EUROPEAN COOPERATION
IN SCIENCE
AND TECHNOLOGY

EURO-COST

SOURCE: ELEN, UCLouvain, Belgium
DEI, University of Bologna, Italy

CA20120 TD(24)08032
Helsinki, Finland
June 17-20, 2024

Learning to Sample Ray Paths for Faster Point-to-Point Ray Tracing

Jérome Eertmans, Nicola Di Cicco, Claude Oestges, Laurent Jacques, Enrico Maria Vitucci, Vittorio Degli Esposti

Jérome Eertmans
ELEN, UCLouvain, Belgium
Place du Levant 3/L5.03.02
B.239, Maxwell Building
1348, Louvain-la-Neuve
Belgium
Phone: N/A
Fax: N/A
Email: jerome.eertmans@uclouvain.be

Learning to Sample Ray Paths for Faster Point-to-Point Ray Tracing

Jérome Eertmans, Nicola Di Cicco, Claude Oestges, Laurent Jacques, Enrico Maria Vitucci, Vittorio Degli Esposti

***Abstract*—Ray Tracing has been extensively utilized in recent years to model radio propagation. Depending on the application, two main variants exist: Ray Launching and Point-to-Point Ray Tracing. In contrast to Ray Launching, Point-to-Point Ray Tracing aims to identify all possible paths between a pair of nodes. This typically entails a greater computational complexity, as the number of paths to be traversed increases exponentially in proportion to the size of the scene and the number of interactions permitted. Nevertheless, only a relatively small proportion of these paths ultimately prove to be valid. In this paper, we present a Machine Learning model that is capable of learning how to sample the valid paths from the set of all possible paths. The complexity of the Machine Learning model is linear with respect to the number of objects in the scene. Moreover, in contrast to recent proposals in the literature, our model is designed to accommodate input scenes of any size and does not depend on electromagnetic properties such as frequency.**

***Index Terms*—Ray Tracing, Machine Learning, Radio Propagation.**

I. Introduction

When performing Point-to-Point (P2P) Ray Tracing (RT), oftentimes the computational complexity is most impacted by the size of the scene or the number of interactions to be simulated. Indeed, the number of path candidates, i.e., the *potential* ray paths to be considered, grows more than linearly with respect to both parameters.

In recent years, numerous papers have investigated the potential applications of Machine Learning (ML) to the modeling of radio propagation. Motivated by the need for reducing the computation time, ML has been primarily utilized for two purposes: (1) the estimation of radio materials from measurements data or RT simulations, or scene reconstruction to later simulate radio propagation [1], [2]; and (2) the learning of wave propagation, based on measurements or simulations, in order to create a less expensive model that can make new predictions when the environment varies [3]–[10]. For the second purpose, which is closely related to our application goal, most models try to learn either a specific, usually fixed-size, input scene, or the significantly complex relationship between some input geometries and the electromagnetic (EM) fields. Since EM fields usually vary with the frequency, the radio materials, and numerous other parameters, these ML models are constrained to operate within a specific frequency range or on a particular scene. Consequently, they are unable to generalize to other scenarios. We hypothesize that ML models are not optimal for the complete replacement of channel modeling tools such as RT. This is particularly true given that the calculation of EM fields is usually straightforward once the ray paths are known. Instead, we believe that ML is best suited for the learning of the most complex parts of the RT pipeline.

In this paper, we propose to address the issue of computational complexity by using an ML model that learns how to identify only the valid ray paths that connect two nodes in a scene, typically a transmitter (TX) and a receiver (RX), and that have a given number of interactions with the environment. A ray path is considered to be valid if (1) fulfills Fermat's principle and (2) it is not obstructed by any other object in the scene. It will be demonstrated that it is possible to prioritize certain paths during the sampling process. One such example is to favor the shortest paths, as they typically contribute the most in terms of received power.

We summarize our major contributions as follows.

