracing for Wireless Channel Modelling and Differentiable Simulations*

[87]: Yang et al. (2024), *R-NeRF: Neural Radiance Fields for Modeling RIS-enabled Wireless Environments*

[88]: Kingma et al. (2015), *Adam: A Method for Stochastic Optimization*

[89]: Griewank (1989), *On automatic differentiation*

[90]: Griewank et al. (2008), *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*

[35]: Ansel et al. (2024), *PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation*

[34]: Abadi et al. (2015), *TensorFlow, Large-scale machine learning on heterogeneous systems*

[36]: Frostig et al. (2018), *Compiling machine learning programs via high-level tracing*

have made these methods accessible in large-scale learning and simulation pipelines.

4.2.1 Mathematical Foundation

At its core, AD applies the chain rule to computer programs. Any programmatic function can be decomposed into a directed acyclic graph (DAG) of elementary operations (addition, multiplication, exponential, trigonometric functions, etc.).

For a composite function $f(g(x))$, the chain rule states

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\mathrm{d}f}{\mathrm{d}g}\frac{\mathrm{d}g}{\mathrm{d}x}. \tag{4.3}$$

For multiple variables, the multivariate chain rule gives

$$\frac{\partial f}{\partial x_i} = \sum_j \frac{\partial f}{\partial y_j}\frac{\partial y_j}{\partial x_i}, \tag{4.4}$$

where y_j are intermediate variables in the computational graph. AD computes these derivatives by traversing the graph either forward or backward.

For a function $\boldsymbol{f} : \mathbb{R}^M \to \mathbb{R}^Q$, let

$$\boldsymbol{J_f}(\boldsymbol{x}) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \in \mathbb{R}^{Q\times M}, \tag{4.5}$$

denote the Jacobian. Forward mode computes Jacobian-vector products (JVPs), whereas reverse mode computes vector-Jacobian products (VJPs). In radio optimization, one often has $Q = 1$ (for example, received power) and large M, which strongly favors reverse mode, as detailed below.

In the notation used throughout this chapter, M is the number of inputs, N is the total number of variables in the computational graph, and the variables $v_{M+1}, \ldots, v_N$ are the intermediate states and outputs produced after the inputs have been initialized.

4.2.2 Forward Mode

Forward mode AD propagates derivative information from inputs to outputs. It answers the question: how does a perturbation of one input affect all outputs?

Dual Numbers

Forward mode is naturally implemented with dual numbers, which extend real numbers with an infinitesimal part,

$$x + \epsilon \dot{x}, \tag{4.6}$$

where $\epsilon^2 = 0$ by definition, and $\dot{x} = \dfrac{\partial x}{\partial x_{in}}$ represents the derivative of x with respect to a chosen input variable x_{in}.

Arithmetic on dual numbers propagates derivatives together with primal values, following the chain rule. For example, for addition and multiplication, we have

$$(a + \epsilon \dot{a}) + (b + \epsilon \dot{b}) = (a + b) + \epsilon(\dot{a} + \dot{b}), \tag{4.7}$$

$$(a + \epsilon \dot{a}) \times (b + \epsilon \dot{b}) = ab + \epsilon(a\dot{b} + \dot{a}b). \tag{4.8}$$

Forward Mode Algorithm

Consider a function $\boldsymbol{f} : \mathbb{R}^M \to \mathbb{R}^Q$ evaluated as a sequence of N variables. The first M variables are inputs x_i, and the subsequent variables v_{M+1} to v_N are obtained through elementary operations ϕ_j. The final Q variables correspond to the outputs.

Let k_j be the number of arguments of operation ϕ_j, and let $p_l(j)$ denote the index of its l-th argument. Forward mode computes primal values v and directional derivatives (tangents) $\dot{v}$, as summarized in Algorithm 1.

Algorithm 1: Forward mode AD.

Input: Input variables $v_i \stackrel{\text{def}}{=} x_i$ for $i = 1, \dots, M$.
Output: Output variables and their derivatives with respect to the target input variable.

Initialize tangents: For $i = 1, \dots, M$, set $\dot{v}_i = 1$ for the target input variable, and 0 for all others.

for $j = M+1, \dots, N$ **do** `// Evaluate intermediate and output variables`

  $v_j = \phi_j(v_{p_1(j)}, \dots, v_{p_{k_j}(j)})$ `// Primals`

  $\dot{v}_j = \sum_{l=1}^{k_j} \frac{\partial \phi_j}{\partial v_{p_l(j)}} \dot{v}_{p_l(j)}$ `// Tangents`

return $(v_{N-Q+1}, \dots, v_N)$, $(\dot{v}_{N-Q+1}, \dots, \dot{v}_N)$

Forward Mode Example

Let us now consider the following two-output, two-input function

$$\boldsymbol{f}(x, y) = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix} = \begin{bmatrix} \cos(y)e^{x^2} + C \\ \sin(y)e^{x^2} + C \end{bmatrix}. \tag{4.9}$$

Figure 4.1 illustrates this procedure when differentiating with respect to x. The tangents are initialized as $\dot{x} = 1$ and $\dot{y} = 0$. Computation then proceeds from left to right, following the chain rule exactly. For example, for the intermediate variable $u = x^2$, we obtain $\dot{u} = 2x\dot{x}$. Because $\dot{y} = 0$, every derivative path that depends only on y vanishes (e.g., $\dot{w}_1 = -\dot{y}\sin(y) = 0$).

Computational Complexity

To recover the full gradient of a scalar objective, forward mode has complexity $\mathcal{O}(M \cdot \text{cost}(f))$, because one JVP pass provides derivatives for one seed direction at a time. It is therefore efficient for Jacobian columns when M is small, but costly in high-dimensional inverse design.

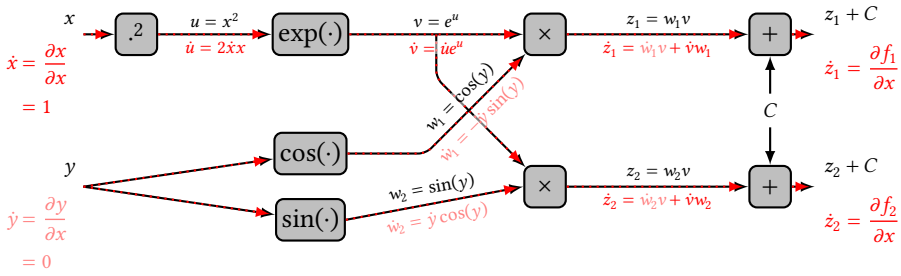


Figure 4.1: Example of forward mode automatic differentiation applied to (4.9). The derivative tracks the chain rule from left to right. Since we differentiate with respect to x, we initialize $\dot{x} = 1$ and $\dot{y} = 0$. Paths originating exclusively from y act as "symbolic zeros" (represented by faded red text), preventing unnecessary computation downstream.

4.2.3 Reverse Mode

Reverse mode AD (widely known as backpropagation [91]) propagates derivative information from outputs to inputs. Its objective is to answer the following question: how do all inputs influence one selected output?

[91]: Rumelhart et al. (1986), *Learning representations by back-propagating errors*

The Adjoint Method

Reverse mode computes "adjoints." For any intermediate variable v_j, its adjoint is the sensitivity of a selected scalar output f with respect to that variable,

$$\bar{v}_j = \frac{\partial f}{\partial v_j}. \tag{4.10}$$

Applying the chain rule from outputs to inputs yields the reverse mode recurrence

$$\bar{v}_i = \sum_{j \in \text{children}(i)} \bar{v}_j \frac{\partial v_j}{\partial v_i}. \tag{4.11}$$

Reverse Mode Algorithm

In reverse mode, we compute adjoints $\bar{v}_i = \frac{\partial y}{\partial v_i}$, representing the sensitivity of a chosen scalar output y to each intermediate variable v_i. Because local partial derivatives $\frac{\partial v_j}{\partial v_i}$ depend on primal intermediates, reverse mode uses two passes: a forward pass that records intermediates (the "tape"), followed by a backward pass.

Let args(j) denote the set of indices i of variables v_i used directly to compute v_j. The reverse pass then yields the gradient of the scalar output with respect to all inputs, as shown in Algorithm 2.

If the objective has several outputs, the same machinery can be applied one output at a time, or with a cotangent vector (representing output sensitivities) to compute a general vector–Jacobian product. In practice, this is the standard way to handle multi-output models in reverse mode without explicitly forming the full Jacobian matrix.

Algorithm 2: Reverse mode AD.

Input: Input variables x_i for $i = 1, \dots, M$.
Output: Gradient with respect to all M inputs.

Forward pass: Evaluate $\boldsymbol{f}$ from inputs to outputs (for $j = M+1, \dots, N$), storing intermediate values v_j in memory (the "tape").
Initialize adjoints: Set $\bar{v}_j = 0$ for all variables $j = 1, \dots, N$. Then, set $\bar{v}_k = 1$ for the target scalar output variable v_k.
for $j = N, N-1, \dots, M+1$ **do** `// Backward Pass`
  for $i \in args(j)$ **do** `// For each argument` v_i `of operation` j
    $\bar{v}_i \mathrel{+}= \bar{v}_j \frac{\partial v_j}{\partial v_i}$ `// Accumulate adjoints`
return $(\bar{v}_1, \dots, \bar{v}_M)$

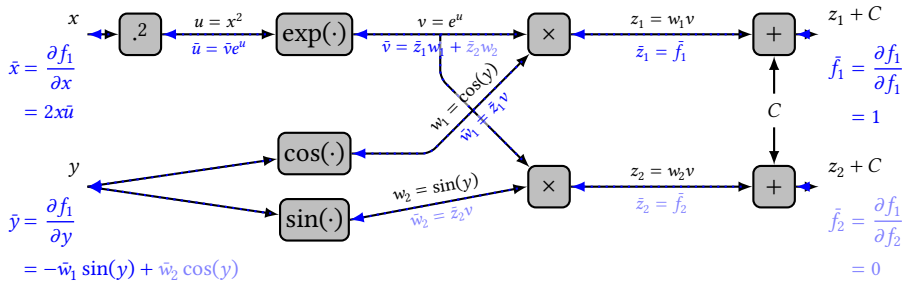


Figure 4.2: Example of reverse mode automatic differentiation applied to (4.9). Computation flows from right to left. By setting $\bar{f}_1 = 1$ and $\bar{f}_2 = 0$, the $\bar{z}_2$ branch becomes a "symbolic zero" (faded blue text), saving compute cycles on unrelated outputs.

Reverse Mode Example

Figure 4.2 shows reverse mode on the same example, computing the gradient of f_1 with respect to both x and y. Initialization sets $\bar{f}_1 = 1$ for the target output and $\bar{f}_2 = 0$ for the non-target output (faded blue text).

Computation proceeds from right to left. At the multiplication node $z_1 = w_1 v$, we obtain $\bar{w}_1 = \bar{z}_1 v$ and $\bar{v} = \bar{z}_1 w_1$. Since node v is shared by z_1 and z_2, its total adjoint is the sum of both contributions: $\bar{v} = \bar{z}_1 w_1 + \bar{z}_2 w_2$. Because $\bar{f}_2 = 0$, the $\bar{z}_2$ branch contributes nothing.

Computational Complexity and Memory Trade-offs

Reverse mode has complexity $\mathcal{O}(Q \cdot \mathrm{cost}(f))$ for Jacobian rows. In the common case $Q = 1$, one backward pass yields the full gradient over all M inputs through a VJP. This is the main reason reverse mode is preferred for high-dimensional radio-network optimization.

This computational advantage comes with a memory cost, because reverse mode stores forward intermediates on a tape. If a traced program has depth D and per-step state

size S, tape memory scales as

$$\mathcal{M}_{\text{tape}} = \mathcal{O}(DS). \tag{4.12}$$

In differentiable ray tracing, D increases with interaction depth, number of candidate paths, and solver iterations. As a result, memory often becomes the primary bottleneck rather than arithmetic throughput. This directly motivates the checkpointing and rematerialization strategies discussed in Chapter 5.

4.2.4 On "Differentiable" vs "Fully Differentiable"

It is useful to distinguish *mechanical differentiability* from *optimization-useful differentiability*. If an implementation uses AD-supported primitives, frameworks can usually execute a backward pass without runtime errors. This mechanical property alone does not guarantee informative gradients.

Ray tracing includes discrete predicates (visibility, intersection validity, finite object extent) that behave as multi-dimensional step functions. For instance, triangle intersection tests rely on inequalities over barycentric coordinates. These predicates induce piecewise-constant regions where received power does not change under small perturbations, so gradients vanish almost everywhere away from boundaries. Consequently, a mechanically differentiable program can remain ineffective for gradient-based optimization unless discontinuities are regularized. This issue is discussed in Section 4.4.

4.3 Building a Differentiable Ray Tracing Pipeline

In modern machine learning stacks, building a differentiable ray tracing pipeline from scratch is often straight-

forward. With Python, operator overloading, and mature AD systems, many derivatives are obtained automatically once the forward model is expressed with differentiable primitives.

The main challenge is therefore not merely to make code differentiable, but to obtain gradients that are numerically stable, physically meaningful, and computationally efficient at scale. In practice, this shifts attention to implementation constraints that strongly influence architecture choices.

4.3.1 New Code vs Legacy Tools

For a new codebase, differentiability can be introduced from the beginning: scene parameters are represented as tensors, geometric operations are written with AD-compatible operators, and gradients are obtained through JVPs or VJPs with little manual intervention. Listings 4.1 to 4.3 illustrate this on a simple JAX example for both gradient and Jacobian evaluation.

Retrofitting an existing ray tracing program is usually harder and can be undesirable from a performance perspective. Legacy engines often combine heterogeneous components (e.g., C/C++ kernels, custom acceleration structures, external solvers, or MATLAB post-processing) that are opaque to AD. Wrapping these components restores forward execution, but not necessarily gradients; correct derivatives then require custom backward rules and careful validation, with additional maintenance and runtime overhead.

This is why recent differentiable ray tracing frameworks often re-implement key kernels in AD-first environments rather than differentiating a pre-existing production simulator end-to-end.

```python
import jax
import jax.numpy as jnp
from jax import Array


# Some function to differentiate
def f(xy: Array, c: float) -> Array:
  x, y = xy
  exp = jnp.exp(x * x)
  f1 = jnp.cos(y) * exp + c
  f2 = jnp.sin(y) * exp + c
  return jnp.array([f1, f2])


xy = jnp.array([1.0, -jnp.pi / 4])
c = 1.0  # constant parameter
print(f(xy, c))
# [ 2.9221153  -0.92211545]
```

Listing 4.1: Various ways to differentiate the example function (4.9) with JAX: function definition. The x and y inputs are represented as a single JAX array, as JAX supports differentiating with respect to array arguments, and the function itself is defined using JAX primitives whose syntax is similar to standard NumPy operations.

```python
# Gradient (w.r.t. xy)
f1 = lambda xy, c: f(xy, c)[0]
print(jax.grad(f1)(xy, c))
# [ 3.844231   1.9221154]
f2 = lambda xy, c: f(xy, c)[1]
print(jax.grad(f2)(xy, c))
# [-3.844231   1.9221154]
```

Listing 4.2: Various ways to differentiate the example function (4.9) with JAX: the `jax.grad` function computes the gradient of a scalar output with respect to all inputs.

```python
# Jacobian matrix (w.r.t. xy)
dfdx, dfdy = jax.jacobian(f)(xy, c).T
print(dfdx)
# [ 3.844231   -3.844231]
print(dfdy)
# [-1.9221154   1.9221154]
```

Listing 4.3: Various ways to differentiate the example function (4.9) with JAX: the `jax.jacobian` function computes the full Jacobian matrix.

4.3.2 Framework-Level Constraints for Differentiable Implementations

Even in newly designed implementations, framework constraints determine what is practical. Each AD framework has its own tracing rules, supported primitives, and performance trade-offs. Understanding these rules helps avoid runtime errors, slow gradients, and excessive memory usage.

JAX is a representative example. Its syntax is NumPy-like, but it imposes stricter constraints on shapes and control flow than many users initially expect. These constraints are especially important in reverse-mode AD, because the backward pass requires access to forward intermediates.

Conceptually, JAX first *traces* a function to build a symbolic computational graph (a JAX expression). This is the program-level counterpart of Figure 4.2: build the forward graph first, then traverse it backward to propagate adjoints. After tracing, JAX may send the graph to the Accelerated Linear Algebra (XLA) compiler to optimize and generate kernels for CPUs, GPUs, or tensor processing units (TPUs). Although optional, this compilation step is often essential for performance and therefore imposes additional structure on code and control flow.

The first issue concerns loops. With regular Python `for` or `while` loops inside compiled code, JAX may unroll iterations when bounds are static, increasing graph size and compilation time. In reverse mode, this also increases memory pressure because many per-iteration intermediates are required by the backward pass.

For iterative operations, such as recursion in the image method, a better formulation is a carry update, e.g.,

$$f(i, \text{carry}) \to \text{carry}', \tag{4.13}$$

implemented with `jax.lax.fori_loop` or `jax.lax.scan` (see Listing 4.4). These primitives keep control flow inside the traced graph, reduce Python overhead, and generally

improve scaling in reverse mode because the computation structure is explicit.

The second issue concerns branching. If the predicate is independent of traced variables, a Python `if` is usually valid because only one branch matters at trace time. If the predicate depends on traced variables, branch selection must be implemented using `jax.lax.cond` (see Listing 4.5). A common alternative is `jnp.where`, but it evaluates both branches elementwise, which can propagate invalid values (e.g., NaNs) into reverse-mode derivatives even when one branch is logically inactive.

Moreover, `jax.lax.while_loop` has an important limitation: in general, it is not reverse-mode differentiable because XLA requires static memory bounds, which conflict with potentially unbounded dynamic loops. When reverse-mode gradients are needed, bounded formulations based on `fori_loop` or `scan` are typically preferred.

Finally, JAX's compilation model imposes static-shape requirements, which can be challenging in ray tracing where the number of candidate paths and interactions varies dynamically. A common remedy is to use fixed-size tensors with activity masks, preserving static shapes while representing variable activity. Another option is to exclude dynamic parts from compilation and compile only selected subroutines, but this can degrade performance and increase memory use because fewer global optimizations are available.

Listing 4.4: JAX's alternative to the Python `for` loop.

```
c = 0.0
for i in range(
  10
):
  c += f(i)


c = jax.lax.scan(
  lambda i, c: (
    c + f(i)
  ),
  0.0,  # Carry
  length=10,
)
```

Listing 4.5: JAX's alternative to the Python `if` statement.

```
if x < 0:
  x = a
else:
  x = b

x = (
  jax.lax.cond(
    x < 0,
    lambda: a,
    lambda: b,
  )
)
```

4.3.3 Non-Differentiable Operations and Custom Gradients

Not every operation in a ray tracing pipeline is differentiable by default. Typical examples include discrete visibility predicates, indexing-heavy lookup tables, hard pruning decisions, and external kernels that appear as black-box calls to the AD engine. To bridge this gap, hybrid

simulation pipelines can be used. For instance, a "semi-differentiable" ray tracing simulation with point clouds was proposed by Vaara *et al.* [92], who combined differentiable ray tracing software (Sionna RT [37]) for computing electromagnetic fields with a non-differentiable engine (NimbusRT [93]) to trace the paths. In this setup, gradients with respect to object positions are unavailable because the ray paths are obtained from the non-differentiable solver.

[92]: Vaara et al. (2026), *Differentiable High-Performance Ray Tracing-Based Simulation of Radio Propagation With Point Clouds*

[37]: Hoydis et al. (2023), *Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling*

[93]: Vaara et al. (2025), *Ray Launching-Based Computation of Exact Paths With Noisy Dense Point Clouds*

When such operations are unavoidable, one must either (i) treat them as non-differentiable and stop gradients, or (ii) provide explicit custom derivative rules (e.g., custom VJP or JVP) consistent with the underlying physics. This choice is critical: incorrect custom gradients can drive optimization toward physically meaningless solutions even when the numerical objective decreases.

4.3.4 Example: Path Optimization and Implicit Differentiation

Custom gradients are particularly useful when the forward model contains an iterative solver [55]. In the Fermat-based path optimization problem of Chapter 3, the ray path is obtained by minimizing a convex objective $L(\boldsymbol{T}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes scene or transceiver parameters (e.g., geometry or antenna positions). The solver updates $\boldsymbol{T}$ until convergence, and many iterations may be required to reach $\boldsymbol{T}^\star(\boldsymbol{\theta})$.

[55]: Eertmans et al. (2025), *Fast, Differentiable, GPU-Accelerated Ray Tracing for Multiple Diffraction and Reflection Paths*

If this procedure is differentiated naively, reverse mode must backpropagate through all iterations. In practice, this leads to substantial compute and memory overhead, because the optimization trajectory must be traversed in both directions. A standard alternative is to apply the implicit function theorem [94, Thm. 7.6], so that gradients of $\boldsymbol{T}^\star$ are obtained from optimality conditions instead of full unrolling.

[94]: Apostol (1957), *Mathematical analysis: a modern approach to advanced calculus*

At convergence, the optimal path $\boldsymbol{T}^\star(\boldsymbol{\theta})$ satisfies

$$\nabla_{\boldsymbol{T}} L(\boldsymbol{T}^\star(\boldsymbol{\theta}); \boldsymbol{\theta}) = \mathbf{0}, \tag{4.14}$$

and total differentiation with respect to $\boldsymbol{\theta}$ gives

$$\frac{\partial}{\partial \boldsymbol{\theta}} \nabla_{\boldsymbol{T}} L(\boldsymbol{T}^\star(\boldsymbol{\theta}); \boldsymbol{\theta}) + \frac{\partial \nabla_{\boldsymbol{T}} L}{\partial \boldsymbol{T}} \frac{\partial \boldsymbol{T}^\star}{\partial \boldsymbol{\theta}} = \mathbf{0}. \tag{4.15}$$

Therefore,

$$\frac{\partial \boldsymbol{T}^\star}{\partial \boldsymbol{\theta}} = -\left[\frac{\partial \nabla_{\boldsymbol{T}} L}{\partial \boldsymbol{T}}\right]^{-1} \frac{\partial}{\partial \boldsymbol{\theta}} \nabla_{\boldsymbol{T}} L. \tag{4.16}$$

In reverse-mode AD, vector–Jacobian products of the form $\boldsymbol{v}^\top \frac{\partial \boldsymbol{T}^\star}{\partial \boldsymbol{\theta}}$ are typically required. These can be computed efficiently through the linear system

$$\boldsymbol{u}^\top = -\boldsymbol{v}^\top \left[\frac{\partial \nabla_{\boldsymbol{T}} L}{\partial \boldsymbol{T}}\right]^{-1}, \tag{4.17}$$

or equivalently

$$\left[\frac{\partial \nabla_{\boldsymbol{T}} L}{\partial \boldsymbol{T}}\right]^\top \boldsymbol{u} = -\boldsymbol{v}, \tag{4.18}$$

where $\frac{\partial \nabla_{\boldsymbol{T}} L}{\partial \boldsymbol{T}}$ is the Hessian of the convex function L and is therefore symmetric positive semidefinite. This yields

$$\boldsymbol{v}^\top \frac{\partial \boldsymbol{T}^\star}{\partial \boldsymbol{\theta}} = \boldsymbol{u}^\top \frac{\partial}{\partial \boldsymbol{\theta}} \nabla_{\boldsymbol{T}} L. \tag{4.19}$$

This approach still relies on AD for the remaining Jacobian–vector terms, but this is not a practical limitation and can be further optimized, as discussed in Chapter 5. Listing 4.6 illustrates this implicit-differentiation strategy in JAX.

While implicit differentiation is a highly efficient technique for computing gradients, it cannot be applied in all cases. First, we assume that the forward model can be expressed as the solution to an optimization problem, which is not the case, or at least not straightforward, in many applications. Second, we must assume convergence (the local optimality condition), which is also not always easy to guarantee, much like in a machine learning model or an

```
# Return path length
def L(T: Array, theta: Array) -> Array: ...


# Return optimal path T_opt
@partial(
  jax.custom_vjp,
  nondiff_argnames=("static_params",),
)
def f(
  theta: Array, *static_params
) -> Array: ...


# Compute optimal path and any intermediate
# values needed for the backward pass
def f_fwd(
  theta: Array, *static_params
) -> Array:
  T_opt = f(theta, *static_params)
  return T_opt, (T_opt, theta)
```

Listing 4.6-I: Example of implicit differentiation for the Fermat-based path tracing method in its convex setting. The path length function `L` is defined as in (3.22), where $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{B})$, and the optimal path $\boldsymbol{T}^\star$ is returned by the `f` function. Static parameters may include the number of iterations, convergence thresholds, or solver hyperparameters; however, they are assumed not to impact the final solution.

iterative solver that has not perfectly converged. When these assumptions are violated, or when the linear system for the backward solve becomes poorly conditioned near grazing or topological-transition regimes, the implicit formulation can become unstable or fail to represent the intended physical branch, making a fallback to explicit differentiation necessary.

```python
# Compute backward pass using intermediate
# values from the forward pass
def f_bwd(
  *static_params,
  fwd_res: tuple[Array, Array],
  cotangent: Array,
) -> tuple[Array]:
  # Convergence should not depend on static
  # parameters, so we can ignore them here
  del static_params
  # Output of the forward pass
  T_opt, theta = fwd_res

  # Gradient w.r.t. T
  def grad_T(T: Array, theta: Array) -> Array:
    return jax.grad(L)(T, theta)

  # Hessian A at (T_opt, theta)
  v = cotangent
  A = jax.hessian(L)(T_opt, theta)
  # Hessian is a positive semi-definite matrix
  u = jax.scipy.linalg.solve(
    A, -v, assume_a="pos"
  )
  # v^T dT*/dtheta = u^T ∂(grad_T L)/∂theta
  _, vjp_fun_theta = jax.vjp(
    lambda theta: grad_T(T_opt, theta), theta
  )
  # Compute gradient w.r.t. theta
  return vjp_fun_theta(u)


# Register custom VJP for f
f.defvjp(f_fwd, f_bwd)
```

Listing 4.6-II: Example of implicit differentiation for the Fermat-based path tracing method in its convex setting (continued). First, the static parameters are ignored to emphasize that the optimal path does not depend on these parameters. Next, the results from the forward pass are unpacked. The gradient of the objective is obtained with `jax.grad`, and the Hessian is obtained with `jax.hessian`. Finally, a linear solve is performed to compute the vector–Jacobian product without unrolling the optimization trajectory. The Hessian's positive semi-definiteness is leveraged to employ a more efficient linear solver.

4.4 Resolving Geometric Discontinuities and Topological Transitions

The shift to differentiable ray tracing enables gradient-based optimization for diverse radio network applications, ranging from high-fidelity radio map calibration [95] to physical-world adversarial attacks via optimized object placement [96]. However, this paradigm exposes a critical structural flaw: the omnipresence of geometric discontinuities and discrete topological transitions.

A fundamental challenge in differentiable ray tracing is the treatment of these discontinuities caused by visibility events and discrete path-topology changes [97, 98]. These discontinuities can hinder gradient-based optimization by creating zero-gradient regions in which no descent direction is available.

[95]: Castro Farfan (2025), *Towards high-fidelity radio coverage maps: calibration using real-world measurements in Sionna RT*

[96]: Han et al. (2025), *Loki: Physical-World Adversarial Attacks on Wireless Indoor Localization via Differentiable Object Placement*

[97]: Fischer et al. (2023), *Plateau-Reduced Differentiable Path Tracing*

[98]: Eertmans et al. (2024), *Fully Differentiable Ray Tracing via Discontinuity Smoothing for Radio Network Optimization*

4.4.1 Sources of Discontinuities

Because ray tracing relies on discrete geometric decisions—for example, whether an object occludes a ray or a triangle intersection test verifies barycentric coordinates—discontinuities in the objective function are common. The resulting objective is piecewise and often locally constant, with limited smooth transitions.

These discontinuities appear whenever branching occurs in the ray tracing process. For example, when a ray transitions from being visible to being occluded, the received power can drop sharply, creating a discontinuity. Similarly, when the path topology changes (e.g., a new reflection or diffraction path appears or disappears), the objective can experience sudden jumps. Surface transitions, such as moving from one material to another, can also introduce discontinuities in the received signal.

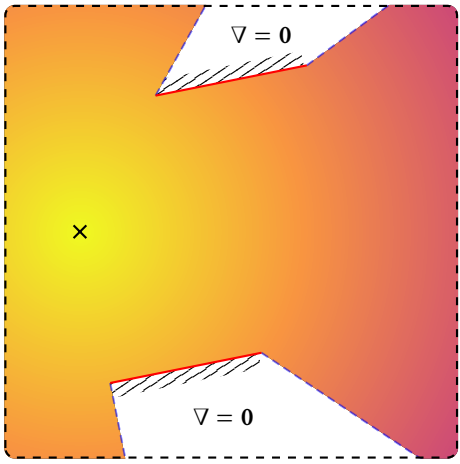


Figure 4.3: Example of discontinuities in ray traced power maps. The color gradient represents received power (dB), where shadow boundaries (dashed blue lines) create sharp discontinuities. These transitions pose challenges for gradient-based optimization. The white regions indicate "deep shadows," as using a finite number of simulated paths eventually results in zero-gradient regions.

Another source of discontinuity is the finite number of simulated paths. In practice, ray tracing algorithms often simulate a limited number of paths due to computational constraints. This can lead to zero-gradient regions in deep shadows, where no paths are simulated and the objective remains constant across a wide neighborhood. This is illustrated in Figure 4.3, where the white regions represent deep shadows with zero gradients.

These discontinuities appear as zero-gradient regions—"plateaus"—or sudden jumps, both of which obstruct gradient-based optimization. A typical visibility predicate can be modeled as a Heaviside step function $H(x)$ over a signed geometric residual x; consequently, the objective changes abruptly when crossing geometric boundaries.

At non-smooth points, the classical gradient may be undefined. In practice, AD frameworks will use a *subgradient*, i.e., a generalized local slope used as a replacement direction for descent, in order to avoid runtime errors [99].

[99]: Bubeck (2015), *Convex Optimization: Algorithms and Complexity*

This helps at some specific points, but it does not solve the deep-shadow problem. When the objective is locally constant over a finite region, the subgradient is still zero, so moving a transmitter within that plateau provides no descent direction. As a result, naive gradient-based meth-

ods can stall until additional smoothing or exploration is introduced.

Three families of methods are commonly used to address this non-differentiability:

- *Stochastic smoothing and plateau reduction* methods, such as Gaussian smoothing, convolve the objective function or the forward operator globally with a continuous kernel to recover nonzero descent directions [97]. Recent wireless localization frameworks apply this idea by convolving the signal-generation process with a Gaussian kernel to overcome sparse gradients [100]. While effective at eliminating plateaus, these methods generally require repetitive Monte Carlo sampling that scales poorly in dense multipath scenes.
- *Edge-sampling* methods sample rays directly on geometric boundaries to estimate derivative contributions from edges with high accuracy [33]. However, because they are designed to differentiate area integrals over pixel footprints in computer graphics rendering, they are fundamentally incompatible with the deterministic point-to-point path-finding setup used in radio propagation.
- *Analytical smoothing (physics-based)* methods replace hard visibility predicates with continuous surrogates at the lowest physical level of the ray tracing engine [98]. Similar techniques have recently been applied to the SBR method for solving inverse scattering problems on complex 3D targets [101]. They are highly practical for radio propagation optimization because they resolve the exact source of the discontinuity and vectorize efficiently on modern accelerators.

[97]: Fischer et al. (2023), *Plateau-Reduced Differentiable Path Tracing*

[100]: Han et al. (2025), *RayLoc: Wireless Indoor Localization via Fully Differentiable Ray-tracing*

[33]: Li et al. (2018), *Differentiable Monte Carlo ray tracing through edge sampling*

[98]: Eertmans et al. (2024), *Fully Differentiable Ray Tracing via Discontinuity Smoothing for Radio Network Optimization*

[101]: Liu et al. (2025), *Fully Differentiable Shooting and Bouncing Ray Method for Solving Inverse Scattering Problems: 3-D PEC Targets*

4.4.2 Discontinuity Smoothing via Approximation Functions

To mitigate abrupt visibility transitions without abandoning exact path formulations, a smoothing function $s(x;\alpha)$ can be introduced to approximate the underlying discrete behavior [98]. Let the visibility or intersection condition be represented by the unit step function $\theta(x)$, where $\theta(x) = 1$ if $x > 0$ and 0 otherwise.

[98]: Eertmans et al. (2024), *Fully Differentiable Ray Tracing via Discontinuity Smoothing for Radio Network Optimization*

The approximation maps a real-valued residual to a continuous value in $[0, 1]$ and satisfies

$$\lim_{\alpha\to\infty} s(x;\alpha) = \theta(x), \tag{4.20}$$

where $\alpha \in \mathbb{R}^+$ controls steepness. In practice, smoothing is often parameterized as

$$s(x;\alpha) = s(\alpha x), \tag{4.21}$$

which directly controls transition width through α.

To maintain reliable optimization behavior, the smoothing function should satisfy the following criteria:

C1) The smoothing map must remain bounded in the unit interval to preserve consistency with Boolean logic,

$$s : \mathbb{R} \to [0, 1]. \tag{4.22}$$

C2) The asymptotic behavior must match the hard predicate so that the smoothing function converges to the original decision rule,

$$\lim_{x\to-\infty} s(x) = 0 \qquad \text{and} \qquad \lim_{x\to+\infty} s(x) = 1. \tag{4.23}$$

C3) The transition must be strictly monotonically increasing to support stable gradient flow and avoid artificial local minima,

$$s(x) \text{ is strictly monotonically increasing.} \tag{4.24}$$

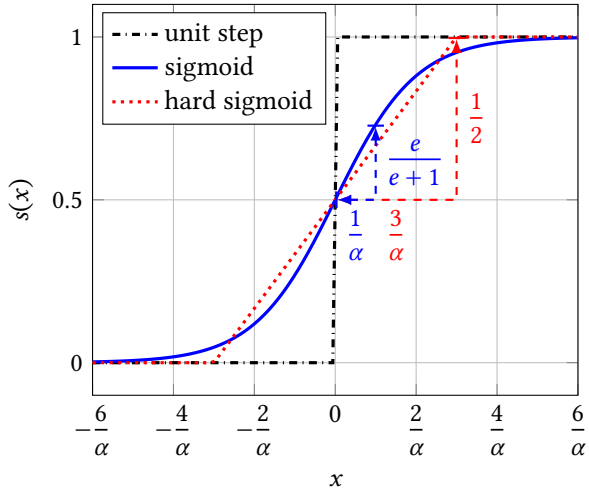


Figure 4.4: Examples of smoothing functions that satisfy the criteria C1–C5. The parameter α controls the transition steepness, with larger values providing a sharper approximation to the step function.

C4) The boundary should be centered so that both sides are treated symmetrically at the discontinuity,

$$s(0) = \frac{1}{2}. \tag{4.25}$$

C5) The profile should be balanced around the boundary to enforce odd symmetry with respect to the centered value,

$$s(x) - s(0) = s(0) - s(-x). \tag{4.26}$$

Examples of Smoothing Functions

Borrowing from machine learning activation functions, two common families satisfy these properties. The first is the parameterized *sigmoid*, defined as

$$\text{sigmoid}(x;\alpha) = \frac{1}{1+e^{-\alpha x}}. \tag{4.27}$$

As $\alpha \to \infty$, $\text{sigmoid}(x;\alpha) \to \theta(x)$. The sigmoid is infinitely differentiable and smooth everywhere, but exponential evaluations can be relatively costly. A faster alternative is

the *hard sigmoid*, a piecewise-linear function, given by

$$\text{hard sigmoid}(x;\alpha) = \frac{\min(\max(0, \alpha x + 3), 6)}{6}. \tag{4.28}$$

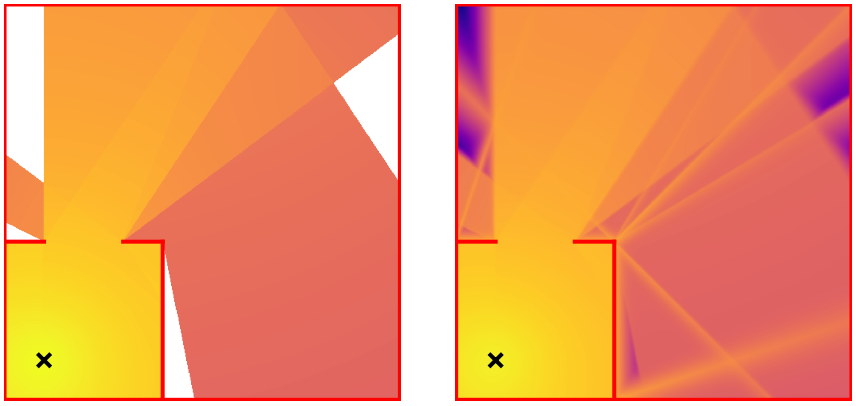
The hard sigmoid is faster to compute but has discontinuous derivatives at $x = \pm 3/\alpha$, bounding its gradient support. The choice depends on the desired trade-off between global smoothness and evaluation cost.

A comparative plot of sigmoid and hard-sigmoid approximations for several α values is reported in Figure 4.4, highlighting the trade-off between transition sharpness and gradient support. Figures 4.5 and 4.6 illustrate the impact of smoothing on ray traced power maps and their gradients, showing reduced plateaus and smoother transitions near shadow boundaries. An important downside of smoothing is that it can introduce leakage through obstacles, which may bias optimization if not properly controlled. These surrogates are therefore best interpreted as numerical regularizations, although they loosely mimic physically smooth effects such as penetration loss, surface roughness, and diffraction-induced transition regions. When those mechanisms matter quantitatively, they should be modeled explicitly rather than absorbed into the smoothing function.

