# DeepMIMO

The Community Database and Toolchain for Ray-tracing Datasets

4.0.0beta is now available

Wireless
Intelligence Lab

AI-RAN™
ALLIANCE

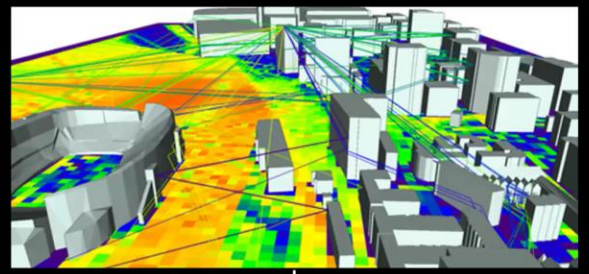
# Seamless, Scalable, Reproducible

DeepMIMO's mission is to accelerate AI wireless research by making ray tracing data easy to access and share at scale.

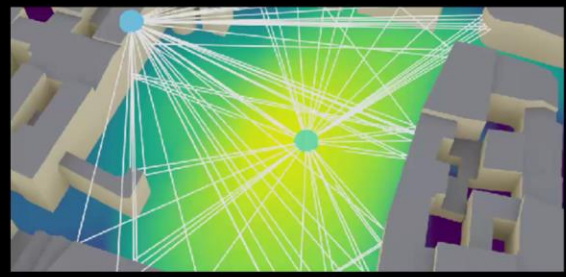
# DeepMIMO Bridges Ray Tracer Datasets and Simulation Tools

NVIDIA AODT

REMCOM InSite



NVIDIA Sionna RT



DeepMIMO Parameters

```
#
```

Unified DeepMIMO Scenario → DeepMIMO Database

DeepMIMO Generator

DeepMIMO Dataset

NVIDIA Sionna

interdigital NeoRadium

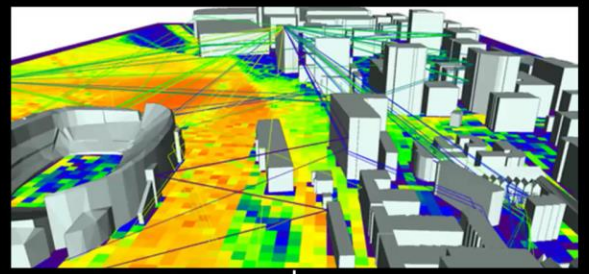
DeepMIMO Processing & Plotting

MathWorks 5G Toolbox

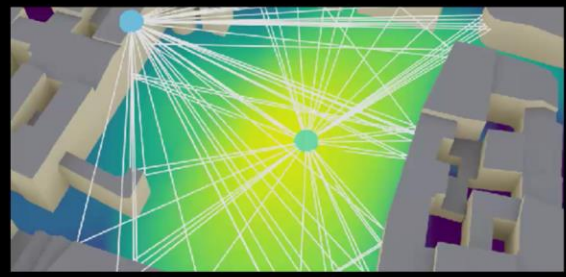
# Convert Ray Tracing Simulations to DeepMIMO

NVIDIA AODT

REMCOM InSite



NVIDIA Sionna RT



*One standard format for all data*

**DeepMIMO Scenario**

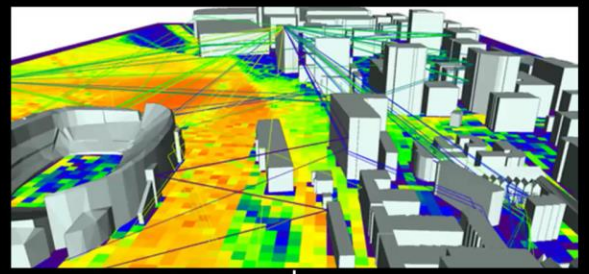
```python
import deepmimo as dm

dm.convert(raytracing_folder)
```

Unlock Interoperability

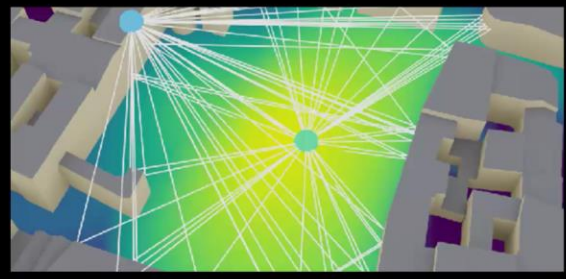
# Convert Ray Tracing Simulations to DeepMIMO

NVIDIA AODT

REMCOM InSite



NVIDIA Sionna RT



*One standard format for all data*

**DeepMIMO Scenario** ⋯▸ DeepMIMO Database

```python
import deepmimo as dm

dm.convert(raytracing_folder)
```

Growing Database of Multiple Ray Tracers

Unlock Interoperability

Enable Machine Learning at Scale

Enable Ray Tracer Comparisons

# Generate Wireless Channel Datasets

```python
import deepmimo as dm

dm.download('asu_campus_3p5')
```

```python
import deepmimo as dm

# Load a scenario
dataset = dm.load('asu_campus_3p5')

# Instantiate channel parameters
params = dm.ChannelParameters()

# Configure BS antenna array
params.bs_antenna.shape = [8, 4]  # 8x4 array

# Configure UE antenna array
params.ue_antenna.shape = [1, 1]  # Single antenna

# Configure OFDM parameters
params.ofdm.bandwidth = 10e6  # 10 MHz

# Generate frequency-domain channels
channels = dataset.compute_channels(params)
```


