**Nvidia Sionna Digital Twin Simulation Report**

David Lee, Vahid Yazdnian, Prof. Yasaman Ghasempour

SWAN Lab

### **Why use Sionna?**

Sionna is a GPU-accelerated open-source library for research in communication systems. It is differentiable and features a lightning-fast ray tracer for radio propagation, a versatile link-level simulator, and system-level simulation capabilities. This means:

1. It is much faster for big computations, especially ones involving parallel math like simulating radio channels or running neural networks.
2. It is designed specifically for researchers working on wireless communication — like 5G, 6G, Wi-Fi, etc.
3. Can scale up from one link to entire networks — multiple transmitters, receivers, base stations, etc.

### **Purpose**

In order to train ML models, we need accurate data points. However, physical measurements are costly and limited. If we can create an accurate digital twin of a wireless communication system, then we can generate accurate training data for the models and then hypertune with the real results also known as synthetic data training.

We want to assess whether the results Sionna generates through ray tracing aligns with the real world results from NIST datasets, particularly the 28 GHz datasets from NYC and Boulder, Colorado.

### **Tools & Documents**

* The codebase for this research is stored in this repository [here](https://github.com/davidklee04/Sionna_Princeton)
* Nvidia’s [Sionna](https://developer.nvidia.com/sionna) package used for ray tracing
* Duke FuNCtions Lab geo2sigmap [repository](https://github.com/functions-lab) used for scene generation
* [NYU 28 GHz Dataset](https://nextg.nist.gov/submissions/47)
* [Boulder 28.5 GHz Dataset](https://nextg.nist.gov/submissions/112)

### **Simulation Pipeline**



1. **Scene Generation** - the geo2sigmap repo generates 3D meshes of urban landscapes including the material properties of materials allowing for simulation of 28 GHz scenes.
   1. Note: when generating scenes, it is the buildings of the selected landscape. Other factors such as foliage, pedestrians, traffic are not accounted for during this process. As such we will see certain rays from a central transmitter (TX) arriving at certain receivers (RX) since it does not know about this interference. This is a false positive error.
2. **TX/RX Placement** - for each dataset we are given the RX locations in GPS coordinates. We first needed to check whether these RXs were being placed accurately. Because we center the scene based on the TX location, when generating the scenes, we needed to make the RX coordinates relative to the TX. However, through this process we result in RX that are inside buildings when in reality, it is placed outside. And as such it will not receive certain signals when it should be (false negative). The geo2sigmap repo ignores the receivers within the buildings but for the purpose of our research, we need these receivers as we have very few data points to compare for accuracy. In order to solve this problem, we populate an X amount of random locations (e.g., 10,000) in a scene, disregard the locations within buildings, and then find the closest coordinate outside of a building to a receiver inside and replace the old coordinates of the receiver with the new location. This speedup is done through a KDTree which supports nearest neighbor searches.
3. **Ray Tracing Simulation** - We now use Sionna’s PathSolver() function to generate ray paths. There are different parameters to tune this ray tracing function:
   1. # Perform ray tracing using PathSolver()
   2. solver = PathSolver()
   3. paths = solver(scene,
   4. max\_depth=2,
   5. los=True,
   6. specular\_reflection=True,
   7. diffuse\_reflection=True,
   8. refraction=False,
   9. samples\_per\_src=int(1e3)
   10. )
4. **Path Gain Computation** - Each RX receives a certain number of rays. With Sionna, we can extract certain features such as the power, AoA azimuth, AoA elevation, AoD azimuth, AoD elevation, delay, etc to compare with the actual results. We have also normalized certain features such as the power (converting to linear scale, normalizing by max value, and then reconverting to logarithmic) and the delay (shift max delay as 0).
5. **Analysis**
   1. For the NYC data we could only compare the power values
   2. For the Boulder data, we were able to compare the power, AoA azimuth, AoA elevation, and delay results.
      1. Because there were more data points, we took the strongest path gain ray to compare the features above.

### NYU

The NYU data is given a word document which was extracted into a csv file in the repository.

#### Data



The NYU dataset contained 74 data points with 6 LOS (L) paths and 68 NLOS (N) paths. As shown by the columns, the only data value that we could compare is the path loss values (PL) and the rows marked with “ - ” indicate that the receiver did not receive any signal and the “ \* ” indicate that this receiver was not considered in the omnidirectional path loss.