Continuation Strategy for Optimization

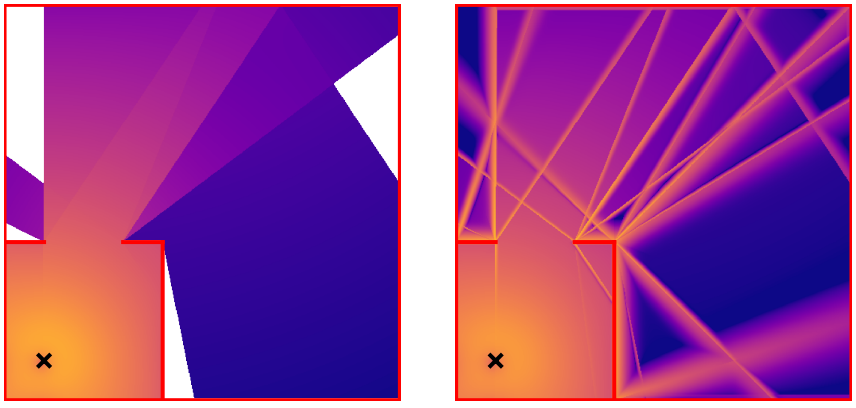
Using a fixed, aggressive smoothing level can bias the final solution. If α remains too small, obstacles become unrealistically translucent and the optimizer may drift away from the physically correct optimum. A practical strategy is therefore continuation: start with strong smoothing (small α) to remove plateaus and obtain informative, long-range gradients, then gradually increase α so that the model hardens and approaches the physical discontinuous regime. A convenient schedule over N optimization steps is

$$\alpha^{(i)} = \alpha_0 \left(\frac{\alpha_f}{\alpha_0}\right)^{\frac{i}{N-1}}, \qquad i = 0, \dots, N-1, \tag{4.29}$$



(a) Exact power map. **(b)** Approximate power map with smoothing.

Figure 4.5: Smoothing effects on the power map (dB) using the sigmoid function with $\alpha = 50$.



(a) Exact gradient map. **(b)** Approximate gradient map with smoothing.

Figure 4.6: Smoothing effects on the gradient of the power shown in Figure 4.5 (log-scale of the gradient norm).

with $\alpha_0 \ll \alpha_f$. This geometric progression provides a smooth transition from global exploration to physically faithful refinement.

Example: Optimizing Transmitter Position

To illustrate the practical effect of smoothing, we consider a simple optimization scenario adapted from [98]. The scene contains a single obstacle, two fixed receivers RX_1 and RX_2, and one transmitter with planar coordinates (x, y) to optimize. To keep the setting minimal and interpretable, only line-of-sight paths are simulated.

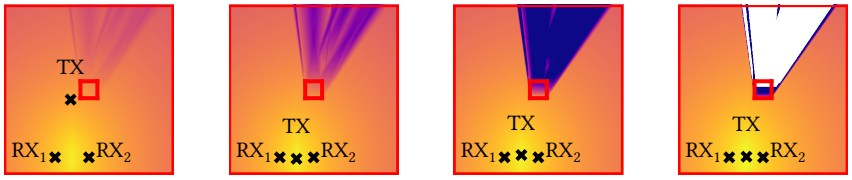
[98]: Eertmans et al. (2024), *Fully Differentiable Ray Tracing via Discontinuity Smoothing for Radio Network Optimization*

The objective is to maximize the weakest received power between the two receivers,

$$F(x, y) = \min\left(P^{\mathrm{RX}_1}(x, y),\ P^{\mathrm{RX}_2}(x, y)\right), \tag{4.30}$$

where $P^{\mathrm{RX}_i}(x, y)$ denotes the received power at receiver i for a transmitter placed at (x, y). This max–min formulation avoids favoring one receiver at the expense of the other and is therefore well suited to coverage-oriented placement.

The transmitter is initialized in a deep shadow region where the exact objective is nearly flat and gradients are uninformative. We then run gradient ascent with the continuation schedule introduced above, starting from strong smoothing ($\alpha_0 = 1$) to recover usable ascent directions and progressively increasing α to approach the physically accurate objective ($\alpha_f = 100$).



(a) After 25 iterations. **(b)** After 50 iterations. **(c)** After 75 iterations. **(d)** After 100 iterations.

Figure 4.7: Optimization trajectory for transmitter placement under the max–min objective F. The four panels show intermediate iterates (25, 50, 75, and 100 iterations). As optimization proceeds, the transmitter exits the initial shadowed region and moves toward a location that improves the worst served receiver. The corresponding smoothing values increase geometrically, illustrating the continuation strategy from smooth exploration to sharper, physically faithful refinement.

4.4.3 Continuous Ray Tracing Framework

Implementing physics-based analytical smoothing fundamentally transforms the discrete ray tracing pipeline into a continuous surrogate process entirely suitable for end-to-end gradient-based optimization.

Continuous Validity Check

A path-validity check is naturally Boolean; for example, $x_0 > x_1$ determines whether a ray clears an obstacle edge. Replacing this hard condition with the smooth surrogate yields $s(x_0 - x_1)$. The validity measure $V(\boldsymbol{X})$ for a candidate ray path $\boldsymbol{X}$ then becomes continuous in $[0, 1]$.

This smoothing enables depth-wise attenuation through walls and edge softening for reflections and occlusions, seamlessly translating geometric properties into continuous path-validity scores. The simulated received electric field adapts directly to this continuum,

$$\mathbf{E}^{\mathrm{RX}} = \sum_{p \in \mathscr{P}} V(\boldsymbol{X}) \tilde{\mathbf{E}}_p^{\mathrm{RX}}, \tag{4.31}$$

where $\mathscr{P}$ is the set of all potential path candidates and $\tilde{\mathbf{E}}_p^{\mathrm{RX}}$ is the electric field contribution from path p assuming no obstacles.

The product of validity scores along a path candidate naturally captures the cumulative effect of multiple interactions, so that paths with more interactions are attenuated more strongly when they approach occlusion. This continuous formulation also allows for smooth transitions between different path topologies, as the validity scores can vary continuously even when the underlying geometric configuration changes discretely. It forms a sort of "confidence score" for each path candidate, which can be used to guide optimization and path selection.

Continuous Path Finding

Beyond line-of-sight visibility tests, exact path finding typically relies on a post-processing check to verify (i) convergence and (ii) that ray path coordinates are well within the object boundaries. As with Boolean visibility gates, replacing these binary convergence predicates with $s(x;\alpha)$ makes the solver behavior fully continuous with respect to the underlying mesh geometry.

This smoothing-based reformulation remains compatible with exact path-finding backends while replacing Boolean gates with differentiable surrogates. In practical network optimization, this allows gradient descent to continue progressing smoothly even when the current iterate is initially placed inside a deeply occluded region that would otherwise yield exactly zero gradients.

4.5 Conclusion

This chapter established the main ingredients required before large-scale implementation: inverse-design motivation, reverse-mode AD, scene parameterization, framework-level differentiation constraints, differentiable path-coordinate recovery, and discontinuity regularization.

These ingredients are necessary but not sufficient for practical deployment. In realistic environments, the dominant

obstacle is computational scale: combinatorial growth of path candidates, expensive intersection workloads, and reverse-mode tape-memory growth.

Chapter 5 therefore focuses on tractability mechanisms for path generation and validation, including pruning, acceleration structures, hardware execution models, and memory-management strategies that make differentiable ray tracing viable in large scenes.

Efficient Path Tracing | 5

The differentiable ray tracing methods introduced in Chapter 4 provide a practical route to gradient-based optimization, but their usefulness in realistic scenarios depends on computational efficiency. In wireless applications—from network planning to beam-management optimization—simulations must handle large, geometrically complex environments under strict runtime constraints.

Moving from classical deterministic propagation models to differentiable, accelerator-based pipelines introduces additional computational requirements. The cost of ray tracing already depends on the number of rays, geometric complexity, interaction depth, and accuracy targets. Differentiability adds the overhead of gradient computation and the memory cost of storing intermediate states for reverse-mode differentiation.

This chapter studies tractability mechanisms across the full simulation pipeline: complexity reduction, visibility-based pruning, acceleration structures, hardware execution, and memory-aware differentiation. The objective is to make differentiable ray tracing usable in large site-specific digital-twin workflows.

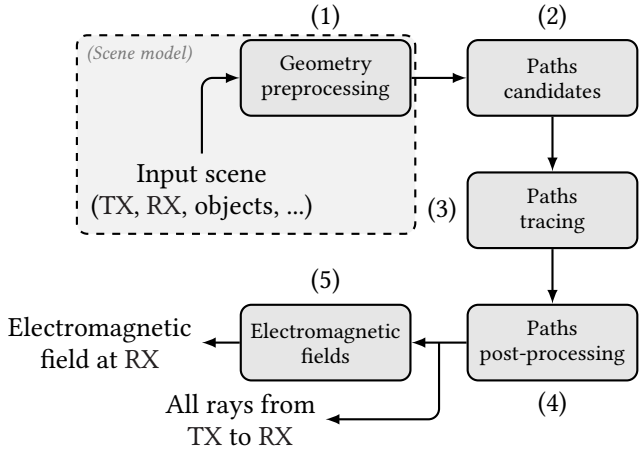


Figure 5.1: High-level view of the typical blocks in modern ray tracers. Some blocks can be subdivided into smaller tasks (e.g., geometry construction), whereas others can be processed jointly to improve performance (e.g., path finding and electromagnetic field computation).

5.1 Scene Representation in Modern Ray Tracing Pipelines

Although scene representation is not directly related to efficient path tracing, it affects all downstream methods. A fundamental prerequisite for any deterministic simulation is the acquisition of an accurate digital environmental model.

For outdoor scenarios, publicly available data such as OpenStreetMap (OSM) Buildings [102] represents building footprints as polygons. When extruded to building height, these form polyhedra that can be perfectly approximated by triangular facets. Most buildings follow this standardized representation; advanced models for specific buildings are exceptions. Similarly, the ground is typically modeled as a planar surface, though elevation data from sources such as NASA [103] enables varying-height terrain representation through triangular facets. Overall, such maps typically have meter-level accuracy. Except in specialized studies, details such as foliage, facade balconies, and glass windows are typically omitted.

[102]: OSM Buildings Contributors (2026), *OSM Buildings: A free and open project for 3D buildings*

[103]: NASA Earthdata (2026), *Digital Elevation/Terrain Model (DEM)*

Indoor environments are commonly modeled using 3D soft-

ware such as Blender. While Blender supports outdoor scenarios, hand-crafted modeling is not scalable. Point clouds offer a practical alternative for modeling new environments rapidly. Point clouds can be acquired inexpensively using LiDAR and similar devices. However, ray tracing requires surface or edge interactions, not point interactions, so point clouds must be preprocessed and transformed into surface elements. For specular reflections, a common approach treats each point as a potential reflection site and verifies that ray paths fall within the first Fresnel zone defined by the transmitter and receiver positions [93]. For diffraction, edges extracted from the point cloud serve as diffracting elements.

[93]: Vaara et al. (2025), *Ray Launching-Based Computation of Exact Paths With Noisy Dense Point Clouds*

Beyond geometric representation, ray tracing requires accurate material properties to compute reflection coefficients, transmission losses, and diffraction. Material characterization is a significant practical challenge, as properties depend on frequency, incidence angle, and material composition.

Common sources for material properties include:

- *Material databases* with pre-measured properties for standard construction materials (concrete, brick, glass, metal, asphalt) at common frequencies [11, Tab. 3].
- *Manufacturer specifications* providing dielectric loss and permittivity data from building material suppliers.
- *In-situ measurements* enabling empirical characterization via transmission or reflection measurements at deployment frequencies.
- *Literature values* from published measurements in prior propagation studies in similar environments.

[11]: International Telecommunication Union (2025), *ITU-R Recommendation P.2040-3: Effects of building materials and structures on radiowave propagation above about 100 MHz*

In practice, material uncertainty is often significant; properties can vary substantially within a single building material category depending on composition, age, and moisture content. For this reason, many studies either employ conservative (high-loss) material assumptions or conduct sensitivity analysis over material parameter ranges. When

constructing digital twins for network optimization, calibration against in-situ measurements is necessary to reduce uncertainty and improve prediction accuracy.

In summary, most ray tracing tools assume environments are represented by triangular meshes with material-specific reflection and transmission models. Consequently, advanced techniques such as MPT are typically impractical for general-purpose scenarios and are reserved for specialized studies with detailed environmental descriptions.

5.2 The Challenge of Exponential Path Generation

In this section, we examine why exact point-to-point ray tracing becomes difficult to scale and why acceleration is essential in realistic scenes. In deterministic propagation modeling, we seek geometric paths between a transmitter and a receiver that satisfy reflection, transmission, and diffraction constraints.

The core challenge is not the recovery of one feasible path, which can often be done efficiently with the image method or Fermat-based formulations (see Chapter 4), but the growth of the candidate search space as scene complexity and interaction order increase. For a scene with N primitives and a maximum interaction order K, the number of possible interaction sequences scales as $\mathcal{O}(N^K)$, making exhaustive search rapidly impractical. In the next subsection, we make this growth explicit through the image-tree viewpoint.

5.2.1 The Image Tree and Candidate Generation

As discussed in Section 3.1.4, candidate-path generation can be represented as a tree (see Figure 5.2) whose size

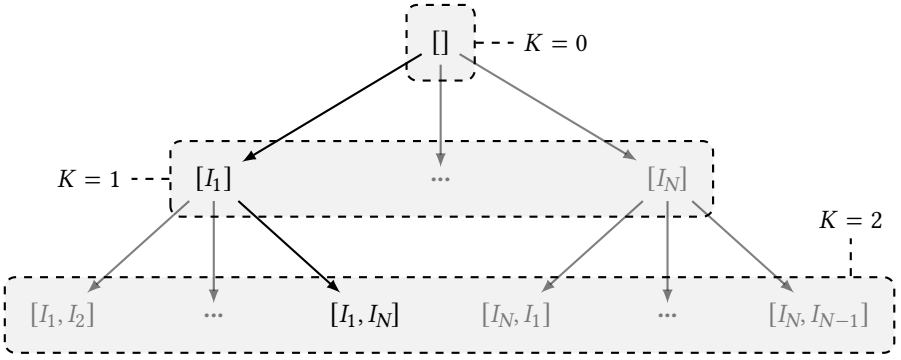


Figure 5.2: Image tree encoding all candidate interaction sequences up to second-order interactions.

grows exponentially with the number of objects and interaction order. In this tree, each node represents an object interaction (e.g., a reflection on a wall), and each edge represents selecting that interaction in the sequence. For reflection-only paths, level 1 connects the transmitter to all N objects, level 2 connects each object to up to $N-1$ others, and so on. For a fixed interaction order K, the tree contains up to $N(N-1)^{K-1}$ leaves (path candidates), which becomes unmanageable even for moderate N and K (see Figure 5.3). Therefore, exhaustive candidate enumeration is infeasible in large scenes or at high interaction orders, motivating early search-space reduction.

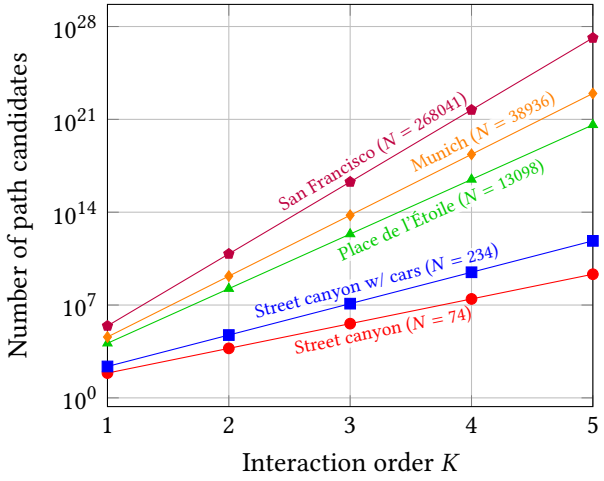


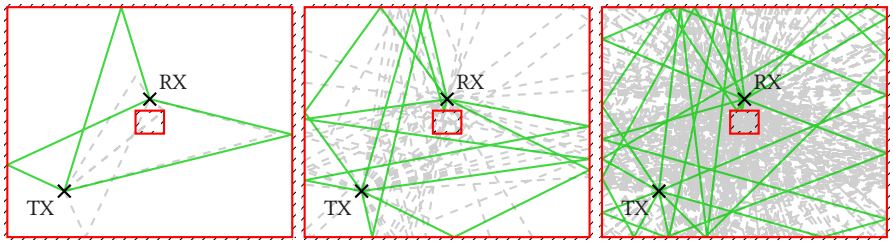
Figure 5.3: Exponential growth of the number of candidate paths with increasing interaction order K. Multiple curves show different environment complexities (N is the number of triangle facets) provided by Sionna RT [37].

5.2.2 Physical Sparsity and Validity Constraints

Most generated sequences in the image tree correspond to physically invalid ray paths. In exact path tracing, this occurs mainly for two reasons. First, a candidate can be *occluded* by objects that are not part of the sequence (the *visibility problem*). Second, interaction points returned by exact methods may violate geometric constraints because those methods often assume infinitely extended primitives.

Even in elementary scenarios, the ratio of valid paths to total candidates is exceedingly low. For instance, in a 2D setup with a single obstacle, only a small fraction of combinatorial candidates satisfy both visibility and geometric constraints (see Figure 5.4). As interaction order increases, this ratio drops sharply, creating an explosion of invalid paths that must be filtered out. In realistic urban scenes with hundreds to thousands of objects, the vast majority of candidates are also invalid: for every million generated candidates, only a few hundred may be valid at second order, and even fewer at higher orders [79]. This combination of strong sparsity and exponential growth motivates

[79]: Eertmans et al. (2026), *Transform-Invariant Generative Ray Path Sampling for Efficient Radio Propagation Modeling*



(a) First-order reflections. **(b)** Second-order reflections. **(c)** Third-order reflections.

Figure 5.4: Illustration of the sparsity of valid ray paths (solid green) compared to the combinatorial explosion of candidates, which leads to many invalid ray paths (dashed gray) in a simple 2D scenario with a single obstacle. As interaction order increases, the ratio of valid paths to candidates decreases dramatically, highlighting the need for pruning strategies.

aggressive pruning before expensive path validation.

5.3 Pruning Strategies

In this section, we discuss strategies to prune the combinatorial search space of candidate paths before expensive path validation. These methods leverage physical constraints, scene structure, and visibility information to eliminate large subsets of invalid candidates early in the process.

5.3.1 Merging Facets and Objects

One effective mitigation strategy is to merge adjacent facets or objects into larger primitives, reducing the total number of scene elements N and therefore the branching factor of the image tree. In urban environments, for example, building facades can be represented as single large planes rather than many small triangles. This reduces the primitive count and simplifies geometric computations for reflections and diffractions. The trade-off is that excessive merging can degrade accuracy, so the strategy must balance computational efficiency and geometric fidelity. DiffeRT [38], for instance, uses virtual merging of consecutive triangles into quadrilaterals, reducing the number of objects by roughly a factor of two.

[38]: Eertmans et al. (2025), *Demonstrating DiffeRT: An Open-Source Library for Optimizing Radio Networks with Differentiable Ray Tracing*

In some scenes, triangles are also grouped into *real objects* (e.g., buildings, trees, and vehicles). This enables object-level path finding rather than triangle-level path finding, further reducing the object count and the branching factor. If objects are convex polyhedra (as in many rectangular buildings), Fermat's principle implies that two consecutive interactions cannot occur on the same object; candidates with such sequences can therefore be pruned a priori. When object grouping is not provided in the scene description, however, this approach requires additional preprocessing to infer object membership, which can be expensive and difficult in scenes with many small or irregular shapes.

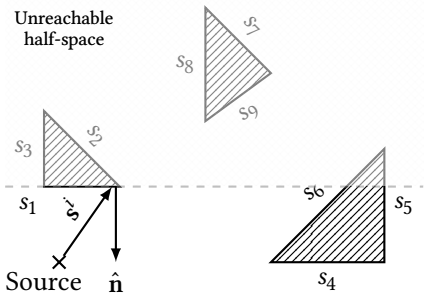


Figure 5.5: Illustration of the masking of unreachable objects after a reflection. The shaded area represents the half-space of unreachable objects after a reflection on surface s_1.

5.3.2 Masking Unreachable Objects After Reflection

After a reflection, the outgoing ray direction is constrained by the surface normal and the incident direction, so only a subset of objects can be reached. Even without the exact reflection point, unreachable objects can be identified by testing whether they lie outside the reflection half-space induced by the incident direction and the surface normal. Candidates requiring reflections toward those objects can then be pruned early, substantially reducing downstream validity checks. This strategy is especially effective in dense scenes. If, on average, half of the objects are unreachable after each reflection (as in a roughly random spatial distribution), the number of candidates is reduced by a factor of 2^K for paths with K interactions. Figure 5.5 illustrates this in 2D: the shaded region denotes objects that are unreachable after a reflection. If surface orientation is known (i.e., interior versus exterior), pruning can be strengthened further by excluding reflections toward interior objects. In Figure 5.5, for example, surfaces s_4 and s_5 are also unreachable after the first reflection on s_1. Computing the half-space and testing object positions against it is computationally inexpensive and can be done either during candidate generation or during scene preprocessing, providing a straightforward way to reduce the number of candidates.

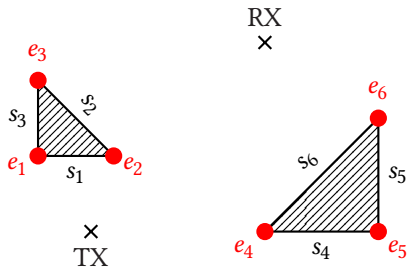


Figure 5.6: 2D scenario with triangular-shaped objects on which reflection or diffraction can occur. Surfaces are colored in black and edges in red.

5.3.3 Visibility Preprocessing

Similar to the previous strategy, visibility relations between objects can be computed offline and used to prune candidates that require transitions between non-visible objects. This approach extends the notion of unreachable objects to the broader case of occlusion (e.g., object shadowing), but it also introduces implementation challenges. While visibility can be solved efficiently (and exactly) in 2D, the problem is more complex in 3D because of the increased geometric complexity and partial occlusion. Even so, coarse visibility analysis can provide significant pruning benefits and is widely used in practical ray tracers [52, 104]. The following subsections present two implementations of visibility preprocessing: transmitter/receiver visibility and inter-object visibility.

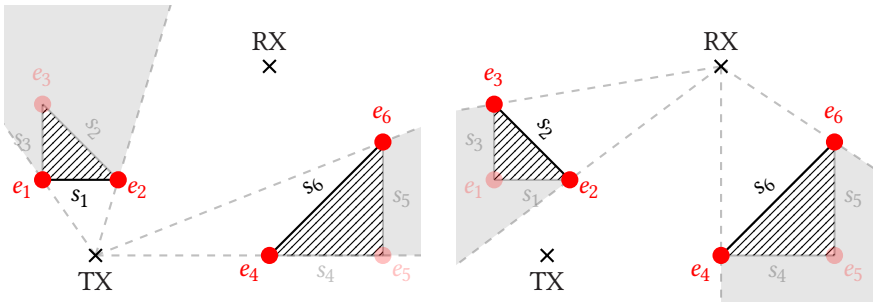
[52]: Aguado Agelet et al. (2000), *Efficient ray-tracing acceleration techniques for radio propagation modeling*

[104]: Lu et al. (2019), *A Discrete Environment-Driven GPU-Based Ray Launching Algorithm*

Transmitter and Receiver Visibility

The most basic form of visibility pruning is to determine which objects are visible from the transmitter and receiver positions. Any candidate path that requires an interaction with an object invisible from either endpoint can be pruned immediately. This simple first step effectively removes candidates that cannot contribute to the received signal.

To determine visibility, ray launching is a common approach. A large number of rays are cast from the transmitter and the receiver toward scene objects. Any object hit



(a) Objects visible from TX. **(b)** Objects visible from RX.

Figure 5.7: Example 2D visibility for the transmitter (TX) and receiver (RX). Objects are faded if they are not visible from the corresponding node.

by at least one ray is considered visible, and the remaining objects are marked as invisible (i.e., unreachable). If a ray intersects another object before reaching a target object, that target is marked as occluded. Figure 5.7 illustrates transmitter and receiver visibility for the scenario in Figure 5.6.

This process is relatively inexpensive and can be performed offline as a preprocessing step because it scales linearly with the number of objects and rays. It can be further accelerated with spatial data structures, briefly discussed in Section 5.5.2.

Finally, ray casting provides an approximation of visibility because it depends on the number and distribution of rays. In practice, a sufficiently large ray set yields good estimates, but edge cases remain: objects may still be incorrectly marked as invisible, especially when they are small or partially occluded. This method is therefore typically used as a pruning heuristic rather than an exact test.

Inter-Object Visibility

While transmitter- and receiver-based visibility pruning effectively removes candidates that involve non-visible objects, it does not capture occlusion between objects them-

selves. In multi-interaction paths, an object visible from the transmitter may still be occluded by another object before the receiver is reached. It is therefore beneficial to compute inter-object visibility as well, typically via ray casting between object pairs. Unlike transmitter/receiver visibility, however, the ray source is no longer a single point: it can be any point on an object's surface. In principle, this requires casting rays from every point on one object to every point on another. In practice, only a limited set of sample points is used. Even then, the method scales quadratically with the number of objects and can be expensive in large scenes. Spatial partitioning can mitigate this cost by restricting tests to nearby object pairs.

The Visibility Matrix

Recalling the graph-based formulation of path-candidate generation from Section 3.1.4, visibility relations can be encoded in an adjacency matrix $\mathcal{G} \in \{0,1\}^{N+2\times N+2}$, where N is the number of scene objects, and the additional two dimensions represent the transmitter and receiver nodes. An entry $\mathcal{G}_{ij}$ equals 1 if objects i and j are mutually visible (i.e., there is an unobstructed line-of-sight), and 0 otherwise. Figure 5.8 shows an example for the scenario in Figure 5.6. Candidate paths can then be generated by graph traversal on $\mathcal{G}$ from the transmitter node to the receiver node. A comparison of candidate counts with and without visibility pruning is provided in Appendix B.

	TX	s_1	s_2	s_3	s_4	s_5	s_6	e_1	e_2	e_3	e_4	e_5	e_6	RX
TX		1					1	1	1		1		1	1
s_1							**1**				**1**			
s_2							**1**				**1**		**1**	1
s_3														
s_4														
s_5														
s_6		**1**	**1**					**1**	**1**	**1**				1
e_1							1				1			
e_2							1				1		1	1
e_3							1				1		1	1
e_4		1	1					1	1	1				1
e_5														
e_6			1						1	1				1
RX														

Figure 5.8: Adjacency matrix, $\mathcal{G}$, generated from the scenario illustrated in Figure 5.9. Each row of this 14×14 matrix refers to the visible objects as seen from the corresponding object. For readability purposes, zeros are discarded. The black coefficients indicate possible reflections, while the red ones are for diffractions.

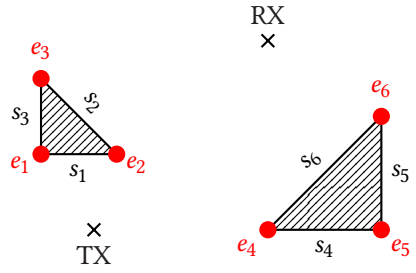


Figure 5.9: 2D scenario with triangular-shaped objects on which reflection or diffraction can occur, repeated here for convenience from Figure 5.6. Surfaces are colored in black and edges in red.

5.3.4 Learning-Based Pruning

As briefly introduced at the end of Chapter 3, machine learning models can be trained to predict the validity likelihood of candidate paths based on geometric features, interaction sequences, and visibility information. These models can be used to rank candidates and prune those below a chosen likelihood threshold before expensive validation. While this approach is promising, it requires careful feature engineering and training-data generation to ensure generalization across diverse environments. In addition, the model-inference overhead must be justified by the reduction in candidate-validation time. This method remains an active research area and has not yet been widely adopted in practical ray tracing pipelines; it is discussed further in Chapter 7.

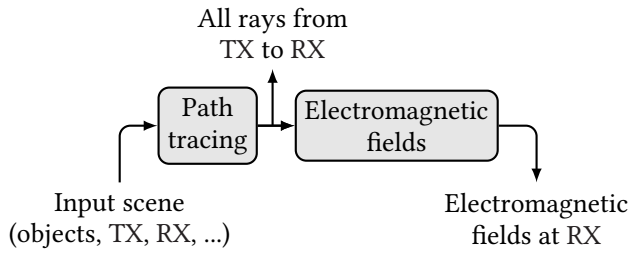


Figure 5.10: Simplified pipeline for dynamic ray tracing: once ray paths are computed, sufficiently small scene changes enable extrapolation of updated paths rather than full recomputation. This is possible because visibility relations between static objects remain unchanged, and only node-to-scene visibility must be updated at runtime.

5.4 Dynamic Scenarios and Continuous Updates

In dynamic scenarios such as vehicle-to-everything (V2X) communication, the environment changes continuously as vehicles and pedestrians move. However, static objects (e.g., buildings) remain unchanged, so visibility relations between them can be precomputed offline. At runtime, only node-to-scene visibility must be updated as transmitters and receivers move. This enables efficient ray-path extrapolation without full recomputation and supports real-time updates in dynamic environments.

As a result, *dynamic ray tracing* techniques have been introduced for such scenarios [51, 105]. These methods leverage precomputed visibility and spatial data structures to update ray paths quickly as nodes move, rather than recomputing the entire path tree from scratch. This is particularly beneficial in high-mobility settings where rapid updates are required.

[51]: Quatresooz et al. (2021), *Tracking of Interaction Points for Improved Dynamic Ray Tracing*

[105]: Bilibashi et al. (2020), *Dynamic Ray Tracing: Introduction and Concept*

5.4.1 Tracking Interaction Points and Doppler Shifts

Instead of computing an independent static snapshot at each time instant, dynamic ray tracing analytically tracks

the motion of interaction points (e.g., reflection and diffraction points) over moving geometry [51]. Assuming the instantaneous velocities of the transmitter ($\mathbf{v}_{TX}$) and receiver ($\mathbf{v}_{RX}$) are known, the velocity of each interaction point can be characterized explicitly [51].

[51]: Quatresooz et al. (2021), *Tracking of Interaction Points for Improved Dynamic Ray Tracing*

For example, the instantaneous velocity of a reflection point on a planar surface can be derived geometrically from the intersection between the reflecting plane and the line connecting the transmitter image (with respect to that plane) and the receiver. Using finite differences on the reflection-point trajectory yields

$$\mathbf{v}_r = \lim_{\Delta t \to 0} \frac{\mathbf{x}_r(t_0 + \Delta t) - \mathbf{x}_r(t_0)}{\Delta t}. \tag{5.1}$$

This geometric tracking enables immediate derivation of channel characteristics, such as Doppler shifts, from a single ray tracing run. For a generalized ray with multiple interactions across K moving scattering points (each with velocity $\mathbf{v}_k$), the Doppler-shifted received frequency is

$$f' = f_0 \prod_{k=1}^{K} \left(\frac{c - \mathbf{v}_k \cdot \hat{\mathbf{k}}_k}{c - \mathbf{v}_{k-1} \cdot \hat{\mathbf{k}}_k} \right), \tag{5.2}$$

where f_0 is the carrier frequency, $\hat{\mathbf{k}}_k$ is the unit vector of segment k, and c is the speed of light.

5.4.2 Visibility Assumptions and the Multipath Lifetime Map

Dynamic ray tracing extrapolation assumes that the active multipath structure remains unchanged for a limited interval, so paths do not need to be recomputed at every time step. This interval is related to the channel "coherence time" (T_c) and can be estimated from measurements [105]. In practice, however, measurement-based estimates are not always reliable because the measurement scenario may differ from the simulated one (geometry, traffic, materials, and mobility patterns).

[105]: Bilibashi et al. (2020), *Dynamic Ray Tracing: Introduction and Concept*

The validity interval also depends on modeling choices: higher interaction orders generally induce more frequent structural changes under mobility, because additional bounces are more sensitive to geometric perturbations. Moreover, expressing validity only in time can be misleading, as time bounds are tied to relative speeds. A spatial criterion is often more informative, since it decouples structural stability from instantaneous velocity.

For this reason, we introduce the multipath lifetime map (MLM)—a geometric tool that characterizes the spatial extent over which a multipath structure remains unchanged [106]. Its full treatment is deferred to Chapter 7; in particular, for single-object mobility (e.g., receiver displacement in a street canyon), these regions can be visualized directly in 2D. Operationally, once the spatial validity region is exited, extrapolation expires and a new full ray tracing snapshot is required to reinitialize path tracking [51].

[106]: Eertmans et al. (2025), *Comparing Differentiable and Dynamic Ray Tracing: Introducing the Multipath Lifetime Map*

[51]: Quatresooz et al. (2021), *Tracking of Interaction Points for Improved Dynamic Ray Tracing*

5.4.3 Dynamic vs. Differentiable Ray Tracing

Dynamic ray tracing and differentiable ray tracing should be viewed as complementary capabilities rather than competing paradigms [106]. Dynamic ray tracing focuses on fast temporal updates of path geometry, whereas differentiable ray tracing provides derivatives with respect to scene or system parameters.

Historically, the first dynamic ray tracing formulations used manually derived symbolic expressions for path-coordinate updates. These derivations can provide strong physical interpretability and low runtime cost, but they scale poorly as scenarios become more complex or when derivatives are needed with respect to richer parameterizations (e.g., a rotating car or multiple moving vehicles).

Automatic differentiation offers a practical alternative: the same dynamic-update pipeline can be differentiated with

respect to local variations without deriving new closed-form expressions for each case. In that sense, differentiable ray tracing can directly support dynamic ray tracing by broadening the set of tractable motion and sensitivity analyses.

5.5 Efficient Ray–Object Intersection

A fundamental operation in ray tracing is the intersection test between rays and scene objects (e.g., triangles). The efficiency of this operation directly impacts the overall performance of the ray tracer, especially when millions of rays must be tested against thousands of objects. The optimal algorithmic approach for this operation depends heavily on the specific geometric representation of the scene's objects. In this section, we present a very popular (and fast) algorithm for ray–triangle intersection and discuss how spatial data structures can further accelerate these computations.

5.5.1 Möller–Trumbore Ray–Triangle Intersection Algorithm

When it comes to ray–triangle intersection, the Möller–Trumbore algorithm [107] is widely adopted because of its efficiency and robustness. It computes the intersection of a ray with a triangle defined by its vertices in 3D space. The algorithm uses barycentric coordinates to determine whether the intersection point lies within the triangle and to compute the distance from the ray origin to the intersection point. Its main advantage is that it minimizes arithmetic operations and avoids unnecessary computation through early rejection tests, making it suitable for real-time and large-scale ray tracing. However, in differentiable ray tracing, such early rejections are not directly compatible with gradient computation through branching

[107]: Möller et al. (1997), *Fast, minimum storage ray-triangle intersection*

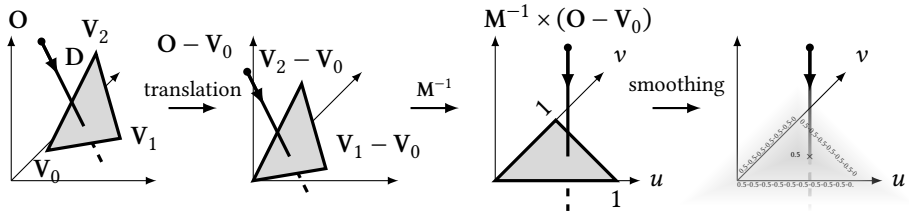


Figure 5.11: Translation and change of basis of the ray origin, adapted from [107, Fig. 1], and smoothing near triangle boundaries.

operations. As a result, Algorithm 3 presents a modified version of the Möller–Trumbore algorithm that aggregates subsequent rejection tests into a single boolean flag. The algorithm also includes optional smoothing, as introduced in Section 4.4, where the returned intersection flag becomes a continuous value between 0 and 1 rather than a binary hit-or-miss indicator. Figure 5.11 illustrates the geometric transformation to barycentric coordinates, as well as the effect of this smoothing near triangle boundaries, where the intersection flag transitions smoothly from 1 (full hit) to 0 (full miss) across a boundary region defined by the tolerance ϵ and smoothing factor α. On triangle edges, the intersection flag always evaluates to $\frac{1}{2}$.

Algorithm 3: Möller–Trumbore algorithm with optional smoothing.

Input: Ray origin $\mathbf{o}$ and direction $\mathbf{d}$, triangle vertices $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$, tolerance ϵ, smoothing factor α (optional), and smoothing function $s(\cdot\, \alpha)$

Output: Distance to intersection t and intersection flag/probability hit

$\mathbf{e}_1 \leftarrow \mathbf{v}_1 - \mathbf{v}_0$, $\mathbf{e}_2 \leftarrow \mathbf{v}_2 - \mathbf{v}_0$ `// Triangle edges`

$\mathbf{h} \leftarrow \mathbf{d} \times \mathbf{e}_2$

$a \leftarrow \mathbf{e}_1 \cdot \mathbf{h}$ `// Determinant`

`// Check ray--triangle parallelism`

if *α is provided* **then**

  hit $\leftarrow s(|a| - \epsilon; \alpha)$

else

  hit $\leftarrow |a| > \epsilon$

$f \leftarrow \frac{1}{a}$ `// In practice, division by zero must be handled with care`

$\mathbf{s} \leftarrow \mathbf{o} - \mathbf{v}_0$

$u \leftarrow f[\mathbf{s} \cdot \mathbf{h}]$ `// 1st barycentric coordinate`

`// Check 1st barycentric coordinate`

if *smoothing is enabled* **then**

  hit $\leftarrow \min(\text{hit}, s(u; \alpha), s(1 - u; \alpha))$

else

  hit $\leftarrow$ hit **and** $u \geq 0$ **and** $u \leq 1$

$\mathbf{q} \leftarrow \mathbf{s} \times \mathbf{e}_1$

$v \leftarrow f[\mathbf{d} \cdot \mathbf{q}]$ `// 2nd barycentric coordinate`

`// Check 2nd barycentric coordinate`

if *smoothing is enabled* **then**

  hit $\leftarrow \min(\text{hit}, s(v; \alpha), s(1 - (u + v); \alpha))$

else

  hit $\leftarrow$ hit **and** $v \geq 0$ **and** $u + v \leq 1$

$t \leftarrow f[\mathbf{e}_2 \cdot \mathbf{q}]$ `// Ray distance to intersection`

`// Check ray depth`

if *smoothing is enabled* **then**

  hit $\leftarrow \min(\text{hit}, s(t - \epsilon; \alpha))$

else

  hit $\leftarrow$ hit **and** $t > \epsilon$

return t, hit

5.5.2 Spatial Subdivision and Bounding Volume Hierarchies

As established in Section 5.2, naive path-candidate generation leads to a combinatorial explosion. Even after aggressive pruning, a practical simulation may still need to evaluate millions of ray paths in environments containing thousands of objects. Testing every ray against every geometric object is computationally intractable and memory-intensive.

