Radio Propagation as a Service: Raytracing-based channel simulation from camera data

Sasan Sharifipour[1], Tuomas Määttä[1], Niklas Vaara[1], Pekka Sangi[1], Lam Huynh[1], Janne Mustaniemi[1], Janne Heikkilä[1], Luis M. Pessoa[2,3], Filipe B. Teixeira[2,3], Miguel Bordallo López[1]
[1] Center for Machine Vision and Signal Analysis, University of Oulu, Finland
[2] Faculdade de Engenharia da Universidade do Porto, [3]INESC TEC, Portugal
Email: sasan.sharifipour@oulu.fi

***Abstract*—This paper introduces a novel service-oriented framework, Radio Propagation as a Service (RPaaS), that bridges the gap between raw sensor data and high-fidelity wireless channel simulations. RPaaS transforms noisy, sensor-derived point clouds into accurate 3D models through robust registration, segmentation, and edge detection. These models then feed into a GPU-accelerated ray tracing engine that computes multi-path propagation effects, while a separate module derives key electromagnetic and channel parameters. All components are orchestrated via a REST API in a Dockerized environment, enabling dynamic reconfiguration based on sensor data conditions. Experimental validation against commercial ray tracing tools and channel measurements demonstrates that our approach provides accurate simulations even in the presence of sensor noise.**

***Index Terms*—Radio Propagation, Ray Tracing, 3D Reconstruction, Service-Oriented Architecture.**

I. Introduction

Accurate characterization of radio channels is essential for the design and optimization of next-generation wireless systems, including emerging 5G, 6G, and beyond. Traditional propagation models often rely on simplistic assumptions or purely stochastic methods that, while computationally tractable, may fail to capture the complex interactions of signals in realistic indoor and outdoor environments. Likewise, geometric ray tracing has historically been performed on static or hand-modeled triangle meshes, which can be labor-intensive to create and insufficiently detailed to account for diffraction edges, small-scale scatterers, or partial occlusions. As wireless systems move toward higher frequencies (millimeter-wave bands) and tighter link budgets, small changes in geometry can produce significant effects on path loss, multipath components, and overall channel performance. There is a need for a more flexible and accurate approach to propagation modeling.

Meanwhile, the availability of affordable RGB-D sensors has opened the door to constructing detailed digital twins of real-world environments. These sensors provide dense point clouds of buildings, rooms, and even dynamic scenes, promising a level of geometric fidelity previously unattainable. However, sensor-derived point clouds are inherently noisy and incomplete, with artifacts arising from reflective surfaces, limited fields of view, or multi-path in the optical domain. Moreover, stitching together multiple scans from different sensor positions requires robust registration algorithms. Without careful preprocessing, labeling of surfaces, and the identification of diffraction edges—downstream ray-tracing calculations can suffer from inaccuracies that overshadow the potential benefits of using real-world data.

To address these challenges, we propose a unified, service-oriented framework called *Radio Propagation as a Service (RPaaS)*, which transforms raw sensor data into high-fidelity 3D models and subsequently computes multi-path radio propagation parameters. Our approach couples a dedicated 3D reconstruction pipeline with a specialized ray tracer, capable of handling dense, noisy point clouds directly. By integrating these components within a single Flask-based REST API, the entire workflow—from sensor data acquisition to final channel estimation—becomes accessible on demand, facilitating rapid experimentation and scalability.

Specifically, we extend a GPU-accelerated ray-launching algorithm designed to compute exact propagation paths even in the presence of noise, embedding it into a larger data processing pipeline that performs robust registration, semantic segmentation, normal computation, and triangle mesh creation, all of which help ensure that the geometry is suitable for precise reflection and diffraction modeling. The results are used in an electromagnetic (EM) computation module to derive path coefficients, channel impulse responses (CIR), and channel state information (CSI). By consolidating these capabilities into a single REST-based service, researchers can obtain accurate channel simulations from raw sensor data streams, even when faced with dynamic or cluttered environments. Our main contributions are summarized as follows:

- **A Service-Oriented Control architecture**, based on a centralized REST API that coordinates Dockerized, modular components, allowing dynamic execution balance.
- **A 3D Reconstruction Pipeline** that produces robust 3D models even under challenging noise conditions, comprising point cloud registration, semantic segmentation, normal estimation, edge identification and mesh creation.
- **An Extension of NimbusRT**, a novel ray launching algorithm that generates coarse paths from noisy dense point clouds is extended to support processed data obtained directly from the 3D Pipeline.
- **An Electromagnetic Computation Module** that computes key channel parameters such path coefficients and CSI based on the ray-traced paths on top of the sensor-based 3D models.

