UCLouvain

# Differentiable Ray Tracing for Radio Propagation

**Private PhD Defense**

Jérome Eertmans

Supervisors: Laurent Jacques & Claude Oestges

Jury: C. Craeye (Chairperson), C. De Vleeschouwer (Secretary),
P. De Doncker (ULB), E. M. Vitucci (UniBo), J. Hoydis (NVIDIA)

ICTEAM, Université catholique de Louvain — May 5, 2026

# 1. What is Ray Tracing for Radio?

# 1. What is Ray Tracing for Radio?

[Illustration: 2D RT scene
TX → reflections → RX]

# 1. What is Ray Tracing for Radio?

- Radio waves propagate through complex environments (cities, indoors, tunnels).

[Illustration: 2D RT scene
TX → reflections → RX]

# 1. What is Ray Tracing for Radio?

- Radio waves propagate through complex environments (cities, indoors, tunnels).
- Ray Tracing (RT) simulates individual ray paths between transmitter and receiver.

# 1. What is Ray Tracing for Radio?

- Radio waves propagate through complex environments (cities, indoors, tunnels).
- Ray Tracing (RT) simulates individual ray paths between transmitter and receiver.
- Each path undergoes interactions: reflection, diffraction, scattering.

# 1. What is Ray Tracing for Radio?

- Radio waves propagate through complex environments (cities, indoors, tunnels).
- Ray Tracing (RT) simulates individual ray paths between transmitter and receiver.
- Each path undergoes interactions: reflection, diffraction, scattering.
- RT provides site-specific channel models used for network planning and 5G/6G design.

# Why Differentiable Ray Tracing?

# Why Differentiable Ray Tracing?

- Differentiable RT allows computing gradients of any output w.r.t. any input parameter.

# Why Differentiable Ray Tracing?

- Differentiable RT allows computing gradients of any output w.r.t. any input parameter.
- Enables inverse problems: antenna placement, material calibration, beamforming.

# Why Differentiable Ray Tracing?

- Differentiable RT allows computing gradients of any output w.r.t. any input parameter.
- Enables inverse problems: antenna placement, material calibration, beamforming.
- End-to-end optimization through the full RT pipeline using automatic differentiation (AD).

# Why Differentiable Ray Tracing?

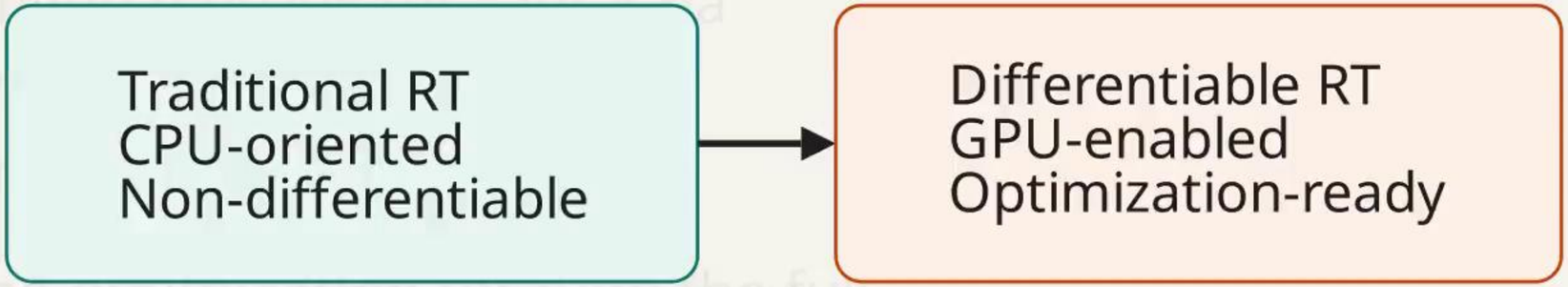
- Differentiable RT allows computing gradients of any output w.r.t. any input parameter.
- Enables inverse problems: antenna placement, material calibration, beamforming.
- End-to-end optimization through the full RT pipeline using automatic differentiation (AD).
- Naturally integrates with machine learning frameworks (JAX, PyTorch).

# Why Differentiable Ray Tracing?

- Differentiable RT allows computing gradients of any output w.r.t. any input parameter.
- Enables inverse problems, such as placement optimization, beamforming, and calibration.
- End-to-end optimization through the full RT pipeline using automatic differentiation (AD).
- Naturally integrates with machine learning frameworks (JAX, PyTorch).

Traditional RT
CPU-oriented
Non-differentiable

# Why Differentiable Ray Tracing?



# Key Challenges

# Key Challenges

- Speed: tracing thousands to millions of path candidates in real time.

# Key Challenges

- Speed: tracing thousands to millions of path candidates in real time.
- Mixed interactions: handling reflection and diffraction in a unified framework.

# Key Challenges

- Speed: tracing thousands to millions of path candidates in real time.
- Mixed interactions: handling reflection and diffraction in a unified framework.
- GPU constraints: avoiding branching, warp divergence, and excessive memory.

# Key Challenges

- Speed: tracing thousands to millions of path candidates in real time.
- Mixed interactions: handling reflection and diffraction in a unified framework.
- GPU constraints: avoiding branching, warp divergence, and excessive memory.
- Differentiability: maintaining end-to-end gradient flow through iterative solvers.

# Key Challenges

- Speed: tracing thousands to millions of path candidates in real time.
- Mixed interactions: handling reflection and diffraction in a unified framework.
- GPU constraints: avoiding branching, warp divergence, and excessive memory.
- Differentiability: maintaining end-to-end g

**This thesis addresses these challenges through three main contributions.**

# PhD Journey: A Timeline

# PhD Journey: A Timeline

# PhD Journey: A Timeline


**① Smoothing** **② ML Path Tracing** **③ Fermat PT**

# Talk Roadmap

1. Context & Motivation
2. Smoothing Technique (EuCAP 2024)
3. ML-Based Generative Path Tracing (ICMLCN 2025)
4. Fermat Path Tracing (EuCAP 2026)
5. Conclusion & Future Directions

# The Path Tracing Problem

# The Path Tracing Problem

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} \sum_{i=0}^{n} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|$$

# The Path Tracing Problem

- Goal: find the path from TX to RX via n interactions that satisfies Fermat's principle.

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} \sum_{i=0}^{n} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|$$

# The Path Tracing Problem

- Goal: find the path from TX to RX via n interactions that satisfies Fermat's principle.
- Fermat's principle: rays follow paths of stationary (extremal) optical length.

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} \sum_{i=0}^{n} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|$$

# The Path Tracing Problem

- Goal: find the path from TX to RX via n interactions that satisfies Fermat's principle.
- Fermat's principle: rays follow paths of stationary (extremal) optical length.
- Each interaction point lies on a surface (reflection) or edge (diffraction).