To significantly reduce the number of objects considered during these queries, modern ray tracers rely on acceleration structures. These spatial data structures organize the scene geometry so that rays can quickly discard large sections of the environment they will definitively not intersect, focusing computational power only on objects in their direct path.

Two historical approaches dominate this space: k-d trees [75] and bounding volume hierarchies (BVHs) [108, 109].

[75]: Bentley (1975), *Multidimensional binary search trees used for associative searching*

[108]: Clark (1976), *Hierarchical geometric models for visible surface algorithms*

[109]: Goldsmith et al. (1987), *Automatic creation of object hierarchies for ray tracing*

k-d Trees: Partitioning Space

A k-d (short for k-dimensional) tree strictly partitions the 3D space itself into non-overlapping regions using axis-aligned splitting planes.

As illustrated in Figure 5.12, each node in the k-d tree represents a spatial plane that divides the space in half. As a ray traverses the tree, it steps through these spatial regions sequentially. Because the spatial partitions never overlap, k-d trees are highly efficient at finding the closest intersection point quickly. They perform exceptionally well on traditional CPU architectures.

However, a significant limitation arises when geometric objects straddle splitting planes: such objects must be duplicated and stored in multiple nodes. On highly parallel architectures like GPUs, this object duplication increases

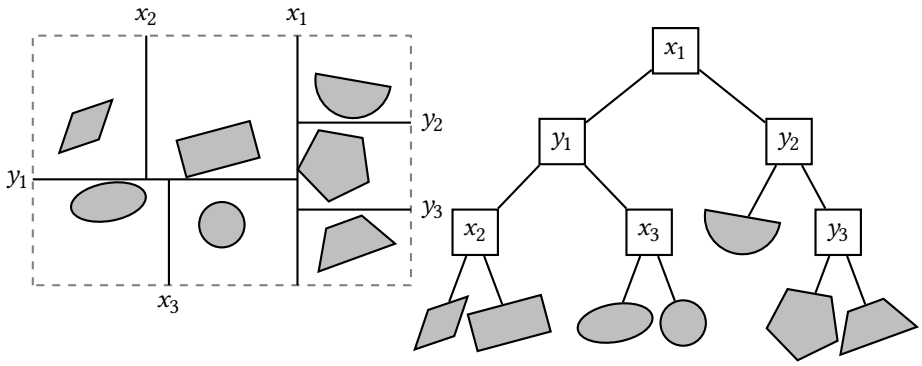


Figure 5.12: Illustration of k-d tree applied to a simple scene with 7 objects. The k-d tree strictly partitions space into non-overlapping regions, where each node contains a splitting plane and its child nodes.

memory overhead and complicates traversal logic, leading to inefficiencies.

Bounding Volume Hierarchy: Partitioning Objects

Unlike k-d trees, a BVH partitions the *objects* rather than the physical space. It organizes the scene primitives into a hierarchy of overlapping bounding boxes.

As illustrated in Figure 5.13, bounding volume hierarchies organize the scene through leaf nodes that contain the actual geometric objects, while parent nodes contain a bounding box that completely encloses all of their children. If a ray misses a parent bounding box, the ray tracer knows it can safely ignore all objects inside it.

Because BVHs partition the object list, each object is referenced exactly once. This zero-duplication, object-centric approach provides predictable memory usage and minimizes divergence, making BVHs the undisputed standard for modern, many-core GPU architectures.

One drawback is that bounding boxes can overlap. If a ray passes through an overlapping region, the traversal

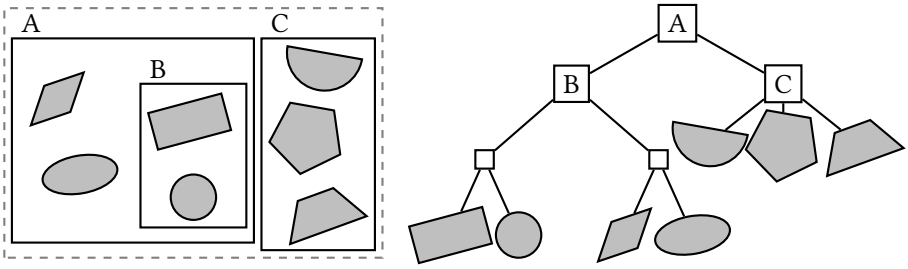


Figure 5.13: Illustration of BVH applied to a simple scene with 7 objects. The BVH organizes primitives into a binary tree of bounding boxes, where each node contains a bounding box that encloses its child nodes.

algorithm must check both branches of the tree, which can sometimes lead to redundant intersection tests.

5.6 Hardware Acceleration and GPU Execution Paradigms

Modern ray tracing benefits profoundly from hardware acceleration, but exploiting these architectures effectively requires understanding fundamental hardware constraints. Unlike traditional compute-focused algorithms, ray tracing accelerated by GPUs is typically *memory-bound* rather than compute-bound; bandwidth and cache efficiency dominate performance far more than raw floating-point throughput. Consequently, acceleration techniques must prioritize improved memory access patterns, reduced warp divergence (caused by divergent execution paths, for example, due to branching), and minimized data movement rather than simply fewer arithmetic operations.

Additionally, GPU compilation imposes strict constraints on program structure. As already discussed in Chapter 4, most GPU frameworks cannot accommodate fully dynamic control flow or variable-sized data structures; compilers require tensor shapes and loop bounds to be known at compile time. This necessitates reformulating ray tracing logic—which naturally involves branching and variable-depth recursion—into fixed-size tensor operations with explicit static parameters. Just-in-time (JIT) compilation, such as that provided by JAX via XLA, is therefore essential for managing this complex branching logic while maintaining performance. It generates specialized kernels for each unique combination of static parameters, enabling both efficient execution and correct gradient computation.

As a practical example, dynamic path-candidate generation via graph traversal must be reformulated into fixed-size batches using padding and chunking operations, and acceleration structures such as BVHs must be designed to hold a fixed number of objects per node rather than variable-length lists. Traversal algorithms similarly process fixed-size batches of rays instead of streaming workloads, accepting that padding may introduce some redundant computation in exchange for deterministic memory

usage and instruction-level parallelism.

5.6.1 Dedicated Ray Tracing Hardware Cores

Modern GPUs provide dedicated ray tracing units, exposed through application programming interfaces (APIs) such as NVIDIA OptiX [32]. Intersection-heavy workloads in radio propagation can be mapped to these APIs for substantial acceleration relative to software-only BVH traversal. Although originally designed for computer graphics, several radio-propagation-oriented ray tracing tools, such as Opal [110], use the NVIDIA OptiX API for efficient path validation. However, NVIDIA OptiX is designed for the forward ray tracing pass only and does not support automatic differentiation. As a result, it cannot be used directly for differentiable ray tracing, and custom ray tracing kernels are required to support both forward and backward passes. Tools such as Mitsuba 3 [111] address this by combining AD frameworks (here, Dr.Jit [112]) with NVIDIA OptiX for efficient differentiable ray tracing on GPUs. Although Mitsuba 3 is designed for computer graphics applications, it provides the necessary building blocks (such as polarization states through Jones vectors) to be adapted to radio propagation by modifying the underlying physical models and interaction types; Sionna RT [37] is a representative example of this adaptation.

[32]: Parker et al. (2010), *OptiX: a general purpose ray tracing engine*

[110]: Egea-Lopez et al. (2021), *Opal: An open source ray-tracing propagation simulator for electromagnetic characterization*

[111]: Jakob et al. (2022), *Mitsuba 3 renderer*

[112]: Jakob et al. (2022), *Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering*

[37]: Hoydis et al. (2023), *Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling*

5.6.2 Just-in-Time Compilation, Gradient Checkpointing, and Rematerialization

When building complex simulation tools such as ray tracers, high-level Python code improves development productivity but is often too slow for large-scale workloads. JIT compilation bridges this gap by translating high-level operations into specialized routines tailored to the target hardware.

In DiffeRT [38], this raw execution speed is achieved through JAX, which relies on the XLA compiler to transform high-level mathematical operations into highly optimized machine code. When a function is transformed using `jax.jit`, JAX executes a tracing pass using abstract tracer objects to record the sequence of operations into an intermediate language known as *jaxpr*. XLA then compiles this intermediate representation.

[38]: Eertmans et al. (2025), *Demonstrating DiffeRT: An Open-Source Library for Optimizing Radio Networks with Differentiable Ray Tracing*

The performance benefits of JIT compilation for path tracing are twofold:

- Through *elimination of Python overhead*, path tracing can execute complex conditional logic and mathematical sequences for millions of independent rays without the interpreter in the execution path, allowing the simulation to run at native hardware speeds.
- Through *operator fusion*, XLA identifies adjacent operations and combines them into a single unified operation. Because of this, intermediate values remain in fast on-chip memory (registers) instead of being repeatedly written to and read from slower global memory, which mitigates bandwidth bottlenecks.

Although JIT compilation is widely used to maximize throughput, large-scale differentiable simulations remain constrained by memory. Tracking every step of a computation for gradient evaluation—for example, to quantify how a small geometric perturbation affects a reflected path—can require storing very large volumes of intermediate data.

When memory limits are reached, gradient checkpointing (or rematerialization) provides a practical solution. Instead of storing all intermediate values, the system retains selected checkpoints and recomputes missing intermediates when needed. This trades additional computation for substantially lower peak memory usage.

Technically, gradient checkpointing and rematerialization address these memory bottlenecks by discarding forward

intermediates and recomputing them during backpropagation. This increases computation but substantially reduces peak memory, a crucial trade-off when evaluating deep computational graphs such as those generated by multi-bounce ray tracing. In the context of the JAX framework, this mechanism is accessed via the `jax.checkpoint` decorator (also aliased as `jax.remat`).

Together, JIT compilation and gradient checkpointing form the foundation of an efficient differentiable ray tracer: `jax.jit` ensures that both the forward pass and the rematerialized backward computations run with high hardware utilization, while `jax.remat` helps peak memory usage scale more gracefully with the number of simulated ray interactions. Importantly, these optimizations remain largely transparent to the user, who can continue writing code in a natural high-level style without manually managing memory or parallelism.

5.6.3 Example: GPU-Aware Path Tracing Formulation

Building on the previous chapters, we now discuss how to reformulate path tracing for efficient GPU execution. In particular, path finding can be cast as a tensor optimization problem solved with a fixed number of custom Broyden–Fletcher–Goldfarb–Shanno (BFGS) updates directly on the GPU, enabling thousands of paths to be processed in parallel [106].

[106]: Eertmans et al. (2025), *Comparing Differentiable and Dynamic Ray Tracing: Introducing the Multipath Lifetime Map*

The key challenge for GPU execution is not the convex formulation itself (introduced in Chapter 3), but enforcing a computation graph with static shape and predictable control flow. In practice, this requires two constraints.

First, every path in a batch must use the same parameter dimension, even when interaction sequences mix reflections and diffractions in arbitrary order. A practical implementation sets $d = 2$ for all interactions and represents edge diffraction with a zero second basis vector

($\boldsymbol{A}_{i,2} = \mathbf{0}$). This preserves a single tensor layout for all candidates and avoids per-path shape changes, which would otherwise trigger recompilation and reduce vectorization efficiency.

Second, the iterative solver must run for a fixed number of iterations. If each path were allowed to stop on its own convergence criterion, parallel execution would still be paced by the slowest path in the batch. A fixed budget yields deterministic latency, stable memory usage, and better hardware utilization, at the cost of a controlled amount of extra work for easy paths.

Because the path-length objective is convex in the considered planar reflection–diffraction setting, quasi-Newton schemes such as BFGS are particularly effective. They exploit gradient information to build a local curvature model while avoiding explicit Hessian construction, yielding a favorable compromise between robustness and throughput for large batches.

Once the problem shape is fixed, additional low-level optimizations become possible. In particular, analytic expressions for the objective gradient can be precomputed and reused in both the forward solver and the custom vector–Jacobian product rule introduced in Chapter 4, reducing autodiff overhead. Likewise, specialized search-direction and step-size updates can be designed for this convex structure, further lowering per-iteration cost.

With these constraints and optimizations combined, Fermat-based path solvers approach image-method runtimes while supporting more general interaction sequences, including mixed reflection–diffraction paths [106]. The full JAX-based implementation of this solver is available at `https://github.com/jeertmans/fpt-jax`.

[106]: Eertmans et al. (2025), *Comparing Differentiable and Dynamic Ray Tracing: Introducing the Multipath Lifetime Map*

5.7 Efficacy of Exact Path Tracing Methods

Evaluating the efficiency of point-to-point ray path-finding algorithms exposes fundamental trade-offs between the types of supported interactions, execution speed, differentiability, and modern hardware acceleration paradigms. In this section, we analyze the performance of various exact and Fermat-based solvers, demonstrated through a quantitative benchmark study.

5.7.1 Baseline Method Evaluation

We compare two primary path-finding approaches, each presenting distinct architectural characteristics:

- The *image method (IM)*, as established in Section 3.1.1, is an exact, non-iterative formulation that guarantees the recovery of all valid paths. It remains extremely fast, but it is strictly limited to purely specular reflections and cannot scale to model diffractions or transmissions.
- The *Fermat-based approaches*, as established in Section 3.1.3 and further studied in this chapter, are iterative optimization-based formulations that can model mixed reflection–diffraction paths.

The min-path-tracing method is not considered in this comparison, as its object-oriented structure makes it inherently sequential and unsuitable for parallel hardware acceleration.

5.7.2 The GPU Implementation Dilemma (Uniformity versus Accuracy)

Implementing Fermat-based optimization on parallel hardware accelerators highlights a key dilemma between execution uniformity and optimization accuracy:

- The method proposed by Carluccio and Albani [54], at least in its extended version to model reflections and diffractions [56], uses a mixed-method solver, combining the exact image method for specular reflections with a Newton-based iterative solver for diffraction edges. Although highly accurate, it dynamically reconstructs the optimization problem based on the specific sequence of interactions along each path. This dynamic control flow introduces heavy branching logic, reducing GPU warp execution efficiency.
- In [106], a uniform computational framework is proposed that treats the entire path as a single optimization problem. By setting a fixed interaction dimension ($d = 2$) for all interactions and padding straight edges with a zero second basis vector ($\boldsymbol{A}_{i,2} = \mathbf{0}$), the solver maintains a static tensor layout. While this eliminates branching and maximizes GPU throughput, it also introduces more variables than strictly necessary, increasing both the computational cost per iteration and the risk of suboptimal convergence for complex paths.

[54]: Carluccio et al. (2008), *An Efficient Ray Tracing Algorithm for Multiple Straight Wedge Diffraction*

[56]: Puggelli et al. (2014), *A Novel Ray Tracing Algorithm for Scenarios Comprising Pre-Ordered Multiple Planar Reflectors, Straight Wedges, and Vertexes*

[106]: Eertmans et al. (2025), *Comparing Differentiable and Dynamic Ray Tracing: Introducing the Multipath Lifetime Map*

5.7.3 Benchmark Findings

We compare several Fermat-based solvers against the exact image method baseline, all implemented within the uniform computational framework described above. The evaluated solvers include a standard gradient descent (GD), the Newton-like approach of Carluccio & Albani (CA), the optimized BFGS method (JE) proposed in [55]—which employs fixed-point iteration to determine the optimal step size at each solver iteration—and a naive limited-memory BFGS (L-BFGS) implementation to highlight the impact of domain-specific optimizations. For diffraction-only scenarios, the image method cannot be used, so an advanced second-order cone program (SOCP)-based solver is used instead (executed on the CPU for reasons detailed below). The simulations were conducted on an NVIDIA

[55]: Eertmans et al. (2025), *Fast, Differentiable, GPU-Accelerated Ray Tracing for Multiple Diffraction and Reflection Paths*

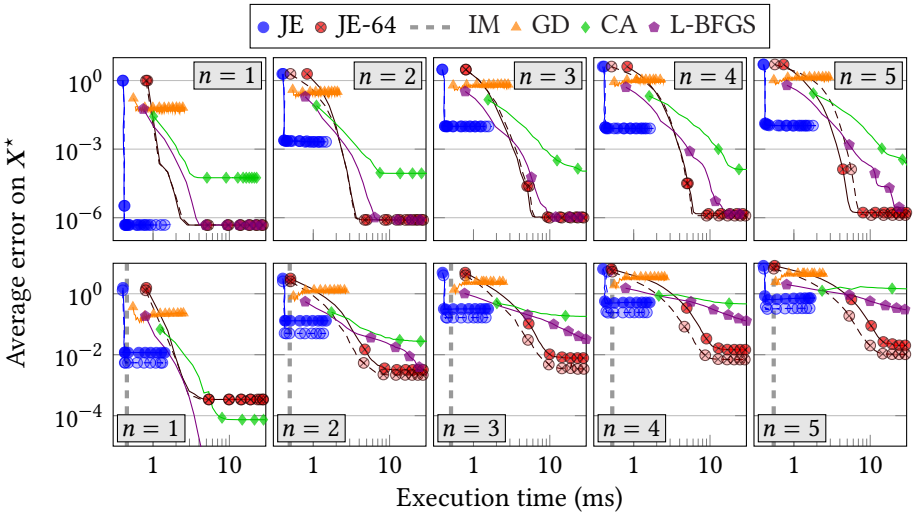


Figure 5.14: Computational time versus solution accuracy across 1000 randomly generated path instances, with interaction order K ranging from 1 to 5, reproduced from [55, Fig. 2]. The upper subplot presents one-dimensional diffraction scenarios, while the lower depicts two-dimensional reflection cases. Lighter dashed curves in the top subplot show diffractions handled with $d = 2$ by our solver alone; comparable methods fail in these configurations. Similarly, the bottom subplot shows hybrid reflection–diffraction paths (lighter dashes) exclusively solved by our method. Two variants are evaluated: the standard configuration and an enhanced version with 64 fixed-point refinements per step (JE-64). The image method serves as a reference baseline for reflection cases, marked with a vertical line indicating its average runtime.

GeForce RTX™ 3070 GPU with 8 GB of memory using single-precision floating-point arithmetic, and the average error was evaluated over 1000 random path scenarios.

Figure 5.14 summarizes the average error on the true geometrical ray path $\boldsymbol{X}^\star$ versus execution time for interaction orders K from 1 to 5. In homogeneous, diffraction-only configurations ($d = 1$, top row), the JE solver achieves the lowest error and fastest convergence across all cases. Increasing the fixed-point iteration budget (JE-64) enables convergence down to machine precision. In reflection-only scenarios (bottom row), the image method remains the most efficient, achieving comparable or better accuracy with runtimes an order of magnitude lower than the iterative alternatives. Nonetheless, among the iterative solvers,

the JE solver consistently delivers higher accuracy in less time than both CA and L-BFGS for $K > 1$. Finally, in mixed reflection–diffraction configurations (dashed curves in the bottom row of Figure 5.14), the JE solver also converges the fastest, but all solvers stagnate around an error rate of 10^{-2}. This is significantly higher than the single-precision machine epsilon (approximately 10^{-6}) and may be insufficient for practical applications requiring high precision.

Speed Gains of Implicit Differentiation

As discussed in Section 4.3.4, differentiating through an iterative solver can be computationally expensive. Implicit differentiation provides an alternative to standard automatic differentiation, enabling much more efficient gradient computations. We investigate this by comparing the execution times of implicit differentiation against standard automatic differentiation under various solver configurations.

As shown in Figure 5.15, implicit differentiation is nearly tenfold faster than standard AD for gradient computation. Crucially, this speedup is maintained regardless of the solver iteration budget (16 versus 128) or the number of fixed-point refinements per step (1 versus 64).

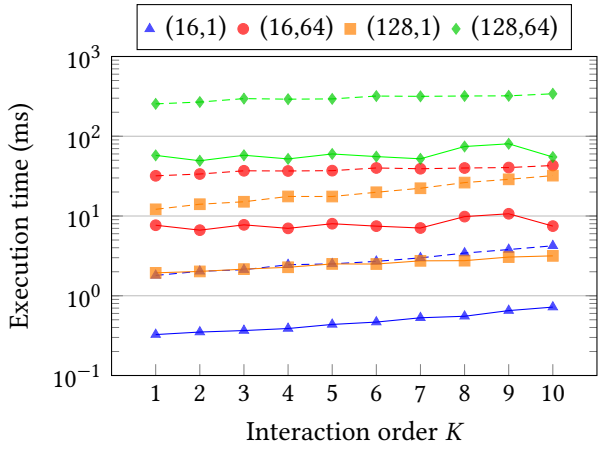


Figure 5.15: Runtime comparison for gradient computation of 1000 path lengths relative to object parameters, contrasting implicit differentiation (solid curves) with AD (dashed curves), plotted against the count of interactions K, with $d = 2$. Results shown for solver configurations employing either 16 or 128 iterations, combined with 1 or 64 fixed-point refinements per step, are reproduced from [55, Fig. 3].

5.7.4 Future Work and Open Questions

Resolving convergence issues in mixed reflection–diffraction paths remains an active area of research:

- Reformulating path-length minimization as a convex conic optimization problem shows promise in resolving mixed-path convergence issues. In fact, on the CPU, the ECOS [113] SOCP solver outperforms all quasi-Newton methods in terms of convergence speed and accuracy. This is likely because recasting the problem as an SOCP regularizes the square-root terms arising from Euclidean distances, transforming the optimization landscape. However, solving SOCPs natively and efficiently on GPUs is a nascent research area, and general-purpose, differentiable, GPU-compatible SOCP solvers are not yet widely available. Furthermore, many CPU solvers are designed for double-precision floating-point arithmetic, which is typically not natively or efficiently supported on GPUs; porting these algorithms to single-precision is often non-trivial.
- Another key question is whether a uniform solver is the most effective choice, or if the overhead of dynamically compiling specialized kernels for each path configuration is justified by the resulting gains in accuracy and performance. If compilation is sufficiently fast and the number of encountered path configurations is small (as is typical in ray tracing tools that limit diffraction or refraction to a single interaction), then the specialized approach proposed in [56] might be preferable.

[113]: Domahidi et al. (2013), *ECOS: An SOCP Solver for Embedded Systems*

[56]: Puggelli et al. (2014), *A Novel Ray Tracing Algorithm for Scenarios Comprising Pre-Ordered Multiple Planar Reflectors, Straight Wedges, and Vertexes*

5.8 Software Landscape and Recommendations

A very relevant question the reader might have at this point is: "which ray tracing software tool should I use?" The an-

swer depends on the specific requirements of the project, such as the need for differentiability, the types of interactions to be modeled, and the available computational resources. In this section, we provide a brief overview of the current software landscape for ray tracing in wireless communications and offer recommendations based on different use cases.

5.8.1 The Landscape

The choice of ray tracing software depends on user constraints, including budget, specific use cases, and performance requirements. While proprietary commercial solutions offer high performance, the open-source ecosystem is relatively small but growing rapidly. Moreover, the number of differentiable ray tracing tools is extremely small; at the time of writing, Sionna RT [37] and DiffeRT [38] are likely the only two open-source differentiable ray tracing frameworks designed for radio propagation. Differentiable ray tracing is a relatively new research area, especially in wireless communications, and building such tools typically requires rewriting existing simulation logic from the ground up to support end-to-end automatic differentiation, as discussed in Section 4.3.1.

[37]: Hoydis et al. (2023), *Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling*

[38]: Eertmans et al. (2025), *Demonstrating DiffeRT: An Open-Source Library for Optimizing Radio Networks with Differentiable Ray Tracing*

5.8.2 Primary Recommendation: Sionna RT

While commercial and, to a lesser extent, open-access ray tracing tools have been extensively used for decades by both the academic and industrial communities, the open-source Sionna link-level simulator and its ray tracing module, Sionna RT, offer a highly comprehensive and efficient solution. Currently, Sionna RT represents the most complete and high-performance library for general-purpose differentiable radio propagation modeling. Since its version 1 release, Sionna RT builds on the differentiable rendering framework Mitsuba 3 [111] and its companion

[111]: Jakob et al. (2022), *Mitsuba 3 renderer*

JIT compiler, Dr.Jit [112], natively leveraging hardware-accelerated specialized ray tracing cores via NVIDIA OptiX to achieve exceptionally fast ray–object intersections on GPUs. Sionna RT supports a wide range of propagation interactions, including reflections, diffractions, and transmissions, and provides a flexible API for defining custom materials and complex 3D scenes. Furthermore, it integrates seamlessly with the broader Sionna simulation framework, enabling end-to-end optimizations of wireless systems that couple physical-layer components with accurate radio propagation channels.

[112]: Jakob et al. (2022), *Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering*

5.8.3 An Alternative: DiffeRT

DiffeRT is an open-source differentiable ray tracing toolbox developed by the author of this book. Building on its 2D predecessor, DiffeRT2d [85], it leverages the JAX [36] framework and the Equinox machine learning library to offer a 3D differentiable ray tracing solution natively scalable to GPUs and other hardware accelerators. Unlike broader frameworks such as Sionna, DiffeRT focuses exclusively on core, geometrical ray tracing functionalities. It features both a low-level API for fine-grained path tracing and a high-level API for electromagnetic field computations, offering users the flexibility to either compute all paths in a scene or trace user-defined subsets of path candidates. It also integrates multiple visualization backends and provides compatibility with Sionna scene files. Unfortunately, DiffeRT is currently unable to match the extensive feature set and execution speed of Sionna RT, the latter being primarily a consequence of its JAX-based architecture rather than the specialized rendering compilation pipeline of Dr.Jit. While DiffeRT's functional design facilitates rapid prototyping, its reliance on JAX and XLA leads to significant compilation overheads for complex scenes compared to dedicated rendering compilers, as discussed in the next subsection.

[85]: Eertmans et al. (2024), *DiffeRT2d: A Differentiable Ray Tracing Python Framework for Radio Propagation*

[36]: Frostig et al. (2018), *Compiling machine learning programs via high-level tracing*

5.8.4 The JIT Compiler Mismatch

The difference in performance between Sionna RT and DiffeRT highlights a fundamental compiler mismatch:

- DiffeRT is built on JAX [36] and compiles code using XLA. JAX and XLA are optimized for machine learning workloads, which typically feature shallow computational graphs with heavy, dense tensor operations (e.g., matrix multiplications). Ray tracing, however, consists of massive, deeply nested graphs filled with lightweight, control-flow-heavy geometric operations (e.g., ray–triangle intersections). Tracing this in JAX results in large graph representations, leading to high compilation times, suboptimal kernel fusion, and a lack of support for dedicated hardware ray tracing cores. Another major issue is that JAX's JIT compiler cannot fuse operations to reduce memory usage during ray tracing, meaning that parallelized ray–triangle intersection tests on large scenes create significant memory overheads and allocation issues.
- Sionna RT leverages Dr.Jit, a JIT compiler specifically designed for differentiable rendering and ray tracing. Dr.Jit naturally compiles ray tracing workloads, merges ray–object intersection passes, and targets hardware ray tracing cores through NVIDIA OptiX, resolving this compiler mismatch.

[36]: Frostig et al. (2018), *Compiling machine learning programs via high-level tracing*

Consequently, a direct benchmark comparison between DiffeRT and Sionna RT is not currently meaningful at large scene scales. Because DiffeRT relies on JAX and XLA, its compilation overhead grows rapidly with scene complexity and interaction depth, which limits its scalability in comparison to the large environments that Sionna RT handles efficiently. Therefore, the two tools should be compared primarily in terms of their design philosophies and targeted use cases rather than via runtime benchmarks.

5.8.5 Niche for DiffeRT

Despite Sionna RT's features and performance advantages, DiffeRT remains highly relevant for specific scenarios. It is particularly suited for theoretical research into ray geometry, custom differentiable formulations, or projects that require tight integration with JAX-based optimization workflows (such as deep reinforcement learning or custom neural network architectures). Thanks to DiffeRT's functional programming approach, exposed lower-level routines, and simplified object representation, users can easily modify the ray tracing logic and compose custom pipelines. This composable structure makes it straightforward to study differentiability with respect to arbitrary parameters—such as building corner coordinates or custom electromagnetic parameters—that are not natively or easily differentiable in Sionna RT due to its deep integration with Mitsuba 3.

5.8.6 Future Outlook

As JIT compiler technologies evolve, the boundary between general machine learning compilers and rendering-specific compilers may become less distinct. Although the current landscape is dominated by Sionna RT, the limitations and tool characteristics described here represent a temporary snapshot in a rapidly evolving ecosystem. Future iterations of general-purpose machine learning compilers may natively support specialized ray tracing hardware cores, potentially narrowing the performance gap between general-purpose machine learning libraries and specialized rendering frameworks, and opening up new possibilities for differentiable radio propagation tools.

5.9 Conclusion

This chapter has addressed the computational bottlenecks that appear when ray tracing is scaled to realistic wire-

less scenarios. We first showed why exhaustive candidate generation is intractable, then introduced pruning mechanisms—facet/object merging, unreachable-object masking, and visibility preprocessing—to reduce the candidate set before expensive validation. We also discussed how these ideas extend to dynamic scenarios, where precomputed static visibility and local path extrapolation can reduce full recomputation frequency.

At the geometric-kernel level, we reviewed efficient ray–object intersection and acceleration structures, highlighting the role of Möller–Trumbore intersection tests and the practical trade-offs between k-d trees and BVHs. At the hardware level, we emphasized that modern GPU ray tracing is largely constrained by memory behavior and control-flow regularity, which motivate static-shape formulations, JIT compilation, and rematerialization strategies for differentiable workloads.

Finally, we showed how Fermat-based path solvers can be reformulated for efficient batched execution by enforcing fixed interaction dimensionality and fixed iteration budgets, while leveraging convexity to use robust quasi-Newton updates such as BFGS. Combined with analytic-gradient reuse and custom update rules, this substantially narrows the runtime gap with image-method baselines while supporting mixed reflection–diffraction paths. Further gains are expected by porting existing SOCP solvers, like ECOS, to GPU programs and differentiable frameworks, allowing the path tracing problem to be rewritten in a way that can be solved more efficiently. These efficiency mechanisms provide the computational basis for Chapter 6, where we move from channel forward modeling in a classical urban street-canyon scenario to inverse tasks such as localization and material calibration.

Using

6 Applications

In this chapter, we combine the path-tracing methods from Chapter 3 with the differentiable formulations from Chapter 4 and the acceleration techniques from Chapter 5 to demonstrate how these tools can be applied to real-world wireless-engineering problems. We focus on a canonical urban street canyon scenario, which is representative of many practical deployment environments, and we explore both forward modeling of the radio channel and inverse estimation problems.

The urban street canyon scenario, illustrated in Figure 6.1, is a simple but representative environment for outdoor propagation in cities. It features a single street flanked by two rows of buildings, creating strong multipath propagation. It consists of 74 triangular facets: 12 per building (two facets per wall) and two for the ground. In the following sections, we focus on line-of-sight and specular-reflection paths and, occasionally, diffraction paths to keep the analysis tractable. Due to the physical limitations of cascading multiple diffractions (discussed in Section 2.6.6), we limit ourselves to first-order diffraction paths. However, the methods and insights presented here can be extended to more complex scenarios with additional propagation phenomena.

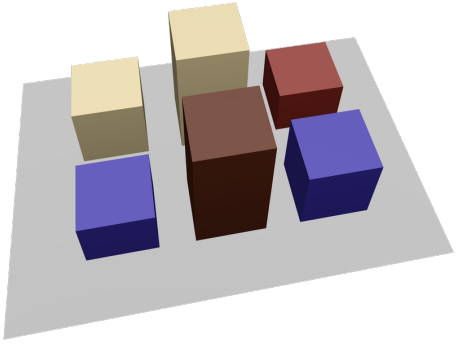


Figure 6.1: Simple street canyon scenario from Sionna RT [37].

6.1 Forward Modeling of the Radio Channel

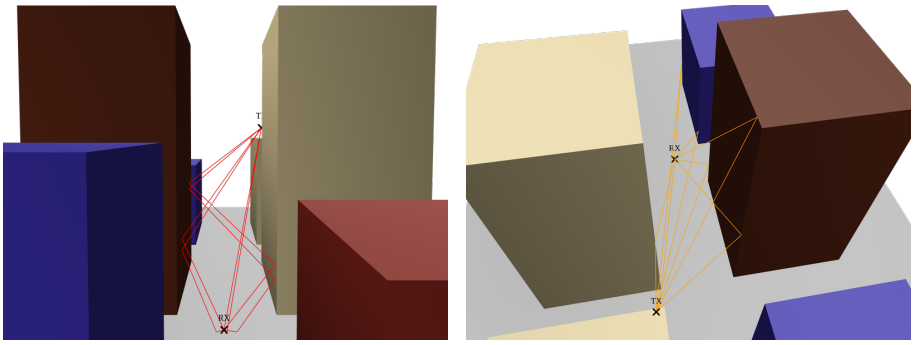
Forward modeling predicts radio-channel observables from known scene geometry, material properties, and network parameters. In practice, this is the core workflow for coverage analysis, network planning, and performance evaluation. Differentiable ray tracing extends this workflow by providing gradients of these observables with respect to both environment and system variables, as introduced in Chapter 4. This enables not only simulation but also gradient-based design and calibration.

6.1.1 Tracing of Propagation Paths

The first task of a ray tracer is to identify physically valid propagation paths between a transmitter and a receiver, subject to geometric visibility constraints and electromagnetic interaction rules.

In the present example, we consider a single transmitter located on top of one building ($\mathbf{x}_{\mathrm{TX}} = \begin{bmatrix} -33 & 11 & 32 \end{bmatrix}^{\top}$ m), similar to a typical base-station location, and a receiver that can be placed anywhere in the scene at an altitude of 1.5 m, representing a typical user-equipment height (e.g., a smartphone held in the user's hand). Because the transmitting antenna is placed on the roof of the building, the line-of-sight path can be easily blocked, which further emphasizes the importance of simulating other propagation mechanisms such as reflections and diffractions. Figure 6.2 shows line-of-sight, up to third-order reflection, and first-order diffraction paths found by the ray tracer for a receiver located at the center of the street, a typical user location in this scenario.

The left panel of Figure 6.2 highlights dominant line-of-sight and specular-reflection trajectories, while the right panel isolates first-order diffraction paths around building



(a) Line-of-sight and reflection paths. **(b)** Diffraction paths.

Figure 6.2: Propagation paths found by the ray tracer for a receiver located at the center of the street.

edges. This decomposition is useful because it already reveals which mechanisms are likely to dominate in different street regions before any field computation is performed.

Even before converting paths into channel coefficients, the geometric path set provides actionable information:

- *Multipath characterization* in which short, low-order interactions are typically the strongest contributors to received power.
- *Physical interpretability*, since each path corresponds to a concrete mechanism (wall reflection, edge diffraction, etc.), which clarifies why channel behavior changes with geometry.
- *Flexible propagation control* that allows selectively enabling or disabling specific propagation mechanisms to study their individual contributions, which is invaluable for validating models against measurements.

6.1.2 From Paths to Channel Coefficients

Once paths are identified, each interaction point is associated with an electromagnetic coefficient (reflection or diffraction), and the full per-path field transfer is assembled as described in Chapter 2 and Chapter 3. We consider a

single transmitter–receiver link at 28 GHz, with isotropic vertically polarized antennas. Scene materials are "concrete", "glass", "wood", "marble", and "brick" (ground, blue, brown, beige, and maroon buildings), with parameters from [11].

[11]: International Telecommunication Union (2025), *ITU-R Recommendation P.2040-3: Effects of building materials and structures on radiowave propagation above about 100 MHz*

For each path p, we denote by $a_p \in \mathbb{C}$ the resulting complex scalar coefficient after interaction modeling, polarization projection, and antenna factors. The associated per-path power contribution is $|a_p|^2$ (up to the normalization adopted in Chapter 2), which is the quantity used in the per-path gain plots below.

This conversion from geometric paths to complex coefficients provides several benefits:

- *Electromagnetic consistency* ensures that path delays, phases, interaction coefficients, and antenna responses are handled in one unified model.
- *Interference fidelity* is preserved because constructive and destructive superposition emerge naturally from complex-path addition.
- *Polarization handling* allows co- and cross-polarized components to be tracked separately when needed.

To model antenna arrays, two main approaches exist. The first approach is to run a single ray tracing simulation treating the array's phase center as a single point, and then reconstruct the response of each element by applying appropriate phase shifts. This phase correction is formalized through the use of steering vectors, which capture the relative phase delays across the array elements for a given path's direction of departure or arrival. Mathematically, a steering vector $\mathbf{a}(\theta, \varphi) \in \mathbb{C}^M$ represents the relative phase shifts of a plane wave arriving at (or departing from) each of the M elements of the array. For instance, for a uniform linear array of M elements aligned along an axis with inter-element spacing d_a and an arrival angle θ relative to the array broadside, the steering vector is expressed as

$$\mathbf{a}(\theta) = \begin{bmatrix} 1 & e^{-jkd_a \sin\theta} & \dots & e^{-jkd_a(M-1)\sin\theta} \end{bmatrix}^\top . \tag{6.1}$$

By aggregating the relative phases, the steering vector maps the single-point ray tracing output to each individual antenna element. The second approach is to run multiple simulations, one per array element, and combine the results. The first approach is more efficient but less flexible, as steering vectors assume the incoming wave is locally planar across the entire array, which may fail to capture path-length differences or element-specific blockage. These effects can be significant for very large arrays or in environments where rays arrive at different angles for different elements. The second approach enables more detailed modeling of array geometry and mutual-coupling effects at the cost of increased computational load. The choice between these approaches depends on the specific application and required accuracy.

