The Journal of Open Source Software

DiffeRT2d: A Differentiable Ray Tracing Python Framework for Radio Propagation

Jérome Eertmans [1], **Claude Oestges** [1], **and Laurent Jacques** [1]

1 ICTEAM, UCLouvain, Belgium

DOI: 10.21105/joss.06915

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @idoby
- @roth-jakob

Submitted: 20 June 2024
Published: 30 June 2024

Figure 1: DiffeRT2d' logo.

Summary

Ray Tracing (RT) is arguably one of the most prevalent methodologies in the field of radio propagation modeling. However, access to RT software is often constrained by its closed-source nature, licensing costs, or the requirement of high-performance computing resources. While this is typically acceptable for large-scale applications, it can present significant limitations for researchers who require more flexibility in their approach, while working on more simple use cases. We present DiffeRT2d, a 2D Open Source differentiable ray tracer that addresses the aforementioned gaps. DiffeRT2d employs the power of JAX (Bradbury et al., 2024) to provide a simple, fast, and differentiable solution. Our library can be utilized to model complex objects, such as reconfigurable intelligent surfaces, or to solve optimization problems that require tracing the paths between one or more pairs of nodes. Moreover, DiffeRT2d adheres to numerous high-quality Open Source standards, including automated testing, documented code and library, and Python type-hinting.

Statement of Need

In the domain of radio propagation modeling, a significant portion of the RT tools available to researchers are either closed-source or locked behind commercial licenses. This restricts accessibility, limits customization, and impedes collaborative advances in the field. Among the limited Open Source alternatives, tools such as PyLayers (Uguen et al., 2014) and Opal (Egea-Lopez et al., 2021) fall short by not offering the capability to easily differentiate code with respect to various parameters. This limitation presents a substantial challenge for tasks involving network optimization, where the ability to efficiently compute gradients is crucial. To our knowledge, SionnaRT (Hoydis et al., 2023) is one of the few radio propagation-oriented ray tracers that incorporates a differentiable framework, leveraging TensorFlow (Abadi et al., 2015) to enable differentiation. Despite its capabilities, SionnaRT's complexity can be a barrier for researchers seeking a straightforward solution for fundamental studies in RT applied to

radio propagation. We believe that researchers need a simple-to-use and highly interpretable RT framework.

DiffeRT2d addresses these shortcomings by providing a comprehensive, Open Source, and easily accessible framework specifically designed for 2D RT. It integrates seamlessly with Python, ensuring ease of use while maintaining robust functionality. By leveraging JAX for automatic differentiation, DiffeRT2d simplifies the process of parameter tuning and optimization, making it an invaluable tool for both academic research and practical applications in wireless communications.

Moreover, in contrast to the majority of other RT tools, DiffeRT2d is capable of supporting a multitude of RT methods. These include the image method (Yun & Iskander, 2015), path minimization based on Fermat's principle (Puggelli et al., 2014), and the Min-Path-Tracing method (MPT) (Eertmans et al., 2023). Each of these methods represents a distinct compromise between speed and the type of interaction that can be simulated, such as reflection or diffraction.

DiffeRT2d democratizes access to advanced RT capabilities, thereby fostering innovation and facilitating rigorous exploration in the field.

Easy to Use Commitment

DiffeRT2d is a 2D RT toolbox that aims to provide a comprehensive solution for path tracing, while avoiding the need to compute electromagnetic (EM) fields. Consequently, we provide a rough approximation of the received power, which ignores the local phase of the wave, to allow the user to focus on higher-level concepts, such as the number of multipath components and the angle of arrival. As an object-oriented package with curated default values, constructing a basic RT scenario can be performed in a minimal number of lines of code while keeping the code extremely expressive.

Moreover, DiffeRT2d is designed to maximize its compatibility with the JAX ecosystem. It provides JAX-compatible objects, which are immutable, differentiable, and jit-in-time compilable. This enables users to leverage the full capabilities of other JAX-related libraries, such as Optax (DeepMind et al., 2020) for optimization problems or Equinox (Kidger & Garcia, 2021) for Machine Learning (ML).

Usage Examples

The documentation contains an example gallery, as well as numerous other usage examples disseminated throughout the application programming interface (API) documentation.

In the following sections, we will highlight a few of the most attractive usages of DiffeRT2d.

Exploring Metasurfaces and More

The primary rationale for employing an object-oriented paradigm is the capacity to generate custom subclasses, enabling the implementation of novel characteristics for a given object. This is exemplified by metasurfaces, which typically exhibit a deviation from the conventional law of specular reflection. Consequently, a distinct procedure must be employed for their treatment.