II. Related Work

Recent efforts in radio communications, such as the CONVERGE project [1], have focused on developing service-oriented frameworks that integrate real-time sensor data with high-fidelity channel simulations. Differentiable ray tracing methods [2], [3] offer gradient-based optimizations, yet they typically rely on preprocessed geometries rather than dynamic sensor inputs. Hybrid approaches—combining techniques such as vector parabolic equations with ray tracing [4], point-scatterer methods for time-varying channels [5], and automated geometry extraction [6]—address specific challenges. However, the efforts have centered around individual components, but still fall short of delivering a unified framework [7].

Ray tracing remains a cornerstone for simulating wireless channels by predicting path loss, multipath effects, and interference [8], [9]. Traditional methods depend on accurate CAD models, which can be inadequate in dynamic or cluttered environments due to incomplete 3D representations. Recent advances in sensor-based digital twins [10] have given rise to systems such as Sionna RT [3] and NimbusRT [11] that seek to bridge this gap, although their effectiveness is still constrained by the quality of noisy input data, which still needs accurate and pre-processed 3D models.

Advances in 3D reconstruction using RGB-D or LiDAR sensors are able to produce various accurate representations—including point clouds, polygonal meshes, and volumetric grids [12], [13]—each offering distinct trade-offs in fidelity and computational demand. Robust segmentation still remains challenging in noisy environments, although recent work employing deep learning techniques, such as SAM3D [14] and OneFormer3D [15], shows promise in overcoming these obstacles. Despite significant progress in individual processing steps, a comprehensive pipeline that consistently delivers ray tracing-ready 3D models from raw sensor data has yet to be achieved.

At the moment, while modular simulation pipelines exist, the integration of real-time sensor processing, advanced 3D reconstruction, and high-fidelity propagation simulation continues to be an open issue.

III. System Architecture

The proposed system adopts a microservice-oriented design in which a single top-level REST API service, called the *RPaaS-API*, orchestrates multiple Dockerized functional components. Each component is also accessible through its own (internal) REST-based interface, thereby isolating dependencies and allowing independent updates or replacements. From the user's perspective, however, all operations—ranging from sensor data acquisition to 3D model generation and final channel computations—are invoked through the unified RPaaS-API. Figure 1 provides an overview of the proposed system. Key micro-services (3D Reconstruction, Ray Tracing and EM Computation) are orchestrated through a single REST endpoint, hiding internal details from the user:

RPaaS API & Service-Oriented Control Layer: A centralized RESTful API, and a local repository serve as the top-level entry point for all simulation requests. Internally, it launches dockerized modules for Point Cloud reconstruction, annotation, ray tracing, and EM computations, coordinating data exchange among them. This design allows users to dynamically configure simulation parameters (such as speed/accuracy trade-offs) and seamlessly integrate additional components in the future. By exposing a uniform set of endpoints (e.g., for uploading sensor data or retrieving channel estimates), the control layer ensures consistency and scalability across the entire pipeline.