In the MIMO case, the scalar coefficient a_p is therefore only a compact single-input single-output (SISO) abstraction. When the array response is kept explicit, the per-path contribution becomes matrix-valued and is naturally expressed through the outer product of the receive and transmit steering vectors, given by

$$\boldsymbol{H}_p(\tau) = a_p \mathbf{a}_{\mathrm{RX}}(\theta_p^{\mathrm{RX}}, \varphi_p^{\mathrm{RX}}) \mathbf{a}_{\mathrm{TX}}^{*}(\theta_p^{\mathrm{TX}}, \varphi_p^{\mathrm{TX}}) \delta(\tau - \tau_p), \quad (6.2)$$

exactly as in the matrix-valued CIR introduced in Section 2.10.2. This outer product aggregates the individual multipath contributions across all transmit and receive element pairs.

6.1.3 The Channel Impulse Response

The paths and their associated coefficients are aggregated into a baseband CIR representation

$$h(\tau) = \sum_{p=1}^{P} a_p \, \delta\big(\tau - \tau_p\big) \quad \text{with} \quad \tau_p = \frac{L_p}{c}. \quad (6.3)$$

Here, L_p is the path length and c is the speed of light. Each impulse encodes one path delay, while its complex weight

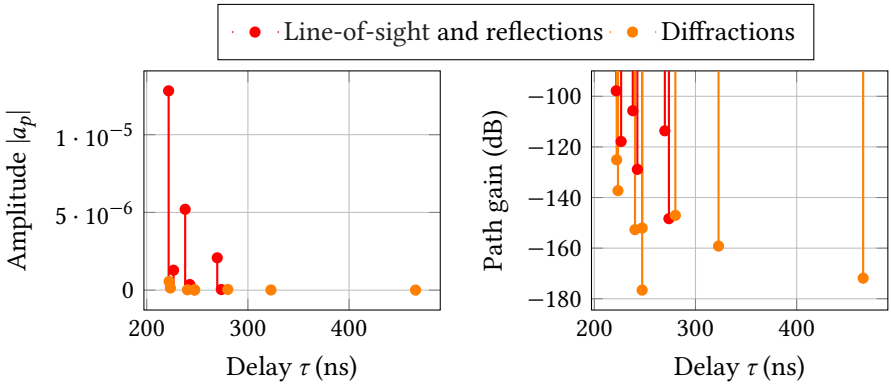


Figure 6.3: Per-path amplitude and gain versus delay for line-of-sight, reflection, and diffraction paths in the urban street canyon scenario.

a_p carries attenuation and phase.

The CIR bridges geometric ray tracing and communication-theoretic channel models. From $h(\tau)$, one can derive delay spread, coherence bandwidth, and frequency selectivity, all in a form directly consumable by link-level simulators.

To compare the individual contributions of reflection and diffraction mechanisms, Figure 6.3 shows per-path amplitudes and gain as a function of delay.

Figure 6.3 confirms the expected structure: early-arriving line-of-sight and reflection components dominate the received power, while diffraction contributes weaker but often critical delayed components that preserve connectivity in partially blocked configurations (see the coverage maps below for a more detailed view). The CIR representation provides several key benefits:

- A *direct link to signal processing*, since the CIR can be used immediately for equalization, modulation, and end-to-end communication-system simulation.
- A *frequency-selectivity analysis* enabled by delay dispersion, which provides indicators such as τ_{RMS} and approximate coherence-bandwidth scales.

- A *mechanism-isolation capability*, where enabling or disabling line-of-sight, reflection, or diffraction terms quantifies their separate impact.

6.1.4 Handling Mobility

With mobile users or dynamic environments, channel quantities become explicitly time dependent. Receiver and scatterer velocities induce Doppler shifts, which broaden the channel spectrum and reduce coherence time.

Two complementary simulation strategies are commonly used. A *snapshot-based* approach repeatedly updates object positions and recomputes valid paths and coefficients. It is physically faithful and captures path birth/death events, but it is costly for long trajectories or fine temporal sampling. A *Doppler-based* approach keeps the geometric path set fixed over a short window and applies analytical pathwise frequency shifts from object velocities. This approximation is much faster, but it remains accurate only while geometry changes are small (typically over a few wavelengths of motion). In practice, the choice follows the required temporal horizon and fidelity.

Doppler Shift and Frequency Translation

For a propagation path p with scattering points on moving objects, the accumulated Doppler shift is

$$f_p^{\mathrm{D}}(t) = -\frac{f_{\mathrm{c}}}{c}\,\mathbf{v}_{\mathrm{RX}} \cdot \hat{\mathbf{d}}_p(t) - \frac{f_{\mathrm{c}}}{c}\,\sum_i \mathbf{v}_i \cdot \hat{\mathbf{k}}_i(t), \qquad (6.4)$$

where $\mathbf{v}_{\mathrm{RX}}$ is the receiver velocity, $\hat{\mathbf{d}}_p$ is the path arrival direction, $\mathbf{v}_i$ is the velocity of the i-th scattering object, and $\hat{\mathbf{k}}_i$ is the outgoing ray direction at that point. The net Doppler shift depends on the relative velocities of transmitter, receiver, and all scattering surfaces. For the receiver, forward motion toward the transmitter induces positive (upward) frequency shifts, while receding motion induces negative

shifts. The same pattern applies to each scattering surface along the path. Multiplying by the carrier frequency f_c scales the shift: at millimeter-wave frequencies (28 GHz), a pedestrian traveling at $5\,\mathrm{m\,s^{-1}}$ can incur Doppler shifts of hundreds of hertz per path, whereas at sub-6 GHz frequencies the shifts are correspondingly smaller.

Coherence Time and Symbol-Rate Constraints

A key implication of Doppler spread is the reduction of *coherence time* T_c, i.e., the time interval over which the channel is approximately stationary. For Rician fading (when a dominant line-of-sight component coexists with multipath scattering), a common approximation is

$$T_c \approx \frac{0.423}{f_{\max}^D}, \tag{6.5}$$

where $f_{\max}^D$ is the maximum Doppler spread. When different paths carry different Doppler shifts, decorrelation accelerates and fewer symbols can reuse the same channel estimate. At high speeds or millimeter-wave carriers, coherence time may drop to microseconds, requiring frequent pilot updates and channel tracking. In low-mobility regimes, longer coherence times permit sparser channel-state-information feedback.

Applications and Benefits

Mobility-aware modeling provides several practical benefits:

- A *realistic temporal model* that captures user motion, traffic, and dynamic blockages explicitly, enabling handover analysis and beam-tracking studies.
- A *Doppler-aware waveform-design workflow* using per-path frequency shifts to quantify spectral broadening and set pilot density, modulation mode, and channel-prediction strategy.

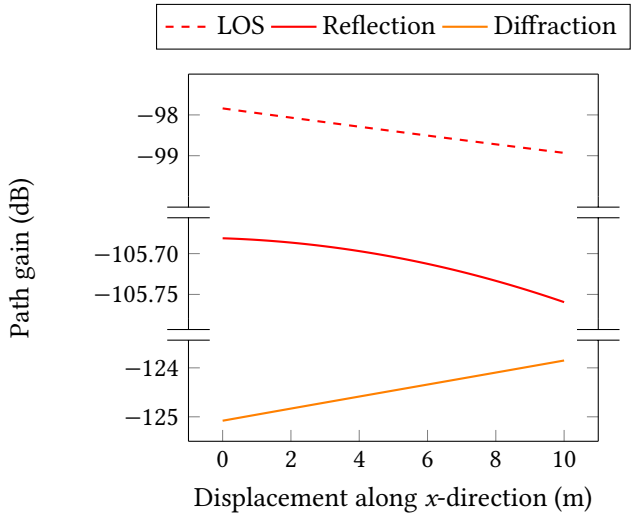


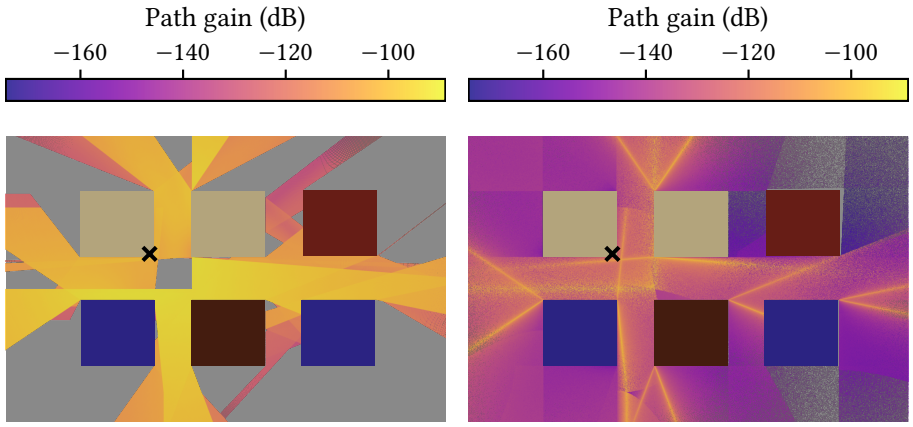
Figure 6.4: The path gain of dominant line-of-sight, reflection, and diffraction paths as a function of distance for a receiver moving along the street. The receiver starts at position $\mathbf{x}_{\mathrm{RX}} = \begin{bmatrix} 25 & 0 & 1.5 \end{bmatrix}^{\top}$ m and moves away from the transmitter toward the positive x-direction.

- A *transient channel model* that reveals path appearance/disappearance and deep fades during short blockage events.
- A *velocity-conditioned statistical perspective*, obtained by aggregating trajectories at different speeds to extract trends in Doppler spread, coherence time, and reliability.

As shown in Figure 6.4, dominant components evolve smoothly but with distinct trends: the line-of-sight term decreases with distance, reflection evolves nonlinearly due to incidence–reflection geometry, and diffraction remains weaker but structured. Even in a static urban scene, these mechanism-specific dynamics justify mobility-aware channel simulation.

6.1.5 Coverage Maps

Coverage maps—also referred to as radio maps—are obtained by evaluating channel-quality metrics over a dense grid of receiver positions. For each point $\mathbf{x}$, ray tracing yields a CIR, from which received power, delays, signal-to-interference-plus-noise ratio (SINR), achievable rate,



(a) Path gain from line-of-sight and reflection paths.

(b) Path gain from diffraction paths.

Figure 6.5: Path-gain coverage maps for the street-canyon scenario, separating line-of-sight and reflection contributions from diffraction contributions.

and outage are computed. Plotting these quantities as heat maps over the scene geometry reveals dead zones, dominant corridors, and deployment trade-offs.

As discussed in Section 2.10.2, these maps use the non-coherent addition of path powers, not the coherent summation of complex voltages. This convention suppresses small-scale fading and makes the maps more robust to tiny geometric perturbations, which is why it is the default for the coverage-map figures in this chapter.

Figure 6.5 highlights mechanism complementarity: the line-of-sight and reflection coverage map exhibits corridor-like high-gain structures driven by specular visibility, while diffraction partially fills non-line-of-sight regions with lower but useful power. Interpreting both maps jointly avoids overestimating coverage holes in specular-only analyses.

Furthermore, the diffraction-only coverage map in Figure 6.5b clearly shows the reflection and incident shadow boundaries (RSBs and ISBs) in the diffracted power, as introduced in Section 2.5.5. Because the diffracted field in

the UTD formulation is mathematically designed to compensate for the step discontinuities of the GO fields, the diffracted power itself exhibits sharp transitions and characteristic spatial patterns along these boundary lines.

Key metrics include the following definitions:

- The *coverage probability* is the fraction of the region with received power above a threshold,

$$p_{\text{coverage}} = \frac{1}{|\mathscr{A}|} \int_{\mathscr{A}} \mathbb{I}\left[P_r(\mathbf{x}) > P_{\min}\right] \mathrm{d}\mathbf{x}, \tag{6.6}$$

  where $\mathscr{A}$ is the region of interest and $P_{\min}$ is receiver sensitivity.
- The *outage probability* is the fraction experiencing insufficient signal-to-interference-plus-noise ratio,

$$p_{\text{outage}} = \frac{1}{|\mathscr{A}|} \int_{\mathscr{A}} \mathbb{I}\left[\text{SINR}(\mathbf{x}) < \gamma_{\min}\right] \mathrm{d}\mathbf{x}, \tag{6.7}$$

  where $\gamma_{\min}$ is the minimum SINR for reliable communication. This metric captures interference and noise, and therefore approximates usable (not only detectable) coverage.
- The *rate coverage* is the fraction of the region capable of supporting a target data rate,

$$p_{\text{rate}} = \frac{1}{|\mathscr{A}|} \int_{\mathscr{A}} \mathbb{I}\left[\log_2\big(1 + \text{SINR}(\mathbf{x})\big) > R_{\min}\right] \mathrm{d}\mathbf{x}, \tag{6.8}$$

  where $R_{\min}$ is the minimum required rate. This user-centric metric links propagation quality to service-level requirements.

Multi-frequency analyses are performed by repeating the same evaluation at each carrier frequency, thereby exposing frequency-dependent spatial heterogeneity.

Coverage-map-based analysis offers several practical benefits:

- A *visual planning aid* through heat maps that expose coverage gaps and candidate sites for base stations, relays, or beam reconfiguration.

- A *quantitative benchmarking tool* based on area-level metrics for objective comparison between candidate deployments.
- A *sensitivity-analysis capability* that recomputes maps under alternative assumptions (blockages, vegetation, antenna patterns) to reveal robustness limits.
- A *backhaul co-design perspective* that uses visibility and coupling patterns to co-optimize access and backhaul placement.

6.1.6 Optimizing Network Parameters

A key advantage of differentiable ray tracing is that network parameters can be optimized directly with gradient-based solvers, avoiding exhaustive search and reducing reliance on heuristics.

Optimizing Transmitter Orientation

In practical deployments, antennas are directive and their orientation can be tuned to steer energy toward target regions. With differentiable ray tracing, gradients of coverage objectives with respect to orientation angles are available automatically, enabling efficient optimization.

Using Euler angles, the transmitter orientation is parameterized by three angles $\boldsymbol{\theta} = (\alpha, \beta, \gamma)$ (roll, pitch, yaw). A common design objective is to maximize the average received power over a target region, i.e.,

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \frac{1}{|\mathscr{A}|} \int_{\mathscr{A}} P_r(\mathbf{x}; \boldsymbol{\theta})\, \mathrm{d}\mathbf{x}, \tag{6.9}$$

where $\mathscr{A}$ is the target area and the integral is approximated in practice by a receiver grid. Optimization is then performed with gradient-based methods (e.g., Adam [88]) using automatic differentiation.

[88]: Kingma et al. (2015), *Adam: A Method for Stochastic Optimization*

In Figure 6.6, the objective increases smoothly from a random initial orientation, indicating stable convergence.

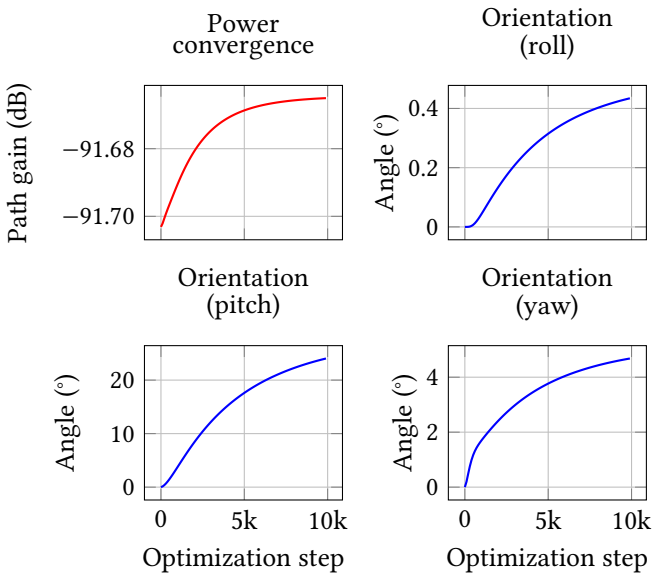


Figure 6.6: Transmitter-orientation optimization: average received power (in dB) and orientation-angle trajectories (roll, pitch, yaw) across optimization steps.

Pitch and yaw show the dominant updates, while roll changes remain limited, consistent with beam steering primarily in elevation and azimuth. The optimized orientation yields a clear increase in the average received power over the target region.

Optimizing Antenna Pattern

Beyond orientation, the radiation pattern itself can be optimized. By tuning pattern-shape parameters, radiated energy can be redistributed to better match scene geometry and user distribution. Denoting the pattern parameters by $\boldsymbol{\theta}$, we can find the antenna pattern that maximizes received power by solving

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \sum_i P_r(\mathbf{x}_i; \boldsymbol{\theta}), \tag{6.10}$$

where the sum is taken over a set of target receiver positions.

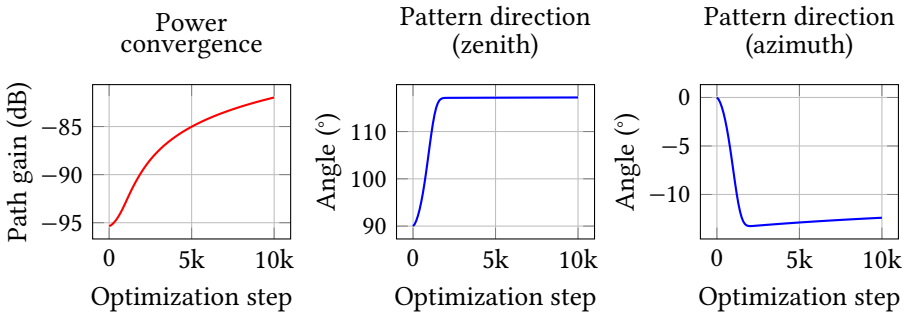


Figure 6.7: Antenna-pattern optimization: received power (in dB) and learned pattern-direction angles (zenith and azimuth) across optimization steps.

This approach supports end-to-end antenna design for a specific deployment rather than generic isolated synthesis. For example, indoor reflective settings and outdoor street canyons induce different optimal patterns. Because the optimization includes scene materials and multipath, learned patterns are often better aligned with in-situ performance.

Representative use cases include the following examples:

- A *directional SINR maximization* objective to optimize phased-array weights toward a user while suppressing known interferers.
- A *scenario-specific compact MIMO design* approach to tailor a small-form-factor array to a target indoor or outdoor deployment.
- A *null-steering strategy* to suppress dominant interferers while preserving useful multipath support.

6.2 Inverse Problems

Inverse problems reverse the standard simulation pipeline: instead of predicting measurements from a known scene, they infer unknown scene or system parameters from measured radio data. In wireless settings, these unknowns

can include user position, material properties, antenna or network configuration, and even geometric scene structure. The practical value is substantial: once calibrated from observations, the same model can improve localization, reduce planning uncertainty, maintain digital twins over time, and support sensing tasks that are difficult with purely data-driven methods.

Historically, such formulations were often limited by non-convex objectives and expensive derivative computation. Differentiable ray tracing changes this by making the full forward model end-to-end differentiable, so gradients of the measurement mismatch with respect to physical parameters are obtained automatically and exploited by modern gradient-based optimizers. Combined with GPU acceleration and automatic differentiation frameworks, this has made large-scale inverse optimization far more tractable and has driven strong recent interest in radio-frequency inverse sensing, including source localization [96, 100, 114], material calibration [95, 115], and free-form 3D reconstruction with generative priors [116]. We focus here on the first two applications, which are more mature and directly relevant to wireless communication.

[96]: Han et al. (2025), *Loki: Physical-World Adversarial Attacks on Wireless Indoor Localization via Differentiable Object Placement*

[100]: Han et al. (2025), *RayLoc: Wireless Indoor Localization via Fully Differentiable Ray-tracing*

[114]: Lequeu (2025), *Wireless indoor source localization as an inverse problem: An optimization approach via ray tracing simulations*

[95]: Castro Farfan (2025), *Towards high-fidelity radio coverage maps: calibration using real-world measurements in Sionna RT*

[115]: Hoydis et al. (2024), *Learning Radio Environments by Differentiable Ray Tracing*

[116]: Han et al. (2026), *Stereo-Fi: Free-form 3D Reconstruction via Generatively Co-Trained Inverse RF Rendering*

6.2.1 Inverse Localization

Inverse localization estimates a receiver position by fitting a geometric model to radio observations collected from anchors at known locations. Unlike classical trilateration, which relies on idealized line-of-sight distances, this formulation explicitly leverages multipath signatures, turning reflected and diffracted components into informative constraints rather than nuisance effects. This model-based approach also differs from *radio fingerprinting*, which matches observed signatures against a precomputed database. While fingerprinting is robust in complex environments, it requires an expensive offline survey and is sensitive to scene changes. In contrast, differentiable ray tracing enables continuous optimization without a static database by solving

the data-fitting problem

$$\mathbf{x}_{\mathrm{RX}} = \arg\min_{\mathbf{x}} \sum_i \|y_i - \hat{y}_i(\mathbf{x})\|^2, \tag{6.11}$$

where y_i represents measured channel features—such as power, time-of-arrival, or angle-of-arrival—and $\hat{y}_i(\mathbf{x})$ denotes the corresponding ray tracing predictions for a candidate position $\mathbf{x}$.

By providing gradients of the measurement mismatch via automatic differentiation, the framework enables efficient iterative updates toward a consistent position estimate. Although the resulting objective is typically non-convex and exhibits piecewise smoothness due to path-set transitions, smoothing strategies can effectively regularize the optimization landscape and improve convergence [114].

[114]: Lequeu (2025), *Wireless indoor source localization as an inverse problem: An optimization approach via ray tracing simulations*

In the example scenario, we consider three transmitters and one receiver. The reference position is

$$\mathbf{x}_{\mathrm{RX}}^{\mathrm{ref}} = \begin{bmatrix} 10 & 0 & 1.5 \end{bmatrix}^\top \mathrm{m}, \tag{6.12}$$

and the initialization is

$$\mathbf{x}_{\mathrm{RX}}^{(0)} = \begin{bmatrix} 0 & -10 & 1.5 \end{bmatrix}^\top \mathrm{m}. \tag{6.13}$$

Multiple transmitters provide spatial diversity and richer multipath constraints. We minimize the mean squared error between measured and predicted received power over all transmitters. Iterative gradient updates converge to a local minimum, as shown in Figure 6.8.

Ray-tracing-based inverse localization provides several key benefits:

- A *multipath-exploitation advantage* that retains useful accuracy in non-line-of-sight settings where classical range-based methods degrade.
- An *environment-awareness signal*, where mismatch patterns can indicate model drift (new obstacles, changed layouts) and trigger recalibration.

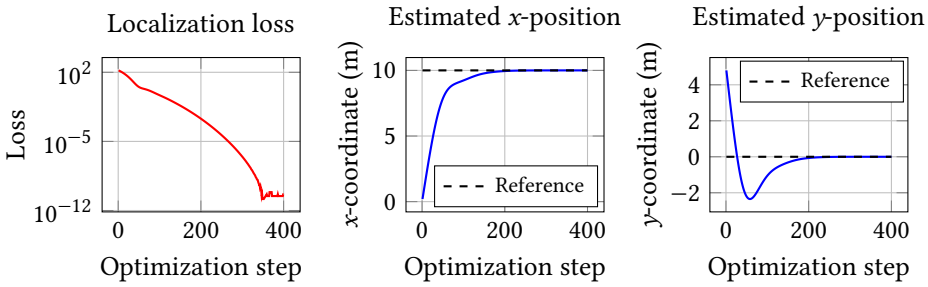


Figure 6.8: Inverse-localization optimization: loss convergence and coordinate trajectories for the receiver position estimate.

- A *differentiable refinement and security-analysis workflow* that uses gradients for both joint model refinement and analysis of adversarial environmental perturbations [96].

[96]: Han et al. (2025), *Loki: Physical-World Adversarial Attacks on Wireless Indoor Localization via Differentiable Object Placement*

A few representative use cases include the following examples:

- *Indoor positioning* in warehouses, factories, and malls where satellite localization is unavailable.
- *Emergency-response applications* that benefit from robust tracking in obstructed or damaged buildings with heavy multipath.
- *Vehicular localization* in urban canyons, where satellite connectivity is intermittent and V2X or cellular signals are informative.

6.2.2 Radio Materials Calibration

Radio-material calibration addresses a central modeling gap: electromagnetic parameters used in simulation are often uncertain, site dependent, and time varying. Relative permittivity, conductivity, and effective roughness directly shape reflection, absorption, and scattering, so even moderate parameter mismatch can produce a significant prediction error. The calibration objective is to recover these

parameters from channel measurements by solving

$$\boldsymbol{\theta}_{mat}^{\star} = \arg\min_{\boldsymbol{\theta}_{mat}} \sum_{i} \| P_{\text{meas},i} - \hat{P}_{\text{sim},i}(\boldsymbol{\theta}_{mat}) \|^2, \tag{6.14}$$

where $\boldsymbol{\theta}_{mat}$ denotes candidate material parameters. Here, $P_{\text{meas},i}$ is observed received power at location i, and $\hat{P}_{\text{sim},i}(\boldsymbol{\theta}_{mat})$ is the corresponding ray tracing prediction. Because the forward model is differentiable, gradients with respect to $\boldsymbol{\theta}_{mat}$ are obtained automatically and used in efficient gradient-based solvers.

This differentiability also enables the joint fitting of not only material parameters but also antenna and scattering-model parameters from channel observations [115].

[115]: Hoydis et al. (2024), *Learning Radio Environments by Differentiable Ray Tracing*

Material calibration can be performed at multiple abstraction levels:

- A *per-material* calibration that fits one ϵ_r and σ pair per material class (e.g., concrete, wood, glass).
- A *per-surface* calibration that fits distinct parameters for non-uniform or weathered surfaces, with neural fields capturing continuous spatial variation.
- A *frequency-dependent* calibration that performs separate or joint fitting across carrier frequencies to capture dispersion.

An illustrative calibration example, inspired by the Sionna RT documentation [37], considers one unknown material (concrete) with unknown conductivity σ. Using a literature value as ground truth ($\sigma = 0.63\,\text{S}\,\text{m}^{-1}$), we minimize the mean squared error between measured and simulated received power over a single location. Starting from an arbitrary low conductivity, the optimization converges toward parameters that better match the observations, eventually reaching the ground truth (Figure 6.9).

[37]: Hoydis et al. (2023), *Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling*

Ray-tracing-based material calibration offers several benefits:

- A *reduction in uncertainty* from site-specific calibration that replaces coarse literature approximations.

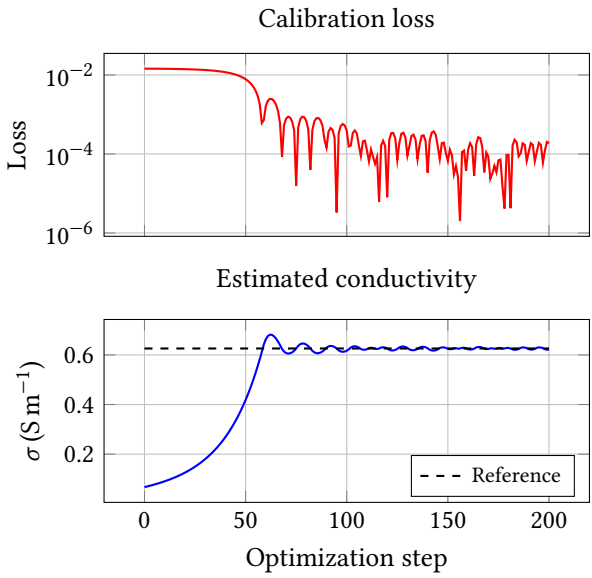


Figure 6.9: Radio-material calibration: loss convergence and estimated conductivity across optimization steps.

- A *digital-twin maintenance pathway* using periodic updates to track aging, moisture, and other environmental drifts.
- A *lower measurement burden* by relying on non-intrusive channel measurements instead of direct material probing.
- A set of *reusable fidelity gains* when calibrated parameters improve downstream studies across scenarios and frequencies.

Practical use cases for radio-material calibration include the following examples:

- *Indoor small-cell planning* that starts by calibrating venue-specific materials before optimizing placement, power, and interference control.
- *Urban macro-cell planning* that builds representative city material libraries from multi-site calibration, including crowd-sourced data at scale [95].
- *Anomaly detection* using persistent simulation/measurement mismatch as an indicator of structural scene changes.

[95]: Castro Farfan (2025), *Towards high-fidelity radio coverage maps: calibration using real-world measurements in Sionna RT*

- *Seasonal adaptation* relying on repeated calibration to track wet/dry and occupancy effects over time.

6.2.3 System Constraints in Practical Deployments

While the forward and inverse workflows demonstrate the power of differentiable ray tracing, their application in real-world networks is subject to several system-level constraints:

- Tight *latency budgets* in real-time radio resource management and beamforming control loops, which operate on millisecond timescales, make simulating a full 3D environment at each step computationally prohibitive, relegating ray tracing to offline planning or surrogate training.
- A carefully tuned *measurement cadence* that dictates how quickly model drift (e.g., due to environmental changes or seasonal variation) can be detected and corrected, balancing the delay of recalibration against network overhead.
- Hard *compute limits* from available hardware resources that constrain the spatial grid resolution for coverage maps, the number of propagation paths, and the interaction depth (e.g., limiting diffraction to one bounce) to avoid memory and execution bottlenecks, particularly in reverse-mode automatic differentiation.
- Inherent *accuracy–cost trade-offs* when choosing modeling options (e.g., whether to include diffuse scattering or detailed antenna patterns), forcing engineers to select physical mechanisms based on the accuracy requirements of the specific application.

6.3 Conclusion

This chapter has illustrated the practical scope of differentiable ray tracing in both forward and inverse workflows. On the forward side, geometric scene descriptions and network settings are converted into channel observables such as CIRs, received power, and coverage maps. Because these quantities are differentiable, the transmitter pose, orientation, and pattern parameters can be optimized with gradient-based solvers instead of trial-and-error tuning. Mobility modeling further extends this pipeline to time-varying channels and Doppler-aware performance analysis.

On the inverse side, localization and material calibration show how measurements can be fused with simulation to infer latent variables and maintain accurate digital twins. Multipath-aware localization improves robustness in the absence of a direct line-of-sight path, while calibration adapts material parameters as deployments evolve. Together, these inverse workflows close the loop between prediction and observation.

While ray tracing techniques are mainly used for offline planning and analysis, their online deployment remains shaped by practical constraints. *Latency budgets* limit the direct use of full ray tracing in real-time control loops, motivating surrogate models. *Measurement cadence* determines how quickly model drift can be detected and corrected. *Compute limits* constrain spatial/angular resolution and propagation richness, requiring selective model complexity. *Accuracy–cost trade-offs* therefore remain central when deciding which mechanisms to include for a given application.

These trade-offs determine whether a workflow is best run offline (planning), periodically (recalibration), or in near real time via learned approximations. The next chapter (Chapter 7) builds on this foundation with learning-assisted path search for scalability and multipath-lifetime analysis for transient behavior.

Advanced Topics in Propagation Modeling

7

After presenting the common ray tracing use cases in Chapter 6, this chapter focuses on advanced research directions in which ray tracing is combined with machine learning to improve computational efficiency and to investigate the spatial structure of multipath propagation.

7.1 Machine-Learning-Assisted Ray Tracing

As already motivated in Chapter 3, ray tracing faces combinatorial challenges with exhaustive candidate enumeration: the number of potential paths grows exponentially with the number of objects and interaction order, while most candidates are physically invalid. Beyond the purely geometrical acceleration techniques presented in Chapter 5, recent advances in machine learning have opened new possibilities across many research fields, and ray tracing is no exception. A natural question arises: can a neural network predict which candidates are likely to be valid, and thus help the ray tracing engine focus its computational resources on promising paths? To answer this question, this section presents a generative path sampler [79, 117] that assists path discovery while keeping deterministic geometric validation and electromagnetic-field evaluation unchanged. In short, the learned model replaces the exhaustive candidate-enumeration stage of the ray tracing pipeline, while geometric validation and electromagnetic evaluation remain physics-based, as illustrated in Figure 7.1.

[79]: Eertmans et al. (2026), *Transform-Invariant Generative Ray Path Sampling for Efficient Radio Propagation Modeling*

[117]: Eertmans et al. (2025), *Towards Generative Ray Path Sampling for Faster Point-to-Point Ray Tracing*

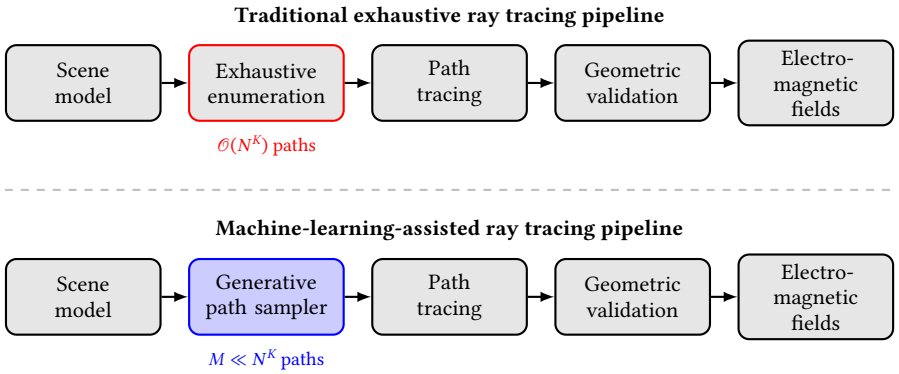


Figure 7.1: Classical versus ML-assisted ray tracing pipelines. The exhaustive candidate enumeration stage is replaced by a generative sampler, while geometric validation and electromagnetic evaluation remain physics-based.

7.1.1 State-of-the-Art Context and Positioning

The application of machine learning to radio propagation modeling has accelerated in recent years, motivated by the fundamental trade-off between efficiency and accuracy. Traditional empirical models are computationally fast but often insufficiently accurate for modern wireless systems, while full-wave electromagnetic solvers and high-fidelity ray tracing are precise yet prohibitively expensive at scale. Machine learning offers a compelling middle ground: approximating propagation phenomena with the speed of neural inference. Moreover, the emergence of differentiable programming frameworks has opened new research directions toward end-to-end optimization of wireless systems, spurring interest in differentiable surrogates that can replace or augment conventional simulation tools.

Current machine learning approaches for radio propagation can be organized into several families based on their design philosophy and architectural choices.

Direct Field and Channel Learning

Much of the existing literature frames propagation modeling as a supervised learning task, in which neural networks map environmental features directly to channel metrics.

Standard MLP and CNN-based approaches Early methods employed multilayer perceptrons (MLPs) to predict path loss or received power from geometric and environmental features [118–120]. For example, MLP-based models demonstrated reasonable accuracy in urban and millimeter-wave settings, while convolutional neural network (CNN)-based approaches were trained on satellite imagery for path loss prediction at 2.6 GHz. However, these approaches typically require extensive training data and struggle to generalize across different environments or operating conditions, as they are inherently tied to the specific frequencies and material properties of their training distributions.

[118]: Wu et al. (2020), *Artificial Neural Network Based Path Loss Prediction for Wireless Communication Network*

[119]: Popoola et al. (2019), *Determination of Neural Network Parameters for Path Loss Prediction in Very High Frequency Wireless Channel*

[120]: Thrane et al. (2020), *Model-Aided Deep Learning Method for Path Loss Prediction in Mobile Communication Systems at 2.6 GHz*

Neural radiance fields (NeRFs) and volumetric representations Inspired by advances in computer graphics, researchers have recently explored using neural radiance fields to represent radio-frequency propagation within complex environments [87, 121, 122]. For instance, NeRF^2 introduces a joint training strategy using both synthetic and real-world data to construct a continuous volumetric channel representation. This concept is extended by R-NeRF to model the dynamic scattering behaviors in RIS-assisted networks. While these approaches can capture complex channel dynamics with impressive precision, they suffer from strict scene-dependence, necessitating a complete, computationally intensive retraining cycle whenever the environment changes. More recently, 3D Gaussian splatting (GS) models such as wireless radiation field (WRF)-GS and radio frequency (RF)-3DGS have significantly reduced this overhead, enabling training within minutes and signal queries in milliseconds [123, 124]. Yet,

[87]: Yang et al. (2024), *R-NeRF: Neural Radiance Fields for Modeling RIS-enabled Wireless Environments*

[121]: Zhao et al. (2023), *NeRF2: Neural Radio-Frequency Radiance Fields*

[122]: Shen et al. (2025), *NeRF-APT: A New NeRF Framework for Wireless Channel Prediction*

[123]: Wen et al. (2025), *Neural Representation for Wireless Radiation Field Reconstruction: A 3D Gaussian Splatting Approach*

[124]: Zhang et al. (2026), *RF-3DGS: Wireless Channel Modeling With Radio Radiance Field and 3D Gaussian Splatting*

these neural surrogates do not substitute for ray tracing engines; rather, they serve as complementary downstream models that depend on ray-traced datasets for initial training. Consequently, developing faster path sampling techniques directly accelerates the generation of the training data required to bootstrap these representations.

Alternative and Hybrid Approaches

Recognizing the challenges of end-to-end field learning, alternative strategies have focused on narrower components of the propagation pipeline or hybrid methods that combine neural networks with physics.

Channel impulse response and trajectory learning Rather than predicting single metrics like path loss, some models target richer intermediate quantities. To estimate the channel impulse response over spatial regions, physics-informed networks have been developed, capturing both spatial and temporal characteristics [77]. Other work formulates trajectory generation as a sequential decision problem, treating the synthesis of ray bounces as a sequence-modeling task [78]. The SANDWICH model, briefly mentioned in Chapter 3, is a fully differentiable surrogate that learns to generate sequences of bounces, aiming to completely replace the ray tracing simulator.