Flexibility

Antenna Array

Doppler & Mobility

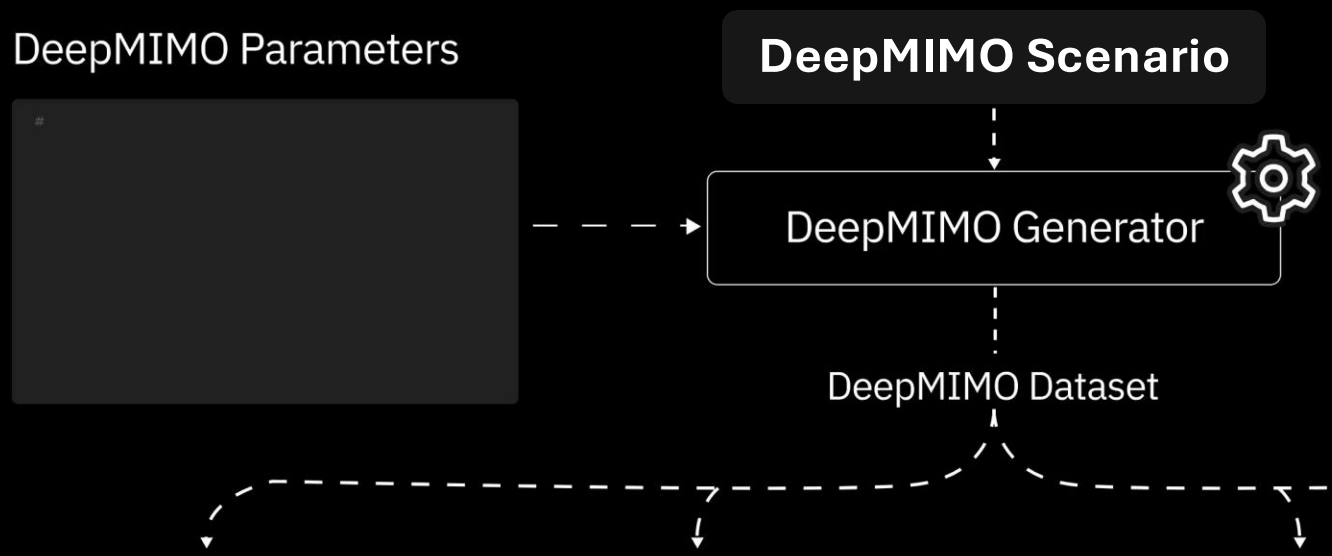
Reproducibility

Accurately defined

Easy to share

# Integrate DeepMIMO with Link/System-level Simulators

```python
from deepmimo.integration import SionnaAdapter
from deepmimo.integration import NeoRadiumAdapter
from deepmimo.integration import Matlab5GAdapter
```




**Portable**
All you need is Python

**Interoperable**
Not locked to one full chain

# Search 100s of Scenarios based on Ray Tracing Parameters

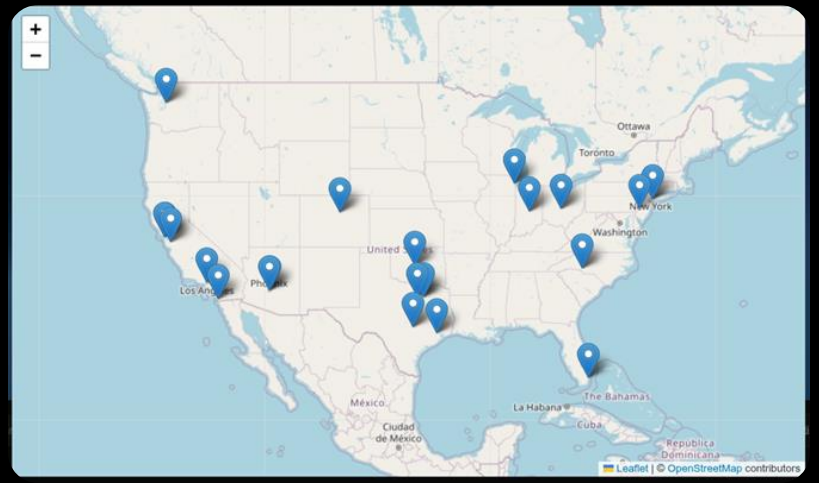
Search by scenario title... | 15 per page | << | < | Page 1 of 11 | > | >> | Advanced Settings

152 TOTAL RESULTS | Copy | SELECT ▼ | Mir | Ma | Mir | Ma | All | All

| SCENARIO | IMAGE | BAND | NUM RX | MAX REFLECTIONS | RAYTRACER | ENVIRONMENT |
|---|---|---|---|---|---|---|
| asu_campus_3p5 |  | sub6 | 85158 | 6 | Insite | outdoor |
| city_0_newyork_3p5 |  | sub6 | 16306 | 3 | Insite | outdoor |
| city_1_losangeles_3p5 |  | sub6 | 19914 | 3 | Insite | outdoor |

DeepMIMO Database

**World Map Available**




+150 Scenarios Indexed by:

Ray-tracing parameters

Location, Environment, etc.