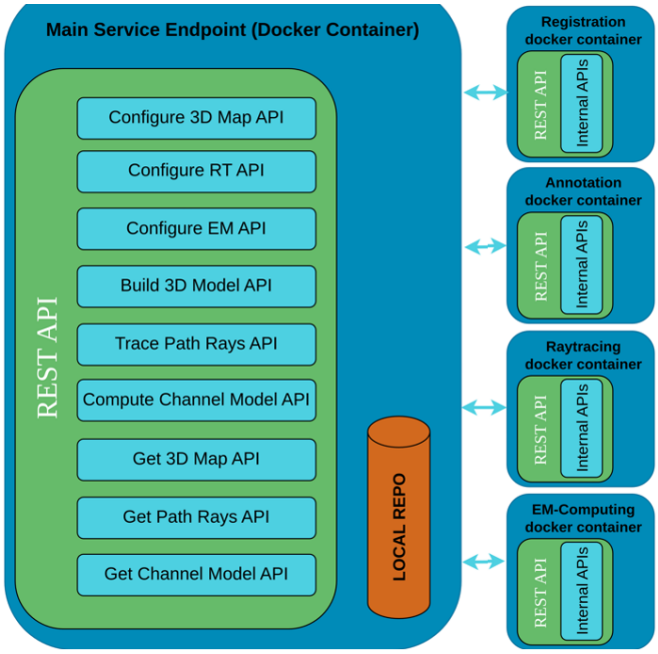

Fig. 1. System Architecture. A Docker container with an external REST API and a local repository orchestrates the control of the dockerizer implementation modules (Registration, Annotation, Raytracing, EM-Computing) through their internal APIs.

Sensor Data Acquisition and Registration: This module handles the ingestion and preliminary registration of raw sensor inputs—such as RGB-D images, LiDAR scans, or multi-view photographs. Depending on the use case, it may also include calibration data or metadata describing sensor positions and orientations. The system stores these inputs in a structured format that a 3D reconstruction module can easily consume. In dynamic scenarios, it can be repeatedly invoked to capture incremental changes in the environment.

Annotation: This modules segments the data and assigns semantic labels to different parts of the 3D model. This step ensure that surfaces, diffraction edges, and material properties are properly annotated for downstream processing in the ray tracing and electromagnetic computation modules.

Raytracing Engine: This module operates on the processed 3D data, tracing paths between transmitters and receivers. It uses an environment-driven ray launching (RL) approach to detect reflections, diffractions, scattering, and reconfigurable intelligent surface paths. By iterating with gradient-descent refinement, it corrects coarse path approximations acquired through RL. The engine exploits GPU acceleration to handle large point clouds efficiently and can adapt to user-specified

parameter (e.g. max number of reflections or diffractions).

EM & Channel Computation Module: Based on the geometric paths from the raytracing engine, this module derives electromagnetic parameters such as path gains, phases, and CIR. It interprets segmentation labels from the 3D model to apply frequency-dependent material properties, enabling realistic modeling of multi-band channels. Users can configure subcarrier spacing or other system parameters to obtain wideband or narrowband channel estimates.

External Interfacing: Finally, external tools—such as higher-layer network simulators, visualization front-ends, or machine learning frameworks—can interact with the pipeline programatically, obtaining results through standardized output formats(e.g., JSON or YAML) returned by the RPaaS API. This interface is especially useful for large-scale studies or iterative workflows, where the radio channel data feeds back into other processes (e.g., coverage optimization or robotic navigation). By supporting a clear external interface, the system can be readily integrated into other simulation environments or digital twin applications.

The modular, microservice-based structure of the RPaaS-API ensures that each component can be developed, updated, or scaled independently. For instance, new segmentation algorithms can be integrated into the *3D Pipeline* container without disrupting the NimbusRT or Sionna modules. Likewise, if an organization prefers an alternative ray tracer, they can replace NimbusRT with minimal changes to the REST interface, as long as the output format remains consistent. This design also facilitates horizontal scaling: in a production environment, one could replicate containers of the most heavily used module—such as the ray tracer—behind a load balancer, allowing multiple user requests to be handled concurrently. The RPaaS-API design brings a cohesive, service-oriented methodology to radio propagation modeling, bridging the gap between noisy sensor data and high-fidelity channel simulations.

A. 3D Pipeline Modules

The pipeline takes RGB-D images or LiDAR scans as input, fusing them into a global 3D representation using point cloud registration. When the input consists of RGB images only, DUSt3R is used for 3D reconstruction. Surface reconstruction is performed using TSDF fusion to generate a triangle mesh. We address partial overlaps, sensor noise, and drift in multi-view captures. This step builds a denser model and filters out spurious measurements.

