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

Towards Generative Ray Path Sampling for Faster Point-to-Point Ray Tracing

Jérome Eertmans[1], Nicola Di Cicco[2], Claude Oestges[1], Laurent Jacques[1], Enrico M. Vitucci[3], Vittorio Degli-Esposti[3]

[1] ICTEAM (Université catholique de Louvain), [2] DEIB (Politecnico di Milano), [3] DEI (University of Bologna) — jerome.eertmans@uclouvain.be

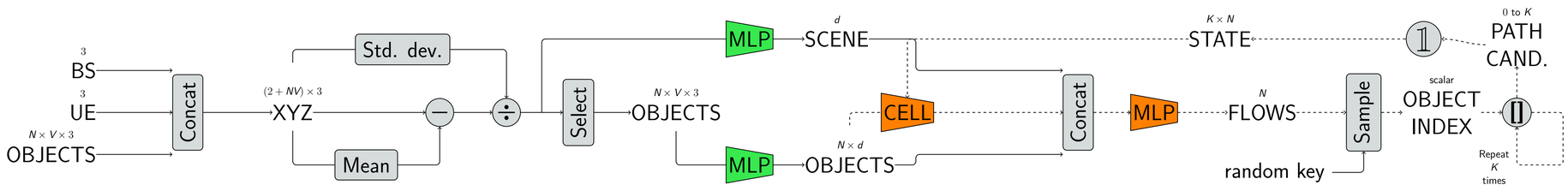

Figure 1: Proposed ML model for sampling path candidates, where each sample is a ray path candidate obtained for an input scene and a given random key.

Highlights

We present a generative Machine Learning (ML) model to avoid the exponential computation time of exhaustive Ray Tracing (RT):

- **Invariant** to $SE(3)$ and scaling transformations
- Works with **arbitrary-sized inputs** (DeepSets)
- Does **not** learn a specific scene
- **Reinforcement learning** (no ground-truth)
- Learns **path candidates** instead of EM fields

Context

Ray Tracing (RT) is a deterministic technique to model wave propagation (Fig. 2).

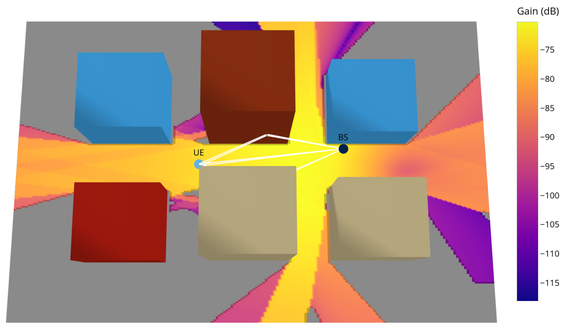

Figure 2: Ray Tracing in an urban scene, scene from [1].

Computing electromagnetic (EM) fields is usually performed in five steps (Fig. 3).

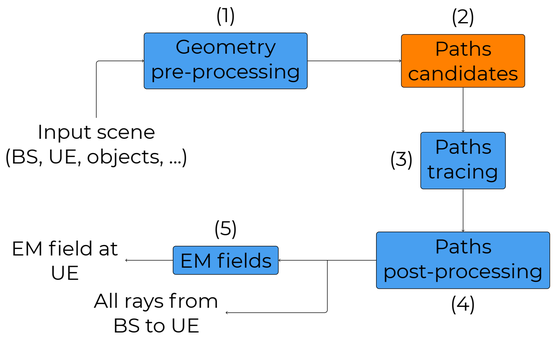

Figure 3: Typical Ray Tracing Pipeline.

In exhaustive RT,

$$\text{number of path candidates} = \mathcal{O}(N^K),$$

with N and K the number of objects and interactions.
However (Fig. 4),

num. of valid paths $\ll$ num. of path candidates.

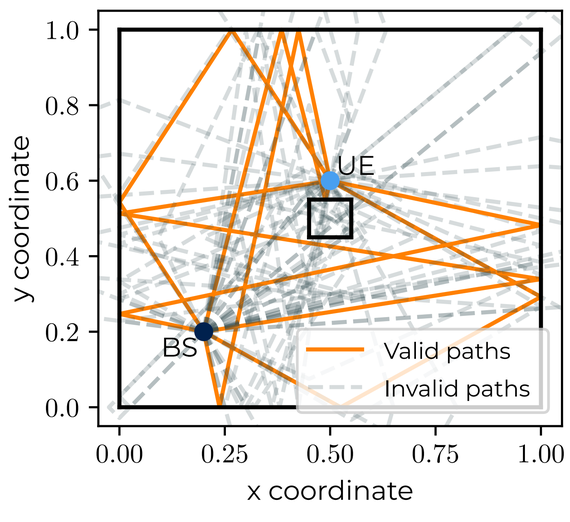

Figure 4: Illustration of the many path candidates vs. few valid paths issue.