1) We propose a **methodology for sampling path candidates based in generative ML**, the first in the field to the best of our knowledge;
2) We develop a **radio-material- and frequency-independent** model that can be trained with minimal on-the-fly simulations;[1]
3) We implement an ML model that can support input scenes of **any size**, which enables generalization to novel scenes unseen at training time.

The remainder of this paper is organized as follows. Section II motivates the use of a sampling-based

We would like to thank to COST Action 20120 INTERACT for funding the Short-Term-Scientific-Mission that initiated this collaboration between the authors of this paper. Part of this research is supported by the FRS-FNRS (QuadSense Project, T.0160.24).

[1]RT simulations are only run for sampled path candidates, nothing more.

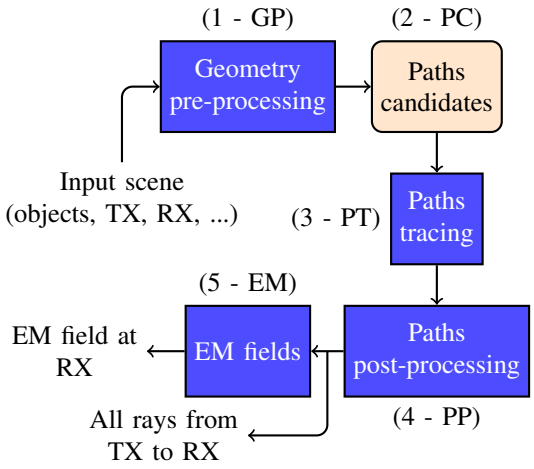

Fig. 1. The pipeline of a typical P2P RT software. Five steps are required to compute the EM fields from a given input scene: (GP) processes the geometry to obtain necessary information (e.g., normal vectors) for the next steps. Then, (PC) enumerates all possible paths candidates, and (PT) traces a geometrical ray path for each of them. Finally, (PP) removes invalid ray paths and (EM) computes the appropriate EM fields. On large scenes, (PC) is usually the performances bottleneck, as the number of path candidates it generates can grow exponentially fast. This is this step we will later replace with an ML model.

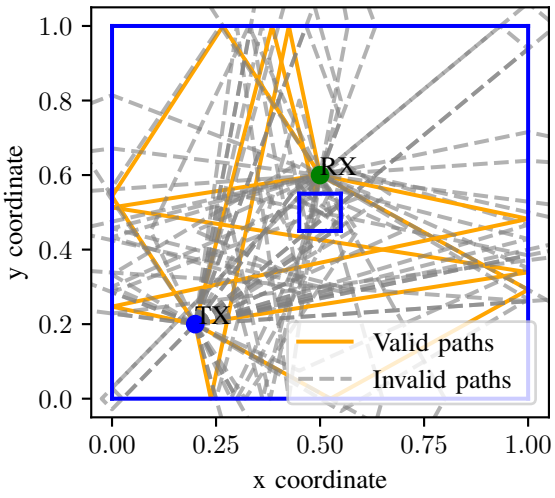

Fig. 2. Illustration of the many path candidates - few valid paths - issue. In this scenario, only 10 % of the path candidates end up creating a valid ray path (orange solid lines) between TX and RX.

approach to reduce the overall simulation time, and introduces the necessary notations. Then, section III discusses the recent applications of ML to radio propagation. Subsequently, section IV presents our model in a general framework, elucidating its functioning and demonstrating its efficiency through a two-dimensional (2D) RT example and preliminary results. Finally, V reiterates the contributions and preliminary outcomes of the model, while also outlining future avenues for research. Our model was implement using our own 2D RT toolbox, DiffeRT2d[2], and a well-documented notebook[3] is available in Open Access for further details.

II. Motivations and Notations

In most P2P RT frameworks, the process of estimating the EM fields in a given scene can be summarized as in Figure 1. It should be noted that the computational complexity of individual steps varies, with path computation being particularly demanding in the context of large scenes.