[77]: Wang et al. (2025), *A Physics-Informed Deep Ray Tracing Network for Regional Channel Impulse Response Estimation*

[78]: Jin et al. (2025), *SANDWICH: Towards an Offline, Differentiable, Fully-Trainable Wireless Neural Ray-Tracing Surrogate*

Neural scene augmentation An alternative strategy enriches the geometric scene description through neural representations. For instance, neural radiance fields can represent complex objects, allowing standard ray tracing engines to query these representations to account for scattering and diffraction effects that are difficult to model with traditional primitives [125].

[125]: Chen et al. (2025), *Radio Frequency Ray Tracing with Neural Object Representation for Enhanced RF Modeling*

Fundamental Limitations of Current Approaches

Despite these advances, direct field-learning methods face several fundamental challenges that limit their adoption in general-purpose simulation tools.

Highly Oscillatory Field-Mapping Problem Models attempting to learn electromagnetic fields directly face a highly sensitive mapping: tiny geometric or frequency perturbations cause significant field fluctuations due to constructive and destructive interference. Learning such highly oscillatory, non-convex spatial functions demands large networks and vast training datasets, often resulting in poor generalization.

Lack of Invariance and Generalization Most direct learning models lack invariance to geometric transformations such as rotation or scaling. Furthermore, models are tightly coupled to their training frequencies and material properties. A model trained at 2.4 GHz in an office environment cannot be applied to 28 GHz street canyons without complete retraining. Only recently have researchers explored geometric invariance properties using graph attention architectures [126].

[126]: Hehn et al. (2024), *Geometric Wireless Simulation with Equivariant Transformers*

Loss of Physical Interpretability End-to-end black-box models discard valuable path-level information. For emerging applications—sensing, localization, and beam management—knowing *which* paths (reflections, diffractions) contribute to the signal is often as important as the final signal strength. This loss of interpretability limits their utility in diagnostic and optimization workflows.

Computational Efficiency Paradox While neural inference can be fast, querying large networks may paradoxically be slower than highly optimized geometric ray tracing, especially in simple scenarios. Practical deployment

requires that inference overhead be negligible relative to the physics engine it assists.

Intelligent Path Sampling: A Complementary Approach

These limitations suggest a different role for machine learning: *guiding* the physics engine rather than replacing it. If neural models learn to *identify* likely valid candidates instead of predicting fields directly, the learning problem becomes less dependent on frequency and material parameters. The model can then focus on geometric regularities, while rigorous field computation remains in the physics solver. This design philosophy motivates the generative path sampler presented next.

7.1.2 Methodology

In this subsection, we show how learning can be integrated into the tracing pipeline while preserving physics-based validation. We proceed through the following components: path construction as sequential sampling, transform-invariant scene encoding, generative flow network (GFlowNet)-based policy learning, training-objective optimization, and sparse-reward stabilization.

From Exhaustive Search to Sequential Sampling

As discussed in earlier chapters, exhaustive point-to-point tracing is equivalent to exploring all branches of the image tree. For a scene with N candidate objects and interaction order K, this exploration scales combinatorially, while only a small fraction of candidates satisfy geometric constraints.

Recent approaches reformulate candidate construction as a sequential decision process. Starting from an empty candidate, one object is appended at each step until a complete

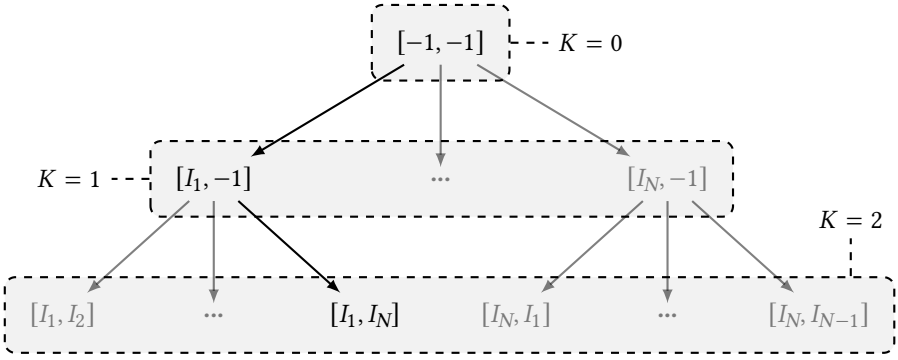


Figure 7.2: Sequential construction of complete path candidates up to interaction order K, adapted from the image tree in Figure 5.2 with incomplete path candidate notation for intermediate states.

length-K sequence is obtained (Figure 7.2). In this framework, a state corresponds to a partial path $\boldsymbol{I}$, and each action selects the next object index to extend that path. The 2D geometry and resulting decision tree in Figure 7.3a and Figure 7.3b (combined in Figure 7.3) illustrate how geometric constraints prune infeasible branches. This reformulation converts the exhaustive graph traversal into policy-guided sampling over the same search space.

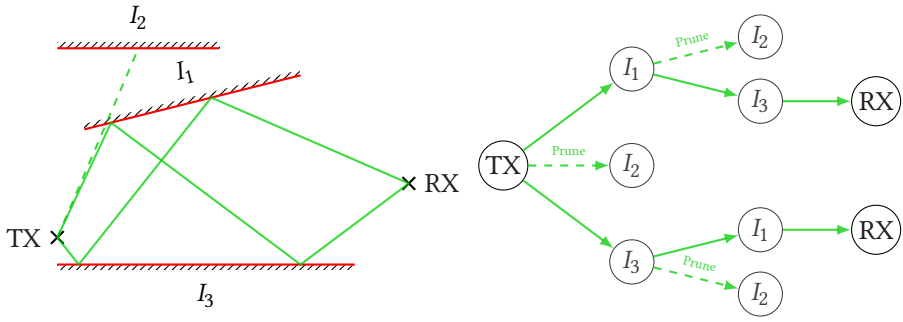
Transform-Invariant Scene Encoding

To reduce sensitivity to absolute coordinates, scene geometry is mapped to a canonical TX-RX frame. With transmitter and receiver positions $\mathbf{x}_{\mathrm{TX}}$ and $\mathbf{x}_{\mathrm{RX}}$, the basis vector aligned with the line-of-sight is defined by

$$\hat{\mathbf{w}} = \frac{\mathbf{x}_{\mathrm{RX}} - \mathbf{x}_{\mathrm{TX}}}{s} \quad \text{with} \quad s = \|\mathbf{x}_{\mathrm{RX}} - \mathbf{x}_{\mathrm{TX}}\|. \tag{7.1}$$

To preserve a consistent vertical orientation, the remaining orthonormal axes are constructed from the global vertical unit vector $\hat{\mathbf{e}}_z = [0, 0, 1]^\top$ as

$$\hat{\mathbf{u}} = \frac{\hat{\mathbf{w}} \times \hat{\mathbf{e}}_z}{\|\hat{\mathbf{w}} \times \hat{\mathbf{e}}_z\|}, \qquad \hat{\mathbf{v}} = \hat{\mathbf{w}} \times \hat{\mathbf{u}}. \tag{7.2}$$



(a) 2D geometry. **(b)** Decision tree.

Figure 7.3: Simple 2D geometry and resulting decision tree for a path tracing scenario where some interactions can be pruned because an object is unreachable.

When $|\hat{\mathbf{w}} \cdot \hat{\mathbf{e}}_z| \approx 1$ (near-vertical TX–RX alignment), this construction becomes singular because $\hat{\mathbf{w}} \times \hat{\mathbf{e}}_z \approx \mathbf{0}$. In that case, a fallback reference axis (e.g., $\hat{\mathbf{e}}_x$) is used to build the orthonormal basis and avoid division by zero.

With the rotation matrix $\boldsymbol{R} = [\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}]^\top$, each scene vertex $\mathbf{x}_i$ is transformed as

$$\mathbf{x}_i' = \boldsymbol{R}\left(\frac{\mathbf{x}_i - \mathbf{x}_{\mathrm{TX}}}{s}\right), \tag{7.3}$$

so that the transmitter maps to $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$ and the receiver to $\begin{bmatrix}0 & 0 & 1\end{bmatrix}^\top$. This normalization improves transfer across scenes by factoring out global translation, scale, and azimuthal orientation.

GFlowNet-Based Path Sampling

The neural architecture (detailed in Figure 7.4) consists of three main components: an object encoder, a state encoder, and a flow head. The object encoder processes the geometric features of each scene primitive (transformed to the canonical frame), which in this case consist only of the object's vertex coordinates. Because the set of candidate ob-

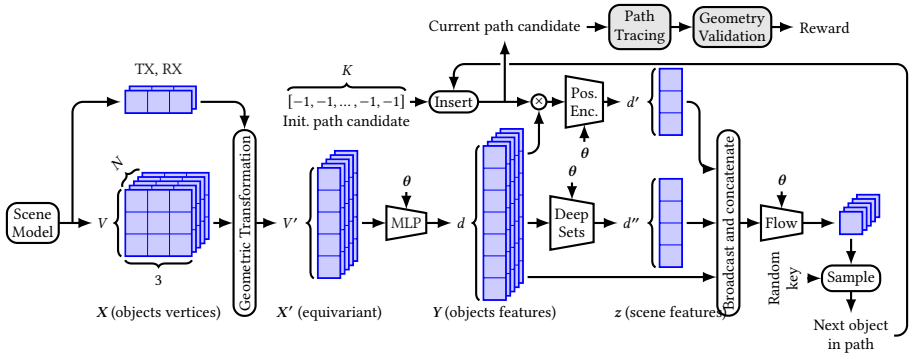


Figure 7.4: Neural architecture of the generative path sampler (object/state/scene encoders + flow head).

jects is inherently unordered, a *Deep Sets* architecture [127] is used to encode the objects into a single scene embedding. This choice is motivated by the need for permutation invariance, ensuring that the model's predictions do not depend on the arbitrary indexing of objects in the scene. Compared to attention-based architectures such as Transformers, Deep Sets offer superior scalability—linear $\mathcal{O}(N)$ complexity versus quadratic $\mathcal{O}(N^2)$—and avoid the need for artificial positional encodings, which is particularly beneficial when handling thousands of objects. Through training, the MLPs within these encoders learn high-level spatial embeddings that capture the relative importance of objects for potential interactions based on their geometry and distance from the TX–RX axis.

[127]: Zaheer et al. (2017), *Deep Sets*

Sampling is modeled with a GFlowNet over partial candidates. The goal is to learn a forward probability distribution (called the *policy*) $\pi(\boldsymbol{I}' \mid \boldsymbol{I})$ such that complete valid paths are sampled with probability mass proportional to a reward function expressed as

$$P(\boldsymbol{I}) \propto R(\boldsymbol{I}), \tag{7.4}$$

for terminal states $\boldsymbol{I}$. In this context, a binary reward captures path validity and is defined as

$$R(\boldsymbol{I}) = \begin{cases} 1, & \text{if path tracing(TX}, \boldsymbol{I}, \text{RX) is geometrically valid,} \\ 0, & \text{otherwise.} \end{cases} \tag{7.5}$$

Although the reference study uses a strictly binary reward, it may be possible to use a continuous reward. For example, the smoothing mechanism described in Section 4.4 could be applied to provide more informative feedback for near-valid candidates, thereby improving training dynamics.

Training enforces flow conservation on visited states through the constraint

$$F(\text{parent}(\boldsymbol{I}) \to \boldsymbol{I}) = R(\boldsymbol{I}) + \sum_{\boldsymbol{I}' \in \mathscr{C}(\boldsymbol{I})} F(\boldsymbol{I} \to \boldsymbol{I}'), \tag{7.6}$$

where $\mathscr{C}(\boldsymbol{I})$ denotes the children of state $\boldsymbol{I}$. The induced policy is then expressed as normalized outgoing flow values

$$\pi(o_i \mid \boldsymbol{I}) = \frac{F(\boldsymbol{I} \to \mathscr{T}(\boldsymbol{I}, o_i))}{\sum_j F\big(\boldsymbol{I} \to \mathscr{T}(\boldsymbol{I}, o_j)\big)}, \tag{7.7}$$

where $\mathscr{T}(\boldsymbol{I}, o_i)$ denotes the child state reached by appending object o_i to state $\boldsymbol{I}$.

This mechanism pushes probability mass toward branches that lead to valid complete paths while preserving stochastic exploration. After successful training, the architecture in Figure 7.4 will only sample geometrically valid candidates. This effect is visible in Figure 7.3, where the policy suppresses invalid branches and concentrates computation on promising ones.

Training Objective and Optimization

In practice, the flow-conservation constraint is optimized through a mean-squared flow-matching loss accumulated over sampled trajectories. At each training iteration, scenes are drawn from a scenario generator (or replay memory), transformed into the canonical frame, and used to sequentially sample one or more candidates up to order K. The

object encoder, scene encoder, path-state encoder, and flow head are trained jointly via gradient descent.

Because rewards are sparse at higher orders, convergence quality depends strongly on exploration and replay design. The training loop in Figure 7.5 therefore combines policy-based sampling, replay sampling, and geometric validation feedback. In this context, the model is trained on a *class of scenes*, which we define as a collection of environments sharing similar geometric and topological statistics. For instance, the urban street-canyon class represents outdoor environments with parallel building rows of varying heights and widths, whereas an indoor-office class would consist of interconnected rooms and corridors. By learning on randomized instances within a class, the model generalizes to unseen snapshots of the same environment type, rather than memorizing a single fixed map.

Training Stabilization Mechanisms

Because valid paths are exceptionally sparse at high interaction orders, stable learning necessitates careful algorithmic design. To this end, five complementary mechanisms have been investigated:

- The *successful-experience replay buffer* allows previously successful scene-path pairs to be revisited during training, mitigating collapse toward trivial low-flow policies.
- The *uniform exploratory policy (ϵ-greedy)* samples actions from a uniform distribution $\mathcal{U}$ with probability ϵ rather than from the learned policy, maintaining discovery of unseen valid branches.
- The *physics-based action masking* removes geometrically impossible next interactions from the policy support by leveraging physical constraints (see Section 5.3.2).
- The *distance-based weighting* reweights outgoing flows using distance priors to bias selection toward

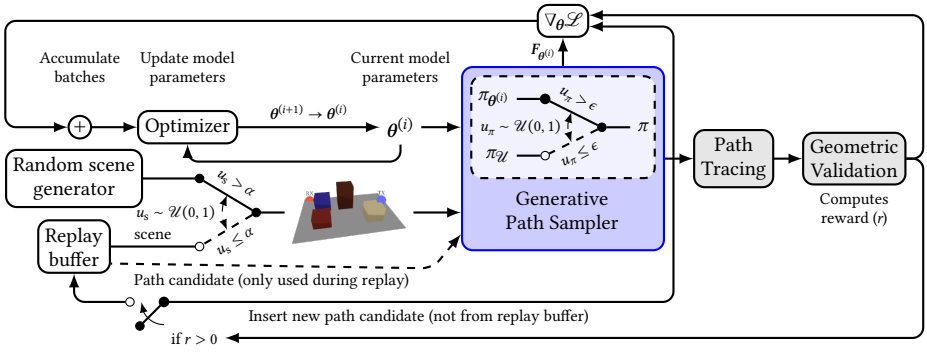


Figure 7.5: Training loop with replay sampling, exploratory policy, and path validation feedback.

geometrically plausible next interactions, though empirical gains are scenario-dependent.

- Finally, the *transmitter–receiver symmetry replay* injects reciprocal path pairs during training to encourage symmetry-aware sampling, although reported improvements are not systematic across all settings.

The overall training procedure is detailed in Figure 7.5, while the action masking and distance-based weighting mechanisms are illustrated in Figure 7.6.

These stabilization mechanisms collectively address sparse rewards and enable the network to discover long-range valid paths even at high interaction orders.

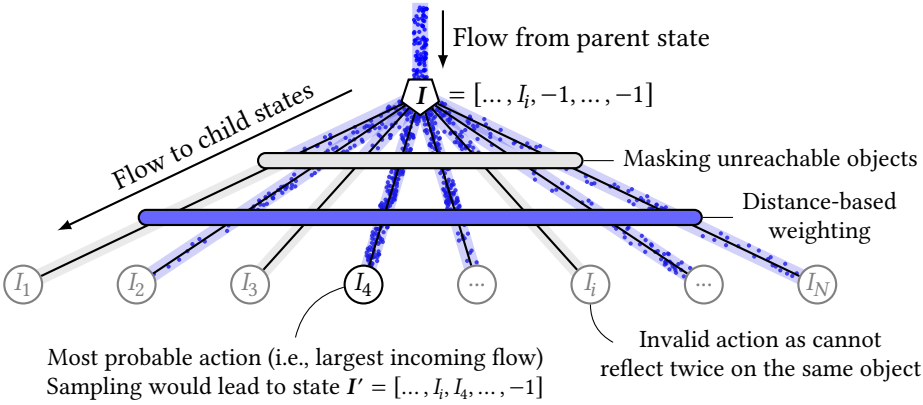


Figure 7.6: Action masking used to prune physically invalid transitions during sampling.

7.1.3 Study Summary and Main Results

The presented framework has been evaluated on the Sionna RT urban street-canyon scenario presented in Chapter 6 (illustrated in Figure 6.1), with randomized scene variations and interactions up to third order. Its training relied on dynamic scene generation to promote learning of geometric regularities rather than memorization of fixed coordinates.

Performance Metrics

Two complementary metrics are used to evaluate the performance of the generative path sampler for a given *sampling budget* M (the maximum number of path candidates generated by the model per receiver):

- The *accuracy* is defined as the ratio of valid sampled candidates to total sampled candidates (up to M), quantifying the precision of the sampler.
- The *hit rate* is the ratio of unique valid paths discovered by the sampler (within the M candidates) to the full set of valid paths (estimated via exhaustive tracing), measuring coverage of the valid-path set.

These metrics capture different behaviors: high accuracy can be achieved by repeatedly sampling a small subset of valid paths, whereas a high hit rate requires path diversity and broader exploration.

Ablation Results

Table 7.1 summarizes the main findings reported in the reference study [79]. The replay buffer is the dominant component for stability: without replay, third-order training frequently collapses (zero accuracy and hit rate in several settings), whereas replay-enabled configurations reach substantially better third-order performance. Exploratory sampling (ϵ-greedy) further improves coverage, with the best third-order hit rate obtained for the combined replay and exploration setting (67.5 %).

[79]: Eertmans et al. (2026), *Transform-Invariant Generative Ray Path Sampling for Efficient Radio Propagation Modeling*

Action masking provides moderate gains, especially at higher orders, while distance-based weighting is generally detrimental in the tested setup, often degrading both accuracy and hit rate. Enforcing explicit TX-RX symmetry does not yield systematic improvements and can hurt third-order performance. For training-scenario ablations, models trained on canyon-focused (referred to as canyon-only (CO)) data with optional ground (OG) achieve the strongest overall performance, reaching a 97.3 % hit rate at $K = 1$, 92.8 % at $K = 2$, and 70.7 % at $K = 3$.

Study of Intermediate Representations and Reciprocal Behavior

In order to understand how the neural network maps spatial environments and to substantiate the interpretability of the generative path sampler, it is helpful to study the model's internal representations. A fundamental question in machine-learning-assisted radio propagation modeling is how physical reciprocity is reflected in the learned features. According to the principles of electromagnetics, swapping the positions of the transmitter and receiver

Table 7.1: Final accuracy and hit rate values for the following ablation studies across interaction orders (K): (A) replay buffer and exploratory policy, (B) action masking and distance-based weighting, (C) symmetry enforcement, and (D) training scenario. Accuracy is the percentage of valid paths among sampled candidates; hit rate is the percentage of valid paths found among all valid paths. The acronyms, ordered by appearance in the table, are: B (baseline), R (replay buffer), ϵ (exploratory policy), AM (action masking), DW (distance-based weighting), RS (replaying symmetrical), CO (canyon-only), OG (optional ground), WS (whole scene).

Study	Case	$K = 1$		$K = 2$		$K = 3$	
		Acc. (%)	**H.R. (%)**	**Acc. (%)**	**H.R. (%)**	**Acc. (%)**	**H.R. (%)**
A	B	88.9	96.0	68.5	89.8	0.00	0.00
	R	86.5	96.8	62.0	93.4	26.2	33.8
	ϵ	**90.1**	96.5	**69.2**	85.8	*0.0*	*0.0*
	R + ϵ	85.2	**97.0**	63.2	**94.0**	**42.9**	**67.5**
B	B	**87.0**	96.5	58.2	94.4	39.7	58.4
	AM	85.2	**97.3**	**60.1**	**94.5**	**45.6**	**65.5**
	DW	7.7	31.8	*0.5*	3.6	30.8	49.3
	AM + DW	*1.0*	*1.0*	*0.9*	3.8	*0.1*	*0.5*
C	B	**86.7**	**97.5**	61.6	93.3	**45.2**	**65.8**
	RS	86.2	97.0	**62.2**	**94.3**	11.6	11.3
D	CO	86.3	96.8	59.0	88.7	38.2	53.0
	CO + OG	**88.6**	**97.3**	**62.0**	**92.8**	**47.0**	**70.7**
	WS	61.8	84.3	37.0	73.1	23.0	34.1
	WS + OG	62.5	85.7	37.0	63.8	27.7	42.4

must yield the exact same physical ray paths. However, since the generative path sampler constructs paths autoregressively from the transmitter, and the geometric transform used in the model is not invariant to transmitter-receiver swaps, sequential object selection is direction-dependent.

To analyze this behavior, we evaluate the trained model on two configurations in a representative 3D street canyon scene: a *basic scene* (with the transmitter and receiver at their original positions) and a *swapped scene* (where the transmitter and receiver positions are exchanged). The scene contains five buildings (named *Building 1* to *Building 5*) and the ground plane (named *Floor*), as illustrated in Figure 7.7. This configuration represents one of the possible randomized variations of the street-canyon scenario shown in Figure 6.1 and is representative of the environ-

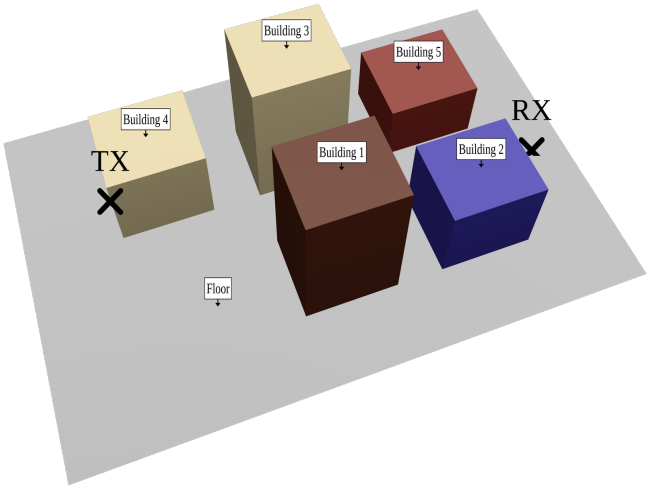


Figure 7.7: 3D idealized street canyon scene with annotated object identifiers used for the study of intermediate representations.

ment class used for model training and testing. In this specific randomized snapshot, one building (*Building 0*) was randomly removed, leaving five buildings instead of the original six.

For each configuration, we extract three key internal outputs from the trained models of interaction orders $K \in \{1, 2, 3\}$: the object embeddings, the scene embeddings, and the step-by-step selection probability vectors.

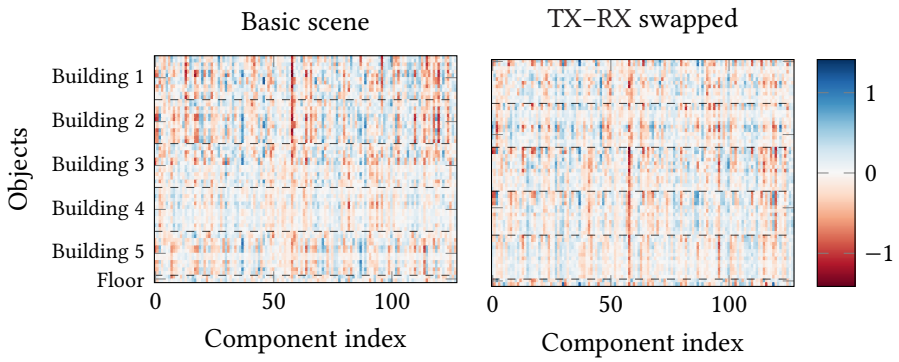
Object Embeddings (Y) The object encoder maps the canonical coordinates of the active triangles in the mesh to a 128-dimensional embedding space. The resulting embedding matrix Y contains one row per triangle in the scene, where the i-th row stores the embedding vector $\boldsymbol{y}_i$ of the corresponding triangle primitive. In other words, each horizontal strip in the matrix represents the learned features of a single object, while the columns correspond to the 128 embedding dimensions. In the canonical frame, the transmitter is always mapped to $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$ and the receiver to $\begin{bmatrix}0 & 0 & 1\end{bmatrix}^\top$. Swapping the transmitter and receiver shifts the origin and reverses the orientation of the reference axis, causing the canonical coordinates of all scene objects to change.

Figure 7.8 compares the resulting object embedding matrices for the basic and swapped scenes across orders 1, 2, and 3. The horizontal dashed lines indicate the boundaries between physical objects, with each object consisting of multiple triangles. These delimitations are only a visual aid: the model receives triangle-level geometry and is not given explicit information about which triangles belong to the same building. We observe that:

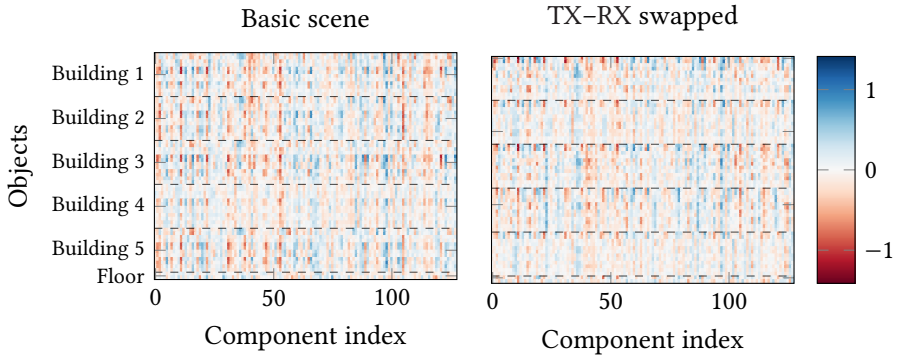
- Triangles belonging to the same object exhibit highly coherent and uniform embedding values across the 128 dimensions. This indicates that the object encoder successfully aggregates local triangle geometries into object-level spatial features.
- Under the TX–RX swap, the activation patterns across the embedding dimensions change completely. This demonstrates that the encoder does not memorize invariant object identities; instead, it learns relative spatial representations tied to the canonical TX–RX propagation axis.

Furthermore, certain objects (such as Building 4 for the basic scene and Building 5 for the swapped scene, in the order $K = 1$ embedding) exhibit faded, near-zero activations across the embedding dimensions. This occurs because these buildings do not lie on any geometrically valid first-order reflection paths between the transmitter and receiver. The object encoder learns to assign neutral, zero-valued feature vectors to these irrelevant components, effectively pruning them from downstream path selection. For $K = 2$, a similar pattern can be observed for the same buildings. The higher the interaction order, the more objects can possibly lie on reflection paths, and hence the less pronounced this effect becomes.

Scene Embeddings The scene encoder combines the individual object embeddings into a single 128-dimensional vector, representing the global scene state. Figure 7.9 stacks the scene embedding vectors for the basic and swapped configurations. Despite the coordinate transformation and

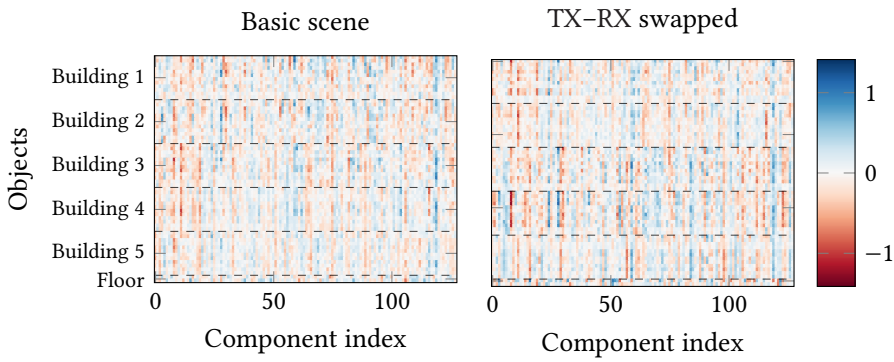


(a) Interaction order $K = 1$ ($r = 0.0493$).



(b) Interaction order $K = 2$ ($r = 0.0250$).

Figure 7.8-I: Comparison of intermediate object embeddings Y for the basic scene (left) and the transmitter–receiver swapped scene (right) across interaction orders $K \in \{1, 2, 3\}$. The colorscale indicates the embedding values. The horizontal dashed lines demarcate boundaries between physical objects (buildings and floor). The distinct activation patterns demonstrate that the learned representations are coordinate-dependent. The Pearson correlation coefficient r reported in each subfigure measures the correlation between the flattened object embedding matrices of the basic and swapped configurations.



(c) Interaction order $K = 3$ ($r = 0.0955$).

Figure 7.8-II: Comparison of intermediate object embeddings Y for the basic scene (left) and the transmitter–receiver swapped scene (right) across interaction orders $K \in \{1, 2, 3\}$ (continued). The colorscale indicates the embedding values. The horizontal dashed lines demarcate boundaries between physical objects (buildings and floor). The distinct activation patterns demonstrate that the learned representations are coordinate-dependent. The Pearson correlation coefficient r reported in each subfigure measures the correlation between the flattened object embedding matrices of the basic and swapped configurations.

the distinct patterns in individual object embeddings, certain dimensions of the scene representation remain partially correlated or stable under the swap. This indicates that the scene encoder preserves global spatial information (such as building density and spacing) while also capturing the orientation of the propagation axis.

To quantify the degree of symmetry preserved by the scene representation, we calculate the Pearson correlation coefficient r between the embedding vector (or matrix) of the basic scene and that of the swapped scene. Specifically, for two flattened embeddings $\boldsymbol{x}$ and $\boldsymbol{y}$ of dimension D, the correlation coefficient is computed as

$$r = \frac{\sum_{i=1}^{D}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{D}(x_i - \bar{x})^2 \sum_{i=1}^{D}(y_i - \bar{y})^2}}, \tag{7.8}$$

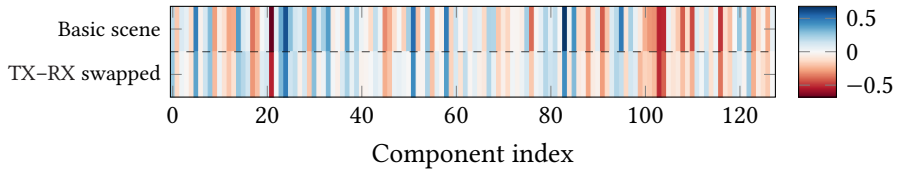
where $\bar{x} = \frac{1}{D}\sum_{i=1}^{D} x_i$ and $\bar{y} = \frac{1}{D}\sum_{i=1}^{D} y_i$ denote their respective sample means. Across the different interaction orders, we find a remarkably high correlation for the scene embeddings: $r = 0.9298$ for $K = 1$, $r = 0.9278$ for $K = 2$, and $r = 0.9321$ for $K = 3$. This strong correlation (≈ 0.93) indicates that the combined scene representation is highly reciprocal and invariant to coordinate reflections. In comparison, the individual object embeddings $\boldsymbol{Y}$ exhibit a very low correlation when flattened ($r = 0.0493$, $r = 0.0250$, and $r = 0.0955$ for $K = 1, 2, 3$, respectively), highlighting that the object-level features remain coordinate-dependent while the pooled scene-level features achieve spatial reciprocity.

Probability Vectors and Path Trajectories The flow head generates selection policies (normalized edge flows) at each step of the path construction. Figure 7.10 visualizes these step-by-step selection probabilities, with red boxes highlighting the selected objects at each interaction step.

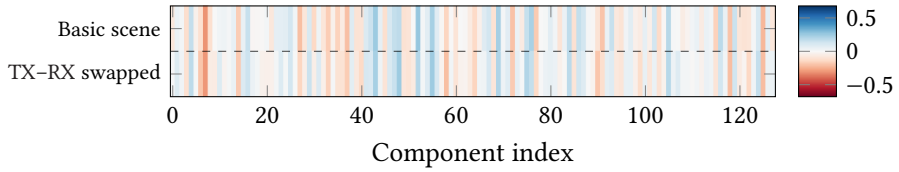
The probability vectors show that the policy is highly sparse, concentrating almost all probability mass on a small subset of candidate objects, which confirms that the model successfully prunes unreachable or invalid parts of the search space.

Although the sampled path trajectories may differ (for example, at order $K = 3$, the basic model samples Building 4 → Building 1 → Floor while the swapped model samples Building 5 → Building 4 → Building 4), this difference is primarily a consequence of the stochastic sampling process during path candidate construction. Indeed, because sampling involves a random selection step, minor floating-point variations in the probability vectors can lead to different discrete path samples, even when using the same random seed. A closer examination of the selection probability vectors at the initial step (interaction index 0, before any object has been chosen) reveals a close match between the basic and swapped configurations across all orders.

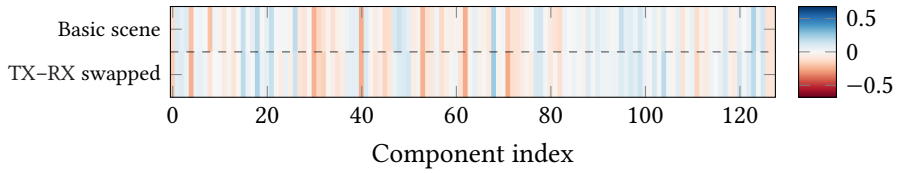
This indicates that the policy network has successfully learned reciprocal initial distributions. However, once a different first object is stochastically sampled, the subsequent probability distributions (at indices 1 and 2) naturally diverge because they are conditioned on different path histories. To fully align these conditional sequential policies and ensure reciprocal symmetry throughout the entire autoregressive generation process, symmetric training strategies, such as the transmitter–receiver symmetry replay described in Section 7.1, remain crucial.



(a) Interaction order $K = 1$ ($r = 0.9298$).

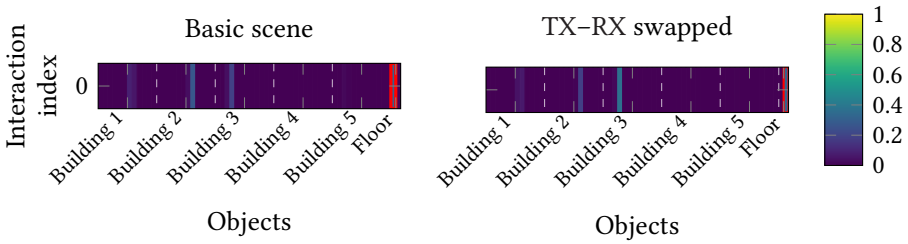


(b) Interaction order $K = 2$ ($r = 0.9278$).



(c) Interaction order $K = 3$ ($r = 0.9321$).

Figure 7.9: Comparison of scene embedding vectors for the basic scene (top row) and the transmitter–receiver swapped scene (bottom row) across interaction orders $K \in \{1, 2, 3\}$. The horizontal dashed line separates the two configurations. The Pearson correlation coefficient r reported in each subfigure measures the correlation between the scene embedding vectors of the basic and swapped configurations. The high correlation (≈ 0.93) between the two configurations shows that global topological structures are preserved despite the coordinate reflection.



(a) Interaction order $K = 1$ (one step).

(b) Interaction order $K = 2$ (two steps).

(c) Interaction order $K = 3$ (three steps).

Figure 7.10: Comparison of selection probability vectors (normalized edge flows) for the basic scene (left) and the transmitter–receiver swapped scene (right) across interaction orders $K \in \{1, 2, 3\}$. The colorscale indicates selection probability. Red boxes outline the selected object at each construction step, and vertical dashed lines denote boundaries between physical objects. The policy exhibits high sparsity, but trajectories differ due to the directional, random nature of path construction.

7.1.4 Generalization to Unseen Scenarios and Sampling-Size Trade-Off

By design, the model is trained on randomized scenes to learn geometric properties rather than memorizing a single map. Although evaluation is performed on unseen snapshots, training and evaluation distributions can remain close when the transmitter and receiver are sampled from similar spatial regions. Complementing the journal results [79], Figure 7.11 reports two additional analyses: (i) hit-rate evolution versus sampling budget M, and (ii) in-distribution versus shifted-distribution evaluation regions.

[79]: Eertmans et al. (2026), *Transform-Invariant Generative Ray Path Sampling for Efficient Radio Propagation Modeling*

Most curves exhibit a saturation behavior: the hit rate increases quickly for small M and then plateaus, typically around $M \approx 10$, consistent with the evaluation budget used in [79]. This indicates diminishing returns from additional samples once dominant valid branches are already captured.

Evaluation on the same sampling region as training generally yields higher hit rates than evaluation on shifted regions, confirming a non-negligible distribution-shift effect. This gap is also visible in the coverage reconstructions of Figure 7.12, with the exhaustive reference in Figure 7.12a, the CO prediction in Figure 7.12b, and the whole scene (WS) prediction in Figure 7.12c. At the same time, models trained on broader spatial regions tend to degrade less under a distribution shift, indicating that training-region diversity improves robustness even when peak in-region performance is lower. Importantly, these coverage maps are defined using a *non-coherent addition* of path powers, as introduced in Section 2.10, to provide a stable estimate of the local mean power.