# Use APIs to Interface with DeepMIMO Database Directly in Code

## Download

```python
import deepmimo as dm

scenario_name = 'asu_campus_3p5'
dm.download(scenario_name)
```

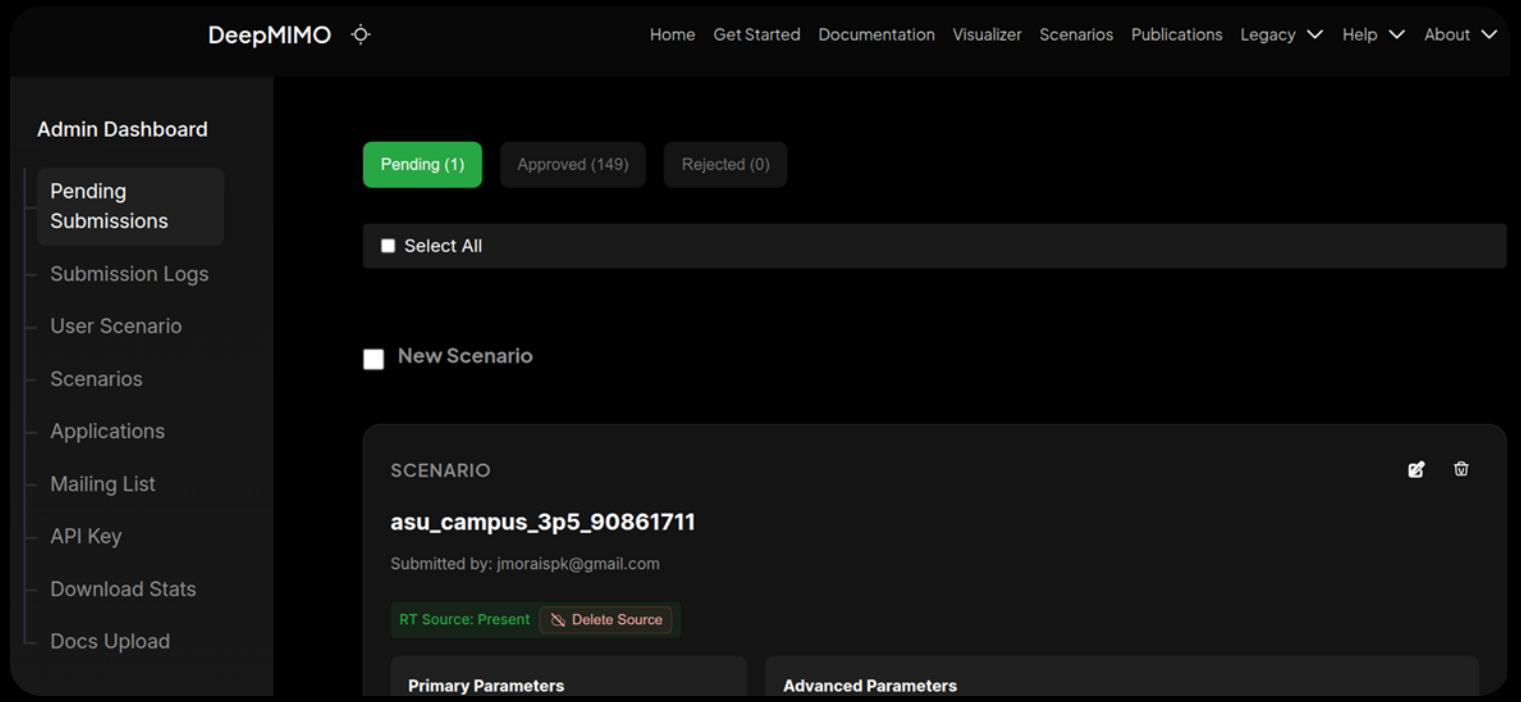
## Search

```python
query = {'maxReflections': {'min': 5}}
search_results = dm.search(query)
```

## Upload

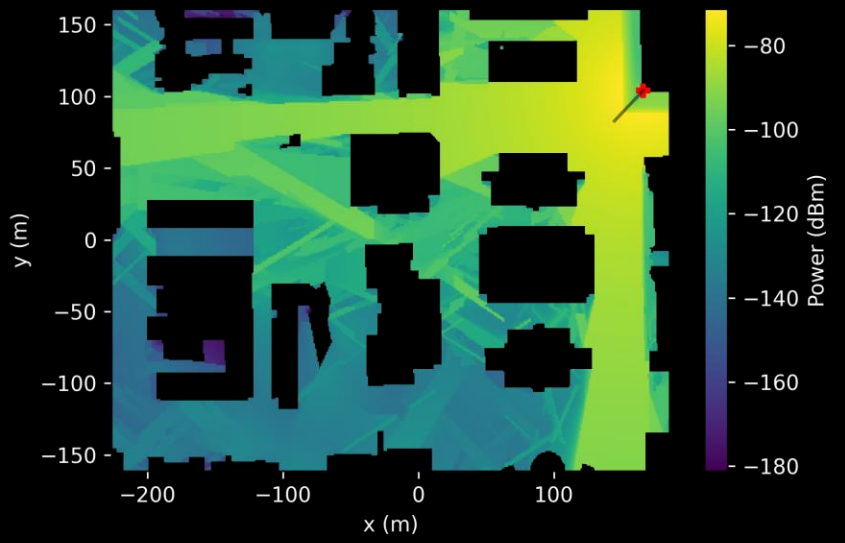
```python
import deepmimo as dm

dm.upload('my_converted_dm_scenario')
```