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} \sum_{i=0}^{n} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|$$

# The Path Tracing Problem

- Goal: find the path from TX to RX via n interactions that satisfies Fermat's principle.
- Fermat's principle: rays follow paths of stationary (extremal) optical length.
- Each interaction point lies on a surface (reflection) or edge (diffraction).
- Unified parametrization: x_i = A_i t_i + b_i (same tensor layout for all types).

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} \sum_{i=0}^{n} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|$$

# Min-Path-Tracing (MPT)

# Min-Path-Tracing (MPT)

**Genesis of MPT**

Created during a student job before the PhD even started — without knowing the method was novel!

# Min-Path-Tracing (MPT)

- Origin: student job in 2020 — porting MATLAB code to Python for Claude Oestges.

**Genesis of MPT**

Created during a student job before the PhD even started — without knowing the method was novel!

# Min-Path-Tracing (MPT)

- Origin: student job in 2020 — porting MATLAB code to Python for Claude Oestges.
- Key idea: optimize path coordinates via gradient descent on path length.

**Genesis of MPT**

Created during a student job before the PhD even started — without knowing the method was novel!

# Min-Path-Tracing (MPT)

- Origin: student job in 2020 — porting MATLAB code to Python for Claude Oestges.
- Key idea: optimize path coordinates via gradient descent on path length.
- First formalized and presented at EuCAP 2023 (Florence).

**Genesis of MPT**

Created during a student job before the PhD even started — without knowing the method was novel!

# Min-Path-Tracing (MPT)

- Origin: student job in 2020 — porting MATLAB code to Python for Claude Oestges.
- Key idea: optimize path coordinates via gradient descent on path length.
- First formalized and presented at EuCAP 2023 (Florence).
- Supports mixed reflection/diffraction sequences.

**Genesis of MPT**

Created during a student job before the PhD even started — without knowing the method was novel!

# Min-Path-Tracing (MPT)

- Origin: student job in 2020 — porting MATLAB code to Python for Claude Oestges.
- Key idea: optimize path coordinates via gradient descent on path length.
- First formalized and presented at EuCAP 2023 (Florence).
- Supports mixed reflection/diffraction sequences.
- Foundation for all subsequent contributions.

**Genesis of MPT**

Created during a student job before the PhD even started — without knowing the method was novel!

# Smoothing for Differentiable RT

# Smoothing for Differentiable RT

[Animation: step function
→ smooth sigmoid]

# Smoothing for Differentiable RT

- Problem: visibility tests use hard if-else conditions → non-differentiable.

[Animation: step function → smooth sigmoid]

# Smoothing for Differentiable RT

- Problem: visibility tests use hard if-else conditions → non-differentiable.
- Solution: replace hard visibility with smooth approximations (e.g., sigmoid).

[Animation: step function → smooth sigmoid]

# Smoothing for Differentiable RT

- Problem: visibility tests use hard if-else conditions → non-differentiable.
- Solution: replace hard visibility with smooth approximations (e.g., sigmoid).
- Smooth union of path candidates: soft selection enables gradient flow.

[Animation: step function → smooth sigmoid]

# Smoothing for Differentiable RT

- Problem: visibility tests use hard if-else conditions → non-differentiable.
- Solution: replace hard visibility with smooth approximations (e.g., sigmoid).
- Smooth union of path candidates: soft selection enables gradient flow.
- Trade-off: smoothing parameter controls accuracy vs. differentiability.

[Animation: step function → smooth sigmoid]

# Smoothing: Key Results

# Smoothing: Key Results

# Smoothing: Key Results

- Presented at EuCAP 2024 in Glasgow.

# Smoothing: Key Results

- Presented at EuCAP 2024 in Glasgow.
- Enables end-to-end gradient computation through the full RT pipeline.

# Smoothing: Key Results

- Presented at EuCAP 2024 in Glasgow.
- Enables end-to-end gradient computation through the full RT pipeline.
- Successfully applied to antenna placement optimization and material calibration.

[Figure: optimization convergence plot]

# Smoothing: Key Results

- Presented at EuCAP 2024 in Glasgow.
- Enables end-to-end gradient computation through the full RT pipeline.
- Successfully applied to antenna placement optimization and material calibration.
- Implemented in DiffeRT2d (open-source 2D RT library).

[Figure: optimization convergence plot]

# Smoothing: Impact & Legacy

# Smoothing: Impact & Legacy

**DiffeRT2d**

Open-source 2D ray tracing library built in Python/JAX. Used for teaching and rapid prototyping.

# Smoothing: Impact & Legacy

- Most cited publication of my PhD work.

**DiffeRT2d**

Open-source 2D ray tracing library built in Python/JAX. Used for teaching and rapid prototyping.

# Smoothing: Impact & Legacy

- Most cited publication of my PhD work.
- Adopted by other research groups for differentiable propagation studies.

**DiffeRT2d**

Open-source 2D ray tracing library built in Python/JAX. Used for teaching and rapid prototyping.

# Smoothing: Impact & Legacy

- Most cited publication of my PhD work.
- Adopted by other research groups for differentiable propagation studies.
- Foundation for DiffeRT2d — a pedagogical 2D RT library in Python/JAX.

**DiffeRT2d**

Open-source 2D ray tracing library built in Python/JAX. Used for teaching and rapid prototyping.

# Smoothing: Impact & Legacy

- Most cited publication of my PhD work.
- Adopted by other research groups for differentiable propagation studies.
- Foundation for DiffeRT2d — a pedagogical 2D RT library in Python/JAX.
- Key enabler of the subsequent ML-based path tracing contribution.

**DiffeRT2d**

Open-source 2D ray tracing library built in Python/JAX. Used for teaching and rapid prototyping.

# Why Machine Learning for Path Tracing?

# Why Machine Learning for Path Tracing?

- Optimization-based methods (MPT, FPT) iterate per path candidate — can be slow.

# Why Machine Learning for Path Tracing?

- Optimization-based methods (MPT, FPT) iterate per path candidate — can be slow.
- Idea: learn to predict valid paths directly from scene geometry, skipping iterations.

# Why Machine Learning for Path Tracing?

- Optimization-based methods (MPT, FPT) iterate per path candidate — can be slow.
- Idea: learn to predict valid paths directly from scene geometry, skipping iterations.
- Generative model: given TX, RX, and scene → predict path interaction points.

# Why Machine Learning for Path Tracing?

- Optimization-based methods (MPT, FPT) iterate per path candidate — can be slow.
- Idea: learn to predict valid paths directly from scene geometry, skipping iterations.
- Generative model: given TX, RX, and scene → predict path interaction points.
- Potential for real-time inference on GPU with learned weights.