#### Results

Path\_solver parameters:

# Perform ray tracing using PathSolver()

def manhattan\_pathsolver(scene):

solver = PathSolver()

paths = solver(

scene,

max\_depth=3,

los=True,

specular\_reflection=True,

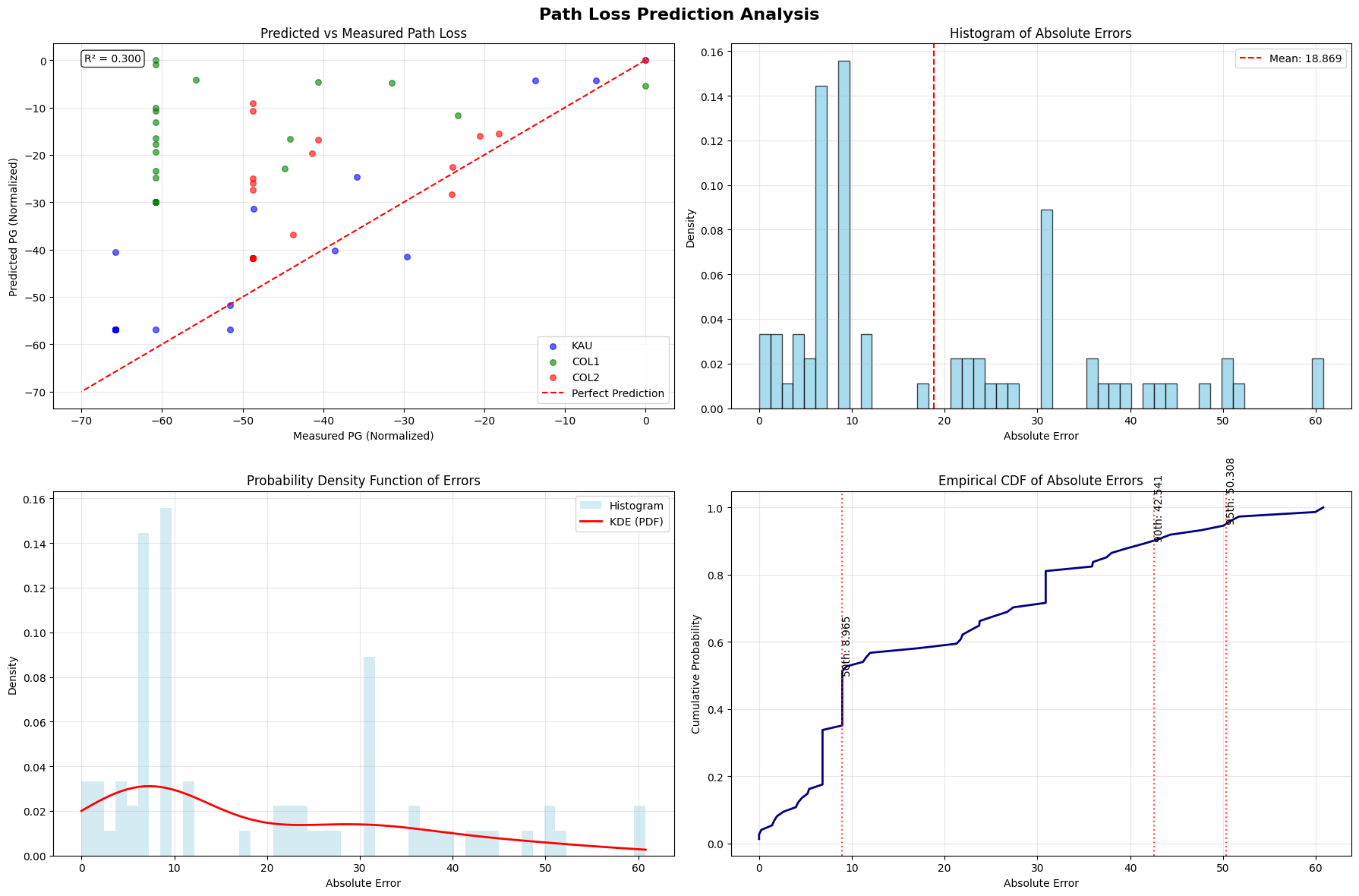
diffuse\_reflection=True,

refraction=False,

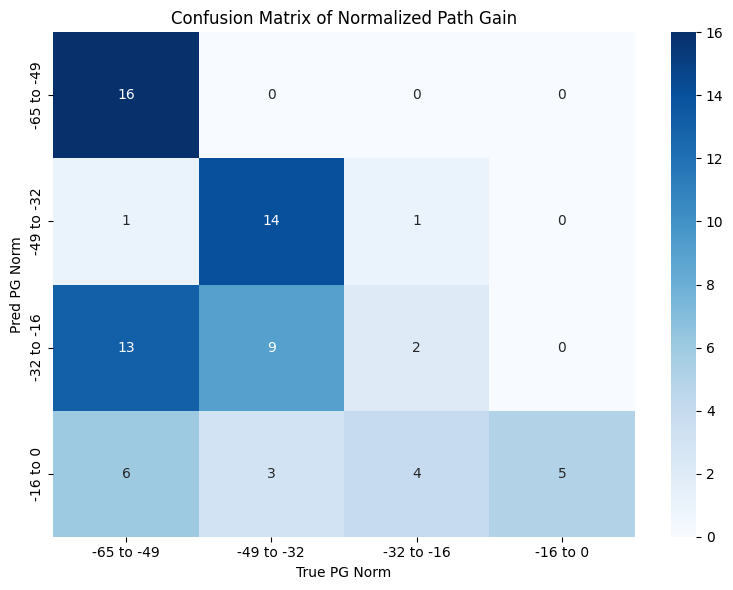
samples\_per\_src=int(1e6)

)

return paths

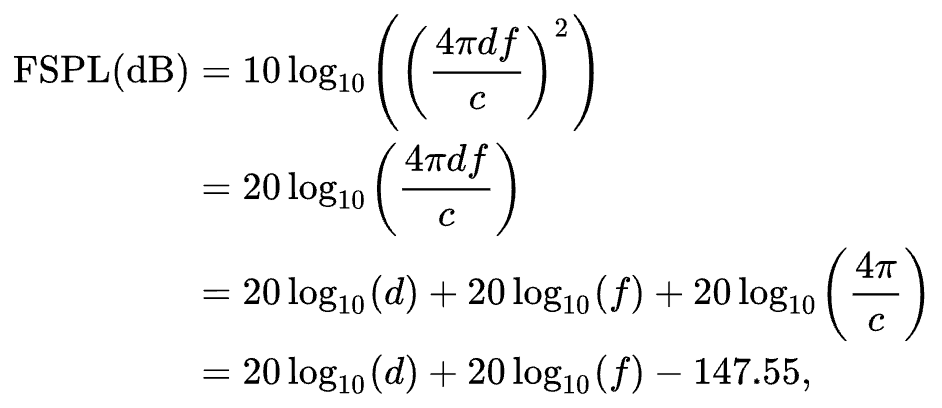


As shown in the first scatter plot, there are more plots located above the perfect prediction line, indicating a greater number of false positives (Receivers in Sionna are picking up a signal while real data cannot). This is actually