Online Dashboard to Manage Your Submissions

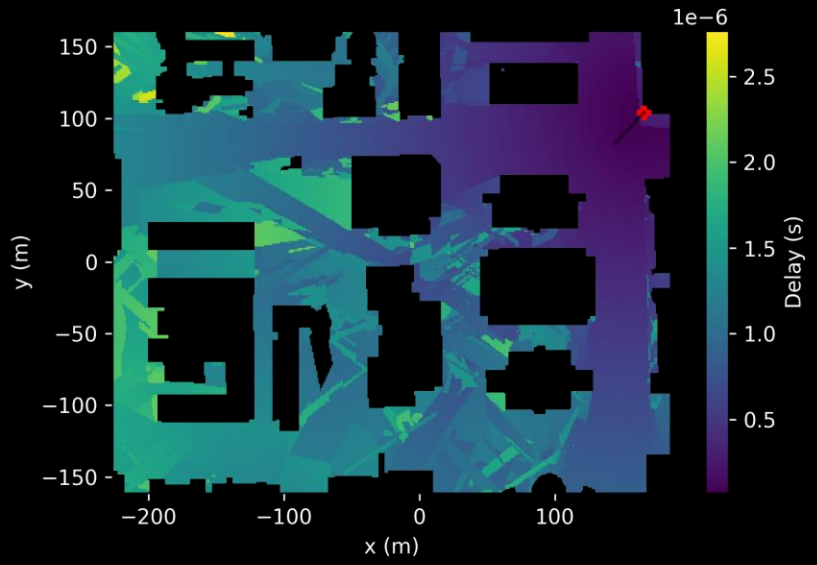


# Leverage Many Built-in Processing and Visualization Tools

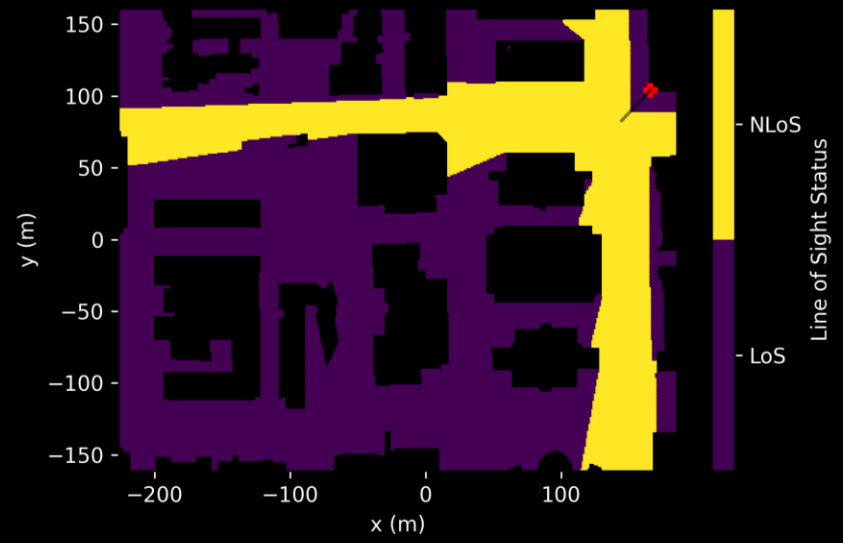
```
dataset.power.plot()
```




```
dataset.delay.plot()
```




```
dataset.los.plot()
```



# Manipulate Data with Matrix-based Objects

```
dataset.power # [n_rx, n_paths]
```

```
dataset.phase # [n_rx, n_paths]
```

```
dataset.delay # [n_rx, n_paths]
```

```
dataset.aoa_az # [n_rx, n_paths]
```

```
dataset.aoa_el # [n_rx, n_paths]
```

```
dataset.aod_az # [n_rx, n_paths]
```

```
dataset.aod_el # [n_rx, n_paths]
```

```
dataset.inter # [n_rx, n_paths]
```

```
dataset.inter_pos # [n_rx, n_paths, n_inter, 3]
```

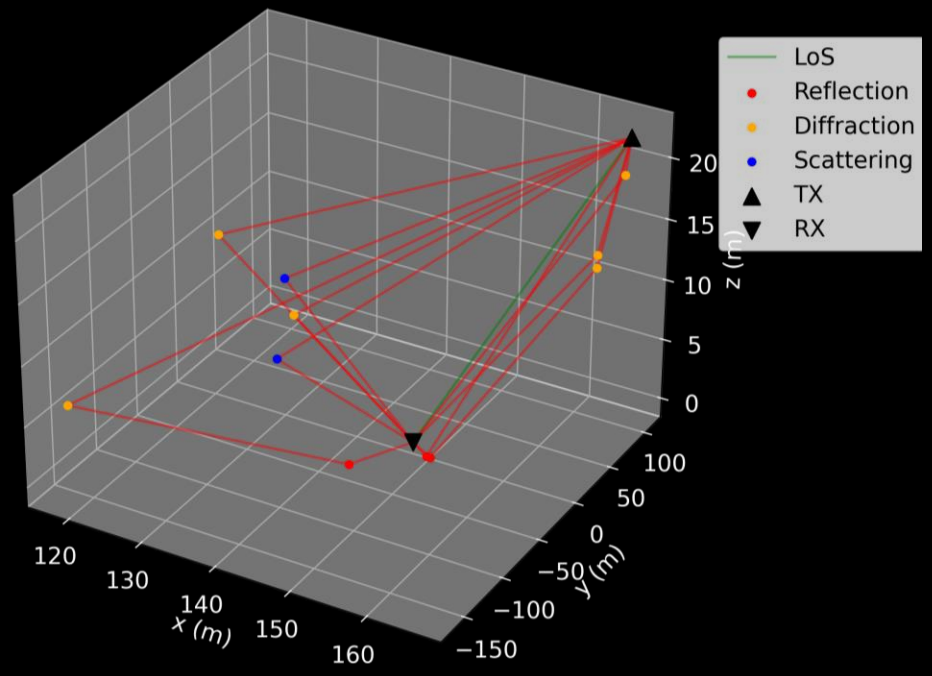
```
dataset.rx_pos # [n_rx, 3]
```

```
dataset.tx_pos # [n_tx, 3]
```