After obtaining this consolidated model, we perform Segmentation to semantically or geometrically assign labels to distinct regions. We employ 3D instance and panoptic segmentation approaches, relying on classical plane extraction in Open3D to isolate large surfaces (e.g. floors, walls and ceilings) in less cluttered scenes, and turn to advanced neural-network method such as SAM3D [14], when dealing with more complex or crowded environments. With a multi-strategy approach, each point is accurately categorized, allowing us to refine or smooth specific segments, attach material properties, and ultimately enhance the interpretability of the 3D data.

With segments defined, we carry out Normal Estimation on each cluster or surface patch to derive accurate per-point orientations. This step mitigates small-scale noise by locally fitting planes or using neighborhood-based principal component analysis (PCA). Proper normal vectors are crucial for predicting physically correct reflection and diffraction angles, thus directly improving the realism of radio propagation paths in high-frequency scenarios.

We employ the normals to conduct Edge Identification to detect sudden changes in surface orientation, targeting corners and boundaries where diffraction plays a key role. Identifying these edges as explicit geometry features ensures they are correctly represented for the raytracing module algorithm, significantly reducing errors in modeling non-line-of-sight (NLoS) components.

The pipeline can accommodate Dynamic or Differential Updates, which allows us to incorporate new sensor data without rebuilding the entire model from scratch. By comparing two point clouds, we can isolate and process only the changed geometry, re-fusing and re-segmenting these updated regions to keep the digital twin in sync with real-world alterations. This capability lets us highlight newly detected or modified objects in the room's virtual representation. Such near-real-time synchronization accounts for the physical environment evolving or becoming more cluttered.

B. Ray Tracing and EM Computation Integration

The NimbusRT ray tracer first computes the geometric paths between TXs and RXs in an annotated 3D point cloud scene produced by our 3D reconstruction pipeline. The segmentation labels available in the annotated point cloud are utilized to remove the duplicate paths. As the underlying geometry has no explicit surface representation and is noisy, a way of distinguishing and removing near identical path trajectories is needed, for example, in the case of specular reflections.

For EM computations, our ray tracer produces the paths in a Sionna compatible format. Each interaction contains an material label acquired from the intersections, which can then be assigned with the electromagnetic properties of choice. The output is a set of path coefficients for each TX-RX pair. This enables that the EM computations can efficiently be calculated for multiple frequencies by only tracing the paths once.

IV. Service Implementation Details

The framework was developed under a Design Science Research (DSR) cycle consisting of (1) problem identification through CONVERGE use-cases, (2) iterative artefact construction, and (3) evaluation against baseline methods. Each iteration followed the build–measure–learn loop: after integrating a module, we measured path-loss error and CIR similarity, refined the module, and repeated. The final artefact (RPaaS) is evaluated against SionnaRT synthetic-mesh results and Wireless InSite simulations.

A. Data Structures and Formats

The system accepts raw sensor inputs as RGB-D images (color and depth stored in PNG format) and preprocessed

3D data as PLY point clouds or triangle meshes. During processing, point clouds are augmented with segmentation labels (object IDs) and normal vectors, and diffraction edges are flagged. Material properties are defined in a YAML file dictionary that maps segmentation labels to electromagnetic ITU materials as defined in Sionna, ensuring that both the geometric and material characteristics are embedded in the internal representation. Final outputs—including computed propagation paths, CIR, and other channel data—are serialized in YAML or JSON formats. Key processing libraries include Open3D [16] (for point cloud manipulation, visualization, and mesh creation) and the Point Cloud Library (PCL) for segmentation and normal estimation.