# Research Collaboration: COST INTERACT

# Research Collaboration: COST INTERACT

# Research Collaboration: COST INTERACT

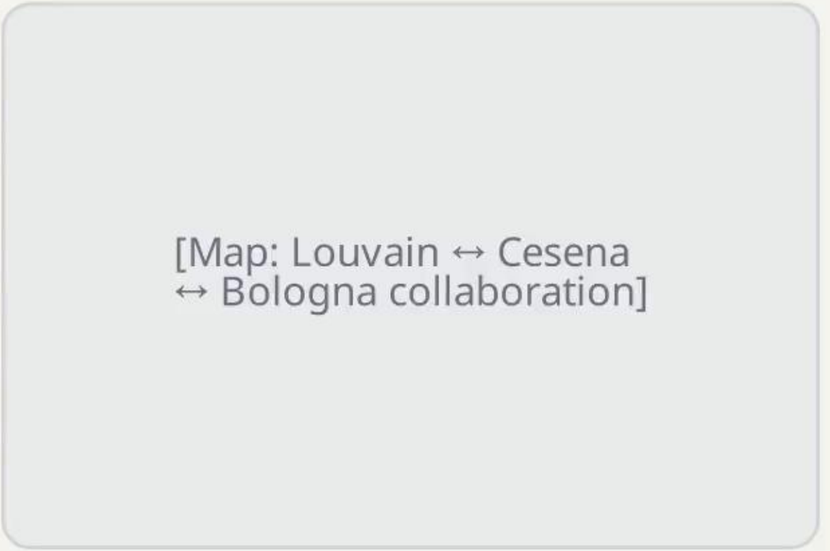
- COST Action CA20120 (INTERACT): European network for radio channel modeling.

# Research Collaboration: COST INTERACT

- COST Action CA20120 (INTERACT): European network for radio channel modeling.
- April 2024: Short-term stay in Cesena, Italy — start of ML project.

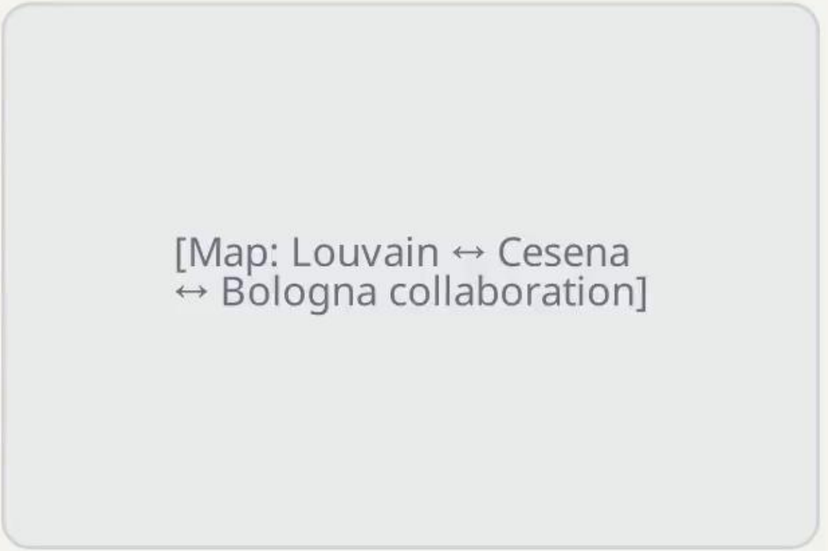
# Research Collaboration: COST INTERACT

- COST Action CA20120 (INTERACT): European network for radio channel modeling.
- April 2024: Short-term stay in Cesena, Italy — start of ML project.
- Sept–Dec 2024: Long stay in Bologna, Italy — developing the generative model.



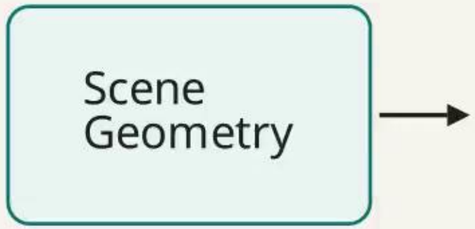
# Research Collaboration: COST INTERACT

- COST Action CA20120 (INTERACT): European network for radio channel modeling.
- April 2024: Short-term stay in Cesena, Italy — start of ML project.
- Sept–Dec 2024: Long stay in Bologna, Italy — developing the generative model.
- Collaboration with Enrico Maria Vitucci and Vittorio Degli-Esposti (University of Bologna).

