German International University **(GIU)** Berlin

Faculty of Information Engineering and Technology

Mobile Communications Project

Ray-Tracing Based Analysis of Urban Wireless Channels in Pankow Using Sionna RT

Abdallah Mohamed – 58-21166
Amr Mohsen – 13001690
Zeina Mohamed – 19002220

Supervisors:
Prof. Mohamed Ashour
Eng. Aly Ahmed

Contents

1 Abstract

This project investigates the propagation characteristics of a realistic urban wireless channel using the Sionna RT ray-tracing engine and a detailed 3D model of the Pankow area in Berlin. The scenario consists of a single transmitter and an array of receivers placed along a street canyon, and the simulation is performed at 3.5 GHz and 28 GHz. The main objectives are to compute path loss as a function of transmitter–receiver distance, extract the power delay profile (PDP) and RMS delay spread, and visualize the dominant propagation paths in the 3D environment using static renders and an interactive notebook-based viewer.

2 Introduction

Future cellular systems increasingly rely on accurate channel models for both sub-6 GHz and millimeter-wave bands. Geometry-based ray tracing provides a physically motivated way to study reflection, diffraction, and scattering in complex urban environments without building a full hardware measurement campaign. In this project, the Pankow scene provided with Sionna RT is used to emulate a realistic street canyon with surrounding buildings, and the channel between a rooftop transmitter and multiple street-level receivers is analyzed.

The study compares propagation at 3.5 GHz and 28 GHz, focusing on large-scale path loss behavior and time dispersion. The simulation pipeline is implemented in Python using Sionna RT and wrapped in a Jupyter notebook that supports interactive 3D visualization of ray paths and device locations. A standalone Python script with the full code is also provided.

3 Scenario and Methodology

3.1 Scene Description

The 3D environment is the Pankow urban model distributed with Sionna RT, which includes building blocks, inner courtyards, and a central street segment. The geometry is imported from the XML description into Sionna, and the global scene bounds and building corridor in the x-direction are computed programmatically using the underlying Mitsuba meshes. This ensures that all transmitter and receiver positions stay inside the visible plane and in front of the building facades.

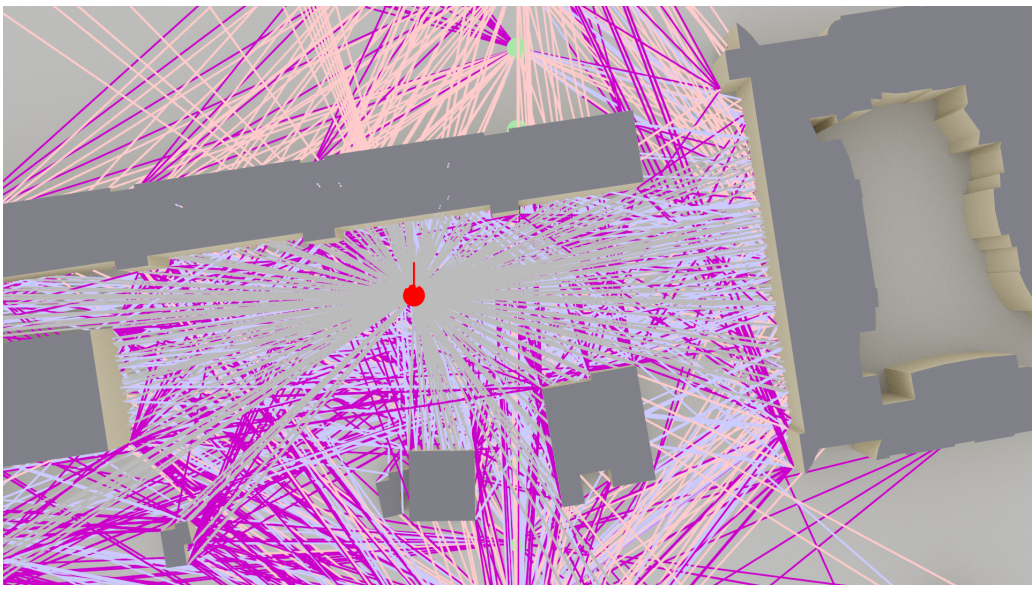

Figure 1: Top-down Sionna RT render of the Pankow scene showing the transmitter (red) and receivers (green) with multi-bounce propagation paths at 3.5 GHz.

3.2 Transmitter and Receiver Configuration

A single transmitter is placed near the center of the building corridor at a height of approximately 10 m, representing a rooftop or lamp-post base station. Fifteen receivers are distributed along a straight line on the street at a height of 1.5 m, mimicking user equipment locations. The line extends from the minimum to the maximum building x-coordinate, with a small margin to avoid receivers falling outside the modeled facades.