Using MPT, which is one of the path tracing methods implemented in DiffeRT2d, we can easily accommodate those surfaces, thanks to the object-oriented structure of the code. We also provide a very simple reflecting intelligent surface (RIS) to this end.

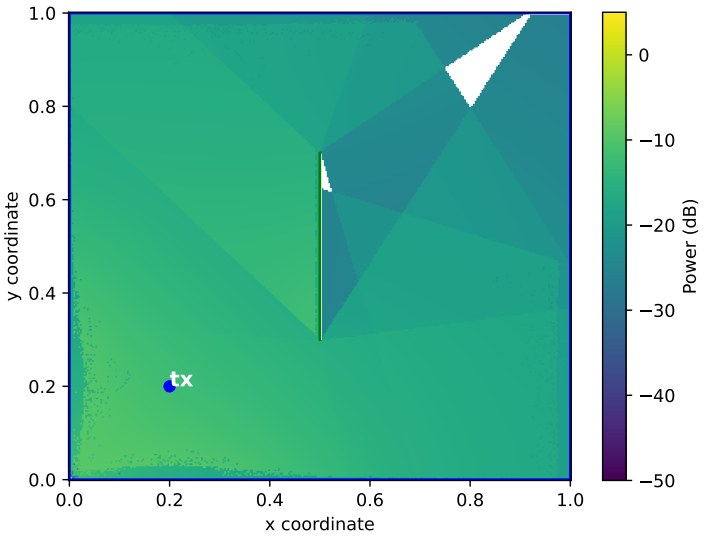

Figure 2: A coverage map for single-reflection paths (i.e., no line-of-sight) in a scene containing a RIS. The RIS, situated in the center, reflects rays at an angle of 45°, as evidenced by the fixed reflection angle of the reflected rays, irrespective of the angle of incidence. The minor noise observed around the edges is attributed to convergence issues with the MPT method, which can be mitigated by increasing the number of minimization steps.

Figure 2 can be reproduced with the following code:

```
import jax
import jax.numpy as jnp
import matplotlib.pyplot as plt

from differt2d.geometry import RIS, MinPath
from differt2d.scene import Scene
from differt2d.utils import P0, received_power

scene = Scene.square_scene()
ris = RIS(
    xys=jnp.array([[0.5, 0.3], [0.5, 0.7]]),
    phi=jnp.pi / 4,
)
scene = scene.add_objects(ris)

fig, ax = plt.subplots()

annotate_kwargs = dict(color="white", fontsize=12, fontweight="bold")

key = jax.random.PRNGKey(1234)
X, Y = scene.grid(n=300)

scene.plot(
    ax,
    transmitters_kwargs=dict(annotate_kwargs=annotate_kwargs),
    receivers=False,
)

P = scene.accumulate_on_receivers_grid_over_paths(
```

```
    X,
    Y,
    fun=received_power,
    path_cls=MinPath,
    order=1,
    reduce_all=True,
    path_cls_kwargs={"steps": 1000},
    key=key,
)

PdB = 10.0 * jnp.log10(P / P0)

im = ax.pcolormesh(
    X,
    Y,
    PdB,
    vmin=-50,
    vmax=5,
    zorder=-1,
)
cbar = fig.colorbar(im, ax=ax)
cbar.ax.set_ylabel("Power (dB)")

ax.set_xlabel("x coordinate")
ax.set_ylabel("y coordinate")
plt.show()
```

Network optimization

In previous work, we presented a smoothing technique (Eertmans, Jacques, et al., 2024) that makes RT differentiable everywhere. The aforementioned technique is available throughout DiffeRT2d via an optional `approx` (for *approximation*) parameter, or via a global config variable.

Figure 3 shows how we used the Adam optimizer (Kingma & Ba, 2017), provided by the Optax library, to successfully solve some optimization problem.

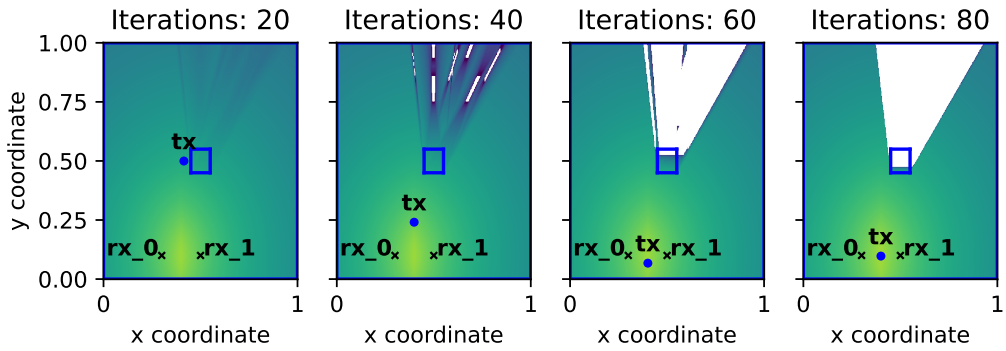

Figure 3: Different numbers of iterations converging towards the maximum of the objective function, see Eertmans, Jacques, et al. (2024) for all details.