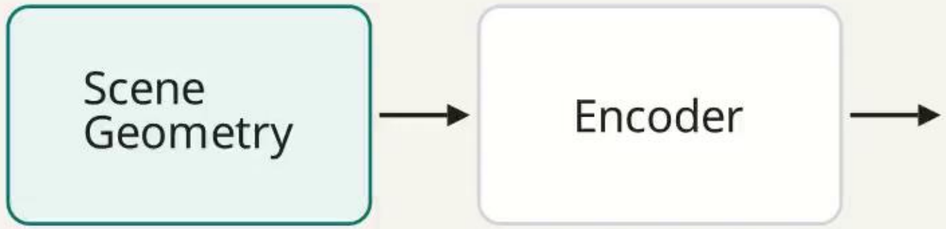


# ML Architecture Overview

# ML Architecture Overview

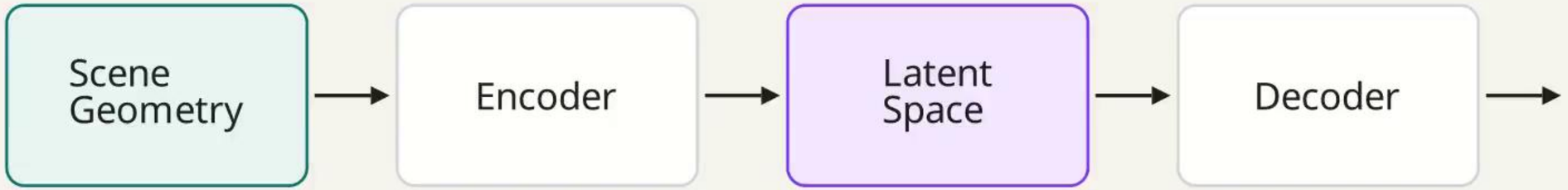


# ML Architecture Overview

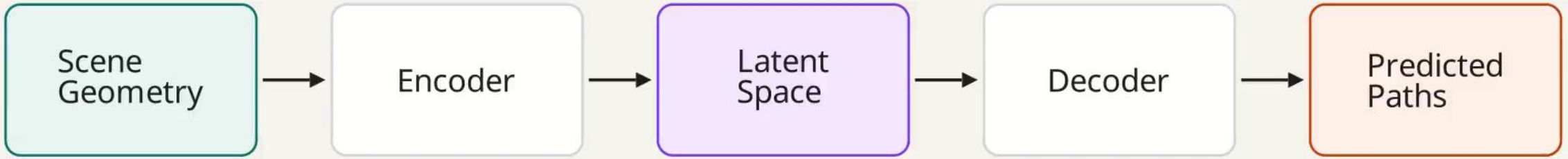


# ML Architecture Overview

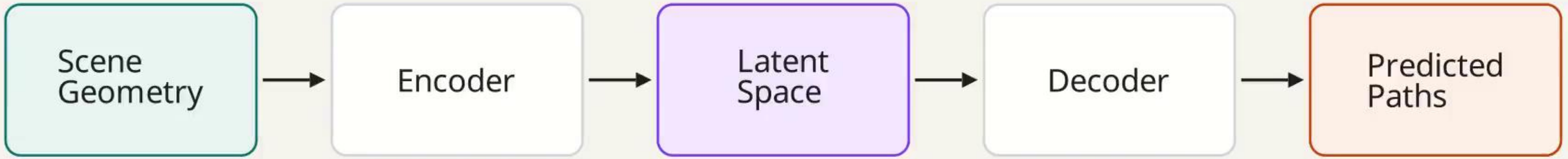
# ML Architecture Overview



# ML Architecture Overview

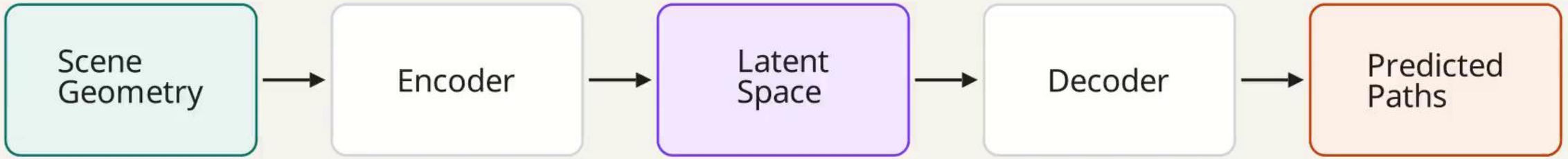


# ML Architecture Overview



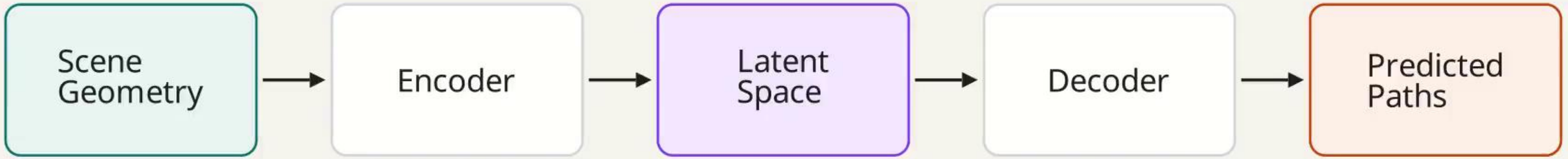

- Input: TX/RX positions + scene geometry description.

# ML Architecture Overview




- Input: TX/RX positions + scene geometry description.
- Output: predicted interaction points for each path candidate.

# ML Architecture Overview




- Input: TX/RX positions + scene geometry description.
- Output: predicted interaction points for each path candidate.
- Training data: generated from conventional RT simulations.

# Training Strategy

# Training Strategy

# Training Strategy

- Training data: large-scale RT simulations on canonical urban scenes.

[Figure: training/validation loss curves]

# Training Strategy

- Training data: large-scale RT simulations on canonical urban scenes.
- Each sample: (scene, TX, RX, interaction type sequence) → ground-truth path coordinates.

[Figure: training/validation loss curves]

# Training Strategy

- Training data: large-scale RT simulations on canonical urban scenes.
- Each sample: (scene, TX, RX, interaction type sequence) → ground-truth path coordinates.
- Loss function: mean squared error on interaction point positions.

[Figure: training/validation loss curves]

# Training Strategy

- Training data: large-scale RT simulations on canonical urban scenes.
- Each sample: (scene, TX, RX, interaction type sequence) → ground-truth path coordinates.
- Loss function: mean squared error on interaction point positions.
- Augmentation: random TX/RX placement, varying scene configurations.

[Figure: training/validation loss curves]

# ML Path Tracing: Results

# ML Path Tracing: Results

# ML Path Tracing: Results

- Significant speedup over iterative methods for large numbers of path candidates.

# ML Path Tracing: Results

- Significant speedup over iterative methods for large numbers of path candidates.
- Accuracy comparable to conventional RT in tested urban scenarios.

# ML Path Tracing: Results

- Significant speedup over iterative methods for large numbers of path candidates.
- Accuracy comparable to conventional RT in tested urban scenarios.
- Generalizes to unseen scene configurations (within the same scene class).

# ML Path Tracing: Results

- Significant speedup over iterative methods for large numbers of path candidates.
- Accuracy comparable to conventional RT in tested urban scenarios.
- Generalizes to unseen scene configurations (within the same scene class).
- Presented at ICMLCN 2025 in Barcelona.

# Journal Paper: npj Wireless Technology

# Journal Paper: npj Wireless Technology

- Extended version submitted to npj Wireless Technology (March 2026).

# Journal Paper: npj Wireless Technology

- Extended version submitted to npj Wireless Technology (March 2026).
- Expanded results with more scene types and ablation studies.

# Journal Paper: npj Wireless Technology

- Extended version submitted to npj Wireless Technology (March 2026).
- Expanded results with more scene types and ablation studies.
- Most comprehensive and recent contribution of the thesis.

# Journal Paper: npj Wireless Technology

- Extended version submitted to npj Wireless Technology (March 2026).
- Expanded results with more scene types and ablation studies.
- Most comprehensive and recent contribution of the thesis.
- Demonstrates potential of ML-assisted path tracing for next-gen networks.

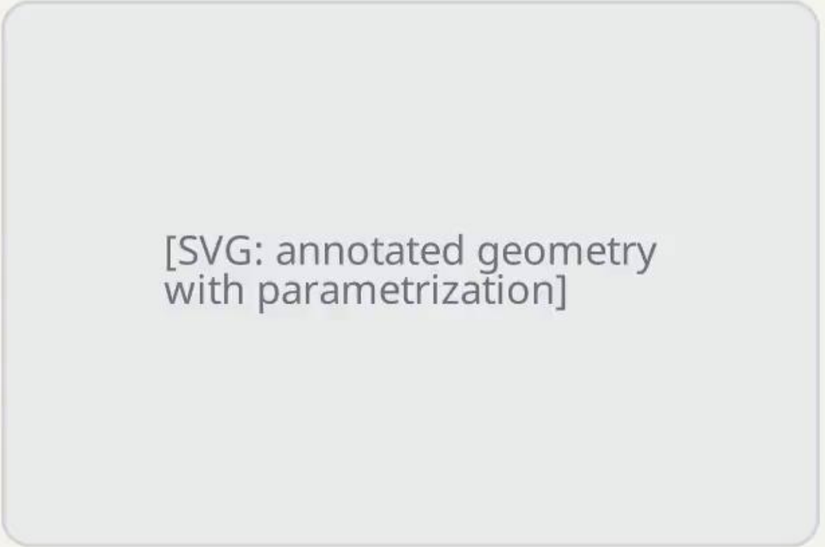
# Journal Paper: npj Wireless Technology

- Extended version submitted to npj Wireless Technology (March 2026).
- Expanded results with more scene types and ablation studies.
- Most comprehensive and recent contribution of the thesis.
- Demonstrates potential of ML-assisted path tracing for next-gen networks

Under review — the most important and comprehensive contribution of this thesis.