A single-element isotropic planar array is used at both the transmitter and receivers, with vertical polarization. This simplifies the antenna pattern while still allowing Sionna RT to compute per-path complex amplitudes and delays.

3.3 Frequencies and Propagation Settings

Two carrier frequencies are simulated: 3.5 GHz representing sub-6 GHz 5G operation, and 28 GHz representing millimeter-wave deployment. For each frequency, the Sionna RT `PathSolver` is configured to include line-of-sight (LOS), specular reflections, refraction, and diffraction up to a limited path depth. The solver returns a set of paths between each transmitter–receiver pair, including complex amplitude and excess delay for every valid ray.

4 Implementation Details

4.1 Python Script and Notebook Structure

The full simulation is implemented in a standalone Python script `Sionna_Code.py`, which:

- Imports and configures Sionna RT, Mitsuba, TensorFlow, and plotting libraries.
- Loads the Pankow XML scene and computes scene and building bounds from Mitsuba meshes.
- Places the transmitter and receiver positions and attaches planar arrays.
- Runs the `PathSolver` for each carrier frequency.
- Computes path loss per receiver from the channel impulse response (CIR).
- Extracts PDP and RMS delay spread for a selected receiver.
- Generates ray-path renders and 2D plots of the results.

For reproducibility and interactive exploration, the same workflow is mirrored in a Jupyter notebook `Interactive_Simulation.ipynb`. The notebook organizes the code into cells for imports, configuration, helper functions, scene loading, device placement, ray tracing, and visualization. One section is dedicated to interactive 3D preview, allowing the user to rotate and zoom the scene while displaying the simulated ray paths.

4.2 Key Code Snippets

Listing 1 shows the main code fragment used to place the transmitter and receivers inside the building corridor and attach the antenna arrays.

Listing 1: Transmitter and receiver placement in the Pankow scene.

```
# Tx at center of building corridor
tx_x_center = 0.5 * (X_BUILD_MIN + X_BUILD_MAX)
tx_pos = clamp_to_bounds([tx_x_center, 0.0, tx_height], mins,
   maxs)
tx_pos = np.array(tx_pos, dtype=float)
tx = Transmitter(name="Tx", position=tx_pos.tolist())
scene.add(tx)

# Rx along the street, limited to building X-range
start_x = X_BUILD_MIN + margin_x
stop_x = X_BUILD_MAX - margin_x

rx_positions = np.linspace(
    start=[start_x, y_rx, z_rx],
    stop=[stop_x, y_rx, z_rx],
    num=num_rx,
)

receivers = []
for i, pos in enumerate(rx_positions):
```

```
    pos_clamped = clamp_to_bounds(pos, mins, maxs)
    rx = Receiver(name=f"Rx_{i}", position=pos_clamped)
    scene.add(rx)
    receivers.append(rx)

scene.tx_array = PlanarArray(
    num_rows=1,
    num_cols=1,
    vertical_spacing=0.5,
    horizontal_spacing=0.5,
    pattern="iso",
    polarization="V",
)
scene.rx_array = scene.tx_array
```

Listing 2 illustrates how the path loss per receiver and PDP are computed from the Sionna RT CIR output.

Listing 2: Path loss and PDP computation from Sionna RT CIR.

```
def path_loss_per_rx(paths):
    a, tau = paths.cir(normalize_delays=False, out_type="numpy")
    power_rx = np.sum(np.abs(a) ** 2, axis=(1, 2, 3, 4, 5))
    pl_db = -10.0 * np.log10(power_rx + 1e-15)
    return pl_db

def pdp_for_rx(paths, rx_index=0):
    a, tau = paths.cir(normalize_delays=False, out_type="numpy")
    a_rx = a[rx_index, 0, 0, 0, :, 0]
    tau_rx = tau[rx_index, 0, :]

    p = np.abs(a_rx) ** 2
    mask = (tau_rx >= 0.0) & (p > 0.0)
    tau_rx = tau_rx[mask]
    p = p[mask]

    idx = np.argsort(tau_rx)
    return tau_rx[idx], p[idx]
```

5 Simulation Results

5.1 Path Loss Versus Distance

Figure 2 shows the path loss between the transmitter and each receiver as a function of distance for 3.5 GHz. The curve exhibits local fluctuations due to constructive and destructive interference between multiple reflected and diffracted components, on top of the general increasing trend with distance.