Specifically, let the size of the scene, i.e., the number of objects, be N, and the number of interactions be the variable K, then the number of possible path candidates is of the order of $\mathcal{O}\left(N^K\right)$. A path candidate is simply an ordered sequence, of length K, indicating the objects with which the ray path should interact. Then, for each of the path candidates, an actual geometric ray path is constructed, e.g., using the image method [11] or other methods, such as the minimization-based ones [12], [13]. Constructing a ray path from a path candidate usually takes a time proportional to the number of interactions, i.e., $\mathcal{O}\left(K\right)$. After that, each ray path is checked (PP) to see if it is an actual valid path or not. This check is usually bounded by $\mathcal{O}\left(NK\right)$ operations, assuming that checks are performed for each ray path segment. Finally, the number of ray paths used to compute EM fields is usually orders of magnitude smaller than the number of possible path candidates.

In city-scale 3D scenes, N can easily exceed 10^6 (e.g., the number of triangular facets), and N^K thus becomes too large either to fit in memory or to be simulated in a reasonable time. There are three principal countermeasures to this fundamental problem: (i) resort to Ray Launching (RL), that launches rays randomly from a transmitter node, e.g., TX, and collects all rays passing in the vicinity of a receiver node, e.g., RX; (ii) use a heuristic approach to reduce the number of path candidates (e.g., the *Fibonacci* method used in SionnaRT [14]); or (iii) compute some kind of visibility tree [15] – or visibility matrix [13] – to avoid generating path candidates that eventually lead to invalid ray paths. Solution (iii), though potentially time-efficient, has the major drawback of being very difficult to implement in practice, since computing the visibility in a 3D scene is itself computationally expensive. Solution (ii) is a hybrid between RL and P2P RT, and may depend on how good the heuristic is.

ML models have already shown very impressive re-

[2]GitHub repository: https://github.com/jeertmans/DiffeRT2d
[3]Jupyter Notebook: https://eertmans.be/r/cost20120-helsinki

sults in generating discrete objects such as graphs [16] and protein structures [17]. Following this line of investigation, instead of relying on a heuristic such as solution (ii), our motivation is to construct an ML model that will learn how to sample valid path candidates in the space of all possible path candidates, $\mathcal{S}$. In essence, our objective is to learn an ML model that constructs a probability-based visibility tree, as in solution (iii).

III. Related Work

Machine learning (ML) methods have already been applied to a variety of radio propagation applications, frequently in combination with ray tracing (RT), most of the time as a means to train the model by generating data. For example, ML methods have been used to predict the propagation of electromagnetic waves in wireless networks [3]–[10]. In many of these cases, the ML model will eventually replace the entire channel modeling tool, thereby eliminating the ability to obtain any intermediate, interpretable outputs, such as those provided by RT in the form of geometrical ray paths. Furthermore, we have identified the following limitation in the existing models: (i) They often necessitate the acquisition of measurements for training, (ii) they depend on a specific frequency and the radio materials from the scene, and, most importantly, (iii) they are typically trained on a fixed scene and cannot generalize to arbitrarily-sized inputs.

As previously stated, we posit that ML is ill-suited to predict the final output of some RT tools, such as coverage maps, as they are often chaotic and depend on an excessive number of parameters. Instead, we propose the creation of an ML-assisted RT pipeline that employs the ML model solely to reduce the overall computational complexity, rather than the deterministic and computationally simple procedures, such as EM coefficients, which can be calculated with certainty once a ray path is known. To the best of our knowledge, the WiNeRT model [10] is the only attempt to address the challenge of path tracing by learning the relationship between incident and outgoing rays for each interaction, e.g., reflections, as well as the EM propagation.

In comparison to existing literature, our model offers the following main novelties:

1) To the best of our knowledge, our model is the first to use ML to sample path candidates;
2) Because our model only samples path candidates, it is therefore radio-material- and frequency-independent;
3) It is capable of accommodating input scenes of any size due to the manner in which the input scene is transformed, as detailed in the subsequent section.
4) Furthermore, the model does not require any a priori ground-truth training set and can be fully trained using any ray tracer, while completely ignoring the EM fields.