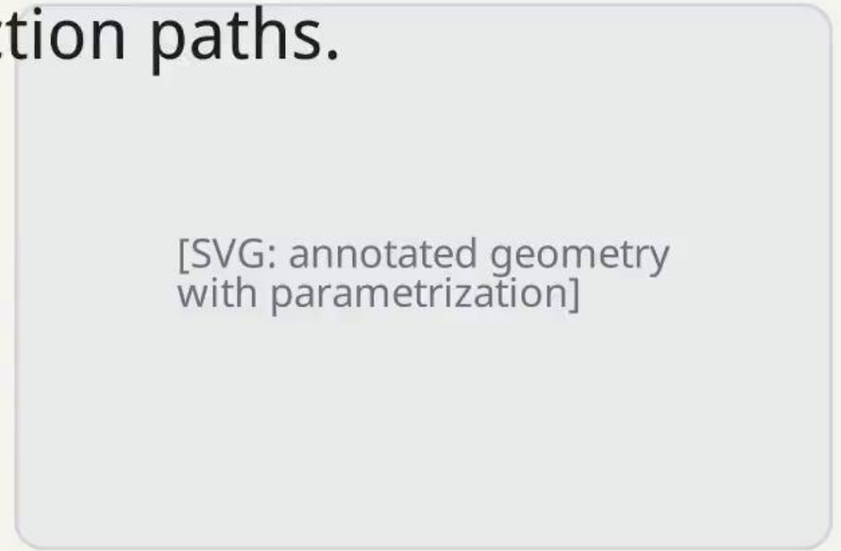
# Fermat Path Tracing: Problem Setup

# Fermat Path Tracing: Problem Setup



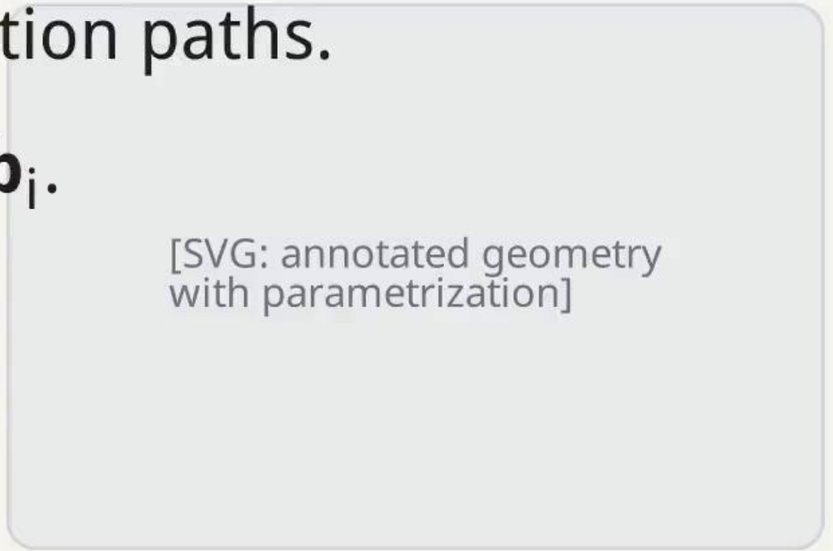
# Fermat Path Tracing: Problem Setup

- Unified formulation for reflection and diffraction paths.



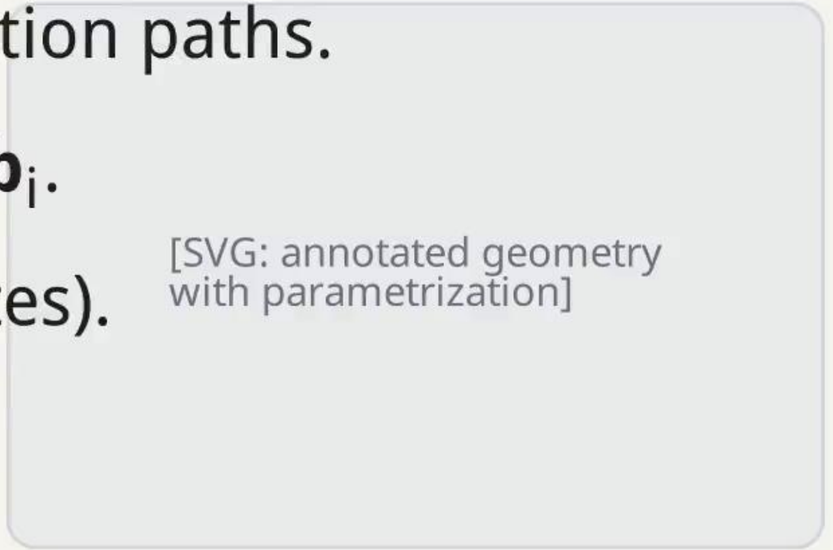
# Fermat Path Tracing: Problem Setup

- Unified formulation for reflection and diffraction paths.
- Parametrize each interaction with $\mathbf{x}_i = \mathbf{A}_i \mathbf{t}_i + \mathbf{b}_i$.



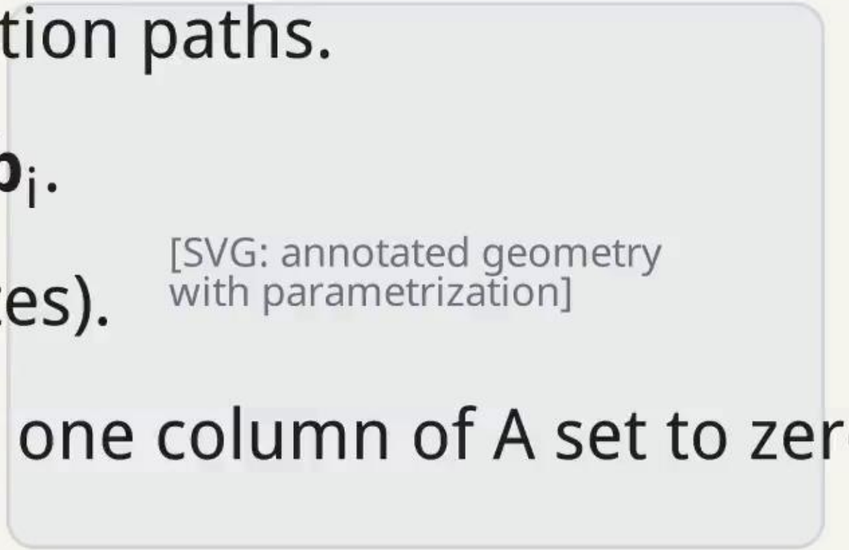
# Fermat Path Tracing: Problem Setup

- Unified formulation for reflection and diffraction paths.
- Parametrize each interaction with $\mathbf{x}_i = \mathbf{A}_i \mathbf{t}_i + \mathbf{b}_i$.
- Reflections: 2D parameter (surface coordinates).