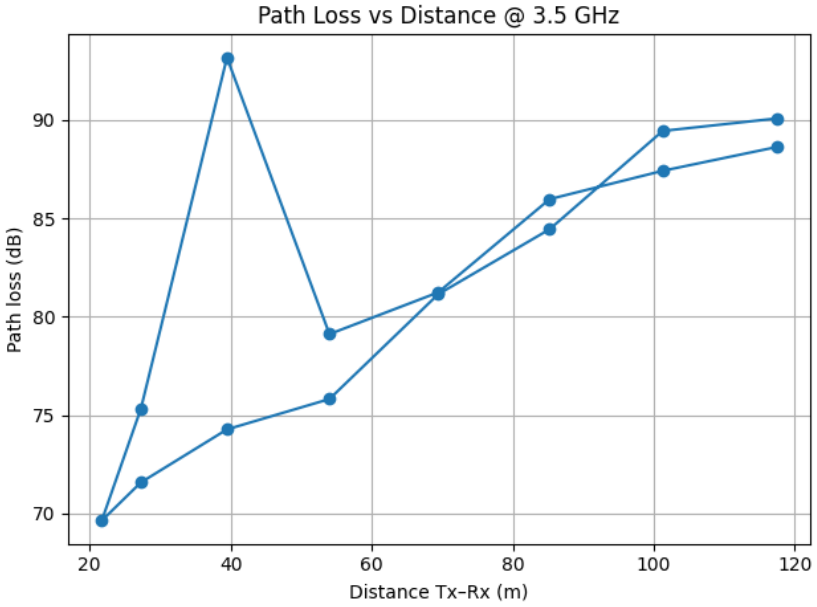

Figure 2: Path loss vs. transmitter–receiver distance at 3.5 GHz.

For 28 GHz, Figure 3 indicates higher absolute path loss and stronger sensitivity to small changes in distance, reflecting increased blockage and higher attenuation at millimeter-wave frequencies.

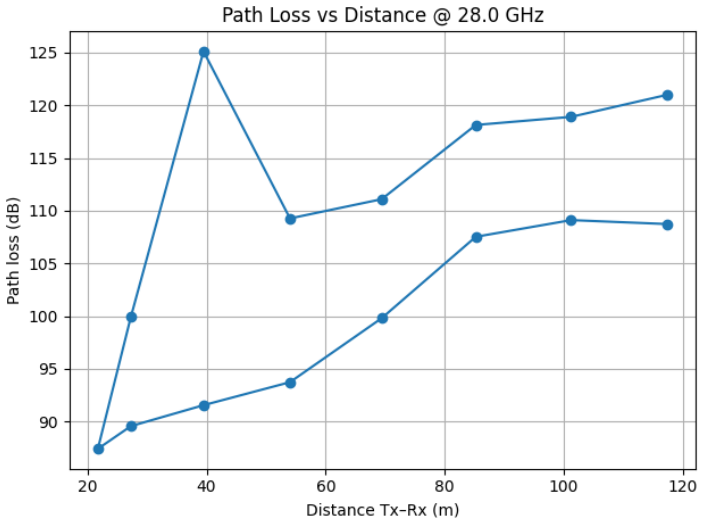

Figure 3: Path loss vs. transmitter–receiver distance at 28 GHz.

5.2 Ray-Path Visualization

The rendered images in Figures 1 and 4 highlight the rich set of multipath components in the Pankow canyon. At 3.5 GHz, more diffracted and longer-range reflected components are visible, whereas at 28 GHz the propagation is dominated by a smaller subset of strong specular paths and the rays are more easily blocked by building edges.