The subsequent section will present the ML model architecture that incorporates all of the aforementioned properties.

IV. Proposed Approach

We illustrate the process of generating all path candidates in Figure 3. For a given number of interactions K, we can construct each possible path candidate by descending from the root node, also referred to as the initial state, and take one of the possible branches. Each branch corresponds to an object in the scene, identified here with a unique letter, and a path candidate is constructed by identifying a list of objects to visit. As a ray path cannot interact with the same path twice in a row, those branches should be **unreachable** (dotted lines). The question mark "?" is used as an indicator of a path being incomplete. We refer to this representation as the *search tree*, in that it is strictly related to the concept of *visibility tree* in RT.

Our approach leverages the principles of Generative Flow Networks (GFlowNets) [18] ML models for learning to efficiently traverse the search tree. Instead of relying on exhaustive search, we model the process of generating a path candidate, $\mathcal{P}$, as *flowing* in a Directed Acyclic Graph (DAG), starting from an initial state, and ending at a terminal state, the latter representing a complete path candidate. Each child state s' has one unique parent state s, e.g., state "AB" is a child state of "A?".

Each terminal state, i.e., a path candidate, is associated with a scalar reward R. Intuitively, greater reward values represent "better" paths. In this paper, we model the reward of some path candidate $\mathcal{P}$ as $R(\mathcal{P}) = 1$ if the path candidate corresponds to a valid ray path, or $R(\mathcal{P}) = 0$ otherwise. The central property of GFlowNets is that, after successful training, the model is expected to sample terminal states, i.e., complete path candidates, with a probability p that is proportional to their corresponding reward, R. That is,

$$p(\mathcal{P}) \propto R(\mathcal{P}). \tag{1}$$

Given our reward definition, the GFlowNet will learn to sample from the distribution of valid paths only.

To achieve this, the model must be trained to respect the following fundamental properties:

1) Each edge in the search graph must be assigned a positive *flow*, $F(s, s') > 0$, where s is the parent state and s' is the child state;
2) Flow conservation between ingoing and outgoing edges must be ensured:

$$\forall s', F(s, s') = R(s') + \sum_{s''} F(s', s''), \tag{2}$$

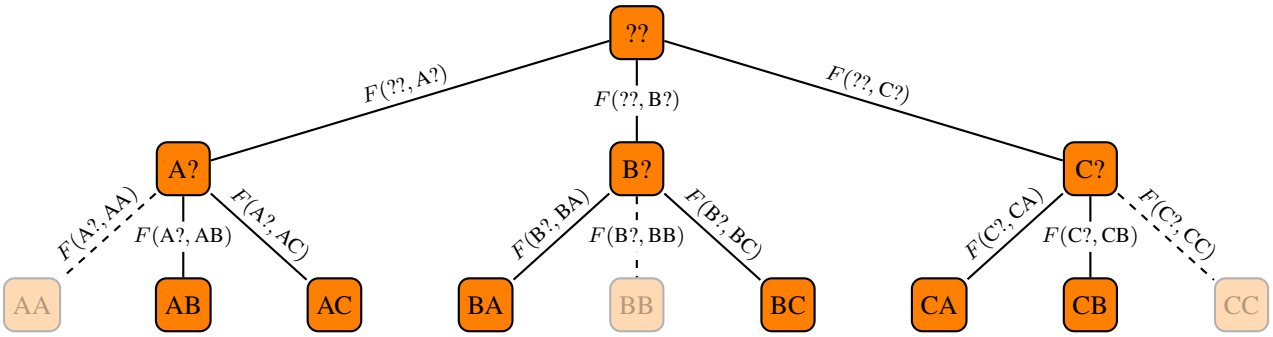

Fig. 3. Flowchart of all possible states for $K = 2$ path candidates in a scene with $N = 3$ objects: A, B, and C. Each state s can flow to three possible child states s', to select interacting objects one at a time. As interacting with the same object twice in a row is **physically unsound**, this state is marked as **unreachable** (faded node and dotted lines). To account for that in the model, the flow is stopped, i.e., set to zero, to prevent reaching those states.