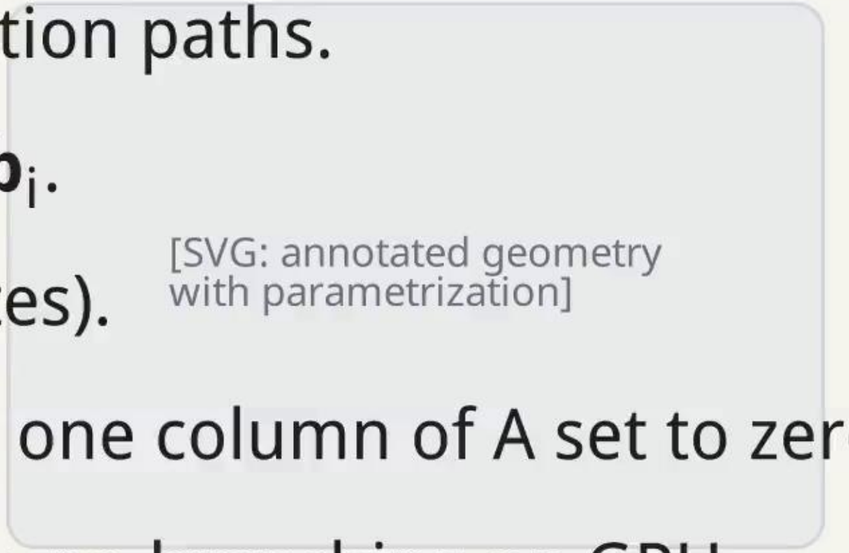
# Fermat Path Tracing: Problem Setup

- Unified formulation for reflection and diffraction paths.
- Parametrize each interaction with $\mathbf{x}_i = \mathbf{A}_i \mathbf{t}_i + \mathbf{b}_i$.
- Reflections: 2D parameter (surface coordinates).
- Diffractions: 1D parameter (edge coordinate, one column of A set to zero).



# Fermat Path Tracing: Problem Setup

- Unified formulation for reflection and diffraction paths.
- Parametrize each interaction with $\mathbf{x}_i = \mathbf{A}_i \mathbf{t}_i + \mathbf{b}_i$.
- Reflections: 2D parameter (surface coordinates).
- Diffractions: 1D parameter (edge coordinate, one column of A set to zero).
- Same tensor shape for all interaction types $\rightarrow$ no branching on GPU.



# BFGS Solver for GPU

# BFGS Solver for GPU

- Quasi-Newton method: approximates Hessian using only gradient information.

# BFGS Solver for GPU

- Quasi-Newton method: approximates Hessian using only gradient information.
- Direction: $\mathbf{p}_k = -\mathbf{B}_k \nabla L(\mathbf{T}_k)$.

# BFGS Solver for GPU

- Quasi-Newton method: approximates Hessian using only gradient information.
- Direction: $\mathbf{p}_k = -\mathbf{B}_k \nabla L(\mathbf{T}_k)$.
- Update: $\mathbf{T}_{k+1} = \mathbf{T}_k + \alpha_k \mathbf{p}_k$.

# BFGS Solver for GPU

- Quasi-Newton method: approximates Hessian using only gradient information.
- Direction: $\mathbf{p}_k = -\mathbf{B}_k \nabla L(\mathbf{T}_k)$.
- Update: $\mathbf{T}_{k+1} = \mathbf{T}_k + \alpha_k \mathbf{p}_k$.
- Fixed K iterations $\rightarrow$ uniform GPU kernel execution (no early stopping).

# BFGS Solver for GPU

- Quasi-Newton method: approximates Hessian using only gradient information.
- Direction: $\mathbf{p}_k = -\mathbf{B}_k \nabla L(\mathbf{T}_k)$.
- Update: $\mathbf{T}_{k+1} = \mathbf{T}_k + \alpha_k \mathbf{p}_k$.
- Fixed K iterations $\rightarrow$ uniform GPU kernel execution (no early stopping).
- More robust than Newton method for mixed reflection/diffraction.

# BFGS Solver for GPU

**Why BFGS over Newton/GD?**

- Newton: ill-conditioned Hessian with zero-padded diffraction.
- GD: slow convergence, no curvature information.
- BFGS: robust, no true Hessian inversion needed.

# Implicit Differentiation

# Implicit Differentiation

- Reverse-mode AD stores all intermediate states → O(K) memory.

# Implicit Differentiation

- Reverse-mode AD stores all intermediate states → O(K) memory.
- Unrolling K iterations is expensive in memory and backward time.

# Implicit Differentiation

- Reverse-mode AD stores all intermediate states → O(K) memory.
- Unrolling K iterations is expensive in memory and backward time.
- Implicit function theorem: use optimality condition at the converged solution.

# Implicit Differentiation

- Reverse-mode AD stores all intermediate states → O(K) memory.
- Unrolling K iterations is expensive in memory and backward time.
- Implicit function theorem: use optimality condition at the converged solution.
- Result: exact gradients without storing intermediate iterations.

# Implicit Differentiation

- Reverse-mode AD stores all intermediate states → O(K) mem
- Unrolling K iteratio memory and back
- Implicit function th condition at the co
- Result: exact gradi intermediate iterations.

**Implicit differentiation**

(1) Optimality condition:

$$\nabla_{\mathbf{T}} L(\mathbf{T}^*; \theta) = \mathbf{0}$$

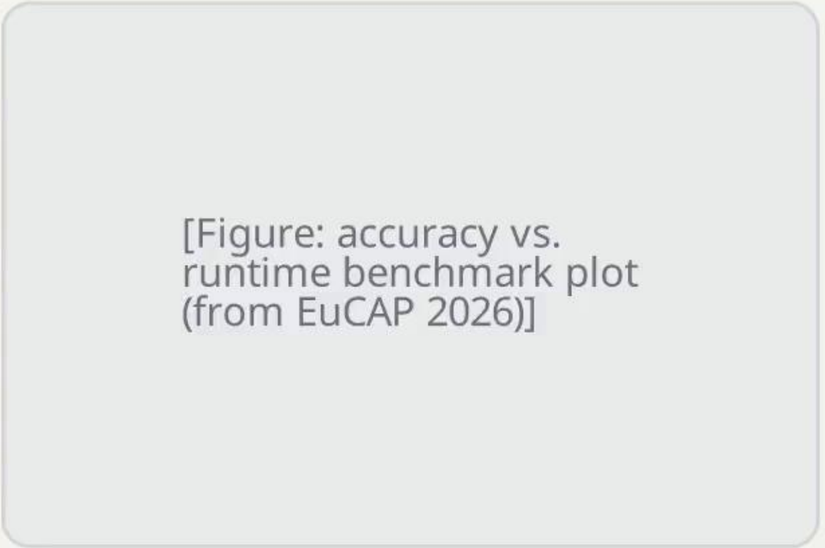
↓

(2) Implicit gradient:

$$\frac{\partial \mathbf{T}^*}{\partial \theta} = -H^{-1} \frac{\partial}{\partial \theta} \nabla_{\mathbf{T}} L$$