# Plot Rays and Scene Information

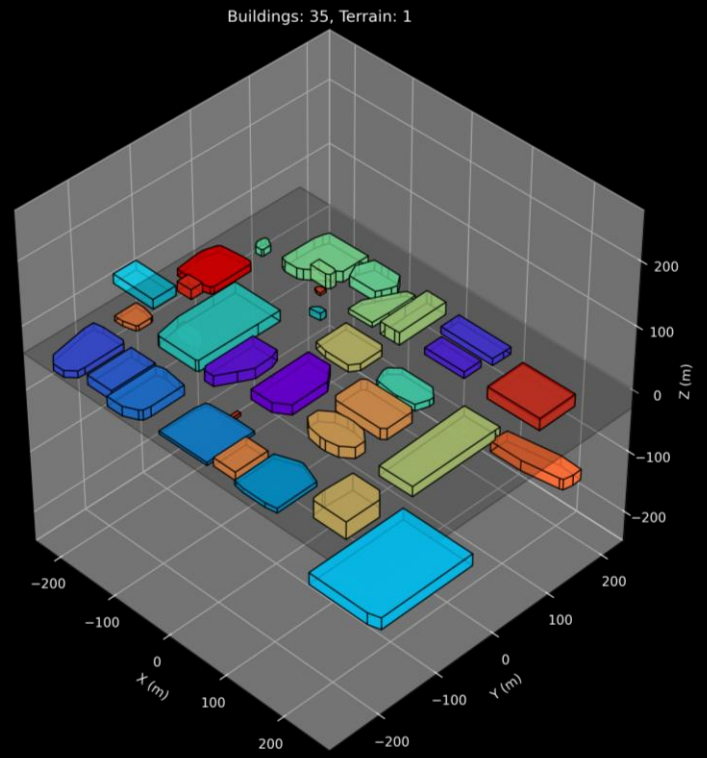
## Rays with Matplotlib

```
dataset.plot_rays()
```