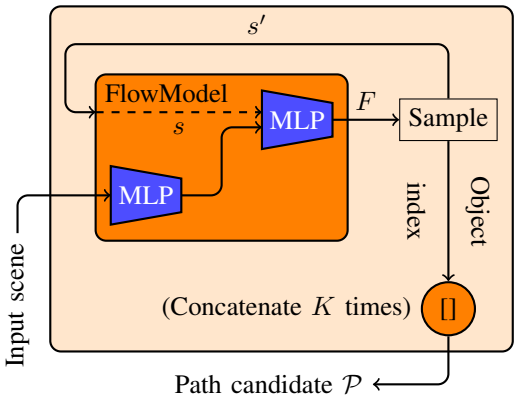

Fig. 4. Simplified representation of the presented ML model that replaces the (PC) stage in Figure 1. Only the two MLPs contain *trainable* weights: the first MLP learns how to map arbitrary many objects to a fixed-sized vector, called *scene embeddings*, and the second learns how to map each object, plus the state and some scene information, to a flow. In this representation, F is therefore the vector of all possible flows from state s.

that is, the sum of output flows, $F(s', s'')$, must be equal to the input flow, $F(s, s')$, minus the reward;

3) The probability of choosing state s' given state s must be defined as

$$p(s'|s) = \frac{F(s, s')}{\sum_{s''} F(s, s'')}, \tag{3}$$

that is, the probability of traversing an edge in the search graph is equal to its flow value normalized over all outgoing edges.

Note that all non-terminal states have a zero reward.

As shown in Figure 4, our model is made of two parts, with the second imbricated in the first. The outer part generates, or samples, the path candidates. The inner part, the *learnable* part, is the flow function F. The flow function is itself implemented as a Deep Sets [19] architecture. Deep Sets have the attractive property of accepting variable-sized collections of items as inputs (in our case, scene objects). This is in contrast with conventional ML models accepting only fixed-size vectors as inputs (e.g., Multi-Layer Perceptrons). As such, our Deep-Sets-based model allows us to support input scenes of arbitrary size. The diversity in the generated path candidates is provided by the random sampling performed between each state, where each possible output state s' is weighted according to its flow $F(s, s')$, see (3).

For training a Deep Sets model, we minimize the GFlowNets loss function, which rewrites (2) as a mean squared error:

$$L(s') = \left(F(s, s') - R(s') - \sum_{s''} F(s', s'') \right)^2. \tag{4}$$

We summarize the training procedure as follows:

1) Generate, for a given random scene, a batch of path candidates;
2) Evaluate the GFlowNet loss (Eq. (4)) for a batch of path candidates, and accumulate the results;
3) Perform a gradient-based update of the ML model, minimizing the GFlowNet loss;

and repeat those steps for a large number of scenes.

Upon successful training, as a consequence of the reward function, the model is expected to sample all valid path candidates with the same probability. As previously suggested, one can bias the model to generate shorter ray paths with a higher probability by changing the reward function accordingly.

A. Application to a 2D scenario

In this section, we present an application of our model to sampling path candidates in the context of 2D P2P RT, with specular-reflection paths only. The complete procedure was developed using our own Open Source RT toolbox, DiffeRT2d.