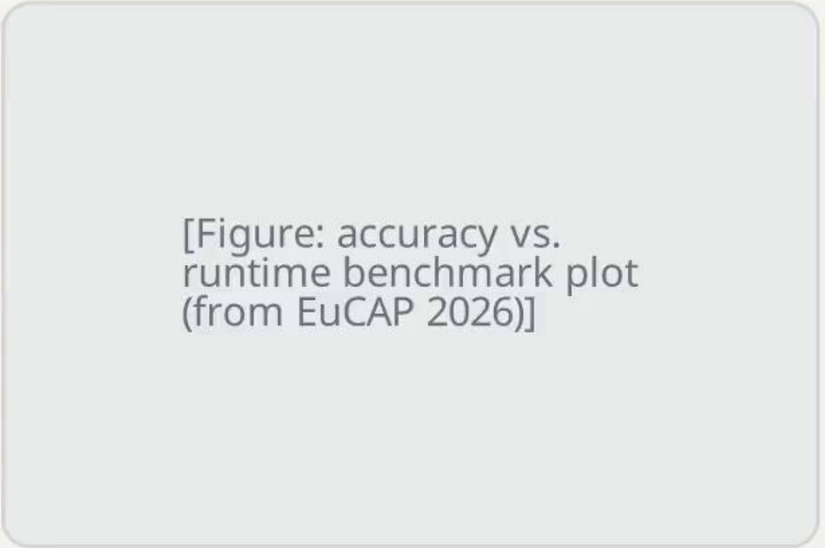
# FPT: Benchmark Results

# FPT: Benchmark Results



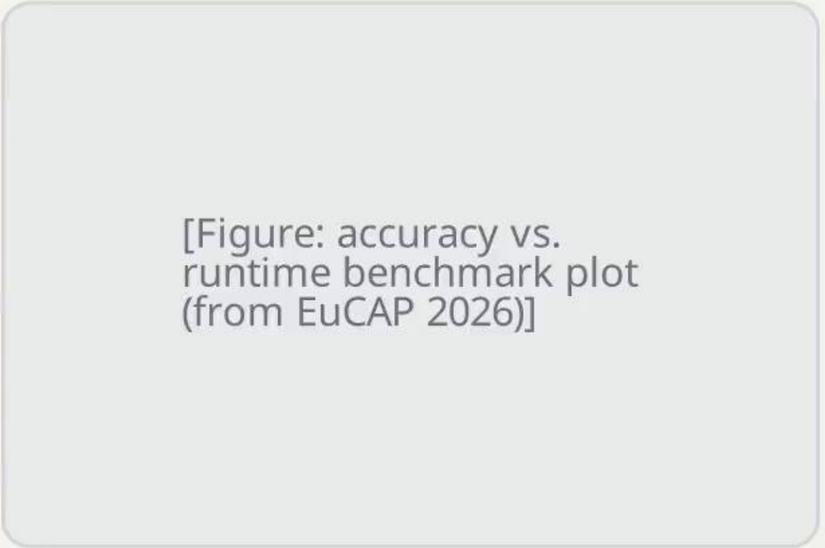
# FPT: Benchmark Results

- Benchmarked on RTX 3070 with 1000 paths in parallel.



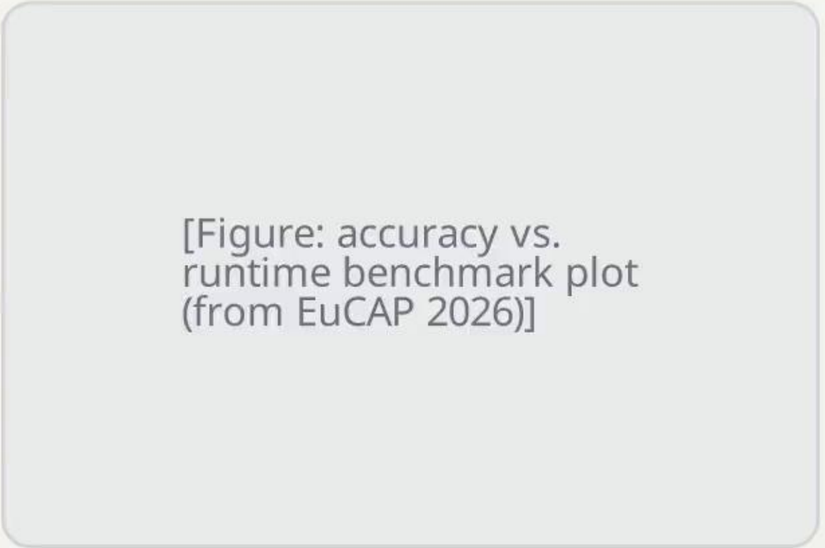
# FPT: Benchmark Results

- Benchmarked on RTX 3070 with 1000 paths in parallel.
- Interactions: n = 1..5 (reflection and diffraction).



# FPT: Benchmark Results

- Benchmarked on RTX 3070 with 1000 paths in parallel.
- Interactions: n = 1..5 (reflection and diffraction).
- Our BFGS solver approaches image-method speed while supporting diffractions.



# FPT: Benchmark Results

- Benchmarked on RTX 3070 with 1000 paths in parallel.
- Interactions: n = 1..5 (reflection and diffraction).
- Our BFGS solver approaches image-method speed while supporting diffractions.
- Accuracy improves with more line-search iterations (ours-64 variant).

[Figure: accuracy vs. runtime benchmark plot (from EuCAP 2026)]

# Open Source: DiffeRT

# Open Source: DiffeRT

**DiffeRT (3D)**

Full 3D ray tracing with JAX + equinox. GPU-accelerated, differentiable.

**DiffeRT2d (2D)**

Lightweight 2D library. Great for teaching and rapid prototyping.

# Open Source: DiffeRT

- DiffeRT: 3D differentiable ray tracing library in Python/JAX.

**DiffeRT (3D)**

Full 3D ray tracing with JAX + equinox. GPU-accelerated, differentiable.

**DiffeRT2d (2D)**

Lightweight 2D library. Great for teaching and rapid prototyping.

# Open Source: DiffeRT

- DiffeRT: 3D differentiable ray tracing library in Python/JAX.
- DiffeRT2d: lightweight 2D version for prototyping and teaching.

**DiffeRT (3D)**

Full 3D ray tracing with JAX + equinox. GPU-accelerated, differentiable.

**DiffeRT2d (2D)**

Lightweight 2D library. Great for teaching and rapid prototyping.

# Open Source: DiffeRT

- DiffeRT: 3D differentiable ray tracing library in Python/JAX.
- DiffeRT2d: lightweight 2D version for prototyping and teaching.
- Both freely available on GitHub under MIT license.