expected because in our scene generation we did not include interfering factors such as foliage due to the difficulty of accurately simulating these elements. So in Sionna, there is a greater chance that a receiver will receive a signal through various paths that do not encounter interference or noise.



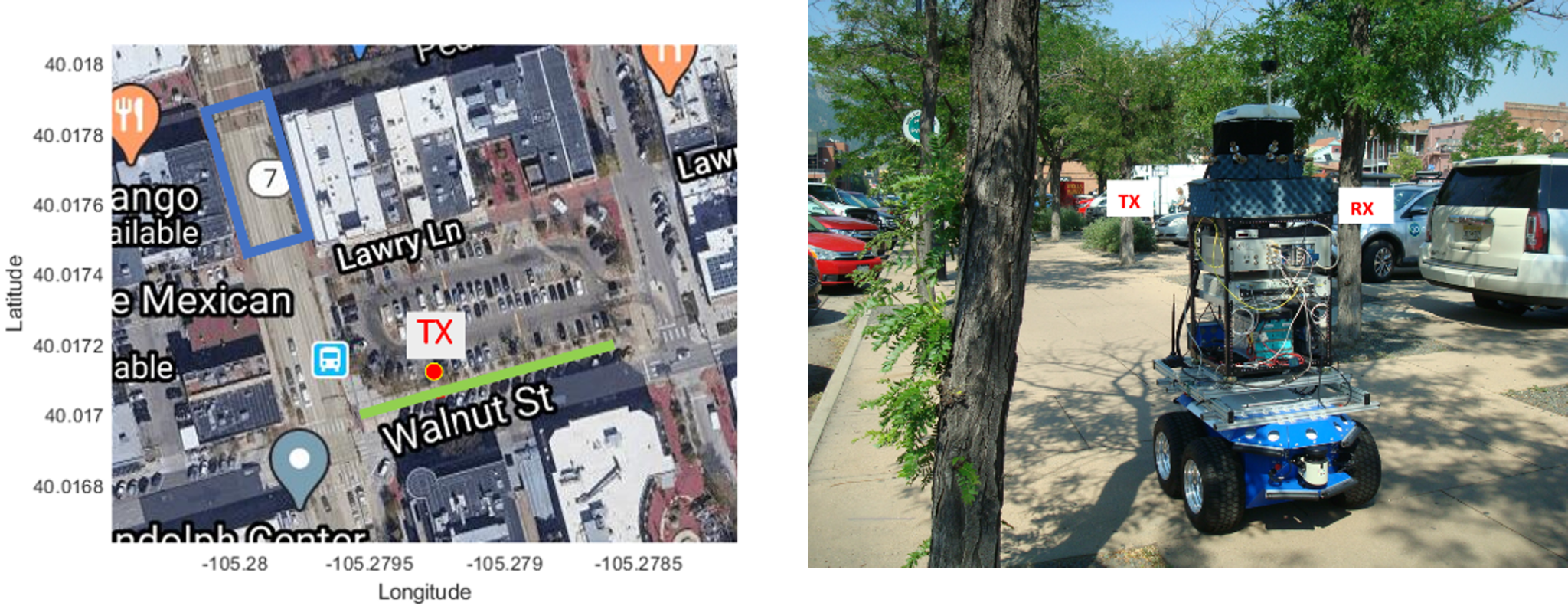
Looking at the confusion matrix we can further illustrate the false positives (bottom left side) and the false negatives (top right side). The bottom right corner is all the LOS receivers (excluding the one that was not considered in the real dataset) so that is a good indicator that Sionna works generally well for LOS paths.