The root mean squared error between the predicted and exhaustive coverage maps in Figure 7.12 is 3.48 dB (and 1.62 dB when restricting evaluation to the canyon) for the CO model, and about 2.41 dB (and 1.36 dB when restricting evaluation to the canyon) for the WS model. The stronger

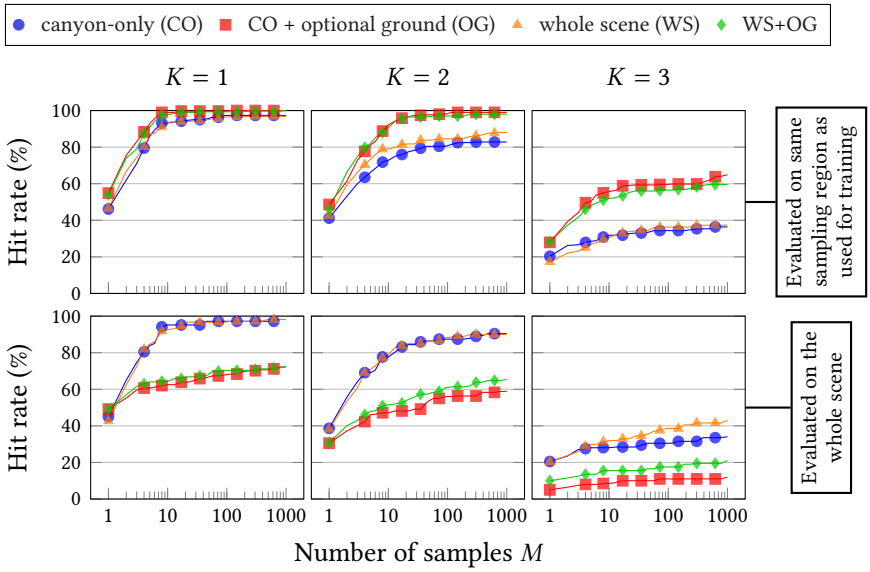
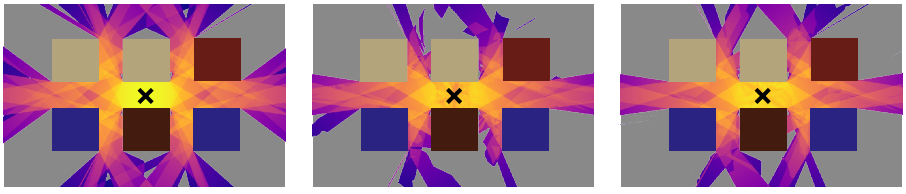
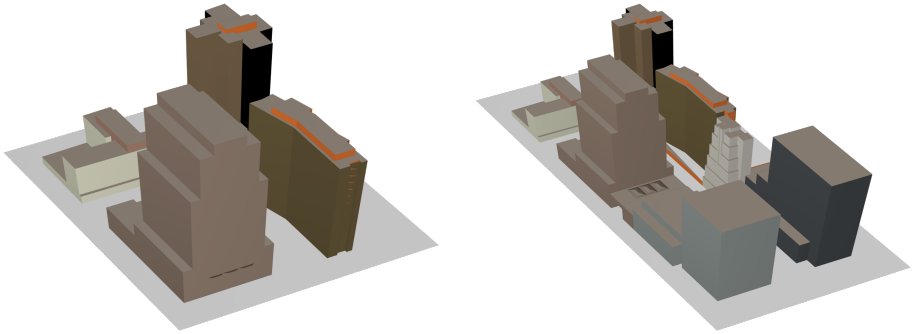


Figure 7.11: Hit rate as a function of the number of samples M for different training and evaluation sampling regions. The top row shows evaluation on the same sampling region as the training dataset, while the bottom row shows evaluation on a different sampling region (except for the whole scene case).



(a) Ground truth (exhaustive). **(b)** Prediction (trained on CO). **(c)** Prediction (trained on WS).

Figure 7.12: Received power map reconstruction in log scale with the generative sampler (sampling size $M = 10$) against the exhaustive ground-truth approach. The left plot shows the ground-truth coverage map obtained with exhaustive ray tracing, while the middle and right plots show the predicted coverage maps obtained with the generative sampler trained on the CO and WS training scenarios, respectively.



(a) Small scene ($N = 400$ triangular facets). **(b)** Medium scene ($N = 805$ triangular facets).

Figure 7.13: Real-world Manhattan street canyon scenes obtained from OpenStreetMap. Both scenes exhibit different building aspect ratios, facade complexity, and street widths compared to the idealized training scenario.

WS performance, including inside the canyon, is consistent with better cross-region generalization.

To assess the generalization capability of the model on unseen urban environments with different structural properties, we evaluate the trained WS model on real-world street-canyon scenarios obtained from OpenStreetMap data in Manhattan, New York. These scenarios exhibit structural characteristics distinct from the training distribution, including different building aspect ratios, facade complexity, street widths, and block organization patterns. Specifically, we select two test environments: a small scene with $N = 400$ triangular facets and a medium scene with $N = 805$ triangular facets, as shown in Figure 7.13. Compared to the procedurally generated canyon training configurations (which contain up to $N = 74$ facets), these real-world geometries represent a genuine out-of-distribution test case. Due to the increased scene complexity, specular reflections are evaluated up to second order ($K = 2$), and the sampling budget is set to $M = 100$ per receiver point to account for the higher density of potential propagation paths.

The resulting ground-truth and predicted coverage maps are compared in Figure 7.14. The evaluation highlights that

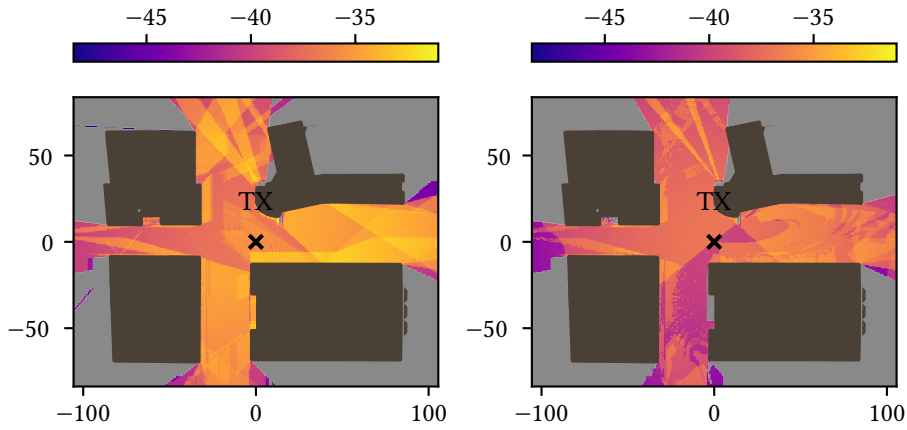
while the model successfully discovers dominant reflection paths along the main street axes, its generalization to the out-of-distribution geometries is imperfect and exhibits significant spatial heterogeneity. The sampler reconstructs the general spatial distribution of the received power field reasonably well, but noticeable discrepancies from the ground truth emerge. Specifically, the prediction quality varies across different spatial regions: in some locations, the model fails to discover reflection paths from specific building facades, whereas in others, paths are only partially recovered. This spatial non-uniformity indicates that although the model has learned robust geometric regularities of path propagation that transfer beyond its procedural training configurations, it has not fully mastered general reflection principles and remains partially adapted to the structural regularities of the training family. Addressing this limitation to enable seamless transfer across arbitrary city layouts represents an important direction for future research, potentially through the use of increased model capacity, visibility-graph representations, or training on more complex scenarios from the outset.

To assess the hardware robustness of the speed-up, the benchmark from [79] is rerun on a different setup: 11^{th} Gen Intel© Core™ i7-1165G7 × 8 (CPU) and NVIDIA Quadro RTX™ 4000 (GPU). Despite the hardware change, the acceleration trend remains consistent (Figure 7.15), with large CPU gains and smaller but still substantial GPU gains. The lower relative gain on GPU stems from the fact that the exhaustive baseline is already heavily parallelized and optimized for high-throughput ray–facet intersections, which partially masks the combinatorial overhead in the relatively small scenes tested. On the CPU, where intersection tests are more sequential, the sampler's ability to prune the candidate space yields a more dramatic reduction in absolute runtime.

[79]: Eertmans et al. (2026), *Transform-Invariant Generative Ray Path Sampling for Efficient Radio Propagation Modeling*

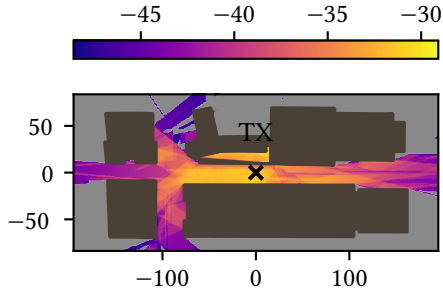
While these results were obtained on scenes with a limited number of buildings (e.g., 6 buildings in the street canyon), they demonstrate a proof of concept for replacing combinatorial candidate enumeration with learned generation.

To scale this approach to city-scale environments with hundreds of buildings, the generative sampler would need to be combined with spatial acceleration structures such as BVHs (see Section 5.5.2). Commercial ray tracing tools often achieve their performance through such structures; however, our focus here is on addressing the fundamental combinatorial bottleneck that remains even with spatial pruning, especially at high interaction orders.

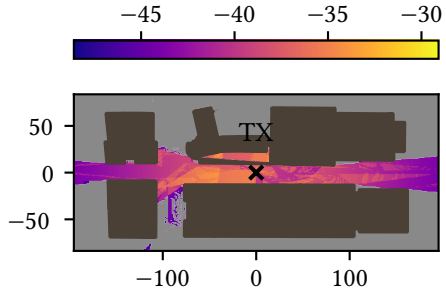


(a) Ground truth G_{gt} (small scene).

(b) Prediction G_{pred} (small scene).



(c) Ground truth G_{gt} (medium scene).



(d) Prediction G_{pred} (medium scene).

Figure 7.14: Coverage map comparison for real, out-of-distribution Manhattan scenarios under a sampling budget of $M = 100$, considering specular reflections up to second order. The plots compare the ground truth path gain (exhaustive) with the sampler's predictions for the small and medium scenes.

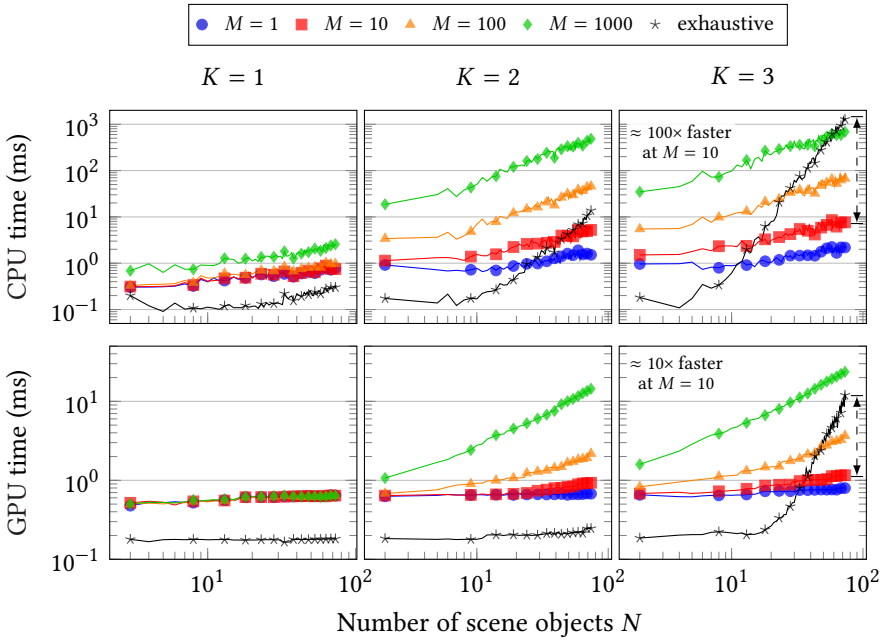


Figure 7.15: Runtime comparison of the generative path sampler against exhaustive ray tracing. The top and bottom rows show CPU and GPU times, respectively, as a function of scene complexity (number of objects) for interaction orders $K \in \{1, 2, 3\}$. Each subplot presents results for different sampling budgets M, demonstrating substantial speedups with the learned approach: approximately 100× on CPU and 10× on GPU at $M = 10$.

7.1.5 Summary and Practical Implications

Machine-learning-assisted path sampling occupies a practical middle ground between exhaustive physical solvers and purely data-driven surrogates. It preserves interpretability and physical validity while addressing the combinatorial bottleneck. The main trade-off is the offline training cost for each scene family. For repeated evaluations in the same class of environments (coverage sweeps, dense optimization loops, repeated TX or RX placements), the amortized runtime gain justifies this upfront investment. Together with the reconstruction errors reported above, these results support learned candidate generation as a robust acceleration layer for high-fidelity ray tracing.

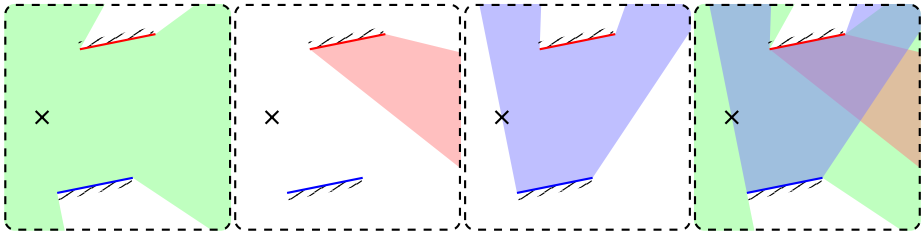
7.2 Study of the Multipath Lifetime Map

Where the previous section focused on computational efficiency through intelligent sampling, this section addresses a complementary question raised in Chapter 5: when and where can a ray tracing snapshot be safely extrapolated in dynamic scenarios? The MLM is a geometric-topological construct that characterizes the validity of snapshot extrapolation in a chosen configuration-parameter space and provides quantitative measures to support hybrid workflows that combine local extrapolation with full recomputation only where necessary [106].

[106]: Eertmans et al. (2025), *Comparing Differentiable and Dynamic Ray Tracing: Introducing the Multipath Lifetime Map*

7.2.1 Methodology

As in the previous section, each ray path is described by a path candidate, i.e., an ordered list of interactions (interaction type and object identity). Let $\mathcal{I}^K$ be the set of all candidates up to interaction order K, and let $\mathcal{V}^K(\boldsymbol{\theta}) \subseteq \mathcal{I}^K$ be the subset that is geometrically valid for a given scene configuration $\boldsymbol{\theta}$ (for example, receiver position, antenna



(a) Line-of-sight only. **(b)** Upper-wall reflection. **(c)** Lower-wall reflection. **(d)** All interactions combined.

Figure 7.16: Illustration of multipath cells for individual interaction families and for the combined valid-path structure. Colors encode unique multipath identities, with up to 11 distinct cells in Figure 7.16d.

tilt, or other dynamic scene parameters). Two scene configurations are treated as equivalent when they lead to the same valid-candidate set,

$$\boldsymbol{\theta}_a \sim \boldsymbol{\theta}_b \iff \mathscr{V}^K(\boldsymbol{\theta}_a) = \mathscr{V}^K(\boldsymbol{\theta}_b). \tag{7.9}$$

A *multipath cell* C_i is a connected region of the considered configuration space in which this valid-candidate set does not change. Figures 7.16a to 7.16c show examples for individual interaction families, while Figure 7.16d (grouped in Figure 7.16) shows the combined valid-path structure.

While MLMs are not limited to exhaustive ray tracing and can also be used with SBR methods, the worst-case number of candidates is still governed by exhaustive enumeration. For a scene with N objects and fixed interaction order K, exhaustive tracing evaluates up to $M = N(N-1)^{K-1}$ candidates. Each cell can therefore be encoded by a binary vector $v_i \in \{0,1\}^M$, where each entry indicates whether a candidate in $\mathscr{I}^K$ is valid.

However, storing dense vectors for all cells is impractical because M grows combinatorially. To reduce memory usage, the implementation processes candidates in chunks and compresses each cell encoding into a persistent hash-based identifier. This approach, similar to the one used to

avoid double-counting in SBR, preserves consistent coloring across snapshots without storing full-length dense vectors, provided that the candidate ordering is consistent across runs and hash collisions do not occur.

Each cell C_i is characterized by two complementary metrics in the chosen configuration-parameter space. The *cell measure* is $S_i = \mathrm{meas}(C_i)$ (which reduces to an area in 2D spatial settings), and the *average minimal inter-cell distance* is $\overline{d_i}$, defined through the parameter-space distance-to-exit

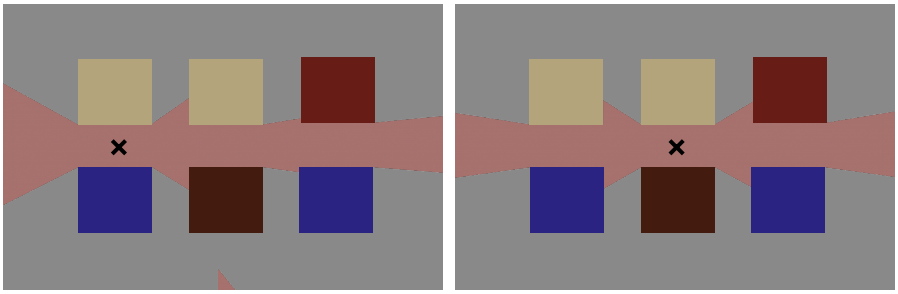
$$d_i(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}' \notin C_i} \mathrm{dist}(\boldsymbol{\theta}, \boldsymbol{\theta}'), \qquad \boldsymbol{\theta} \in C_i, \tag{7.10}$$

where $\overline{d_i}$ is the average of $d_i(\boldsymbol{\theta})$ over C_i in the considered parameter space. Importantly, the units of S_i and $\overline{d_i}$ depend on the configuration variables used to build the MLM: they are not intrinsically m^2 and m. For example, if the configuration variable is an antenna rotation angle, the corresponding distance-like quantity is angular and the associated units follow that choice. In mobility settings such as a moving receiver, the configuration space is spatial, so S_i and $\overline{d_i}$ naturally take the units m^2 and m, and a practical local time-based validity horizon is approximated as $t_i \approx \overline{d_i}/v$.

With the exception of parameter values located at the intersection of multiple cells, adjacent cells differ by exactly one candidate in their encoding. This implies a Hamming distance of one between neighboring binary vectors. This structural property naturally leads to incremental path-update strategies at cell boundaries.

7.2.2 Example Multipath Lifetime Maps

To generate an MLM for receiver mobility in a given scenario, we sweep the receiver across a spatial grid and compute the valid-candidate set at each location. Here, we evaluate a grid of 600×600 receiver positions at a height of 1.5 m, with the transmitter fixed at a height of 32 m, in the street-canyon example. Because this example focuses



(a) Initial TX position. **(b)** Intermediate TX position.

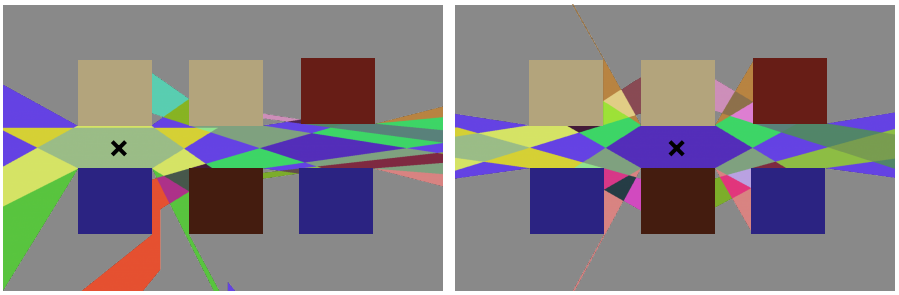
Figure 7.17: Example MLMs across transmitter positions for line-of-sight paths. Identical colors indicate identical multipath-cell identities across transmitter positions.

on mobility induced by receiver-position variations, the resulting MLMs and related metrics can be visualized directly in 2D spatial maps. Figures 7.17 to 7.19 report order-specific MLMs for line-of-sight, first-order reflections, and second-order reflections, respectively.

These MLMs differ from those reported in [106], where the analysis was performed for the combined line-of-sight and first-order set. The order-separated views shown here provide finer structural insight: line-of-sight MLMs form broad and relatively simple regions, first-order MLMs exhibit stronger fragmentation driven by reflection-visibility boundaries, and second-order MLMs show the highest spatial complexity due to compounded interaction constraints.

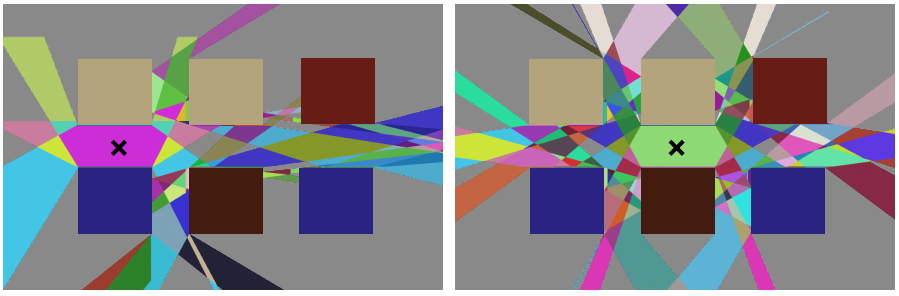
[106]: Eertmans et al. (2025), *Comparing Differentiable and Dynamic Ray Tracing: Introducing the Multipath Lifetime Map*

Quantitative statistics of the proposed metrics (cell-area distributions and average minimal inter-cell distances) are reported in [106].



(a) Initial TX position. **(b)** Intermediate TX position.

Figure 7.18: Example MLMs across transmitter positions for first-order reflection paths. Identical colors indicate identical multipath-cell identities across transmitter positions.



(a) Initial TX position. **(b)** Intermediate TX position.

Figure 7.19: Example MLMs across transmitter positions for second-order reflection paths. Identical colors indicate identical multipath-cell identities across transmitter positions.

7.2.3 Practical Guidance and Extensions

The MLM reframes extrapolation validity as a geometric question measurable directly from path topology. This yields practical guidance: if trajectories remain within large cells, dynamic ray tracing via snapshot extrapolation is efficient; if trajectories cross many cell boundaries, full recomputation may become preferable.

Beyond receiver mobility, the methodology extends to other dynamic variables (moving reflectors, reconfigurable elements, and beam-state changes) and supports adaptive simulation strategies that balance fidelity and runtime. This perspective naturally connects to the chapter conclusion, which synthesizes both contributions.

7.3 Conclusion

This chapter has explored two complementary research directions that extend ray tracing beyond its classical deterministic formulation. First, the machine-learning-assisted path sampler shows how neural networks can complement physics-based simulation: instead of learning electromagnetic fields directly, the model identifies promising partial paths and delegates physical evaluation to geometric and electromagnetic solvers. The reported speedups—approximately 100× on CPU and 10× on GPU at evaluation budgets of $M = 10$ samples—make high-fidelity multipath analysis tractable even in demanding scenarios. This efficiency, however, requires offline training for each scene family and is therefore most beneficial when many evaluations are amortized over the same environment class.

Second, the MLM emphasizes reliability and validity. By characterizing where path topology remains stable, the MLM enables hybrid workflows that combine fast extrapolation with full recomputation only at critical boundaries.

This geometry-driven perspective aligns with physical intuition and provides actionable guidance for motion planning and simulation scheduling.

Together, these studies support a broader principle for computational propagation: machine learning and geometric analysis are most effective when they augment—rather than replace—fundamental physics. This chapter therefore closes the technical development of this book by establishing both an acceleration paradigm (learned path generation) and a validity paradigm (topology-aware extrapolation). Future work could further refine the generative sampler by integrating electromagnetic material properties (such as complex permittivity) into the object features, allowing the model to selectively prune interactions with highly lossy or diffuse surfaces.

In the next and final chapter, we will conclude on what we have seen and discuss the potential implications for future research and applications in differentiable ray tracing.

Conclusion and Perspectives | 8

Throughout this book, we have moved from the physics of electromagnetic waves to practical methods for simulating radio propagation in complex environments. The primary objective was to connect electromagnetic theory, numerical modeling, and optimization tools within a single framework.

In this final chapter, we summarize the main ideas, clarify what remains outside the scope, discuss current limitations, and highlight future research directions.

8.1 Summary of What We Have Seen

The central message of this book is that radio propagation can be modeled accurately enough for many engineering tasks by combining high-frequency electromagnetic approximations with efficient geometric algorithms.

8.1.1 What This Book Covered

At the physical level, we began with Maxwell's equations and introduced the assumptions leading to high-frequency methods. In this regime, geometrical optics (GO) and uniform theory of diffraction (UTD) provide the primary tools for modeling reflection, transmission, and diffraction in large-scale scenes.

At the algorithmic level, we studied how to find valid propagation paths in environments containing numerous objects and potential interactions. This is a challenging combinatorial problem, with a complexity that grows rapidly with the number of surfaces and the interaction order. We discussed several approaches, including the image method,

min-path-tracing (MPT), and variational methods based on Fermat's principle.

Finally, we transitioned from forward simulation to inverse design using differentiable ray tracing. By computing both channel values and their gradients, we can optimize geometry, material parameters, and transceiver settings using gradient-based methods.

8.1.2 What We Did Not Cover

Even with robust models such as GO and UTD, real environments remain more complex than our usual assumptions suggest. In practice, many scenes cannot be reduced to smooth polygonal surfaces without losing important physical effects.

Advanced scattering and facade details represent one important example. Specular reflection is a good approximation only when surface roughness is small compared to the wavelength. At millimeter-wave and sub-terahertz frequencies, this condition is often violated, and diffuse scattering becomes increasingly important. Real facades also feature windows, balconies, and recesses that can significantly affect propagation. These details can create directional reflections and interference patterns that are not captured by simple flat-wall models. Better scattering models are needed to represent these effects without making simulations prohibitively expensive.

Another major gap concerns point clouds versus meshes. LiDAR and photogrammetry pipelines naturally produce large point clouds, while most ray tracers expect triangular meshes. Converting point clouds to clean meshes is computationally expensive and can eliminate sharp edges critical for diffraction. Direct ray tracing on point clouds is possible, but it remains an active research topic, particularly for large outdoor scenes.

Vegetation modeling was also only touched upon indirectly. Trees and shrubs possess complex geometries, and

their leaves and branches often have dimensions comparable to the wavelength at high frequencies. As a result, foliage behaves like a volumetric scattering medium rather than a collection of simple surfaces. Wind also renders the channel time-varying, producing significant amplitude and phase fluctuations.

8.2 Limitations and Future Directions

Before closing, it is important to acknowledge that the methods presented in this book possess practical limitations. Many open problems and active research directions remain in radio propagation modeling, including several directly related to the work presented here.

8.2.1 Open Problems

A key practical limitation lies not only in *how precise* a solver is in theory, but also in *how reliable* the input data are in practice. Research often pushes ray tracing toward more complete physical models. Examples include near-field effects for large arrays, additional diffraction mechanisms, and interactions with reconfigurable intelligent surface (RIS). While these extensions are valuable and scientifically significant, they also increase computational cost and implementation complexity.

In real deployments, however, geometric and material discrepancies often dominate the error budget. Small errors in building positions can move reflection points significantly, and uncertain material parameters can alter the predicted overall channel gain by several decibels. Such uncertainties can outweigh the benefits of extremely high-order asymptotic corrections.

This leads to a core open problem: how to align model complexity with input-data quality. When geometry and

materials are uncertain, extremely high interaction orders may offer little practical benefit. In many industrial workflows, resources are better spent on calibration, robust parameter estimation, and higher-quality environmental data than on increasingly complex high-order models.

8.2.2 Future Research Directions

Hardware acceleration is already changing what is practical in ray tracing. Modern graphics processing units (GPUs) render large-scale simulations much faster than before and reshape algorithmic priorities.

Modern GPUs offer massive parallelism and dedicated ray-intersection hardware. At the same time, just-in-time (JIT)-compiled frameworks such as JAX, PyTorch, and TensorFlow make this hardware more accessible from high-level programming environments. This combination enables highly accelerated simulations for scenes that were previously too expensive to evaluate regularly.

These performance gains also prompt a reassessment of dynamic ray tracing methods. Many classic dynamic techniques were designed to avoid recomputing paths from scratch. On GPUs, however, regular and massively parallel workloads are often more efficient than complex stateful updates with branch-heavy logic. In certain regimes, full recomputation may therefore prove both simpler and faster.

While machine learning surrogates are promising, completely replacing physics-based ray tracing remains challenging. Radio channels are highly sensitive to position and frequency, and even small displacements or frequency shifts can produce large fading variations. Consequently, purely data-driven models may struggle to generalize across scenes and frequency bands. A more pragmatic approach is therefore to combine learned components with physics-based solvers, rather than treating them as mutually exclusive alternatives.

Ultimately, the primary opportunity lies in pairing robust physical models with modern hardware and differentiable optimization.

8.3 Closing Remarks

In conclusion, practical radio propagation modeling requires both physical rigor and engineering pragmatism. High-frequency asymptotic models, efficient path solvers, and differentiable optimization now form a coherent toolbox for many tasks, provided that we remain explicit about the uncertainties in geometry, materials, and measurements.

Progress in this field will likely stem from higher-quality data, improved calibration, and closer integration of physics-based and learning-based components, rather than from isolated advancements in a single layer of the pipeline.

Further Readings

To supplement the material covered in this book, readers are encouraged to explore the following resources:

- *Antennas and Propagation for Wireless Communication Systems* by Saunders and Aragón-Zavala [2], for a comprehensive introduction to the physics of radio propagation and antenna theory.
- *Introduction to the Uniform Geometrical Theory of Diffraction* by McNamara, Pistorius, and Malherbe [41], for everything one needs to know about GO and UTD.
- *Sionna RT: Technical Report* by Aoudia et al. [70], for a shorter, practical introduction to ray tracing for radio propagation modeling.

Appendix

Getting Started with Path Tracing: Tutorial

A

This appendix provides a step-by-step tutorial on performing (differentiable) path tracing based on the advanced path tracing tutorial available in the documentation of the DiffeRT Python library [38]. It explores the library's lower-level API and the underlying logic used to perform exact path tracing. We first focus on the image method in a simple scene, then illustrate scaling limits and pruning mechanisms in larger environments, and finally present ray launching as an alternative strategy. Notation and terminology follow Chapter 3 and Chapter 5. The differentiability aspects of the implementation are not covered here, as they are automatically handled by the library's use of JAX as the underlying computational framework. For more details on the differentiable components, readers are encouraged to consult the library's documentation and source code.

[38]: Eertmans et al. (2025), *Demonstrating DiffeRT: An Open-Source Library for Optimizing Radio Networks with Differentiable Ray Tracing*

A.1 Example on a Simple Scene

Before diving into a complex scene, it is helpful to use a very simple setup to understand the fundamental steps of path tracing.

Note that the high-level `TriangleScene.compute_paths` method implicitly handles all the logic presented in this section, as well as the optimizations and post-processing steps omitted here for clarity.

A.1.1 Scene Loading and Setup

The first step is to define the propagation environment. In modern ray tracers, this is typically done by loading a mesh file (e.g., `.obj` or `.ply`) that represents the physical geometry, such as buildings or terrain. Listing A.1 shows

```python
# Very simple scene with two buildings
mesh_file = DIR / "two_buildings.obj"
mesh = TriangleMesh.load_obj(str(mesh_file))

tx = jnp.array([0.0, 4.9352, 22.0])
rx = jnp.array([0.0, 10.034, 1.50])
```

Listing A.1: Loading a simple scene in DiffeRT.

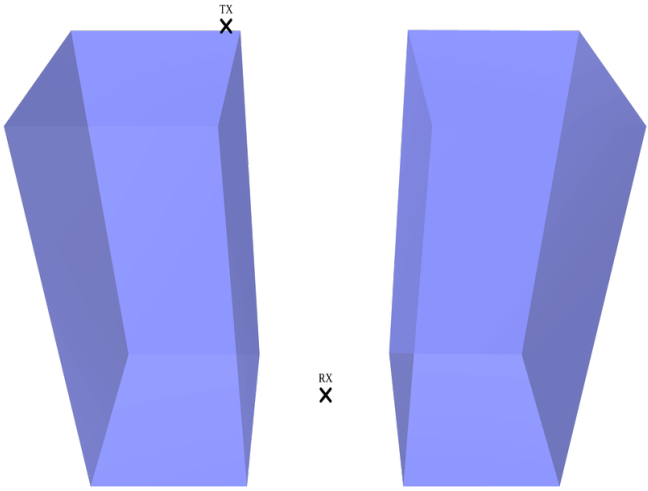


Figure A.1: Simple scene with two buildings, a TX, and an RX.

how to load a simple scene with two buildings and define the transmitter and receiver positions. The mesh is loaded into a `TriangleMesh` structure, which contains vertices and face indices:

1. This structure can also define sub-meshes to group primitives (e.g., walls and ground), radio materials, and face colors.
2. The positions of the transmitter ($\mathbf{x}_{\mathrm{TX}}$) and receiver ($\mathbf{x}_{\mathrm{RX}}$) are defined as coordinates in $\mathbb{R}^3$. The antenna properties are not relevant for path tracing, so they are not considered here.

For more advanced scenarios, DiffeRT also supports loading scenes from the Mitsuba3 XML [111] format, as used by Sionna RT [37].

[111]: Jakob et al. (2022), *Mitsuba 3 renderer*

[37]: Hoydis et al. (2023), *Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling*

A.1.2 How We Trace Rays: The Graph Analogy

Ray tracing can be implemented in many ways, depending on the desired performance, required accuracy, and geometry representation. Here, we implement exhaustive (or *exact*) ray tracing by generating all possible paths from TX to RX that undergo up to a maximum number of interactions (e.g., reflections) with the environment.

As first discussed in Chapter 3, one way to frame this is to represent the problem as a graph. The goal is to find paths from the TX node to the RX node, possibly visiting intermediate nodes corresponding to specific primitives (triangles) in the scene. A graph algorithm thus generates a list of path candidates. A candidate is not a physical path with 3D coordinates, but rather an ordered list of interactions to visit. A deeper discussion of how to generate candidates is provided in Appendix B.

It is then the role of a path tracing method (e.g., the image method) to determine the exact coordinates of that path. For the formal definition of path candidates and exact-coordinate recovery, see Chapter 3.

A.1.3 Solving for Path Vertices using the Image Method

For each generated path candidate, we must calculate the exact coordinates of the reflection points. The mathematical derivation of the image method (forward image construction and backward intersection pass) is detailed in Chapter 3; here we only show the corresponding implementation.

If we manually select a subset of triangles (e.g., from the red surface and the green surface), we can trace paths of increasing order (e.g., TX → RX, TX → red → RX, TX → red → green → RX).

Listing A.2: Manual path tracing with the image method for orders 0, 1, and 2.

```
select = [
  8,  # Red
  9,  # Red
  22,  # Green
  23,  # Green
]  # In practice, you will never hard-code the indices yourself

vertices = mesh.vertices
triangles = mesh.triangles[select, :]

select = jnp.array(
  select[::2],
  dtype=int,
)  # We actually only need one triangle per plane, so [8, 22]

# Iterate through path candidates
#                         ┌> order 0
#                         |           ┌> order 1
#                         |           |           ┌> order 2
for path_candidate in [select[:0], select[:1], select[:2]]:
  # 1 - Prepare input arrays
  mirror_vertices = mesh.vertices[
    mesh.triangles[path_candidate, 0], :
  ]
  mirror_normals = mesh.normals[path_candidate, :]

  # 2 - Trace paths
  path = image_method(tx, rx, mirror_vertices, mirror_normals)

  # 3 - ??

  # 4 - Obtain final valid paths
  full_path = jnp.concatenate(
    (
      tx[None, :],
      path,
      rx[None, :],
    ),
  )
```

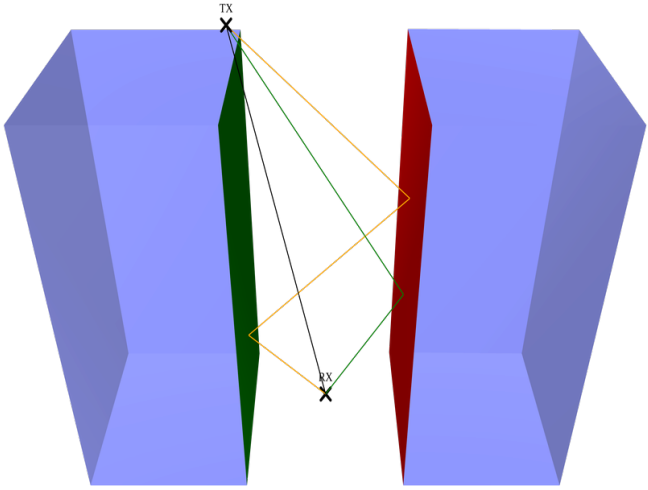


Figure A.2: Paths traced with the Image Method for order 0 (line-of-sight), order 1, and order 2.

A.1.4 Generating All Path Candidates

Manually identifying surfaces and generating all possible path candidates rapidly becomes tedious as the number of surfaces or the path order increases. For this purpose, DiffeRT provides `generate_all_path_candidates`. Written in Rust for performance, this function can generate millions of path candidates per second for a fixed maximum reflection order K. See Section 3.1.4 for a complexity discussion and Section 5.2 for the image-tree perspective. In the simplified case of specular reflections, a path candidate can be formally defined as a sequence of triangle (also referred to as primitive) indices

$$\mathbf{I} = [p_1, p_2, \dots, p_k, \dots, p_K], \tag{A.1}$$

where K is the path order (number of interactions) and p_k is the index of the primitive involved in the k-th interaction. The aforementioned function generates all possible combinations of primitives for a fixed path order, excluding path candidates with two consecutive identical primitives (as reflecting twice off the same flat surface is not possible), resulting in a comprehensive list of path candidates that can be further processed to determine their validity and

physical parameters.

For more details on generating candidates, refer to Appendix B.