B. API Endpoints and Workflow

In the application flow, a user uploads raw sensor data (e.g., RGB-D images) to the RPaaS-API, which then delegates preprocessing tasks to a dedicated 3D pipeline container running libraries such as Open3D. This pipeline merges and cleans point clouds, estimates normals, segments geometry, and optionally constructs triangle meshes. Once the annotated 3D environment is available, the RPaaS-API can trigger the raytracing module (NimbusRT) in another container to compute multi-path trajectories directly on the processed data. Finally, the EM and channel modeling module, built on Sionna, translates these geometric paths into CIR, CSI, or other relevant wireless metrics. Each module has its own REST endpoints internally, but the user only interacts with a coherent set of high-level operations exposed by the RPaaS-API.

By structuring the platform in this manner, we decouple specialized functionality (e.g., 3D reconstruction & ray tracing) while maintaining a consistent interface. This approach simplifies integration and testing while also facilitates scaling: for instance, one could replicate the raytracing containers if many simultaneous propagation requests arrive. The service-oriented architecture enables seamless integration with external tools, making it suitable for large-scale network simulations.

From a user's perspective, the RPaaS-API provides configuration endpoints such as `POST /Configure_3DMap`, `/Configure_RT`, and `/Configure_EM` to set reconstruction and simulation parameters. Execution is triggered via `POST /Build_3D`, `/Trace_PathRays`, and `/Compute_Channel`, each returning a `jobID`. Results can be retrieved using `GET` requests (e.g., `/Get3DMap`, `/GetPathRays`). This modular workflow supports step-wise control and monitoring of the entire propagation pipeline.

Figure 2 illustrates the typical data flow within the RPaaS-API. Upon receiving a request to reconstruct a 3D environment, the service forwards the input files (e.g., sensor data) to the *3D Pipeline* container. Once the pipeline completes its tasks (registration, segmentation, etc.), it stores the resulting annotated point cloud or mesh in a shared volume or database. A subsequent call to trace radio paths triggers NimbusRT in another container, which reads the geometry, performs ray launching, and produces a set of multi-path components. Finally, a call to the EM computation endpoint launches Sionna,

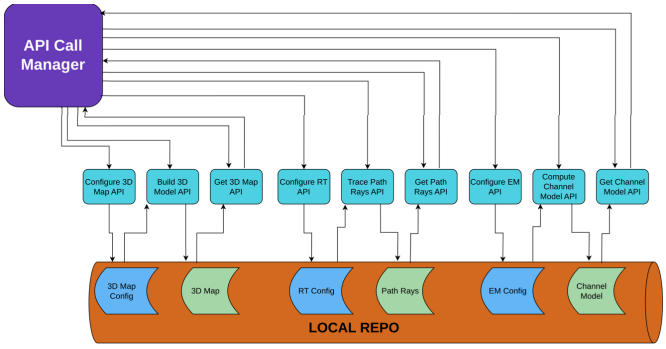

Fig. 2. High-level sequence of configuration, execution and data retrieval calls in RPaaS-API.

which processes those multi-path components to compute CIR or frequency responses. All intermediate and final results are accessible via standard HTTP `GET` requests.

V. Results and Discussion

To demonstrate the effectiveness of our pipeline, we evaluate a corridor scene captured using RGB-D sensors. The initial point cloud contains noise, missing data, and misalignment, which our reconstruction pipeline refines through fusion, segmentation, normal estimation, and diffraction edge detection. Figure 3, shows a step-by-step visualization of this process.

Using the same corridor scene, we validated our results against available real-D band channel measurements [17]. Additionally we compare the paths generated by our tool against those created using a commercial radio propagation simulation tool (Wireless Insite), but from manually created synthetic mesh of the same environment. Detailed comparisons are shown in previous work [11]. In this work, we extend the analysis by evaluating power delay profiles (PDPs) obtained from our ray tracer operating on the noisy reconstructed point cloud. We compare these results with PDPs generated using SionnaRT, which relies on a manually synthetic created mesh, as well as with the real measurements. Figure 4 shows that our system produces PDPs that closely match both the SionnaRT-based simulation and the measured data, further validating the feasibility of using reconstructed point clouds for ray tracing-based radio propagation modeling.