**DiffeRT (3D)**

Full 3D ray tracing with JAX + equinox. GPU-accelerated, differentiable.

**DiffeRT2d (2D)**

Lightweight 2D library. Great for teaching and rapid prototyping.

# Open Source: DiffeRT

- DiffeRT: 3D differentiable ray tracing library in Python/JAX.
- DiffeRT2d: lightweight 2D version for prototyping and teaching.
- Both freely available on GitHub under MIT license.
- Designed for reproducibility and research extensibility.

**DiffeRT (3D)**

Full 3D ray tracing with JAX + equinox. GPU-accelerated, differentiable.

**DiffeRT2d (2D)**

Lightweight 2D library. Great for teaching and rapid prototyping.

# Open Source: DiffeRT

- DiffeRT: 3D differentiable ray tracing library in Python/JAX.
- DiffeRT2d: lightweight 2D version for prototyping and teaching.
- Both freely available on GitHub under MIT license.
- Designed for reproducibility and research extensibility.
- Used by multiple research groups worldwide.

**DiffeRT (3D)**

Full 3D ray tracing with JAX + equinox. GPU-accelerated, differentiable.

**DiffeRT2d (2D)**

Lightweight 2D library. Great for teaching and rapid prototyping.

# Summary of Contributions

## ① Smoothing Technique

Continuous relaxation of hard visibility → fully differentiable RT.

## ② ML Generative Path Tracing

Learned model predicts paths directly → real-time inference potential.

## ③ Fermat Path Tracing (FPT)

Unified BFGS solver for refl. + diffr. with implicit differentiation.

# Summary of Contributions

## ① Smoothing Technique

Continuous relaxation of hard visibility → fully differentiable RT.

## ② ML Generative Path Tracing

Learned model predicts paths directly → real-time inference potential.

## ③ Fermat Path Tracing (FPT)

Unified BFGS solver for refl. + diffr. with implicit differentiation.

Cross-cutting: open-source tools (DiffeRT, DiffeRT2d) and COST INTERACT contributions.

# Most Proud Achievements

# Most Proud Achievements

- Built DiffeRT from scratch — a full 3D differentiable RT library used by the community.

# Most Proud Achievements

- Built DiffeRT from scratch — a full 3D differentiable RT library used by the community.
- International collaborations through COST INTERACT (Italy, Dublin, Lille, ...).

# Most Proud Achievements

- Built DiffeRT from scratch — a full 3D differentiable RT library used by the community.
- International collaborations through COST INTERACT (Italy, Dublin, Lille, ...).
- Created Manim Slides — an open-source tool for animated presentations (used right now!).

# Most Proud Achievements

- Built DiffeRT from scratch — a full 3D differentiable RT library used by the community.
- International collaborations through COST INTERACT (Italy, Dublin, Lille, ...).
- Created Manim Slides — an open-source tool for animated presentations (used right now!).
- Contributed a chapter to the COST INTERACT book (Lille, 2025).

# Most Proud Achievements

- Built DiffeRT from scratch — a full 3D differentiable RT library used by the community.
- International collaborations through COST INTERACT (Italy, Dublin, Lille, ...).
- Created Manim Slides — an open-source tool for animated presentations (used right now!).
- Contributed a chapter to the COST INTERACT book (Lille, 2025).
- Bridging communities: radio propagation, optimization, and machine learning.

# Future Research Directions

# Future Research Directions

- SOCP formulations for stronger convergence guarantees in FPT.

# Future Research Directions

- SOCP formulations for stronger convergence guarantees in FPT.
- Port high-precision conic solvers to practical GPU kernels.

# Future Research Directions

- SOCP formulations for stronger convergence guarantees in FPT.
- Port high-precision conic solvers to practical GPU kernels.
- Combine ML and optimization: use ML predictions as warm start for FPT.

# Future Research Directions

- SOCP formulations for stronger convergence guarantees in FPT.
- Port high-precision conic solvers to practical GPU kernels.
- Combine ML and optimization: use ML predictions as warm start for FPT.
- Extend to scattering and more complex interaction types.

# Future Research Directions

- SOCP formulations for stronger convergence guarantees in FPT.
- Port high-precision conic solvers to practical GPU kernels.
- Combine ML and optimization: use ML predictions as warm start for FPT.
- Extend to scattering and more complex interaction types.
- Integration with digital twin platforms for real-time network optimization.

# Future Research Directions

- SOCP formulations for stronger convergence guarantees in FPT.
- Port high-precision conic solvers to practical GPU kernels.
- Combine ML and optimization: use ML predictions as warm start for FPT.
- Extend to scattering and more complex interaction types.
- I
time network optimization.

Key bottleneck: open GPU solvers are still lagging behind theory.

# Thank you!

## Happy to take your questions.

github.com/jeertmans/DiffeRT

github.com/jeertmans/DiffeRT2d

jeertmans.github.io

Made with Manim Slides (open-source tool)

# Other Contributions

# Other Contributions

- Multipath Lifetime Map (MLM): visual tool for analyzing dynamic scene multipath structure (EuCAP 2025).

# Other Contributions

- Multipath Lifetime Map (MLM): visual tool for analyzing dynamic scene multipath structure (EuCAP 2025).
- DiffeRT2d: 2D differentiable RT library for pedagogical and prototyping use.

# Other Contributions

- Multipath Lifetime Map (MLM): visual tool for analyzing dynamic scene multipath structure (EuCAP 2025).
- DiffeRT2d: 2D differentiable RT library for pedagogical and prototyping use.
- COST INTERACT book chapter (Lille, 2025).

# Other Contributions

- Multipath Lifetime Map (MLM): visual tool for analyzing dynamic scene multipath structure (EuCAP 2025).
- DiffeRT2d: 2D differentiable RT library for pedagogical and prototyping use.
- COST INTERACT book chapter (Lille, 2025).
- Multiple conference presentations: EuCAP (×4), ICMLCN, SITB, COST meetings.

# Other Contributions

- Multipath Lifetime Map (MLM): visual tool for analyzing dynamic scene multipath structure (EuCAP 2025).
- DiffeRT2d: 2D differentiable RT library for pedagogical and prototyping use.
- COST INTERACT book chapter (Lille, 2025).
- Multiple conference presentations: EuCAP (×4), ICMLCN, SITB, COST meetings.
- Supervision and mentorship of master students.

# Publications

[C1] EuCAP 2023 — Min-Path-Tracing (Florence)

[C2] EuCAP 2024 — Smoothing technique (Glasgow)

[C3] EuCAP 2025 — Multipath Lifetime Map (Stockholm)

[C4] ICMLCN 2025 — ML-based path tracing (Barcelona)

[C5] EuCAP 2026 — Fermat Path Tracing (Dublin)

[J1] npj Wireless Technology 2026 — ML-assisted RT (submitted)

[B1] COST INTERACT book chapter (2025)