A.1.5 Path Validation

Generating all possible candidates has one major side effect that we already discussed: many paths will be invalid (e.g., they might pass through a building or miss the mirror entirely). Furthermore, because the image method assumes infinite planes for its calculations, we must validate our paths against a series of geometric checks, consistent with Section 3.1.4.

After computing the exact coordinates of the reflection points for each candidate, we apply a series of checks to filter out invalid paths:

1. A *finite triangle check* verifies that each computed reflection point lies within the boundaries of its corresponding finite triangle.
2. An *interaction order check* ensures that all interactions respect the physical laws of reflection (e.g., incident and reflected rays must hit the same side of the mirror).
3. An *occlusion check (ray–triangle intersection)* detects line-of-sight obstructions. Even if a reflection point is valid, the ray traversing the scene must not be blocked by any other object.

As mentioned earlier, running this manually is cumbersome. However, we can accomplish all of this with a single call to `TriangleScene.compute_paths`, as shown in Listing A.4.

Listing A.3-I: All ray tracing steps assembled: candidate generation, geometry inputs, and path computation.

```
# [num_triangles 3 3]
all_triangle_vertices = mesh.triangle_vertices

num_triangles = mesh.num_triangles

for order in range(5):
  # 1 - Prepare input arrays
  # [num_path_candidates order]
  path_candidates = generate_all_path_candidates(
    num_triangles, order
  )
  num_path_candidates = path_candidates.shape[0]

  # [num_path_candidates order 3]
  triangles = jnp.take(mesh.triangles, path_candidates, axis=0)

  # [num_path_candidates order 3 3]
  triangle_vertices = jnp.take(mesh.vertices, triangles, axis=0)

  # [num_path_candidates order 3]
  mirror_vertices = triangle_vertices[
    ...,
    0,  # Only one vertex per triangle is needed
    :,
  ]
  # [num_path_candidates order 3]
  mirror_normals = jnp.take(
    mesh.normals, path_candidates, axis=0
  )

  # 2 - Trace paths
  # [num_path_candidates order 3]
  paths = image_method(tx, rx, mirror_vertices, mirror_normals)
```

Listing A.3-II: All ray tracing steps assembled (continued): first validity check.

```
# 3 - Remove invalid paths
# 3.1 - Remove paths with vertices outside triangles
# [num_path_candidates order]
mask = triangles_contain_vertices_assuming_inside_same_plane(
  triangle_vertices,
  paths,
)
# [num_path_candidates]
mask = jnp.all(mask, axis=-1)

# [num_paths_inter order+2 3]
full_paths = assemble_paths(
  tx,
  paths[mask, ...],
  rx,
)
```

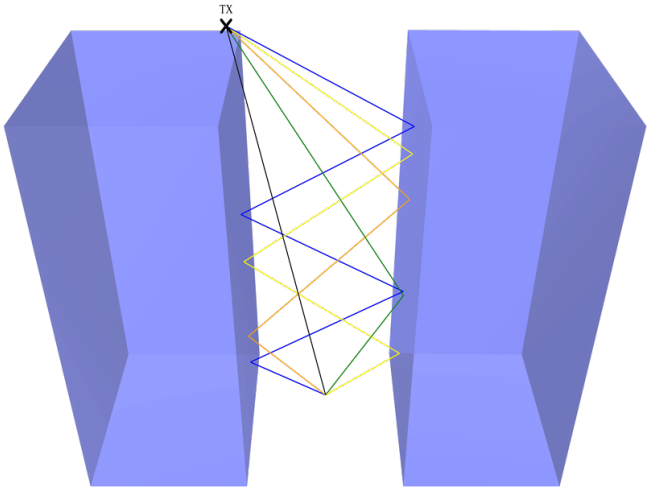


Figure A.3: Validated paths after applying boundary, interaction-side, and occlusion checks.

Listing A.3-III: All ray tracing steps assembled (continued): occlusion filtering and final path extraction.

```
# 3.2 - Remove paths with vertices not on the same side
# [num_paths_inter order]
mask = consecutive_vertices_are_on_same_side_of_mirrors(
  full_paths,
  mirror_vertices[mask, ...],
  mirror_normals[mask, ...],
)

# [num_paths_inter]
mask = jnp.all(
  mask, axis=-1
)  # We will actually remove them later

# 3.3 - Remove paths that are obstructed by other objects
# [num_paths_inter order+1 3]
ray_origins = full_paths[..., :-1, :]
# [num_paths_inter order+1 3]
ray_directions = jnp.diff(full_paths, axis=-2)

# [num_paths_inter order+1 num_triangles]
t, hit = rays_intersect_triangles(
  ray_origins[..., None, :],
  ray_directions[..., None, :],
  all_triangle_vertices[None, None, ...],
)
# In theory, we could do t < 1.0
# (because t == 1.0 means we are perfectly on a surface),
# but numerical errors require us to use some tolerance
tol = 1e-4
# [num_paths_inter order+1 num_triangles]
intersect = (t < (1.0 - tol)) & hit
#  [num_paths_inter]
intersect = jnp.any(intersect, axis=(-1, -2))
#  [num_paths_inter]
mask = mask & ~intersect

# 4 - Obtain final valid paths and plot
#  [num_paths_final]
full_paths = full_paths[mask, ...]
```

```
scene = TriangleScene(
  mesh=mesh,
  transmitters=tx,
  receivers=rx,
)
paths = scene.compute_paths(
  order=...,
)
```

Listing A.4: Computing ray paths with the high-level API.

A.2 Scaling to Complex Scenes

While exact path tracing works gracefully on simple scenes, it scales poorly to environments with millions of primitives—like modern city models.

A.2.1 The Combinatorial Explosion

The function `generate_all_path_candidates` returns an array of size $N_{\text{triangles}} \times (N_{\text{triangles}} - 1)^{\text{order}-1} \times \text{order}$, leading directly to the combinatorial bottleneck previously discussed in Chapter 5. For instance, a modestly sized city scene can quickly generate hundreds of millions of second-order paths that require evaluation.

For large scenes, the size of the candidate array quickly exceeds the available memory. To address this issue, DiffeRT offers an iterator variant called `generate_all_path_candidates_chunks_iter` that yields smaller arrays in chunks. While this solves the memory exhaustion issue, it does not reduce the massive computational load of tracing and checking these paths. Therefore, we must attempt to reduce the number of path candidates *before* generating them.

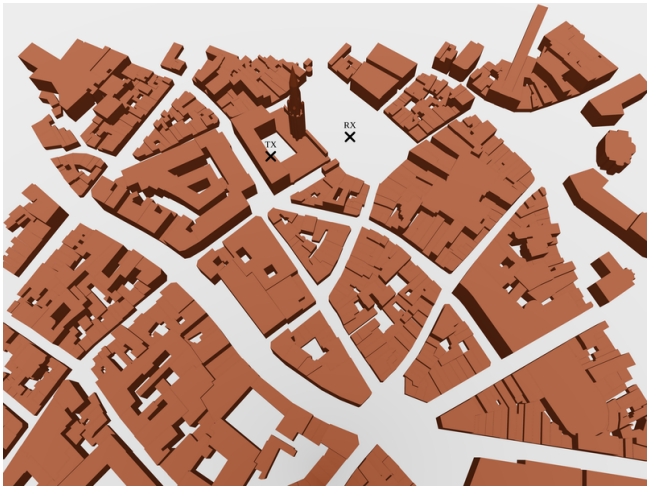


Figure A.4: Large urban scene—Grand Place of Brussels—used to illustrate combinatorial growth and pruning strategies.

A.2.2 Assuming Quadrilaterals

In many cases, architectural scenes primarily consist of quadrilaterals that have been trivially split into two triangles. Configuring the library to assume quadrilaterals via `mesh.set_assume_quads(True)` effectively halves the number of individual primitives. Since the number of candidates scales with $N_{\text{triangles}} \times (N_{\text{triangles}} - 1)^{\text{order}-1}$, halving the primitives reduces the total candidates by nearly a factor of 2^{order}.

A summary of the number of path candidates for second-order reflection paths and the effect of assuming quadrilaterals is provided below for the large urban scene shown in Figure A.4.

```
# Number of primitives (triangles): 14 206
# Number of path candidates: 201 796 230
# Number of primitives (assume_quads=True): 7 103
# Number of path candidates: 50 445 506
```

Listing A.5: Estimating triangle visibility from TX and recoloring the mesh.

```
tx = jnp.array([-40.0, 75, 30.0])

default_color = jnp.array([[0.2, 0.2, 0.2]])  # Hidden, black
visible_color = jnp.array([[1.0, 0.2, 0.2]])  # Visible, red
visible_triangles = triangles_visible_from_vertices(
  tx,
  mesh.triangle_vertices,
)
mesh = mesh.set_face_colors(default_color)
mesh = mesh.set_face_colors(
  mesh.face_colors.at[visible_triangles].set(visible_color)
)
```

A.2.3 Determining Transmitter Visibility for Graph Pruning

A highly effective way to prune path candidates is to recognize that the transmitter (or receiver) cannot physically reach all objects in the scene. By assessing which triangles are visible from TX using `triangles_visible_from_vertices`, we can construct a visibility vector, shrinking the pool of potential target nodes in the graph. This corresponds to the visibility preprocessing strategy of Chapter 5.

The percentage of visible triangles can be quite low in dense urban scenes, leading to a dramatic reduction in path candidates. Example numbers for the large urban scene are provided below.

```
# Percentage of visible triangles: 17.72%
# Percentage of visible quadrilaterals: 26.81%
# Number of path candidates: 13 522 208
```

Moreover, if the geometry is static, we can precompute the visibility vector of *every* triangle to every other triangle, constructing a massive adjacency (visibility) matrix. Instead of using a `CompleteGraph`, this matrix allows us to instantiate a `DiGraph` representation (see Appendix B

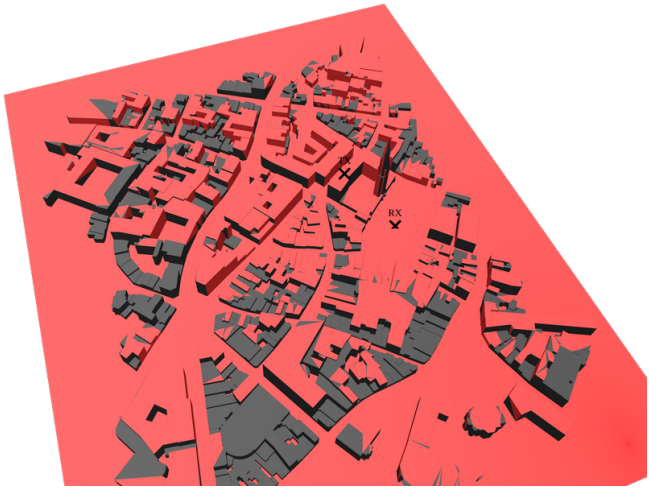


Figure A.5: Triangles visible from TX (red) versus hidden triangles (dark) for the large scene.

for more details). By explicitly disconnecting mutually invisible primitives within the topological graph, we dramatically prune the theoretical path candidates, preventing wasted computations on segments that are physically obstructed. However, this approach becomes infeasible for very large scenes due to the quadratic growth of the adjacency matrix size with the number of primitives, and because the precomputed visibility, which relies on ray casting, may not be exact.

A.3 An Alternative: Ray Launching

Eventually, any exact method, even heavily pruned, encounters a practical limit. Highly dense scenes or requests for high-order interactions will cause the sheer number of path candidates to overwhelm optimizations. In these scenarios, ray launching methods, such as SBR, serve as a viable alternative (see also the inexact methods overview in Chapter 3).

Unlike exact graph-based methods, SBR launches a fixed, manageable number of mathematical rays radially outward from TX. These rays traverse the environment and

Listing A.6-I: Simplified SBR implementation: setup and ray launch parameters.

```
num_rays = int(1e6)
max_dist = 0.5**2  # Squared distance (to avoid sqrt)
max_order = 2

# [num_path_candidates order 3]
frustum = viewing_frustum(
  tx, triangle_vertices.reshape(-1, 3)
)  # This avoids launching rays where there are no objects

# [num_rays 3]
ray_origins = jnp.broadcast_to(tx, (num_rays, 3))
ray_directions = fibonacci_lattice(num_rays, frustum=frustum)
```

undergo a constrained number of reflections off consecutive surfaces. Before each reflection event, the trajectory is scanned to determine whether it passes within an acceptable radius of the receiver (RX). If so, the sequence of reflections is harvested and implicitly formulated as a path, often followed by a localized final "correction" to align the endpoint strictly with RX. To avoid casting rays in directions that are unlikely to yield valid paths, launching can be guided by a coarse visibility analysis or by importance sampling based on the antenna pattern. Here, the rays are launched within the boundaries of the scene, as determined by the viewing frustum of the transmitter. Listing A.6 shows a simplified implementation. For efficient looping, the implementation relies on JAX's `jax.lax.scan` function.

Because path candidates are never explicitly generated beforehand, SBR smoothly bypasses the combinatorial explosion. The computational cost roughly scales with the chosen resolution of launched rays instead of strictly with the mesh fidelity. Within DiffeRT, this behavior can be natively invoked by providing `method='sbr'` when calling `TriangleScene.compute_paths`.

Listing A.6-II: Simplified SBR implementation (continued): computing the next bounce (scan function).

```
def scan_fun(ray_origins_directions_and_valids, _unused_input):
  # Unpack values from previous iteration
  ray_origins, ray_directions, valid_rays = (
    ray_origins_directions_and_valids
  )

  # 1 - Compute next intersection with triangles

  # [num_rays]
  triangles, t_hit = first_triangles_hit_by_rays(
    ray_origins,
    ray_directions,
    triangle_vertices,
  )
  # The above may generate infinite values,
  # so we will need to be careful with those
```

Listing A.6-III: Simplified SBR implementation (continued): detecting intersections with reception sphere (scan function).

```
  # 2 - Check if the rays pass near RX

  # [num_rays 3]
  ray_origins_to_rx = rx - ray_origins

  # [num_rays]
  # note: the fact that ray directions have unit length
  #       allows for some simplifications.
  ray_distances_to_rx = jnp.square(
    jnp.cross(ray_directions, ray_origins_to_rx)
  ).sum(axis=-1)  # Squared distance from rays to RX
  # Distance (scaled by ray directions) from
  # RX projected onto rays to ray origins
  t_rx = jnp.sum(ray_directions * ray_origins_to_rx, axis=-1)
  masks = jnp.where(
    (t_rx < t_hit) & (t_rx > 0) & valid_rays,
    # Check if RX is between origin and first triangle hit
    ray_distances_to_rx
    < max_dist,  # Check if RX is close enough
    False,
  )  # Whether rays pass near RX
```

Listing A.6-IV: Simplified SBR implementation (continued): reflecting rays (scan function).

```
  # 3 - Update rays

  # [num_rays 3]
  mirror_normals = jnp.take(mesh.normals, triangles, axis=0)

  ray_origins += t_hit[..., None] * ray_directions
  ray_directions = (
    ray_directions
    - 2.0
    * jnp.sum(
      ray_directions * mirror_normals, axis=-1, keepdims=True
    )
    * mirror_normals
  )
  # We mark rays that left the scene
  # i.e., when they no longer hit any object (t_hit is +inf.)
  valid_rays = valid_rays & jnp.isfinite(t_hit)

  return (ray_origins, ray_directions, valid_rays), (
    ray_origins,
    masks,
  )
```

Listing A.6-V: Simplified SBR implementation (continued): scan execution and path assembly.

```
# We mark rays that left the scene as invalid
valid_rays = jnp.ones(num_rays, dtype=bool)

# [max_order+1 num_rays 3], [max_order+1 num_rays]
_, (paths, masks) = jax.lax.scan(
  scan_fun,
  (ray_origins, ray_directions, valid_rays),
  length=max_order + 1,
)

# We swap 'max_order' and 'num_rays' axes
# [num_rays max_order+1 3], [num_rays max_order+1]
paths = jnp.moveaxis(paths, 0, 1)
masks = jnp.moveaxis(masks, 0, 1)

for order in range(max_order + 1):
  full_paths = assemble_paths(
    tx,
    # [num_valid_rays order 3]
    paths[masks[..., order], :order, :],
    rx,
  )
```

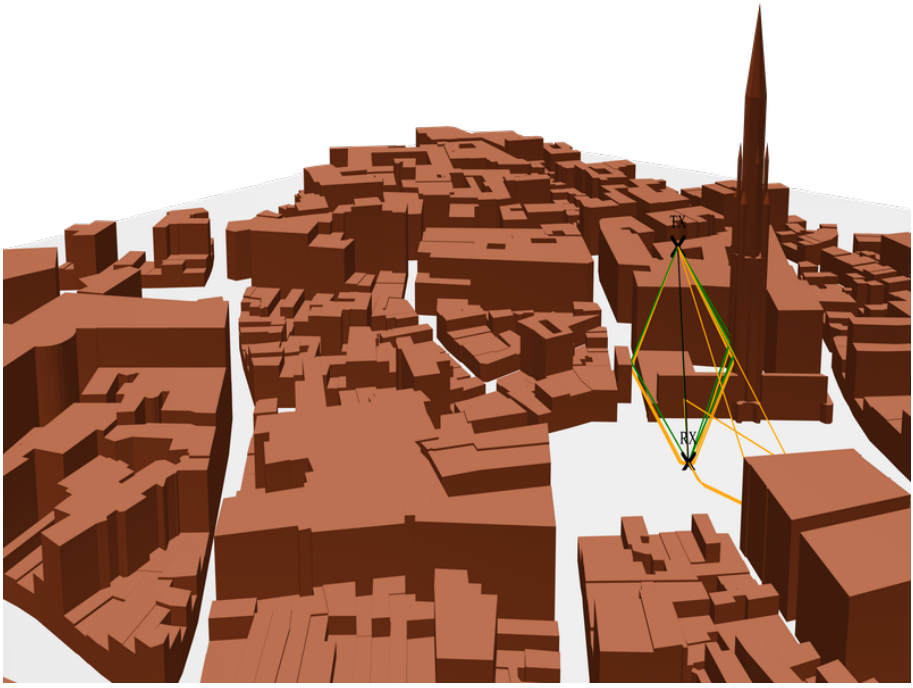


Figure A.6: Up to second-order reflection paths found with the SBR method that pass near RX in the large urban scene. Note that the double-counting problem is not addressed here, and duplicate paths are not removed.

A.4 Summary

This tutorial demonstrates, in executable form, the workflow discussed in Chapter 3 and Chapter 5: candidate generation, exact-coordinate recovery, validation, pruning, and an inexact alternative via SBR. Ultimately, the DiffeRT pipeline abstracts these low-level routines to produce validated geometric trajectories, which parameterize the computation of channel impulse responses, delays, and signal power. For further insight and fully reproducible notebooks, readers are invited to consult the library's online documentation.

Generating Path Candidates

B

As discussed in Chapter 3 and Appendix A, deterministic (exact) ray tracing can be decomposed into two tasks: generating path candidates and then solving for the exact coordinates of each candidate. As a reminder, a path candidate is an ordered interaction sequence between a TX and an RX; geometric validity is verified only afterward (see Section 3.1.4).

To optimize this computationally intensive process, modern differentiable ray tracers such as DiffeRT and Sionna RT usually partition it into two stages:

1. *Candidate generation*, to identify ordered sequences of primitives (e.g., triangles) that a ray could potentially traverse.
2. *Path tracing*, to compute the exact coordinate path for each sequence.

This appendix focuses only on the first stage: path candidate generation. In this formulation, listing candidates is equivalent to enumerating paths in a directed graph that connects scene objects with the TX and RX nodes, as illustrated in Figure 3.10 in Chapter 3.

B.1 Scene with Known Visibility

Let us consider again the simple 2D scene in Figure B.1. In preprocessed environments or simplified geometries, visibility between objects may be known or trivially computed.

This visibility is encoded as an adjacency matrix (visibility matrix), consistent with what we saw in Chapter 5. A value of one indicates an unobstructed line-of-sight between two graph nodes.

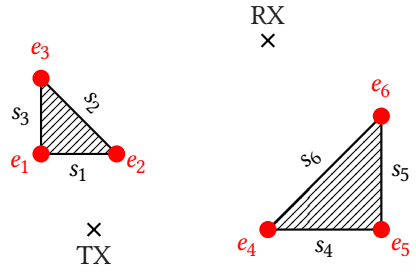


Figure B.1: 2D scenario with triangular-shaped objects on which reflection or diffraction can occur, repeated here for convenience from Figure 5.6. Surfaces are colored black and edges are colored red.

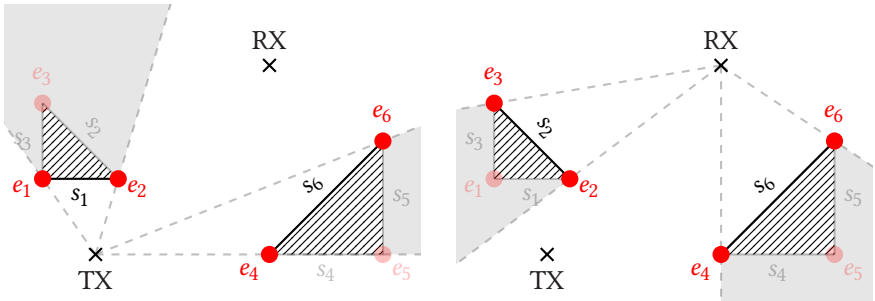
	TX	s_1	s_2	s_3	s_4	s_5	s_6	e_1	e_2	e_3	e_4	e_5	e_6	RX
TX		1					1	1	1		1		1	1
s_1							**1**				**1**			
s_2							**1**				**1**		**1**	1
s_3														
s_4														
s_5														
s_6		**1**	**1**					**1**	**1**	**1**				1
e_1							1				1			
e_2							1				1		1	1
e_3							1				1		1	1
e_4		1	1					1	1	1				1
e_5														
e_6			1						1	1				1
RX														

Figure B.2: Adjacency matrix, $\mathcal{G}$, generated from the scenario illustrated in Figure B.1, repeated here for convenience from Figure 5.8. Each row of this 14×14 matrix refers to the visible objects as seen from the corresponding object. For readability, zeros are omitted. The black coefficients indicate a possible reflection, while the red ones indicate diffraction.

If we omit the endpoint nodes (TX and RX), the matrix becomes symmetric because pairwise line-of-sight visibility is bidirectional. Endpoint-specific visibility maps are shown in Figure B.3.

Using a directed graph (`DiGraph` in DiffeRT's core library), we build the interaction graph from this adjacency matrix and enumerate candidates between the TX and the RX. Because the matrix is sparse, the candidate set is typically much smaller than in a complete-graph model.

The output from Listing B.2 is shown in Listing B.3.



(a) Objects visible from TX.

(b) Objects visible from RX.

Figure B.3: Example 2D visibility for the transmitter (TX) and receiver (RX), repeated here for convenience from Figure 5.7. Objects are faded if they are not visible from the corresponding node.

Listing B.1: Constructing a directed graph from a known visibility matrix.

```
adjacency_matrix = np.array(
  [
    [0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1],
    [0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
  ],
).astype(bool)

# We can test that our adjacency matrix, without TX and RX,
# is indeed symmetric
sub_adjacency_matrix = adjacency_matrix[1:-1, 1:-1]
assert np.all(sub_adjacency_matrix == sub_adjacency_matrix.T)

di_graph = DiGraph.from_adjacency_matrix(adjacency_matrix)
```

```
from_ = 0  # TX
to = 13  # RX

for i, path in enumerate(
  di_graph.all_paths(from_, to, depth=4)
):
  print(f"#{i + 1:03d}: {path}")
```

Listing B.2: Enumerating known-visibility path candidates with `DiGraph`.

```
#001: [ 0  1  6 13]
#002: [ 0  1 10 13]
#003: [ 0  6  2 13]
#004: [ 0  6  8 13]
#005: [ 0  6  9 13]
#006: [ 0  7  6 13]
#007: [ 0  7 10 13]
#008: [ 0  8  6 13]
#009: [ 0  8 10 13]
#010: [ 0  8 12 13]
#011: [ 0 10  2 13]
#012: [ 0 10  8 13]
#013: [ 0 10  9 13]
#014: [ 0 12  2 13]
#015: [ 0 12  8 13]
#016: [ 0 12  9 13]
```

Listing B.3: Output from Listing B.2.

B.2 Scene with Unknown Visibility

In many practical scenarios, exact visibility preprocessing is costly, especially in dense 3D environments (see Chapter 5).

Consequently, one may assume that every object can connect to every other object, i.e., a *complete graph*. This is a worst-case candidate generator: it is simple and highly parallel, but it deliberately over-generates candidates that must later be rejected by geometric validity checks (Section 3.1.4).

DiffeRT adopts this approach through the `CompleteGraph` class, which generates path indices an order of magnitude

faster than an equivalent `DiGraph`. However, when modeling the TX and the RX, we generally do not want these terminal nodes to appear multiple times in a path: TX should only originate paths, and no node should connect back to TX. Similarly, RX should only terminate paths. To model this behavior with `CompleteGraph`, one must:

1. Initialize a complete graph strictly *without* the TX and the RX nodes.
2. Generate path permutations programmatically (via `all_paths`), specifying that the respective start and terminal indices (`from_` and `to`) fall outside the graph's core logic.

Because the implementation is optimized for complete graphs, it assumes that `from_` is connected to all nodes in the graph, and symmetrically, that any node can connect to `to`.

Listing B.4: Generating paths using `CompleteGraph` under unknown visibility assumptions.

```
complete_graph = CompleteGraph(
  12  # number of objects
)

from_ = 12  # Can be anything >= 12
to = 13  # Can be anything >= 12 and != from_

for i, path in enumerate(
  complete_graph.all_paths(from_, to, depth=4)
):
  print(f"#{i + 1:03d}: {path}")
```

The (truncated) output from Listing B.4 is shown in Listing B.5. Note that the candidate set is much larger than in the `DiGraph` case (132 candidates vs. 16).

```
#001: [12  0  1 13]
#002: [12  0  2 13]
#003: [12  0  3 13]
#004: [12  0  4 13]
#005: [12  0  5 13]
#006: [12  0  6 13]
#007: [12  0  7 13]
#008: [12  0  8 13]
#009: [12  0  9 13]
#010: [12  0 10 13]
#011: [12  0 11 13]
#012: [12  1  0 13]
% ... 108 intermediate paths omitted ...
#121: [12 10 11 13]
#122: [12 11  0 13]
#123: [12 11  1 13]
#124: [12 11  2 13]
#125: [12 11  3 13]
#126: [12 11  4 13]
#127: [12 11  5 13]
#128: [12 11  6 13]
#129: [12 11  7 13]
#130: [12 11  8 13]
#131: [12 11  9 13]
#132: [12 11 10 13]
```

Listing B.5: Output from Listing B.4.

B.3 Beware of the Number of Path Candidates

The dominant limitation is combinatorial growth (Section 3.1.4, Section 5.2): the number of path candidates increases exponentially with the interaction order K.

For the complete-graph setting used here (with the TX and the RX handled as endpoints outside the core graph), the number of candidates for a fixed interaction order K is

$$N \times (N-1)^{K-1}, \tag{B.1}$$

where N is the number of scene objects in the complete

graph. This fixed-order term is consistent with the growth discussed in Section 3.1.4.

This induces the same trade-off emphasized in Chapter 5: a `DiGraph` with visibility pruning reduces the candidate count, while a `CompleteGraph` avoids preprocessing but incurs larger combinatorial growth.

Because full `CompleteGraph` implementations demand substantial memory allocations for higher interaction orders, tracing algorithms must either estimate a filtered visibility matrix—even if imperfect compared to mathematically exact ray tracing—or explicitly manage memory by iterating over candidate branches in smaller chunks. DiffeRT facilitates this through specialized block-processing methods (e.g., `*chunks_iter` methods in `differt_core.rt`), allowing continuous generation without catastrophic memory overflow.

Finally, to illustrate this scaling divergence, we compare the number of candidate paths produced by the directed graph (with exact visibility) and the complete graph across increasing depths.

Listing B.6: Comparing candidate counts between `DiGraph` and `CompleteGraph`.

```
for depth in [2, 3, 4, 5, 6, 7]:
  num_paths_di_graph = sum(
    1 for _ in di_graph.all_paths(0, 13, depth=depth)
  )
  num_paths_complete_graph = len(
    complete_graph.all_paths(12, 13, depth=depth)
  )
  print(
    f"{depth = }: {num_paths_di_graph:6d} (DiGraph) "
    f"vs {num_paths_complete_graph:6d} (CompleteGraph)",
  )
```

The output of Listing B.6 is shown in Listing B.7, demonstrating the exponential growth difference between the two graph models as depth increases.

Listing B.7: Output of Listing B.6.

```
depth = 2:      1 (DiGraph) vs      1 (CompleteGraph)
depth = 3:      4 (DiGraph) vs     12 (CompleteGraph)
depth = 4:     16 (DiGraph) vs    132 (CompleteGraph)
depth = 5:     56 (DiGraph) vs   1452 (CompleteGraph)
depth = 6:    192 (DiGraph) vs  15972 (CompleteGraph)
depth = 7:    680 (DiGraph) vs 175692 (CompleteGraph)
```

Bibliography

Here are the references in citation order.

[1] David J. Griffiths. *Introduction to Electrodynamics*. 5th. Cambridge University Press, 2024 (cited on pages 2, 8).

[2] Simon R. Saunders and Alejandro Aragón-Zavala. *Antennas and Propagation for Wireless Communication Systems*. 3rd. John Wiley & Sons, 2024 (cited on pages 2, 3, 5–7, 18, 19, 23, 26, 28, 30, 277).

[3] Claude E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x (cited on page 3).

[4] *ITU-R Recommendation V.431: Nomenclature of the frequency and wavelength bands used in telecommunications*. Standard. Available at https://www.itu.int/rec/R-REC-V.431/en. International Telecommunication Union, 2025 (cited on page 6).

[5] James C. Maxwell. "VIII. A dynamical theory of the electromagnetic field". In: *Philosophical Transactions of the Royal Society of London* 155 (1865), pp. 459–512. DOI: 10.1098/rstl.1865.0008 (cited on page 7).

[6] Constantine A. Balanis. *Advanced Engineering Electromagnetics*. 3rd. John Wiley & Sons, 2024 (cited on page 24).

[7] Nanfang Yu et al. "Light Propagation with Phase Discontinuities: Generalized Laws of Reflection and Refraction". In: *Science* 334.6054 (2011), pp. 333–337. DOI: 10.1126/science.1210713 (cited on page 33).

[8] Jiancheng An, Mérouane Debbah, Tie J. Cui, Zhi N. Chen, and Chau Yuen. "Emerging Technologies in Intelligent Metasurfaces: Shaping the Future of Wireless Communications". In: *IEEE Transactions on Antennas and Propagation* (2025), pp. 1–1. DOI: 10.1109/TAP.2025.3571069 (cited on pages 33, 150).

[9] Emil Björnson, Henk Wymeersch, Bho Matthiesen, Petar Popovski, Luca Sanguinetti, and Elisabeth de Carvalho. "Reconfigurable Intelligent Surfaces: A signal processing perspective with wireless applications". In: *IEEE Signal Processing Magazine* 39.2 (2022), pp. 135–158. DOI: 10.1109/MSP.2021.3130549 (cited on page 34).

[10] Harald T. Friis. "A note on a simple transmission formula". In: *Proceedings of the IRE* 34.5 (1946), pp. 254–256. DOI: 10.1109/JRPROC.1946.234568 (cited on page 34).

[11] International Telecommunication Union. *ITU-R Recommendation P.2040-3: Effects of building materials and structures on radiowave propagation above about 100 MHz*. Standard. Available at https://www.itu.int/rec/R-REC-P.2040/en. International Telecommunication Union, 2025 (cited on pages 37, 106, 107, 179, 218).

[12] George Everest and East India Company. *An account of the measurement of two sections of the meridional arc of India: bounded by the parallels of 18° 3' 15"; 24° 7' 11"; & 29° 30' 48"*. London: Printed by order of the Court of directors of the Honourable East-India Company, by J. & H. Cox, 1847 (cited on page 41).

[13] Alexander E. Conrady. *Applied optics and optical design, pt. I*. London: Oxford University Press, H. Milford, 1929 (cited on page 41).

[14] Otto Klemperer. *Electron Optics*. Cambridge University Press, 1939 (cited on page 42).

[15] Hugo Lichte. "Über den Einfluss horizontaler Temperaturschichtung des Seewassers auf die Reichweite von Unterwasserschallsignalen". In: *Physikalische Zeitschrift* 20.17 (1919), pp. 385–389 (cited on page 42).

[16] National Defense Research Committee (NDRC). "Physics of Sound in the Sea". In: *Summary Technical Report of Division 6* 8 (1946) (cited on page 42).

[17] Henry G. Booker. "Propagation of wave-packets incident obliquely upon a stratified doubly refracting ionosphere". In: *Philosophical Transactions of the Royal Society of London, Series A: Mathematical and Physical Sciences* 237.781 (1938), pp. 411–451. DOI: 10.1098/rsta.1938.0012 (cited on page 42).

[18] George Millington. "Ray-path characteristics in the ionosphere". In: *Proceedings of the IEE - Part IV: Institution Monographs* 101 (7 1954), pp. 235–249. DOI: 10.1049/pi-4.1954.0029 (cited on page 42).

[19] Jenifer Haselgrove. "Ray Theory and a New Method for Ray Tracing". In: *Report of the Conference held at the Cavendish Laboratory, Cambridge, September, 1954*. The Physical Society, 1955 (cited on page 42).

[20] Arthur Appel. "Some techniques for shading machine renderings of solids". In: *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*. AFIPS '68 (Spring). Atlantic City, New Jersey: Association for Computing Machinery, 1968, pp. 37–45. DOI: 10.1145/1468075.1468082 (cited on pages 42, 43).

[21] Turner Whitted. "An improved illumination model for shaded display". In: *Communications of the ACM* 23.6 (1980), pp. 343–349. DOI: 10.1145/358876.358882 (cited on page 43).

[22] Bishnu S. Atal and Manfred R. Schroeder. "Study of Sound Decay Using Ray-Tracing Techniques on a Digital Computer". In: *The Journal of the Acoustical Society of America* 41.6 (1967), pp. 1598–1598. DOI: 10.1121/1.2143658 (cited on page 43).

[23] Asbjørn Krokstad, Svein Strøm, and Svein Sørsdal. "Calculating the acoustical room response by the use of a ray tracing technique". In: *Journal of Sound and Vibration* 8.1 (1968), pp. 118–125. DOI: 10.1016/0022-460X(68)90198-3 (cited on page 43).

[24] Bruce R. Julian and David Gubbins. "Three-dimensional seismic ray tracing". In: *Journal of Geophysics* 43.1 (1977), pp. 95–113 (cited on page 43).

[25] Vlastislav Červený and František Hron. "The ray series method and dynamic ray tracing system for three-dimensional inhomogeneous media". In: *Bulletin of the Seismological Society of America* 70.1 (1980), pp. 47–77. DOI: 10.1785/BSSA0700010047 (cited on page 43).

[26] Hao Ling, Ri-Chee Chou, and Shung-Wu Lee. "Shooting and bouncing rays: calculating the RCS of an arbitrarily shaped cavity". In: *IEEE Transactions on Antennas and Propagation* 37.2 (1989), pp. 194–205. DOI: 10.1109/8.18706 (cited on pages 44, 137).

[27] Kurt R. Shaubackh, Nathaniel J. Davis, and Theodore S. Rappaport. "A ray tracing method for predicting path loss and delay spread in microcellular environments". In: 2 (1992), pp. 932–935. DOI: 10.1109/VETEC.1992.245274 (cited on page 44).

[28] Zhengqing Yun and Magdy F. Iskander. "Ray Tracing for Radio Propagation Modeling: Principles and Applications". In: *IEEE Access* 3 (2015), pp. 1089–1100. DOI: 10.1109/ACCESS.2015.2453991 (cited on page 44).

[29] Joseph B. Keller. "Geometrical Theory of Diffraction". In: *Journal of the Optical Society of America* 52.2 (1962), pp. 116–130. DOI: 10.1364/JOSA.52.000116 (cited on pages 44, 45, 82).

[30] Robert G. Kouyoumjian and Prabhakar H. Pathak. "A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface". In: *Proceedings of the IEEE* 62.11 (1974), pp. 1448–1461. DOI: 10.1109/PROC.1974.9651 (cited on pages 44, 74, 86).

[31] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. "Ray tracing on programmable graphics hardware". In: *ACM Transactions on Graphics* 21.3 (2002), pp. 703–712. DOI: 10.1145/566654.566640 (cited on page 44).

[32] Steven G. Parker et al. "OptiX: a general purpose ray tracing engine". In: *ACM Transactions on Graphics* 29.4 (2010). DOI: 10.1145/1778765.1778803 (cited on pages 45, 200).

[33] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. "Differentiable Monte Carlo ray tracing through edge sampling". In: *ACM Transaction on Graphics* 37.6 (2018). DOI: 10.1145/3272127.3275109 (cited on pages 45, 168).

[34] Martín Abadi et al. *TensorFlow, Large-scale machine learning on heterogeneous systems*. 2015. DOI: 10.5281/zenodo.4724125 (cited on pages 46, 150).

[35] Jason Ansel et al. "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation". In: *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. Association for Computing Machinery, 2024. DOI: 10.1145/3620665.3640366 (cited on pages 46, 150).

[36] Roy Frostig, Matthew Johnson, and Chris Leary. "Compiling machine learning programs via high-level tracing". In: *SysML Conference 2018*. Available at https://mlsys.org/Conferences/doc/2018/146.pdf. 2018 (cited on pages 46, 150, 210, 211).

[37] Jakob Hoydis et al. "Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling". In: *2023 IEEE Globecom Workshops (GC Wkshps)*. 2023, pp. 317–321. DOI: 10.1109/GCWkshps58843.2023.10465179 (cited on pages 46, 148, 162, 182, 200, 209, 215, 232, 280).