For a mid-size scene such as the example corridor, the integrated 3D reconstruction, segmentation, and processing pipeline typically completes in about one minute on a mid-budget nVidia RTX4070 GPU. Once the base model is built, new RGB-D images can be incorporated to the model at roughly 7 frames per second, enabling near-real-time updates. Meanwhile, the ray tracing and electromagnetic (EM) computation stages—responsible for simulating complex propagation paths—run at approximately 20 frames per second, even for reasonably complex scenes. Key performance considerations include the heavy reliance on GPU acceleration for both reconstruction and ray tracing. The system's modular design, implemented through Dockerized containers, allows each processing stage (3D reconstruction, ray tracing, and EM computation) to be scaled independently. This containerization

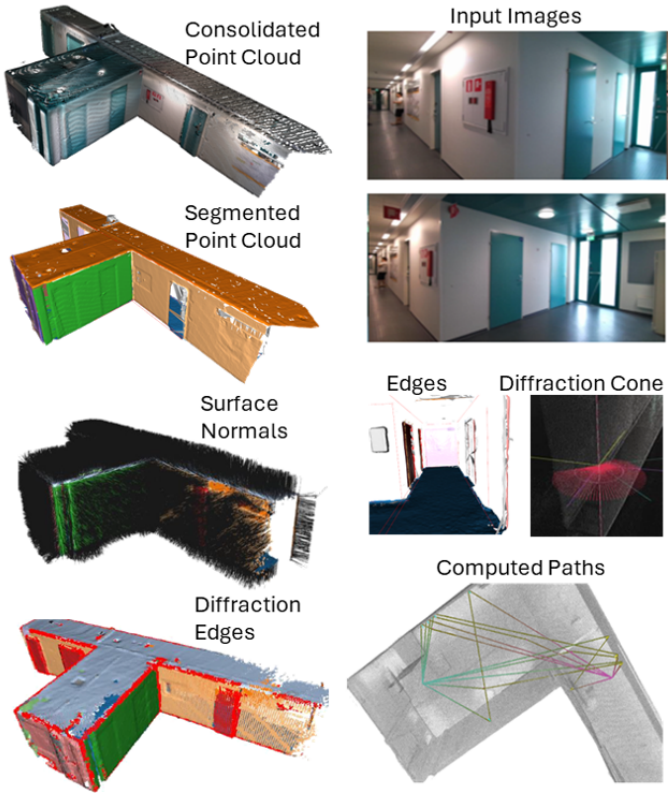

Fig. 3. Example results from the pipeline applied to a simple corridor. A Consolidated Point Cloud, where raw sensor data is fused into a global representation while mitigating noise and occlusions is followed by Segmentation to distinguish large structures from clutter. Surface Normals define point orientations used to identify Diffraction Edges. The Input Images provide scene context, while the Edges, Diffraction Cone, and Computed Paths illustrate diffraction-aware radio signal propagation from transmitter (Tx) to receiver (Rx).

facilitates parallel processing and dynamic load balancing, ensuring that the service can handle multiple concurrent calls or sessions efficiently. Our experiments show that even with mid-range hardware, the pipeline achieves robust performance, and docker orchestration further simplifies scaling across multiple nodes when higher throughput is required.

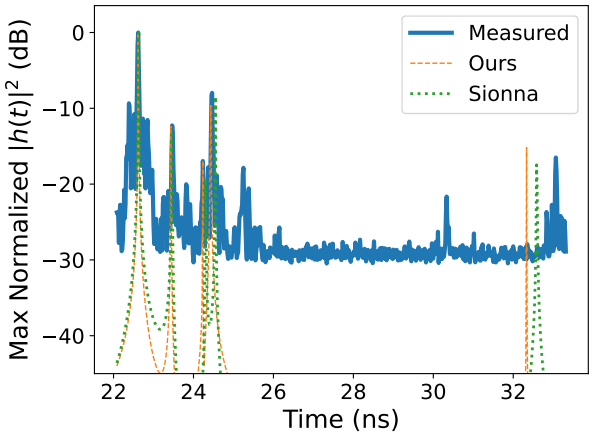

Fig. 4. Comparison of the measured and simulated power delay profiles.