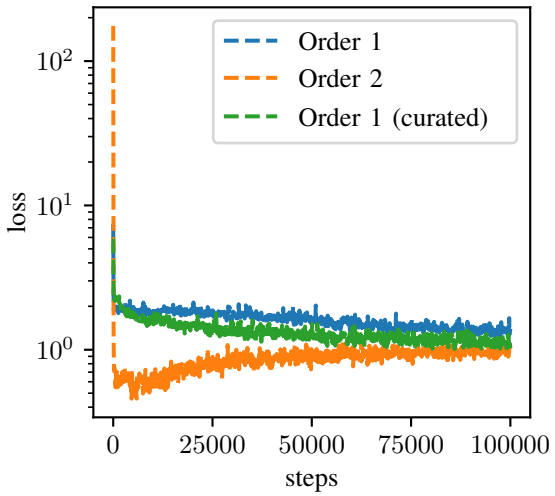

Fig. 5. Comparison of the validation losses between the three trained models. All the trainings were initiated with the same parameters. The validation dataset is shared between models $K = 1$ and $K = 2$, but the curated model has a different validation dataset, as it excludes scenes with no valid ray path.

1) Training set: We generate our training set by randomly sampling modified versions of the base scene presented in Figure 2:

- The TX and RX positions are randomized;
- A random number of walls is removed;
- Each wall is rotated around the center of the scene by a random amount, different for each wall.

From this training set, we sample 100 scenes that we will use as a validation set, to measure the evolution of the loss function throughout the learning steps.

2) Results: The present results were obtained by performing 10^5 steps with the Adam optimizer [20] (learning rate of 3×10^{-5}). The Deep Sets network has 3 linear layers with a hidden size of 500, and the number of embeddings used to represent the scene is 100. The detailed implementation can be found in the provided notebook. The experiments are run for single-bounce reflection ($K = 1$) and double-bounce reflection ($K = 2$). We also trained a model for $K = 1$ on a *curated* training set, where scenes with not valid paths are removed. For $K = 2$, the use of a curated did not yield significant changes, and thus this approach is excluded from the results.

Figure 5 compares the evolution of the validation losses throughout the training steps. Figure 6 illustrates the efficacy of the curated model in avoiding the sampling of invalid path candidates, while successfully sampling the valid ones across multiple scenes. Finally, Figure 7 highlights the fact that the model failed to learn how to sample path candidates, resulting in the generation of a greater number of invalid path candidates than valid ones.

3) Discussion: The results obtained are highly promising, particularly given the relatively small size of the model and the scope for further improvement. With regard to $K = 1$, it can be observed that learning appears to converge to a low loss, which is corroborated during path generation. As anticipated, the majority of the generated candidates result in a valid path. Furthermore, superior outcomes are observed on the curated dataset. This can be explained by the fact that cases where the scene does not allow a valid path are not interesting from a training point of view, since the only solution for the model is to bias the flows towards zero, resulting in a degenerate solution.

For the case $K = 2$, we observe very poor results in terms of generated paths, despite a decrease in loss through iterations. This may indicate that the current loss function is not an effective indicator of the model's performance, and we plan to define an accuracy function in the future. Additionally, a higher K often results in a lower percentage of valid paths, leading to a sparse reward function that can be challenging to learn from, as the majority of generated samples are not rewarded.

As previously stated, the current ML model is composed of two small-sized MLPs. While this approach enables rapid sampling and may be effective in relatively simple scenes, it may be necessary to increase the size and depth of the learnable layers to better handle large and more complex scenes.

B. Extending the model

While our model has been tested thus far in a two-dimensional scenario, it is readily adaptable to three dimensions. For instance, rather than xy-coordinates pairs, the input scene could be represented by a list of triangular facets, each described by a triplet of xyz-coordinates. Additionally, since we are not attempting to compute the path coordinates, it is possible to incorporate other types of interactions, such as scattering, by rewarding the corresponding ray path. Upon successful training, the model will be able to identify and sample path candidates that lead to ray paths containing scattering.