Vahid asked me to look at a false positive case to see if we can extract any more information from Sionna. In particular, we looked at the first column from the left and the second row from the top (KAU scene receiver 8). The distance between the KAU transmitter and the receiver is about 94.3 meters. Using the free space path loss equation shown below, we get a value



### Boulder

#### Data



The Boulder dataset included two measurement types: NLOS (in blue) and LOS (in green). The dataset included information on each received ray’s power, AoA azimuth, AoA elevation, and delay.

#### Result

Pathsolver parameters:

# Configure simulation parameters

scene.frequency = 28.5e9

scene.synthetic\_array = True # Optimize for array calculations

for radio\_material in scene.radio\_materials.values():

radio\_material.scattering\_coefficient = 0.4

# Perform ray tracing using PathSolver()

solver = PathSolver()

paths = solver(scene,

max\_depth=2,

los=True,

specular\_reflection=True,

diffuse\_reflection=True,

refraction=False,

samples\_per\_src=int(1e3)

See [slidedeck](https://www.canva.com/design/DAGuaUN17ns/VfYcMpu_p8CSYOCabqevrA/edit?utm_content=DAGuaUN17ns&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton) for results.

Things to note:

* For the prediction analysis, we are looking at the strongest path gain computed by each part (Sionna and the Boulder data) and comparing its corresponding data points.
* For the individual receiver profiles
  + AoA azimuth and elevation angles have been shifted by 90 degrees to align with their corresponding profiles.
  + The delay profile has been normalized where the strongest path gain is considered the 0 delay and every delay for a receiver is shifted.
* LOS comparisons are appearing quite aligned, NLOS not so much but the results are not as bad as expected.

### Insights

From Sionna’s PathSolver() the refraction factor allows rays to pass through buildings. However, because we are dealing with high frequency signals this is not the best way to simulate the scenes. We also do not generate as many rays when refraction is turned on and diffuse reflection is turned off. However, when we turn on diffuse reflection this allows scattering to occur. This actually generates more simulation rays and by adding a thresholding layer to the power rays we generate results that are more accurate with less false negatives. One reason is that with diffuse reflection, there is an opportunity for the ray tracer to probabilistically define the behavior of the em wave interactions with different buildings so we generate more results. This parameter is one factor that the Duke team has overlooked in their work.

Sionna does pretty well with line of sight (LOS) ray tracing, producing accurate profiles for power, AOA, and delay for both NYC and Boulder data. Additionally, the stronger the received signal, the more aligned the results became. Weaker signals had predictions that were not as aligned. But overall, even the NLOS sites for the Boulder data were on point. The NYC data contained less details and there were more urban interference factors so we could not extrapolate as much accurate data.

As a leader in my Christian fellowship, this role comes with much sacrifice and humility as I challenge myself to truly take care of the people on this campus. The difficult part is that so many people come from various backgrounds and I cannot always understand their perspectives. But as I’ve come to have very honest conversations with students, I’ve come to not only know about them but also create genuine relationships with them. I may not be the most intellectually inclined on the team, but I know how to communicate with different people and bridge commonalities from two diverse sides. In finance roles, there are a lot of numbers involved. But behind those numbers are also people who are connected as well.