VI. Conclusion

This paper proposed a unified RPaaS-API that orchestrates components based on Open3D-based pipeline, NimbusRT, and Sionna to deliver accurate channel estimation from noisy point clouds. Its novelty lies in integrating sensor-driven 3D modeling with service-based ray tracing, addressing key challenges for 5G/6G, digital twins, and high-frequency applications. Planned enhancements include more robust dynamic updates, improved performance, and broader validation, while future expansions may extend support to larger environments, advanced wave phenomena, and additional sensing modalities such as LiDAR, radar or radio-based sensing. The open, microservice-based design a novel convergent collaboration between radio propagation and computer vision research, facilitating upcoming real-world trials and the evolution of RPaaS alongside emerging 6G technologies.

Acknowledgment

The research supported by Academy Finland 6G Flagship (369116), Horizon Europe CONVERGE (101094831) and Business Finland WiSeCom (3630/31/2024) projects.

References

[1] F. B. Teixeira, M. Ricardo, ..., and L. M. Pessoa, "Converge: A vision-radio research infrastructure towards 6g and beyond," in *Europ. Conf. on Networks and Communications (EuCNC)*. IEEE, 2024, pp. 1015–1020.
[2] J. Eertmans, E. M. Vittuci, V. Degli-Esposti, L. Jacques, and C. Oestges, "Comparing differentiable and dynamic ray tracing: Introducing the multipath lifetime map," *arXiv preprint arXiv:2410.14535*, 2024.
[3] 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," in *IEEE Globecom Workshops*, 2023.
[4] X. Zhang, N. Sood, J. Siu, and C. D. Sarris, "Efficient propagation modeling in railway environments using a hybrid vector parabolic equation/ray-tracing method," in *2015 IEEE International Symposium on Antennas and Propagation*, 7 2015, pp. 1680–1681.
[5] N. Leonor, D. Ferreira, R. Caldeirinha, and T. Fernandes, "Ray tracing based model using point scatterers for time-varying radio channels," *Conftele, Castelo Branco, Portugal*, pp. 1–4, 2013.
[6] S. Tadik, R. Bhattacharjea, J. Corgan, D. Johnson, J. Van der Merwe, and G. D. Durgin, "Opengert: Open source automated geometry extraction with geometric and electromagnetic sensitivity analyses for ray-tracing propagation models," *arXiv preprint arXiv:2501.06945*, 2025.
[7] T. Nishio, Y. Koda, J. Park, M. Bennis, and K. Doppler, "When wireless communications meet computer vision in beyond 5g," *IEEE Communications Standards Magazine*, vol. 5, no. 2, p. 76–83, Jun. 2021.
[8] A. F. Molisch, *Wireless communications*. John Wiley & Sons, 2012.
[9] C. A. Balanis, *Antenna theory*. John Wiley & Sons, 2015.
[10] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji, "Digital twin modeling," *Journal of Manufacturing Systems*, vol. 64, pp. 372–389, 2022.
[11] N. Vaara, P. Sangi, M. Bordallo López, and J. Heikkilä, "Ray launching-based computation of exact paths with noisy dense point clouds," *IEEE Transactions on Antennas and Propagation*, 2025.
[12] T. Samavati and M. Soryani, "Deep learning-based 3d reconstruction: a survey," *Artificial Intelligence Review*, vol. 56, no. 9, p. 9175, 2023.
[13] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," 2019. [Online]. Available: https://arxiv.org/abs/1812.03828
[14] Y. Yang, X. Wu, T. He, H. Zhao, and X. Liu, "Sam3d: Segment anything in 3d scenes," *arXiv preprint arXiv:2306.03908*, 2023.
[15] M. Kolodiazhnyi, A. Vorontsova, A. Konushin, and D. Rukhovich, "Oneformer3d: One transformer for unified point cloud segmentation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2024.
[16] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.
[17] J. Kokkoniemi, V. Hovinen, K. Nevala, and M. Juntti, "Initial results on d band channel measurements in los and nlos office corridor environment," in *European Conf. on Antennas and Propagation (EuCAP)*. IEEE, 2022.