V. Conclusion and Future Work

We presented a novel, relatively simple ML model that learns how to sample paths. When properly trained, it could potentially remove the exponential complexity behind the usual P2P RT toolboxes. The model learns how to prioritize some branches in the complex search tree that represents the set of all possible path candidates. However, we do not yet measure the potential benefit in computational time, and that is a future work. In the

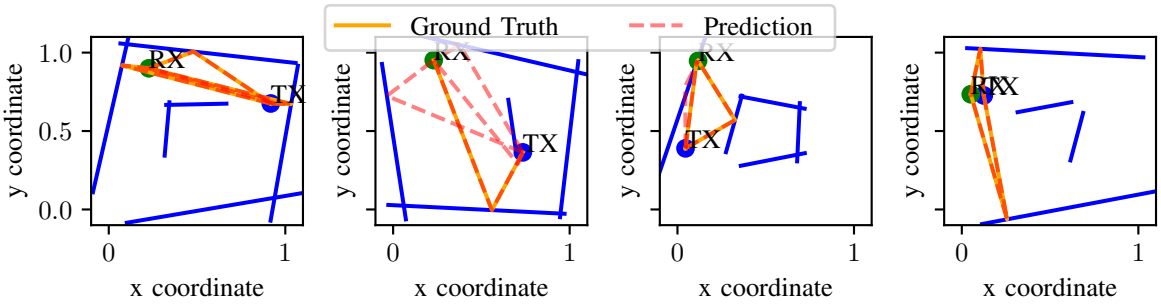

Fig. 6. The following figures illustrate the sampling of single-reflection path candidates on different random scenes following the learning process. For each scene, 10 paths are generated, and the corresponding ray path is displayed (red dashed lines). The ground truth is displayed for comparison (yellow solid line). When a red path (prediction) overlaps with a yellow path (ground truth), this indicates that the path candidate leads to a valid path, as desired. When the model generates the same path candidate several times, the corresponding ray path is displayed only once. In three of the four scenes, the model sampled all possible valid paths, and none were invalid. In contrast, the second scene contained invalid paths.

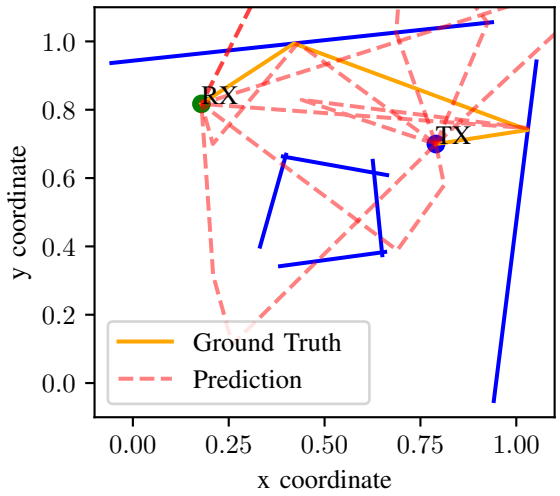

Fig. 7. As illustrated in Figure 6, the present figure illustrates how the double-reflection path candidates are sampled in a random scene. However, in contrast to the previous example, the model fails to sample any of the valid path candidates.

case of paths with a single reflection, namely $K = 1$, the results obtained demonstrate that our model is capable of learning to generate valid candidates, despite the diversity of the scenes. However, preliminary results for $K = 2$ have shown the limits of our model, which is likely linked to its simplicity. Consequently, there are several avenues for improvement, including:

- The subject of sparse reward functions, which has already been studied in the literature [21], [22]; The use of our recent smoothing framework [23] to have a continuous and differentiable reward function, instead of a discrete one;
- The general shaping of the model, including the size of the layers, the pre-training of the model on lower values of K, and so forth, are all avenues for improvement.

Ultimately, the long-term objective is to apply our ML model to the modeling of complex 3D scenes, as in the SionnaRT tool, where reducing the complexity of P2P RT could result in significant savings in computational time.

References

[1] F. Zhang, C. Zhou, C. Brennan, R. Wang, Y. Li, G. Xia, Z. Zhao, and Y. Xiao, "A radio wave propagation modeling method based on high-precision 3-d mapping in urban scenarios," *IEEE Transactions on Antennas and Propagation*, vol. 72, no. 3, pp. 2712–2722, 2024.