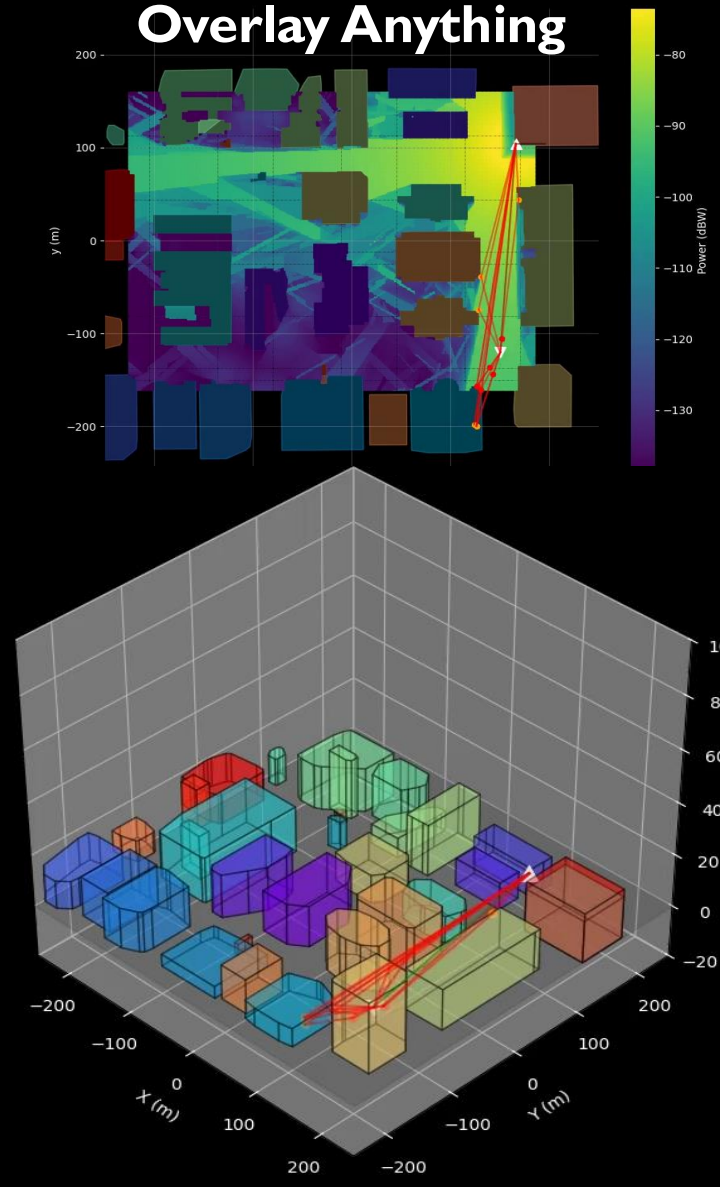



## Objects by Material in the Scene

```
dataset.scene.plot()
```



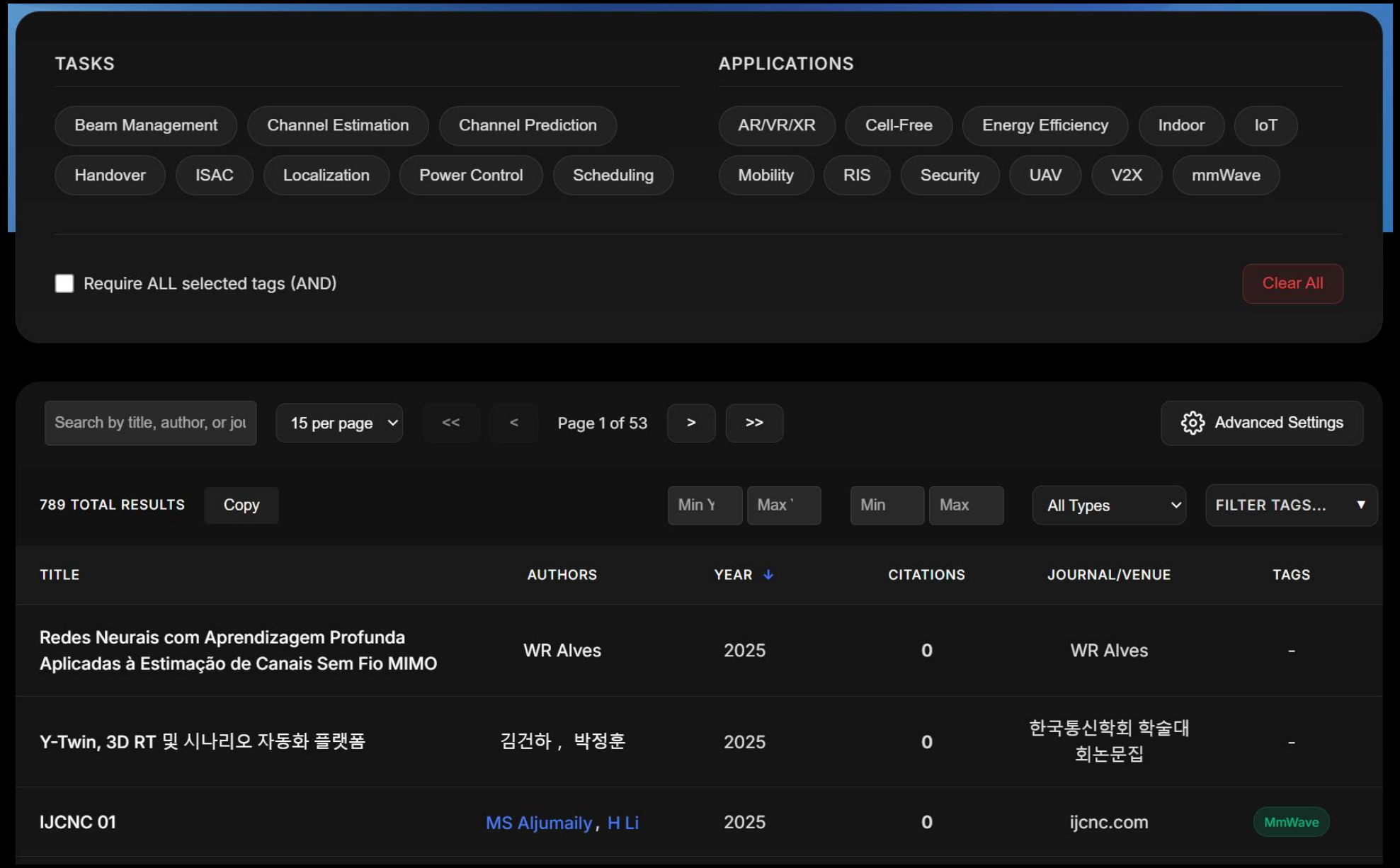

## Overlay Anything



# Train and Benchmark Machine Learning Models on Hundreds of Scenarios in a Few Lines of Code

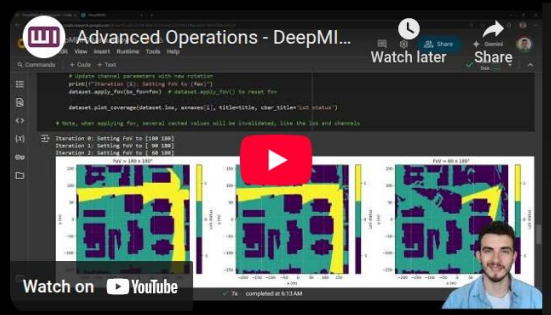
```python
import deepmimo as dm

scenarios = dm.search({
  'bands': ['sub6'], 'environment': 'outdoor'
})

for scenario in scenarios:
    dataset = dm.load(scenario)

    # Channel Compression
    model.train(dataset.channels)

    # Coverage Predicton
    model.train(dataset.rx_pos, dataset.power)

    # CSI Localization
    model.train(dataset.rx_pos, dataset.channels)

    # LoS Identification
    model.train(dataset.los, dataset.power)
    model.train(dataset.los, dataset.channels)

    # Position-aided Beam Prediction
    beams = get_codebook(dataset.channels, n_beams=32)
    model.train(dataset.rx_pos, beams.max(axis=1))
```