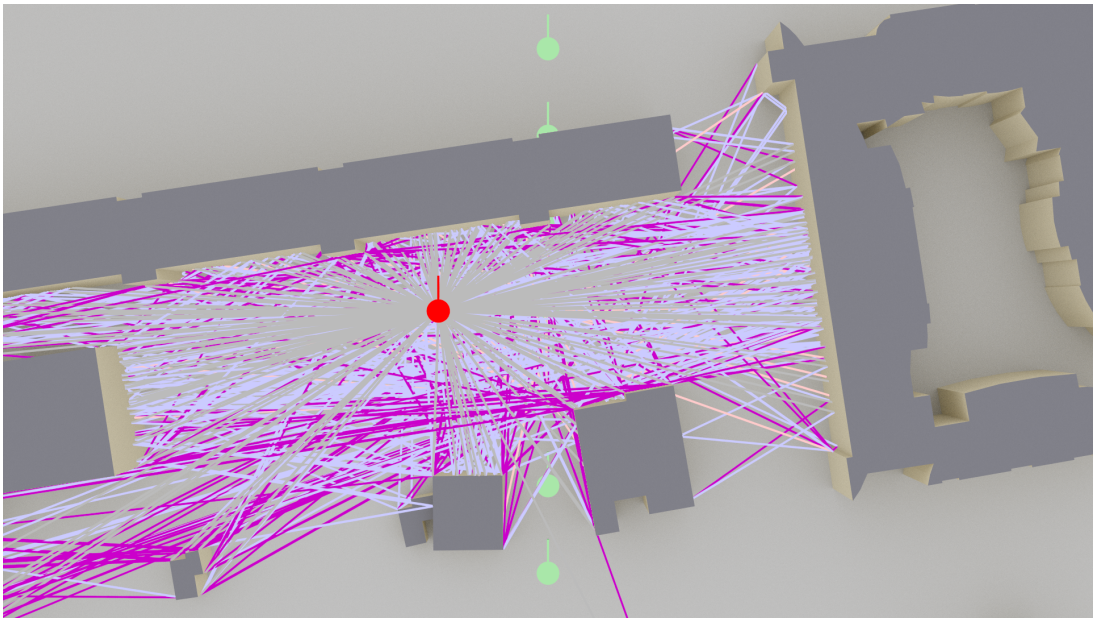

Figure 4: Sionna RT ray tracing for 28 GHz in the Pankow scene.

5.3 Power Delay Profile and RMS Delay Spread

The PDP for a representative receiver at 3.5 GHz is shown in Figure 5. The earliest paths correspond to LOS and strong reflections, while later, weaker components originate from longer reflection and diffraction paths. The RMS delay spread is computed from the PDP and reported in the figure subtitle.

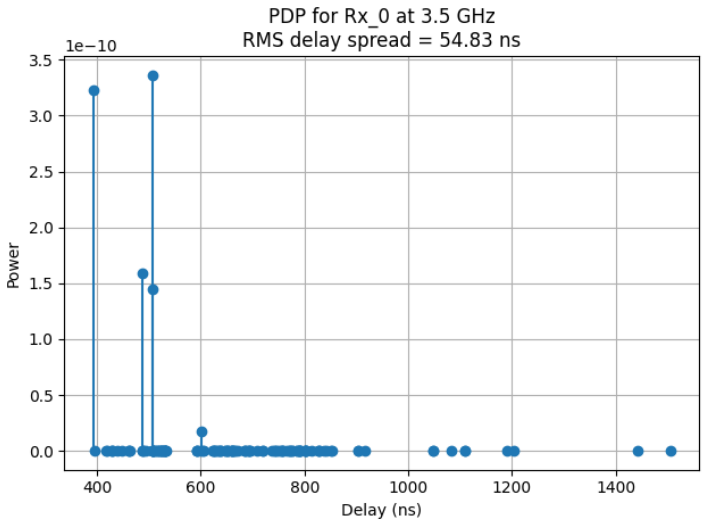

Figure 5: Power delay profile and RMS delay spread for receiver 0 at 3.5 GHz.

6 Discussion

The results confirm the expected behavior of urban street-canyon channels. At 3.5 GHz, the path loss is moderate, and the PDP shows several significant multipath components, leading to a non-negligible RMS delay spread that must be handled by the physical layer waveform. At 28 GHz, path loss increases and the propagation becomes more sensitive to line-of-sight and strong specular reflections, which is consistent with the more directional nature of millimeter-wave channels.

The interactive notebook is particularly useful for understanding which specific buildings and corners contribute to each cluster of rays. By rotating the 3D view, one can visually link PDP peaks and path-loss fluctuations to certain reflections and diffractions in the geometry, providing intuition that is hard to obtain from plots alone.

7 Conclusion and Provided Material

This project demonstrated a complete Python-based workflow for simulating and analyzing an urban wireless channel using Sionna RT. The implementation covers geometric preprocessing, robust placement of devices using scene bounds, multi-frequency ray tracing, path-loss extraction, and time-dispersion analysis. The generated figures illustrate path loss behavior, ray distributions, and PDP characteristics for the Pankow scenario.

To support reproducibility and interactive exploration, two artifacts are supplied with this report:

- A standalone Python script `Sionna_Code.py` containing the full simulation code.
- A Jupyter notebook `Interactive_Simulation.ipynb` that organizes the workflow into cells and provides an interactive 3D visualization of the scene and ray paths.

These resources allow other students to reproduce the plots, modify scenario parameters, and extend the analysis to additional frequencies, antenna configurations, or user trajectories.

References

[1] Sionna RT Documentation, *Geometry-Based Wireless Channel Simulation Library*. Available: `https://nvidia.github.io/sionna/`

[2] 3GPP, *TR 38.901: Study on channel model for frequencies from 0.5 to 100 GHz*.

[3] Sionna Example Scenes, *Pankow Urban Environment*.

[4] Mitsuba Renderer, *Physically Based Renderer and Scene Representation*.