Existing Solutions

To accelerate RT, two main approaches are used:

- **Ray Launching**: only a subset of paths is traced, but it is not guaranteed to find all valid paths
- **EM fields prediction with ML**: models are trained on measurements or simulations to predict EM fields

Issues with most ML models

- EM fields are **chaotic** as they vary very fast, and **depend** on the frequency and radio-materials
- They are **scene-specific**
- They require a **ground-truth**

This Work

We propose a new ML model (Fig. 1) to sample path candidates based on the GFLowNets framework (Fig. 5).

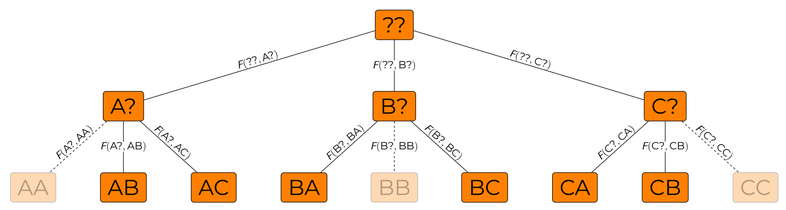

Figure 5: Path candidates generative seen as flowing through a network.

- **Path candidates**: ordered sequence of object interactions
- **1-to-1 correspondence**: each path candidate corresponds to a unique ray path

Flow-Based Path Generation

Generating a path candidate can be seen as flowing through a network, where each node (state) adds an object interaction (Fig. 5).

- The model learns a flow function F that maps state s to state s'
- Each child state s' is sampled with a probability proportional to $F(s, s')$
- **Terminal states** are completed path candidates, each receiving a reward $R(s')$

Customizable Reward

The reward function can be customized to prioritize certain paths. E.g., a boolean function $R(s') \triangleq$ is_valid_ray_path(s') can be used to reward paths that lead to valid ray paths.

By minimizing the loss function:

$$L(s') = \left(F(s, s') - R(s') - \sum_{s''} F(s', s'') \right)^2, \qquad (1)$$

the model learns to sample path candidates that lead to higher rewards (e.g., valid ray paths).

Training Procedure

Train samples are generated by sampling a modified version of a base scene:

- BS and UE are randomly placed in the scene
- A random number of objects is kept

At **each step**:

1. A batch of path candidates is generated
2. The path candidates are traced with a Ray Tracing simulator
3. The corresponding loss (1) is computed
4. A gradient-based update is performed

Results and Future Work

We trained our model on an urban scene (Fig. 2) to sample first-order ($K = 1$) and second-order ($K = 2$) reflection paths.

Table 1: Simulation parameters.

# features	MLPs' depth	batch	# steps
100/100/500	3	100	100k

We evaluated the model on two metrics:

- **Accuracy**: % of generated samples leading to valid ray paths
- **Hit rate**: % of all valid ray paths found in the generated samples

on a validation set of **100 random scenes**, sampling **10 path candidates** for each.

Table 2: Our model vs. random sampling and ongoing work.

K		Random	**This**	Ongoing
1	Acc.	3 %	**38 %**	17 %
	Hit rate	25 %	78 %	**90 %**
2	Acc.	0.03 %	8 %	**14%**
	Hit rate	0.2 %	30 %	**84%**

In the future, we will:

- Train on more **diverse** and **complex** scenes
- Compare **coverage maps** generated with and without the model
- Evaluate **actual computation gains**
- Study **non-sparse** reward functions

Open Access Tutorial & Code

The model is implemented using our open-source Differentiable RT Python library *DiffeRT* [2] and a detailed tutorial is available on our GitHub repository, see QR codes.

Tutorial

GitHub

References

[1] Jakob Hoydis, Sebastian Cammerer, Fayçal Ait Aoudia, et al. *Sionna*. Version 1.0.2. https://nvlabs.github.io/sionna/. 2022.

[2] Jérome Eertmans. *DiffeRT*. Version 0.1.0. https://github.com/jeertmans/DiffeRT. 2025.