# Find and Reproduce Research with the Database of Publications



# And More ...

## Tutorials




### Advanced Operations

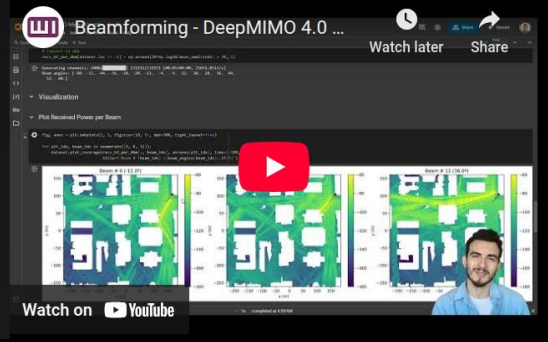
FoV analysis for receivers

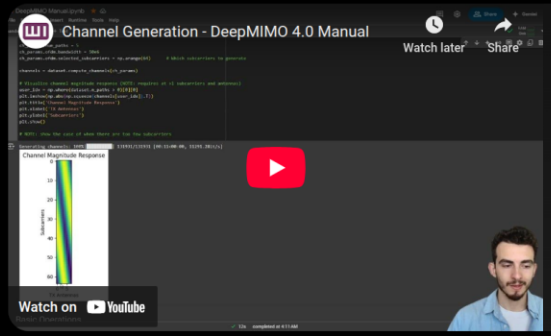
Watch Tutorial

### Beamforming

Calculate received power with beamforming, beamforming visualization methods

Watch Tutorial



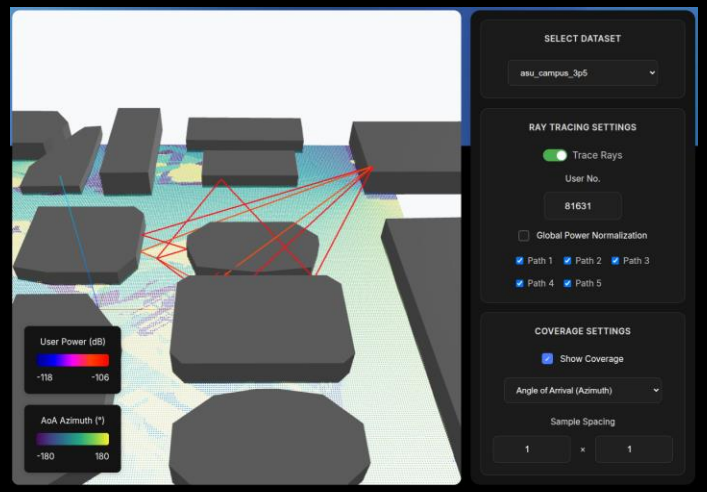



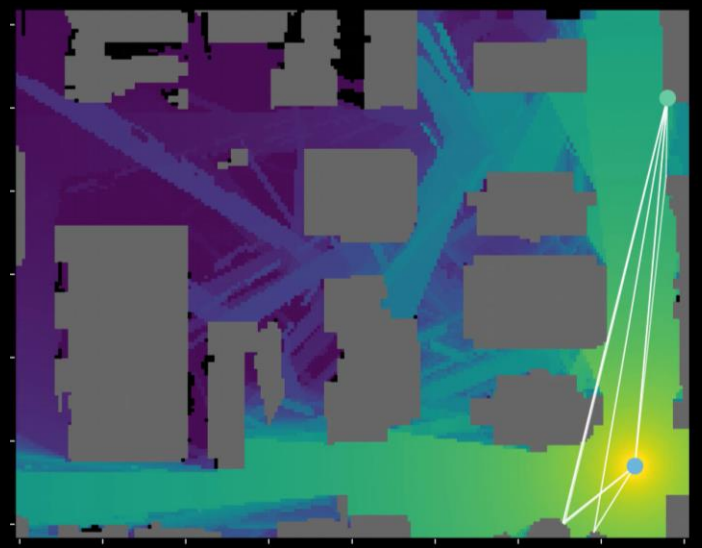

### Channel Generation

Configuring channel generation and generating time-domain and OFDM channel responses