[38] Jérome Eertmans, Claude Oestges, and Laurent Jacques. "Demonstrating DiffeRT: An Open-Source Library for Optimizing Radio Networks with Differentiable Ray Tracing". In: *2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*. 2025, pp. 1–2. DOI: 10.1109/ICMLCN64995.2025.11139997 (cited on pages 46, 148, 184, 201, 209, 279).

[39] Rudolf K. Lüneburg. *Mathematical Theory of Optics*. Notes were later published in 1964 by University of California Press. 1944 (cited on page 47).

[40] Morris Kline. "An asymptotic solution of Maxwell's equations". In: *Communications on Pure and Applied Mathematics* 4.2-3 (1951), pp. 225–262. DOI: 10.1002/cpa.3160040203 (cited on page 47).

[41] Derek A. McNamara, Carl W. I. Pistorius, and Johannes A. G. Malherbe. *Introduction to the Uniform Geometrical Theory of Diffraction*. Artech House, 1990 (cited on pages 47, 51, 52, 56, 74, 75, 81, 84, 85, 89, 92, 93, 277).

[42] R. Clark Jones. "A New Calculus for the Treatment of Optical Systems. I. Description and Discussion of the Calculus". In: *Journal of the Optical Society of America* 31.7 (1941), pp. 488–493. DOI: 10.1364/JOSA.31.000488 (cited on page 67).

[43] Shung-Wu Lee, Yahya Rahmat-Samii, and Ronald C. Menendez. "GTD, ray field, and comments on two papers". In: *IEEE Transactions on Antennas and Propagation* 26.2 (1978), pp. 352–354. DOI: 10.1109/TAP.1978.1141839 (cited on page 93).

[44] Vittorio Degli-Esposti, Enrico M. Vitucci, Marco Di Renzo, and Sergei A. Tretyakov. "Reradiation and Scattering From a Reconfigurable Intelligent Surface: A General Macroscopic Model". In: *IEEE Transactions on Antennas and Propagation* 70.10 (2022), pp. 8691–8706. DOI: 10.1109/TAP.2022.3149660 (cited on pages 95, 99–101, 104, 105).

[45] Vittorio Degli-Esposti. "A diffuse scattering model for urban propagation prediction". In: *IEEE Transactions on Antennas and Propagation* 49.7 (2001), pp. 1111–1113. DOI: 10.1109/8.933491 (cited on page 94).

[46] Vittorio Degli-Esposti, Franco Fuschini, Enrico M. Vitucci, and Gabriele Falciasecca. "Measurement and Modelling of Scattering From Buildings". In: *IEEE Transactions on Antennas and Propagation* 55.1 (2007), pp. 143–153. DOI: 10.1109/TAP.2006.888422 (cited on pages 94, 96).

[47] Vittorio Degli-Esposti, Veli-Matti Kolmonen, Enrico M. Vitucci, and Pertti Vainikainen. "Analysis and Modeling on co- and Cross-Polarized Urban Radio Propagation for Dual-Polarized MIMO Wireless Systems". In: *IEEE Transactions on Antennas and Propagation* 59.11 (2011), pp. 4247–4256. DOI: 10.1109/TAP.2011.2164226 (cited on pages 96–98).

[48] Enrico M. Vitucci, Matteo Albani, Silvi Kodra, Marina Barbiroli, and Vittorio Degli-Esposti. "An Efficient Ray-Based Modeling Approach for Scattering From Reconfigurable Intelligent Surfaces". In: *IEEE Transactions on Antennas and Propagation* 72.3 (2024), pp. 2673–2685. DOI: 10.1109/TAP.2024.3359288 (cited on pages 99, 101–103).

[49] Jont B. Allen and David A. Berkley. "Image method for efficiently simulating small-room acoustics". In: *The Journal of the Acoustical Society of America* 65.4 (1979), pp. 943–950. DOI: 10.1121/1.382599 (cited on page 119).

[50] Jeffrey Borish. "Extension of the image model to arbitrary polyhedra". In: *The Journal of the Acoustical Society of America* 75.6 (1984), pp. 1827–1836. DOI: 10.1121/1.390983 (cited on page 119).

[51] Florian Quatresooz, Simon Demey, and Claude Oestges. "Tracking of Interaction Points for Improved Dynamic Ray Tracing". In: *IEEE Transactions on Vehicular Technology* 70.7 (2021), pp. 6291–6301. DOI: 10.1109/TVT.2021.3081766 (cited on pages 121, 190–192).

[52] Fernando Aguado Agelet, Arno Formella, Jose M. Hernando Rabanos, Fernando Isasi de Vicente, and Fernando Perez Fontan. "Efficient ray-tracing acceleration techniques for radio propagation modeling". In: *IEEE Transactions on Vehicular Technology* 49.6 (2000), pp. 2089–2104. DOI: 10.1109/25.901880 (cited on pages 121, 140, 186).

[53] Jérome Eertmans, Claude Oestges, and Laurent Jacques. "Min-Path-Tracing: A Diffraction Aware Alternative to Image Method in Ray Tracing". In: *2023 17th European Conference on Antennas and Propagation (EuCAP)*. 2023, pp. 1–5. DOI: 10.23919/EuCAP57121.2023.10132934 (cited on page 121).

[54] Giorgio Carluccio and Matteo Albani. "An Efficient Ray Tracing Algorithm for Multiple Straight Wedge Diffraction". In: *IEEE Transactions on Antennas and Propagation* 56.11 (2008), pp. 3534–3542. DOI: 10.1109/TAP.2008.2005540 (cited on pages 130, 133, 205).

[55] Jérome Eertmans, Sophie Lequeu, Benoît Legat, Laurent Jacques, and Claude Oestges. "Fast, Differentiable, GPU-Accelerated Ray Tracing for Multiple Diffraction and Reflection Paths". Accepted at EuCAP 2026 - 20th European Conference on Antennas and Propagation. 2025. DOI: 10.48550/arXiv.2510.16172 (cited on pages 130, 133, 162, 205–207).

[56] Federico Puggelli, Giorgio Carluccio, and Matteo Albani. "A Novel Ray Tracing Algorithm for Scenarios Comprising Pre-Ordered Multiple Planar Reflectors, Straight Wedges, and Vertexes". In: *IEEE Transactions on Antennas and Propagation* 62.8 (2014), pp. 4336–4341. DOI: 10.1109/TAP.2014.2323961 (cited on pages 133, 205, 208).

[57] Armin Erraji, Jonas Stienen, and Michael Vorländer. "The image edge model". In: *Acta Acustica* 5 (2021), p. 17. DOI: 10.1051/aacus/2021010 (cited on page 133).

[58] Shin-Hou Chen and Shyh-Kang Jeng. "All SBR/image approach to indoor radio propagation modeling". In: *IEEE Antennas and Propagation Society International Symposium. 1995 Digest.* Vol. 4. 1995, pp. 1952–1955. DOI: 10.1109/APS.1995.530974 (cited on page 138).

[59] Benjamin Keinert, Matthias Innmann, Michael Sänger, and Marc Stamminger. "Spherical Fibonacci Mapping". In: *ACM Transactions on Graphics* 34.6 (2015), pp. 1–7. DOI: 10.1145/2816795.2818131 (cited on page 138).

[60] Stephen Kasdorf, Blake Troksa, Cam Key, Jake Harmon, and Branislav M. Notaroš. "Advancing Accuracy of Shooting and Bouncing Rays Method for Ray-Tracing Propagation Modeling Based on Novel Approaches to Ray Cone Angle Calculation". In: *IEEE Transactions on Antennas and Propagation* 69.8 (2021), pp. 4808–4815. DOI: 10.1109/TAP.2021.3060051 (cited on page 138).

[61] Dan Shi, Junjian Bi, Zhiliang Tan, and Yougang Gao. "Site-specific wave propagation prediction with improved shooting and bouncing ray tracing method". In: *2015 1st URSI Atlantic Radio Science Conference (URSI AT-RASC).* 2015, pp. 1–1. DOI: 10.1109/URSI-AT-RASC.2015.7303040 (cited on page 138).

[62] Gregory D. Durgin, Neal Patwari, and Theodore S. Rappaport. "Improved 3D ray launching method for wireless propagation prediction". In: *Electronics Letters* 33 (16 1997), pp. 1412–1413. DOI: 10.1049/el:19970928 (cited on page 139).

[63] Roman Novak. "Inconsistent Rays in Propagation Prediction by Ray Launching in Rectangular Tunnels". In: *IEEE Access* 10 (2022), pp. 127056–127066. DOI: 10.1109/ACCESS.2022.3223868 (cited on page 139).

[64] Zhengqing Yun, Magdy F. Iskander, and Zhijun Zhang. "Development of a new shooting-and-bouncing ray (SBR) tracing method that avoids ray double counting". In: *IEEE Antennas and Propagation Society International Symposium.* Vol. 1. 2001, pp. 464–467. DOI: 10.1109/APS.2001.958892 (cited on page 140).

[65] Roman Novak. "Bloom Filter for Double-Counting Avoidance in Radio Frequency Ray Tracing". In: *IEEE Transactions on Antennas and Propagation* 67.4 (2019), pp. 2176–2190. DOI: 10.1109/TAP.2019.2905780 (cited on page 140).

[66] George Liang and Henry L. Bertoni. "A new approach to 3-D ray tracing for propagation prediction in cities". In: *IEEE Transactions on Antennas and Propagation* 46.6 (1998), pp. 853–863. DOI: 10.1109/8.686774 (cited on page 140).

[67] Jundong Tan, Zhuo Su, and Yunliang Long. "A Full 3-D GPU-based Beam-Tracing Method for Complex Indoor Environments Propagation Modeling". In: *IEEE Transactions on Antennas and Propagation* 63.6 (2015), pp. 2705–2718. DOI: 10.1109/TAP.2015.2415036 (cited on page 141).

[68] Te-Shun Wang and Chang-Fa Yang. "A global ray-tube tracing method to determine signal variations in urban areas for mobile communications". In: *IEEE Antennas and Propagation Society International Symposium*. Vol. 2. 2003, pp. 922–925. DOI: 10.1109/APS.2003.1219312 (cited on page 141).

[69] Iuliia Tropkina, Alexander Pyattaev, Yekaterina Sadovaya, and Sergey Andreev. "Modeling of SHF/EHF Radio-Wave Scattering for Curved Surfaces With Voxel Cone Tracing". In: *IEEE Antennas and Wireless Propagation Letters* 21.2 (2022), pp. 426–430. DOI: 10.1109/LAWP.2021.3134953 (cited on page 141).

[70] Fayçal A. Aoudia, Jakob Hoydis, Merlin Nimier-David, Baptiste Nicolet, Sebastian Cammerer, and Alexander Keller. "Sionna RT: Technical Report". 2025. DOI: 10.48550/arXiv.2504.21719 (cited on pages 142, 277).

[71] Eric Veach. "Robust Monte Carlo Methods for Light Transport Simulation". Available at https://graphics.stanford.edu/papers/veach_thesis/thesis-bw.pdf. PhD thesis. Stanford University, 1997 (cited on pages 142, 143).

[72] Mehmet Mert Taygur. "Wave Propagation Simulations by Bidirectional Ray-Tracing and Investigations on Ray Based Channel Modeling for Massive MIMO". Available at https://mediatum.ub.tum.de/doc/1691763/1691763.pdf. PhD thesis. TUM School of Computation, Information and Technology, Technical University of Munich, 2023 (cited on page 143).

[73] Eric Veach and Leonidas J. Guibas. "Metropolis Light Transport". In: *Proceedings of SIGGRAPH 97*. Addison Wesley, 1997, pp. 65–76. DOI: 10.1145/258734.258775 (cited on page 143).

[74] Arne Schmitz and Leif Kobbelt. "Wave propagation using the photon path map". In: *Proceedings of the 3rd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks*. PE-WASUN '06. Terromolinos, Spain: Association for Computing Machinery, 2006, pp. 158–161. DOI: 10.1145/1163610.1163638 (cited on page 144).

[75] Jon L. Bentley. "Multidimensional binary search trees used for associative searching". In: *Communications of the ACM* 18.9 (1975), pp. 509–517. DOI: 10.1145/361002.361007 (cited on pages 144, 196).

[76] Manjuladevi Vasudevan and Murat Yuksel. "Machine Learning for Radio Propagation Modeling: A Comprehensive Survey". In: *IEEE Open Journal of the Communications Society* 5 (2024), pp. 5123–5153. DOI: 10.1109/OJCOMS.2024.3446457 (cited on pages 144, 145).

[77] Shuchen Wang et al. "A Physics-Informed Deep Ray Tracing Network for Regional Channel Impulse Response Estimation". In: *IEEE Transactions on Wireless Communications* 24.7 (2025), pp. 5811–5824. DOI: 10.1109/TWC.2025.3549498 (cited on pages 144, 239).

[78] Yifei Jin, Ali Maatouk, Sarunas Girdzijauskas, Shugong Xu, Leandros Tassiulas, and Rex Ying. "SANDWICH: Towards an Offline, Differentiable, Fully-Trainable Wireless Neural Ray-Tracing Surrogate". In: *2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*. 2025, pp. 1–7. DOI: 10.1109/ICMLCN64995.2025.11139897 (cited on pages 144, 239).

[79] Jérome Eertmans, Enrico M. Vitucci, Vittorio Degli-Esposti, Nicola Di Cicco, Laurent Jacques, and Claude Oestges. "Transform-Invariant Generative Ray Path Sampling for Efficient Radio Propagation Modeling". Submitted to npj Wireless Technology. 2026. DOI: 10.48550/arXiv.2603.01655 (cited on pages 145, 182, 236, 249, 259, 262).

[80] Cheng-Xiang Wang, Jie Huang, Haiming Wang, Xiqi Gao, Xiaohu You, and Yang Hao. "6G Wireless Channel Measurements and Models: Trends and Challenges". In: *IEEE Vehicular Technology Magazine* 15.4 (2020), pp. 22–32. DOI: 10.1109/MVT.2020.3018436 (cited on page 148).

[81] Mostafa Zaman Chowdhury, Mohammad Shahjalal, Shakil Ahmed, and Yeong Min Jang. "The role of massive MIMO in 6G wireless: advancements, challenges, and opportunities". In: *IEEE Open Journal of the Communications Society* 1 (2020), pp. 957–975. DOI: 10.1109/OJCOMS.2020.3010270 (cited on page 148).

[82] Marco Di Renzo et al. "Smart radio environments empowered by reconfigurable AI meta-surfaces: an idea whose time has come". In: *EURASIP Journal on Wireless Communications and Networking* 2019.129 (2019), pp. 1–20. DOI: 10.1186/s13638-019-1438-9 (cited on page 148).

[83] Qingqing Wu and Rui Zhang. "Towards Smart and Reconfigurable Environment: Intelligent Reflecting Surface Aided Wireless Network". In: *IEEE Communications Magazine* 58.1 (2020), pp. 106–112. DOI: 10.1109/MCOM.001.1900107 (cited on page 148).

[84] Paolo Testolina, Michele Polese, Pedram Johari, and Tommaso Melodia. "Boston Twin: the Boston Digital Twin for Ray-Tracing in 6G Networks". In: *Proceedings of the 15th ACM Multimedia Systems Conference*. MMSys '24. Bari, Italy: Association for Computing Machinery, 2024, pp. 441–447. DOI: 10.1145/3625468.3652190 (cited on page 148).

[85] Jérome Eertmans, Claude Oestges, and Laurent Jacques. "DiffeRT2d: A Differentiable Ray Tracing Python Framework for Radio Propagation". In: *Journal of Open Source Software* 9.98 (2024). The Open Journal publishes brief articles that undergo an open peer-review process on GitHub issues., p. 6915. DOI: 10.21105/joss.06915 (cited on pages 148, 210).

[86] Tribhuvanesh Orekondy, Pratik Kumar, Shreya Kadambi, Hao Ye, Joseph Soriaga, and Arash Behboodi. "WiNeRT: Towards Neural Ray Tracing for Wireless Channel Modelling and Differentiable Simulations". In: *The Eleventh International Conference on Learning Representations*. Available at https://openreview.net/forum?id=tPKKXeW33YU. 2023 (cited on page 150).

[87] Huiying Yang, Zihan Jin, Chenhao Wu, Rujing Xiong, Robert Caiming Qiu, and Zenan Ling. *R-NeRF: Neural Radiance Fields for Modeling RIS-enabled Wireless Environments*. 2024. URL: https://arxiv.org/abs/2405.11541 (cited on pages 150, 238).

[88] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *The Third International Conference on Learning Representations*. 2015. DOI: 10.48550/arXiv.1412.6980 (cited on pages 150, 226).

[89] Andreas Griewank. "On automatic differentiation". In: *Mathematical programming (Tokyo, 1988)*. Vol. 6. Math. Appl. (Japanese Ser.) Available at https://softlib.rice.edu/pub/CRPC-TRs/reports/CRPC-TR89003.pdf. SCIPRESS, Tokyo, 1989, pp. 83–107 (cited on page 150).

[90] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. 2nd. Society for Industrial and Applied Mathematics, 2008 (cited on page 150).

[91] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. DOI: 10.1038/323533a0 (cited on page 154).

[92] Niklas Vaara, Pekka Sangi, Miguel Bordallo López, and Janne Heikkilä. "Differentiable High-Performance Ray Tracing-Based Simulation of Radio Propagation With Point Clouds". In: *IEEE Antennas and Wireless Propagation Letters* (2026), pp. 1–5. DOI: 10.1109/LAWP.2026.3670638 (cited on page 162).

[93] Niklas Vaara, Pekka Sangi, Miguel Bordallo López, and Janne Heikkilä. "Ray Launching-Based Computation of Exact Paths With Noisy Dense Point Clouds". In: *IEEE Transactions on Antennas and Propagation* 73.5 (2025), pp. 3270–3283. DOI: 10.1109/TAP.2025.3546110 (cited on pages 162, 179).

[94] Tom M. Apostol. *Mathematical analysis: a modern approach to advanced calculus*. Reading, Massachusetts: Addison Wesley, 1957 (cited on page 162).

[95] Kevinn Giancarlo Castro Farfan. "Towards high-fidelity radio coverage maps: calibration using real-world measurements in Sionna RT". Available at https://hdl.handle.net/2117/452369. MA thesis. UPC, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Departament d'Arquitectura de Computadors, 2025 (cited on pages 166, 229, 233).

[96] Xueqiang Han, Jinyang Huang, Meng Li, Chao Cai, and Tianyue Zheng. "Loki: Physical-World Adversarial Attacks on Wireless Indoor Localization via Differentiable Object Placement". In: *IEEE Transactions on Information Forensics and Security* 20 (2025), pp. 13386–13400. DOI: 10.1109/TIFS.2025.3642536 (cited on pages 166, 229, 231).

[97] Michael Fischer and Tobias Ritschel. "Plateau-Reduced Differentiable Path Tracing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4285–4294. DOI: 10.48550/arXiv.2211.17263 (cited on pages 166, 168).

[98] Jérome Eertmans, Laurent Jacques, and Claude Oestges. "Fully Differentiable Ray Tracing via Discontinuity Smoothing for Radio Network Optimization". In: *2024 18th European Conference on Antennas and Propagation (EuCAP)*. 2024, pp. 1–5. DOI: 10.23919/EuCAP60739.2024.10501570 (cited on pages 166, 168, 169, 173).

[99] Sébastien Bubeck. "Convex Optimization: Algorithms and Complexity". In: *Foundations and Trends in Machine Learning* 8.3–4 (2015), pp. 231–357. DOI: 10.1561/2200000050 (cited on page 167).

[100] Xueqiang Han, Tianyue Zheng, Tony Xiao Han, and Jun Luo. *RayLoc: Wireless Indoor Localization via Fully Differentiable Ray-tracing*. 2025. DOI: 11.48550/arXiv.2501.17881 (cited on pages 168, 229).

[101] Daoqi Liu, Maokun Li, Fan Yang, and Shenheng Xu. "Fully Differentiable Shooting and Bouncing Ray Method for Solving Inverse Scattering Problems: 3-D PEC Targets". In: *2025 International Applied Computational Electromagnetics Society Symposium (ACES-China)*. 2025, pp. 1–3. DOI: 10.23919/ACES-China66523.2025.11333258 (cited on page 168).

[102] OSM Buildings Contributors. *OSM Buildings: A free and open project for 3D buildings*. Data by OpenStreetMap contributors. OSM Buildings. 2026. URL: https://osmbuildings.org/ (visited on 03/25/2026) (cited on page 178).

[103] NASA Earthdata. *Digital Elevation/Terrain Model (DEM)*. Version March 19, 2026. NASA. 2026. URL: https://www.earthdata.nasa.gov/topics/land-surface/digital-elevation-terrain-model-dem (visited on 03/25/2026) (cited on page 178).

[104] Jonathan S. Lu et al. "A Discrete Environment-Driven GPU-Based Ray Launching Algorithm". In: *IEEE Transactions on Antennas and Propagation* 67.2 (2019), pp. 1180–1192. DOI: 10.1109/TAP.2018.2880036 (cited on page 186).

[105] Denis Bilibashi, Enrico M. Vitucci, and Vittorio Degli-Esposti. "Dynamic Ray Tracing: Introduction and Concept". In: *2020 14th European Conference on Antennas and Propagation (EuCAP)*. 2020, pp. 1–5. DOI: 10.23919/EuCAP48036.2020.9135577 (cited on pages 190, 191).

[106] Jérome Eertmans, Enrico M. Vitucci, Vittorio Degli-Esposti, Laurent Jacques, and Claude Oestges. "Comparing Differentiable and Dynamic Ray Tracing: Introducing the Multipath Lifetime Map". In: *2025 19th European Conference on Antennas and Propagation (EuCAP)*. 2025, pp. 1–5. DOI: 10.23919/EuCAP63536.2025.10999736 (cited on pages 192, 202, 203, 205, 266, 269).

[107] Tomas Möller and Ben Trumbore. "Fast, minimum storage ray-triangle intersection". In: *Journal of Graphics Tools* 2.1 (1997), pp. 21–28. DOI: 10.1080/10867651.1997.10487468 (cited on pages 193, 194).

[108] James H. Clark. "Hierarchical geometric models for visible surface algorithms". In: *Communications of the ACM* 19.10 (1976), pp. 547–554. DOI: 10.1145/360349.360354 (cited on page 196).

[109] Jeffrey Goldsmith and John Salmon. "Automatic creation of object hierarchies for ray tracing". In: *IEEE Computer Graphics and Applications*. Vol. 7. 5. 1987, pp. 14–20. DOI: 10.1109/MCG.1987.276983 (cited on page 196).

[110] Esteban Egea-Lopez, Jose Maria Molina-Garcia-Pardo, Martine Lienard, and Pierre Degauque. "Opal: An open source ray-tracing propagation simulator for electromagnetic characterization". In: *PLOS ONE* 16.11 (2021), pp. 1–19. DOI: 10.1371/journal.pone.0260060 (cited on page 200).

[111] Wenzel Jakob et al. *Mitsuba 3 renderer*. Version 3.8.0. Available at https://mitsuba-renderer.org. 2022 (cited on pages 200, 209, 280).

[112] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. "Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering". In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 41.4 (2022). DOI: 10.1145/3528223.3530099 (cited on pages 200, 210).

[113] Alexander Domahidi, Eric Chu, and Stephen Boyd. "ECOS: An SOCP Solver for Embedded Systems". In: *2013 European Control Conference (ECC)*. Zurich: IEEE, 2013, pp. 3071–3076. DOI: 10.23919/ECC.2013.6669541 (cited on page 208).

[114] Sophie Lequeu. "Wireless indoor source localization as an inverse problem: An optimization approach via ray tracing simulations". Available at https://hdl.handle.net/2078.2/42843. MA thesis. École polytechnique de Louvain, 2025 (cited on pages 229, 230).

[115] Jakob Hoydis et al. "Learning Radio Environments by Differentiable Ray Tracing". In: *IEEE Transactions on Machine Learning in Communications and Networking* 2 (2024), pp. 1527–1539. DOI: 10.1109/TMLCN.2024.3474639 (cited on pages 229, 232).

[116] Xueqiang Han, Tianyue Zheng, and Jun Luo. "Stereo-Fi: Free-form 3D Reconstruction via Generatively Co-Trained Inverse RF Rendering". In: Accepted at SenSys '26 - 24th ACM Conference on Embedded Networked Sensor Systems. 2026, pp. 1–14 (cited on page 229).

[117] Jérome Eertmans, Nicola Di Cicco, Claude Oestges, Laurent Jacques, Enrico M. Vitucci, and Vittorio Degli-Esposti. "Towards Generative Ray Path Sampling for Faster Point-to-Point Ray Tracing". In: *2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*. 2025, pp. 1–6. DOI: 10.1109/ICMLCN64995.2025.11140249 (cited on page 236).

[118] Lina Wu et al. "Artificial Neural Network Based Path Loss Prediction for Wireless Communication Network". In: *IEEE Access* 8 (2020), pp. 199523–199538. DOI: 10.1109/ACCESS.2020.3035209 (cited on page 238).

[119] Segun I. Popoola et al. "Determination of Neural Network Parameters for Path Loss Prediction in Very High Frequency Wireless Channel". In: *IEEE Access* 7 (2019), pp. 150462–150483. DOI: 10.1109/ACCESS.2019.2947009 (cited on page 238).

[120] Jakob Thrane, Darko Zibar, and Henrik Lehrmann Christiansen. "Model-Aided Deep Learning Method for Path Loss Prediction in Mobile Communication Systems at 2.6 GHz". In: *IEEE Access* 8 (2020), pp. 7925–7936. DOI: 10.1109/ACCESS.2020.2964103 (cited on page 238).

[121] Xiaopeng Zhao, Zhenlin An, Qingrui Pan, and Lei Yang. "NeRF2: Neural Radio-Frequency Radiance Fields". In: *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. ACM MobiCom '23. Madrid, Spain: Association for Computing Machinery, 2023. DOI: 10.1145/3570361.3592527 (cited on page 238).

[122] Jingzhou Shen, Tianya Zhao, Yanzhao Wu, and Xuyu Wang. "NeRF-APT: A New NeRF Framework for Wireless Channel Prediction". In: *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2025, pp. 1–6. DOI: 10.1109/INFOCOMWKSHPS65812.2025.11152993 (cited on page 238).

[123] Chaozheng Wen, Jingwen Tong, Yingdong Hu, Zehong Lin, and Jun Zhang. "Neural Representation for Wireless Radiation Field Reconstruction: A 3D Gaussian Splatting Approach". In: *IEEE Transactions on Wireless Communications* 25 (2025), pp. 7490–7504. DOI: 10.1109/TWC.2025.3631663 (cited on page 238).

[124] Lihao Zhang, Haijian Sun, Samuel Berweger, Camillo Gentile, and Rose Qingyang Hu. "RF-3DGS: Wireless Channel Modeling With Radio Radiance Field and 3D Gaussian Splatting". In: *IEEE Transactions on Wireless Communications* 25 (2026), pp. 10419–10433. DOI: 10.1109/TWC.2026.3652154 (cited on page 238).

[125] Xingyu Chen, Zihao Feng, Kun Qian, and Xinyu Zhang. "Radio Frequency Ray Tracing with Neural Object Representation for Enhanced RF Modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2025, pp. 21339–21348 (cited on page 239).

[126] Thomas Hehn, Markus Peschl, Tribhuvanesh Orekondy, Arash Behboodi, and Johann Brehmer. "Geometric Wireless Simulation with Equivariant Transformers". In: *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*. 2024 (cited on page 240).

[127] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. "Deep Sets". In: *Advances in Neural Information Processing Systems*. Vol. 30. Red Hook, NY, USA: Curran Associates, Inc., 2017 (cited on page 244).

Nomenclature

The following list describes several symbols that are commonly used within the body of the document.

Physics constants

c	Speed of light in a vacuum	$299\,792\,458\ \mathrm{m\,s^{-1}}$
ϵ_0	Vacuum permittivity	$8.854\,187\,818\,8 \times 10^{-12}\ \mathrm{F\,m^{-1}}$
μ_0	Vacuum permeability	$4\pi \times 10^{-7}\ \mathrm{H\,m^{-1}}$
Z	Intrinsic impedance of a medium	Ω
Z_0	Free-space impedance	$376.730\,313\,412\ \Omega$

Notation

a, A	Scalar quantity	italic lowercase or uppercase letter
$\boldsymbol{a}$	Vector quantity	bold-italic lowercase letter
$\boldsymbol{A}$	Matrix or tensor quantity	bold-italic uppercase letter
$\mathbf{a}$, $\mathbf{A}$	2- or 3-dimensional vector	bold lowercase or uppercase letter
$\hat{\mathbf{n}}$	Normal vector	bold lowercase letter with hat
$\underline{\mathbf{R}}$	Polarization dyadic (2×2 matrix)	bold uppercase letter with underline
a_i	i-th scalar component of vector $\boldsymbol{a}$	
$\boldsymbol{a}_i$	i-th vector component of matrix $\boldsymbol{A}$	
$\boldsymbol{a}_{i,j}$	Indexed (row then column) scalar component of matrix $\boldsymbol{A}$	
a^*	Complex conjugate (and transpose for matrices)	
$a^\star$	Solution to an optimization problem	
$\boldsymbol{\theta}^{(i)}$	Vector of (model) parameters at iteration i	

$\bar{x}$	Average or mean value of x	
$\cdot$	Vector dot product	
$\times$	Vector cross product	
∇	Vector differential operator	
$\nabla\cdot$	Divergence operator	
$\nabla\times$	Curl operator	
∇^2	Laplacian operator	
$\nabla\nabla$	Hessian matrix operator	
∂	Partial derivative operator	
$\boldsymbol{Ab}$	Matrix-vector product	
$\boldsymbol{AB}$	Matrix-matrix product	
$\mathcal{O}$	Big O notation for algorithmic complexity	

Special symbols

j	Imaginary unit	
Re	Real part of a complex number	
Im	Imaginary part of a complex number	
f	Frequency	Hz
τ	Delay	s
ω	Angular frequency	$\mathrm{rad\,s^{-1}}$
θ	Polar angle, $\theta \in [0, \pi]$, measured from the local z-axis	rad
φ	Azimuthal angle, $\varphi \in [0, 2\pi]$, measured from the local x-axis	rad
σ	Surface conductivity	S
ρ_v	Electric volume charge density	$\mathrm{C\,m^{-3}}$

ϵ	Permittivity	$\mathrm{F\,m^{-1}}$
ϵ_r	Relative Permittivity	
Z	Intrinsic impedance	Ω
v	Phase velocity	$\mathrm{m\,s^{-1}}$
λ	Wavelength	m
k	Wavenumber	$\mathrm{rad\,m^{-1}}$
d_f	Fraunhofer distance	m
$\mathbf{r}$	Position vector	m
$\mathscr{E}$	Electric field (time domain)	$\mathrm{V\,m^{-1}}$
$\mathbf{E}$	Electric field (phasor domain)	$\mathrm{V\,m^{-1}}$
$\mathbf{B}$	Magnetic flux density	T
$\mathbf{H}$	Magnetic field strength	$\mathrm{A\,m^{-1}}$
$\mathbf{D}$	Electric displacement field	$\mathrm{C\,m^{-2}}$
$\mathbf{J}$	Current density	$\mathrm{A\,m^{-2}}$
$\mathbf{S}$	Poynting vector	$\mathrm{W\,m^{-2}}$
$\mathbf{r}$	Position vector	m
Ψ	Eikonal (phase) function	m
$\mathbf{P}$	Point of observation	m
$\mathbf{x}_r$	Point of reflection on a surface	m
$\mathbf{x}_e$	Point of diffraction on an edge	m
$\rho_{1,2}$	Principal radii of wavefront curvature	m

Acronyms

AD automatic differentiation 150–155, 157, 158, 160, 161, 163, 167, 175, 200, 207
API application programming interface 200, 210, 279, 288

BDPT bidirectional path tracing 143
BFGS Broyden–Fletcher–Goldfarb–Shanno 202, 203, 205, 213
BVH bounding volume hierarchy 196–200, 213, 263

CA Carluccio & Albani 205–207
CGWA Contact Group on "Wavelets and Applications" ix
CIR channel impulse response 111, 144, 219, 220, 223, 235
CNN convolutional neural network 238
CO canyon-only 249, 250, 259, 260
COST European Cooperation in Science and Technology vii, ix
CPU central processing unit 44, 45, 160, 196, 205, 208, 262, 265, 271

DAG directed acyclic graph 151

EHF extra high frequency 6, 7, 141
EIRP effective isotropic radiated power 38
ER effective roughness 94
EuCAP European Conference on Antennas and Propagation ix, xi

FDTD finite-difference time-domain 39
FEM finite element method 39
FNRS Fonds de la Recherche Scientifique - FNRS ix

GD gradient descent 205, 206
GFlowNet generative flow network 241, 243, 244
GO geometrical optics xiii, 40, 50–53, 55, 57, 60, 61, 70, 71, 75–77, 82, 86, 94, 102, 104, 105, 225, 273, 274, 277
GPU graphics processing unit vi, xi, xv, 44–46, 133, 160, 196, 197, 199, 200, 202, 205, 206, 208, 210, 213, 229, 262, 265, 271, 276
GS Gaussian splatting 238
GTD geometrical theory of diffraction 32, 44, 68, 82, 84, 86, 89, 90, 93

HF high frequency 6, 7

ICMLCN International Conference on Machine Learning for Communications and Networking ix
IM image method 204, 206
INTERACT Intelligence-Enabling Radio Communications for Seamless Inclusive Interactions vii, ix
ISB incident shadow boundary 77, 79, 83–85, 89, 92, 224

JIT just-in-time x, 46, 199–202, 210–213, 276, *see* just-in-time compilation
JVP Jacobian-vector product 151, 153, 158, 162

L-BFGS limited-memory BFGS 205–207
LF low frequency 6, 7

MC Monte Carlo 142, 143, 145
MF medium frequency 6, 7
MIMO multiple-input multiple-output 111, 219
MLM multipath lifetime map vi, xi, 192, 266–271
MLP multilayer perceptron 238, 244
MLT Metropolis light transport 143
MoM method of moments 39
MPT min-path-tracing vi, xi, 121, 122, 124–127, 129, 134, 145, 147, 180, 204, 274

NDRC National Defense Research Committee 42
NeRF neural radiance field 238

OG optional ground 249, 250, 260
OSM OpenStreetMap 178

PEC perfect electrical conductor 73, 80, 81, 87
PO physical optics 44, 60, 61, 104, 105

RF radio frequency 238
RIS reconfigurable intelligent surface 33, 34, 99–104, 109, 113, 114, 122, 127, 148–150, 238, 275
RSB reflected shadow boundary 77, 79, 83–86, 89, 92, 224
RX receiver 41, 110, 112, 119–123, 126, 134, 173, 174, 178, 183, 186, 187, 189–191, 219, 221, 223, 230, 242–245, 249, 252–254, 257, 258, 266, 280, 281, 292, 296, 298–300, 302, 303

SAL Silicon Austria Labs ix
SBR shooting and bouncing rays 44, 45, 137–145, 168, 267, 268, 291–297
SHF super high frequency 6, 7, 141

SINR signal-to-interference-plus-noise ratio 223, 225, 228
SISO single-input single-output 219
SOCP second-order cone program 205, 208, 213
SOFAR sound fixing and ranging 42

TE transverse electric 80, 97, 106
TEM transverse electromagnetic 12, 48
TM transverse magnetic 80, 97, 106
TPU tensor processing unit 160
TX transmitter 41, 110, 112, 119–123, 126, 134, 178, 183, 186, 187, 189–191, 216, 219, 242–245, 249, 252–254, 257, 258, 266, 269, 270, 280, 281, 290, 291, 298–300, 302, 303

UHF ultra high frequency 6, 7
UTD uniform theory of diffraction xiv, 32, 40, 44, 61, 68, 77, 86, 89, 91–93, 114, 225, 273, 274, 277

V2X vehicle-to-everything 190, 231
VHF very high frequency 6, 7
VJP vector-Jacobian product 151, 156, 158, 162
VLF very low frequency 6, 7
VPL vertical plane launching 140, 145

WRF wireless radiation field 238
WS whole scene 250, 259–261

XLA Accelerated Linear Algebra 160, 161, 199, 201, 210, 211
XPD cross-polarization discrimination 98

Glossary

far field is the region far from an antenna, where the electromagnetic field behaves as a simple radiating wave and the wavefronts can be approximated as locally planar 12, 19, 20, 58, 61–64

free space refers to a perfect vacuum, but in practice is often used to refer to air as they have similar electromagnetic properties at radio frequencies 2, 5, 6, 9, 12, 13, 17–19, 22, 23, 33–35, 38, 39, 61, 102, 106, 108–110, 130

geodesy is the science of measuring and understanding the Earth's geometric shape, orientation in space, and gravity field 41

just-in-time compilation is the process of compiling code upon its first execution, in order to generate optimized machine code for subsequent executions 324

line-of-sight is the region where a direct path exists between the transmitter and receiver, without any obstacles in between 3, 22, 27, 34, 35, 134, 138, 173, 175, 188, 215–217, 220–224, 229, 230, 235, 242, 267, 269, 283, 284, 298, 299

near field is the region close to an antenna, where the electromagnetic field does not behave as a simple radiating wave and is dominated by reactive fields 12, 18, 61, 275

rasterization is the process of converting vector graphics into a raster image (pixels or dots), usually for output on a video display or printer 44

shadow region is a spatial region where a signal, whether direct or indirect, emitted from a source is blocked by an obstacle 27, 29, 42, 60, 77, 84, 86, 91, 173

wavelength is the spatial period of a wave (in m), defined as the distance over which the wave's shape repeats 5, 6, 14, 19, 26–28, 33, 34, 39, 46, 60, 65, 73, 93, 99, 105, 108, 110, 117, 221, 274, 275, 326

wavenumber is the spatial frequency of a wave (in m^{-1}), defined as the number of wavelengths per unit distance 11, 47, 75, 102, 108