The code to reproduce the above results can be found in the GitHub repository.

Machine Learning

In Eertmans, Oestges, et al. (2024), presented at a scientific meeting in Helsinki, June 2024, as part of the European Cooperation in Science and Technology (COST) action *INTERACT* (CA20120), we developed an ML model that learns how to sample path candidates to accelerate RT in general.

The model and its training were implemented using the DiffeRT2d library, and a detailed notebook is available online.

Stability and releases

A significant amount of effort has been invested in the documentation and testing of our code. All public functions are annotated, primarily through the use of the jaxtyping library (Kidger, 2024), which enables both static and dynamic type checking. Furthermore, we aim to maintain a code coverage metric of 100%.

Our project adheres to semantic versioning, and we document all significant changes in a changelog file.

Target Audience

The intended audience for this software is researchers engaged in the field of radio propagation who are interested in simulating relatively simple scenarios. In such cases, the ease of use, flexibility, and interpretability of the software are of greater importance than performing city-scale simulations or computing electromagnetic fields[1] with high accuracy.

Acknowledgments

We would like to acknowledge the work from all contributors of the JAX ecosystem, especially Patrick Kidger for the jaxtyping and Equinox packages.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2024). *JAX: Composable transformations of Python+NumPy programs* (Version 0.4.28). http://github.com/google/jax

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., … Viola, F. (2020). *The DeepMind JAX Ecosystem*. http://github.com/google-deepmind

Eertmans, J., Jacques, L., & Oestges, C. (2024). Fully differentiable ray tracing via discontinuity smoothing for radio network optimization. *2024 18th European Conference on Antennas and Propagation (EuCAP)*, 1–5. https://doi.org/10.23919/EuCAP60739.2024.10501570

[1] While this is currently not part of our API, we do not omit the possibility to include more complex EM routines in the future.

Eertmans, J., Oestges, C., & Jacques, L. (2023). Min-Path-Tracing: A diffraction aware alternative to image method in ray tracing. *2023 17th European Conference on Antennas and Propagation (EuCAP)*, 1–5. https://doi.org/10.23919/EuCAP57121.2023.10132934

Eertmans, J., Oestges, C., Jacques, L., & others. (2024). Learning to sample ray paths for faster point-to-point ray tracing. In *COST INTERACT 8th Meeting (Helsinki, from 2024/06/17 to 2024/06/20)*. http:// hdl.handle.net/2078/288635

Egea-Lopez, E., Molina-Garcia-Pardo, J. M., Lienard, M., & Degauque, P. (2021). Opal: An open source ray-tracing propagation simulator for electromagnetic characterization. *PLOS ONE*, *16*(11), 1–19. https://doi.org/10.1371/journal.pone.0260060

Hoydis, J., Aoudia, F. A., Cammerer, S., Nimier-David, M., Binder, N., Marcus, G., & Keller, A. (2023). Sionna RT: Differentiable ray tracing for radio propagation modeling. *2023 IEEE Globecom Workshops (GC Wkshps)*, 317–321. https://doi.org/10.1109/GCWkshps58843.2023.10465179

Kidger, P. (2024). *jaxtyping: Type annotations and runtime checking for shape and dtype of JAX arrays, and PyTrees* (Version 0.2.29). http://github.com/patrick-kidger/jaxtyping

Kidger, P., & Garcia, C. (2021). Equinox: Neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming Workshop at Neural Information Processing Systems 2021*.

Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization*. https://arxiv.org/abs/1412.6980

Puggelli, F., Carluccio, G., & Albani, M. (2014). A novel ray tracing algorithm for scenarios comprising pre-ordered multiple planar reflectors, straight wedges, and vertexes. *IEEE Transactions on Antennas and Propagation*, *62*(8), 4336–4341. https://doi.org/10.1109/TAP.2014.2323961

Uguen, B., Amiot, N., Laaraiedh, M., Mhedhbi, M., Avrillon, S., Burghelea, R., Plouhinec, E., Talom, F. T., Chaluyman, T., & Lei, Y. (2014). *Advanced radio channel simulator* (Version 0.5). http://github.com/pylayers/pylayers

Yun, Z., & Iskander, M. F. (2015). Ray tracing for radio propagation modeling: Principles and applications. *IEEE Access*, *3*, 1089–1100. https://doi.org/10.1109/ACCESS.2015.2453991