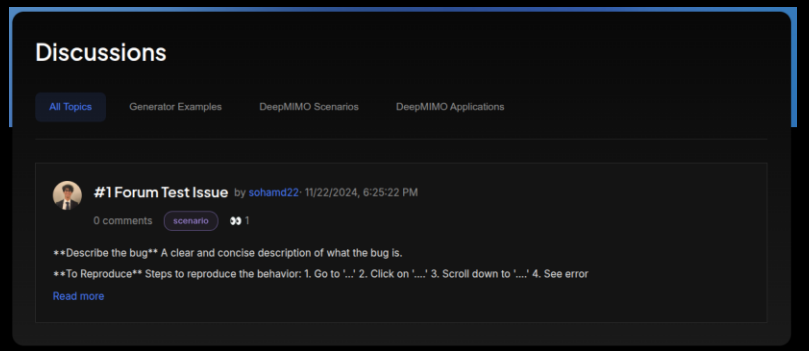
Watch Tutorial

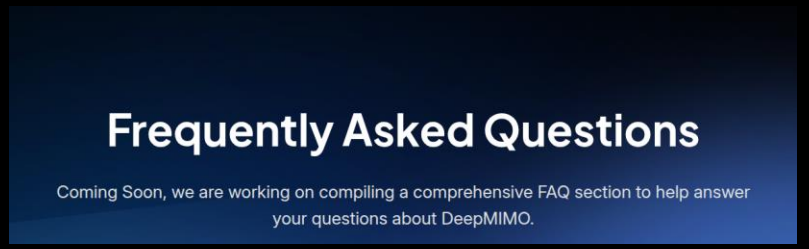
## Online Visualizer

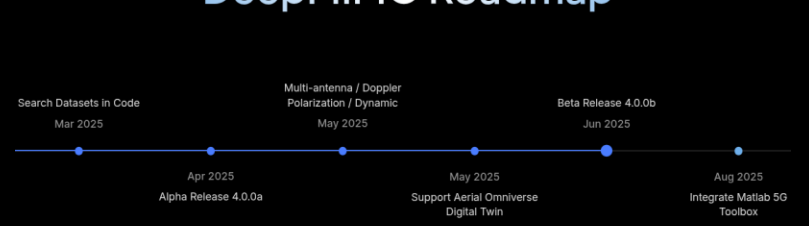


## Forum, FAQ, Roadmap

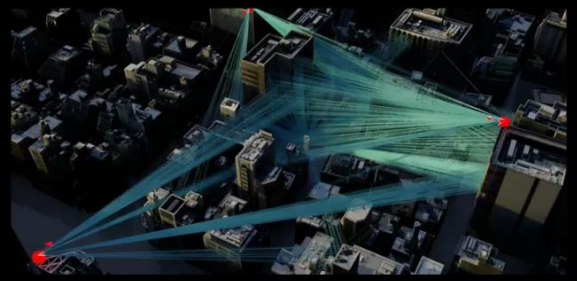


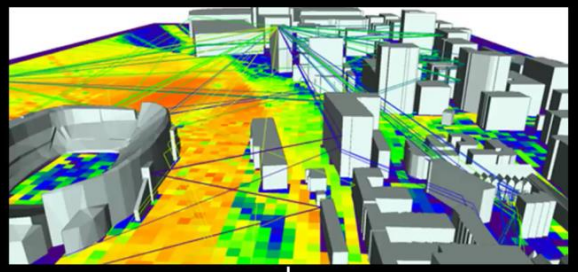



# Enhancement 1: OAI Integration

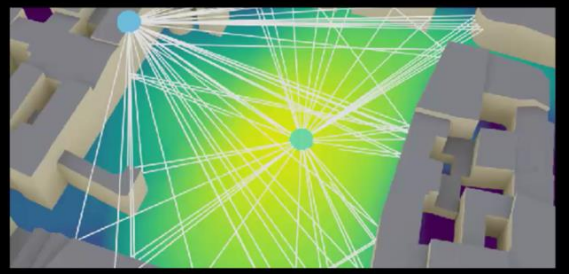
NVIDIA AODT



REMCOM InSite



NVIDIA Sionna RT



DeepMIMO Parameters

Unified DeepMIMO Scenario

DeepMIMO Generator

DeepMIMO Dataset

NVIDIA Sionna

interdigital NeoRadium

DeepMIMO Processing & Plotting

MathWorks 5G Toolbox

OPEN AIR INTERFACE

# Enhancement 2: Raytracing Pipelines

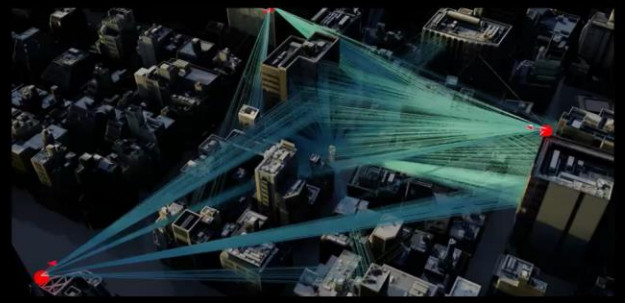
**Raytracer agnostic**

1. Scene Extraction — **e.g.: Open Street Maps, …**
2. Scene Processing — **e.g.: Assing Materials → Surface,..**

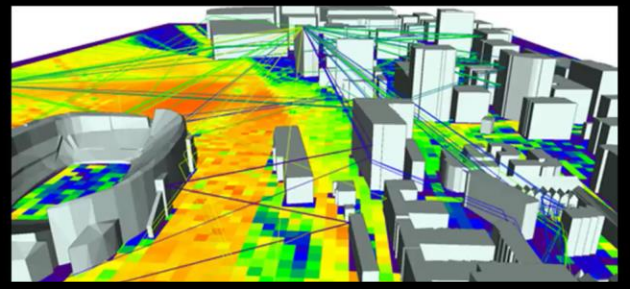
**Raytracer dependent**

3. Scene Export — **e.g.: Write scene in XML, USD, …**
4. Raytracer Configuration — **e.g.: Number of Diffractions, …**

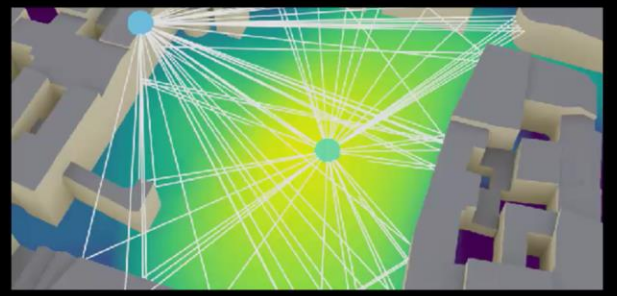
NVIDIA AODT



REMCOM InSite



NVIDIA Sionna RT



Available Now

# deepmimo.net

```
pip install --pre deepmimo
```