[2] J. Hoydis, F. A. Aoudia, S. Cammerer, F. Euchner, M. Nimier-David, S. ten Brink, and A. Keller, "Learning radio environments by differentiable ray tracing," 2023.

[3] K. Ahmad and S. Hussain, "Machine learning approaches for radio propagation modeling in urban vehicular channels," *IEEE Access*, vol. 10, pp. 113 690–113 698, 2022.

[4] S. K. Vankayala, S. Kumar, I. Roy, D. Thirumulanathan, S. Yoon, and I. S. Kanakaraj, "Radio map estimation using a generative adversarial network and related business aspects," in *2021 24th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2021, pp. 1–6.

[5] C. Ding and I. W.-H. Ho, "Digital-twin-enabled city-model-aware deep learning for dynamic channel estimation in urban vehicular environments," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1604–1612, 2022.

[6] S. Bakirtzis, K. Qiu, J. Zhang, and I. Wassell, "Deepray: Deep learning meets ray-tracing," in *2022 16th European Conference on Antennas and Propagation (EuCAP)*, 2022, pp. 1–5.

[7] M. Yin, Y. Hu, T. Azzino, S. Kang, M. Mezzavilla, and S. Rangan, "Wireless channel prediction in partially observed environments," 2022.

[8] K. Mao, Q. Zhu, M. Song, H. Li, B. Ning, G. F. Pedersen, and W. Fan, "Machine-learning-based 3-d channel modeling for u2v mmwave communications," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 592–17 607, 2022.

[9] Q. Zhu, F. Bai, M. Pang, J. Li, W. Zhong, X. Chen, and K. Mao, "Geometry-based stochastic line-of-sight probability model for a2g channels under urban scenarios," *IEEE Transactions on Antennas and Propagation*, vol. 70, no. 7, pp. 5784–5794, 2022.

[10] T. Orekondy, P. Kumar, S. Kadambi, H. Ye, J. Soriaga, and A. Behboodi, "WineRT: Towards neural ray tracing for wireless channel modelling and differentiable simulations," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=tPKKXeW33YU

[11] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015.

[12] F. Puggelli, G. Carluccio, and M. Albani, "A novel ray tracing algorithm for scenarios comprising pre-ordered multiple planar reflectors, straight wedges, and vertexes," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 8, pp. 4336–4341, 2014.

[13] J. Eertmans, C. Oestges, and L. 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.

[14] J. Hoydis, F. A. Aoudia, S. Cammerer, M. Nimier-David, N. Binder, G. Marcus, and A. Keller, "Sionna rt: Differentiable ray tracing for radio propagation modeling," 2023.

[15] F. Fuschini, E. M. Vitucci, M. Barbiroli, G. Falciasecca, and V. Degli-Esposti, "Ray tracing propagation modeling for future small-cell and indoor applications: A review of current techniques," *Radio Science*, vol. 50, no. 6, pp. 469–485, 2015. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015RS005659

[16] Y. Zhu, Y. Du, Y. Wang, Y. Xu, J. Zhang, Q. Liu, and S. Wu, "A survey on deep graph generation: Methods and applications," in *The First Learning on Graphs Conference*, 2022. [Online]. Available: https://openreview.net/forum?id=Im8G9R1boQi

[17] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[18] Y. Bengio, S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, and E. Bengio, "Gflownet foundations," 2023.

[19] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep sets," 2018.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[21] L. Pan, D. Zhang, A. Courville, L. Huang, and Y. Bengio, "Generative augmented flow networks," 2022.

[22] X. Ji, X. Zhang, W. Xi, H. Wang, O. Gadyatskaya, and Y. Li, "Meta generative flow networks with personalization for task-specific adaptation," 2023.

[23] J. Eertmans, L. Jacques, and C. 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.