Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in
Ingegneria Informatica, delle Comunicazioni ed Elettronica

ELABORATO FINALE

SVILUPPO DI UN DIGITAL TWIN PER L'ANALISI DI COPERTURA WIRELESS IN AMBIENTI URBANI

Supervisore
Fabrizio Granelli

Laureando
Di Lauro Antonio

Anno accademico 2023/2024

Ringraziamenti

Nota
sezione Ringraziamenti opzionale

INDICE

Sommario

La crescente diffusione dei dispositivi tecnologici e l'aumento delle richieste di connettività degli ultimi anni hanno reso sempre più complesso garantire una copertura ottimale nelle aree urbane densamente popolate.
Con l'avvento del 5G e la futura implementazione del 6G, è diventato cruciale analizzare in anticipo l'infrastruttura di rete, pianificando il corretto posizionamento delle antenne e ottimizzando la propagazione del segnale. Infatti, la capacità di simulare accuratamente le condizioni reali in un ambiente virtuale rappresenta un passo fondamentale per migliorare la qualità del servizio offerto.

In questo contesto, la mia tesi si propone di sviluppare un Digital Twin per simulare la propagazione del segnale radio telefonico in un'area urbana, utilizzando Nvidia Sionna, una libreria avanzata ottimizzata per il calcolo parallelo su GPU. Nvidia Sionna, integrata con Python e TensorFlow, offre potenti strumenti di machine learning e ray tracing differenziabile per modellare il comportamento delle onde elettromagnetiche in ambienti complessi. Inoltre, il progetto sfrutta la modellazione 3D di Blender per ricostruire le strutture urbane partendo dai dati di OpenStreetMap (OSM), per poi simulare la propagazione del segnale attraverso l'ambiente digitale generato.

L'obiettivo principale è stato lo sviluppo di un software che, combinando queste tecnologie, permettesse in modo automatico la visualizzazione, l'analisi e la simulazione delle coperture radio in ambienti urbani. Il processo si articola in diverse fasi: dalla selezione e importazione dei dati georeferenziati da OpenStreetMap, alla modellazione 3D con Blender, fino alla simulazione della propagazione delle onde elettromagnetiche con Nvidia Sionna, per arrivare alla generazione di mappe di copertura che rappresentano visivamente l'intensità del segnale nelle aree analizzate.

Più precisamente, il progetto prevede la selezione di un'area quadrata specifica attraverso un modello OpenStreetMap, che rappresenta un campione della regione di interesse. I dati dell'area vengono esportati in formato OSM e successivamente importati in Blender per la creazione di un modello 3D accurato. In seguito, il modello viene trasformato in un file XML compatibile con Mitsuba, da cui vengono estratte le informazioni necessarie per l'interpretazione in Nvidia Sionna. Questo file XML costituisce l'input per la simulazione, dove viene impostato uno scenario con un'antenna trasmittente (TX) e un ricevitore (RX). Durante la simulazione, vengono calcolati e visualizzati i percorsi del segnale tra trasmettitore e ricevitore, i quali aumentano in complessità man mano che la simulazione procede. Nvidia Sionna viene impiegato per generare mappe di copertura che mostrano graficamente le aree a più alta potenza del segnale, in base alla distanza dagli ostacoli. Per ciascun punto nell'area simulata, viene calcolato un valore in decibel, che rappresenta la potenza del segnale in quel punto e viene salvato in un file CSV (Comma-Separeted Values), il quale è un formato di file spesso utilizzato per importare ed esportare file nei database, in cui ogni valore è separato da un delimitatore, generalmente una virgola. Ciò permette di monitorare i dati in tempo reale e confrontare la copertura simulata con la posizione attuale dell'utente, restituendo la stima più accurata. Le simulazioni sono state effettuate su cinque punti chiave nel centro di Trento: Piazza Pasi, Residenza Galileo, Santa Maria Maggiore, Palazzo Quetta e Palazzo Tabarelli. I dati ottenuti dalle simulazioni sono stati poi confrontati con misurazioni reali effettuate sul campo, al fine di validare l'accuratezza del modello sviluppato.

L'integrazione con Python e TensorFlow ha permesso di automatizzare molti processi, rendendo le simulazioni più efficienti e precise, e lasciando come unico intervento umano la selezione dell'area da analizzare. Nvidia Sionna ha gestito la simulazione della propagazione del segnale, calcolando i percorsi delle onde radio in funzione degli ostacoli fisici presenti, come edifici e altre strutture. Il modello ha tenuto conto di riflessioni, diffrazioni e rifrazioni delle onde elettromagnetiche, rendendo possibile la simulazione realistica del comportamento del segnale in ambienti urbani complessi.

Il progetto ha raggiunto diversi risultati degni di nota, tra cui la creazione di un Digital Twin dettagliato di un'area urbana utilizzando Blender e la creazione di mappe di copertura accurate, le quali hanno permesso di identificare le aree con segnale debole e suggerire interventi per migliorare la distribuzione delle antenne. Le

simulazioni hanno mostrato una buona correlazione con i dati reali, anche se sono state rilevate alcune discrepanze dovute a fattori locali non modellati, come la presenza di vegetazione o condizioni atmosferiche variabili. Nonostante ciò, il modello sviluppato ha dimostrato un alto livello di affidabilità e precisione, confermando l'efficacia dell'approccio utilizzato per simulare reti wireless di prossima generazione.

In conclusione, questo lavoro di tesi ha contribuito allo sviluppo di un metodo innovativo per la simulazione della copertura del segnale radio in ambienti urbani. Il Digital Twin realizzato rappresenta uno strumento potente per l'analisi e l'ottimizzazione delle reti di telecomunicazione, e potrebbe essere utilizzato per migliorare l'infrastruttura delle reti 5G e 6G. La validazione del modello con misurazioni sul campo ha confermato l'affidabilità delle simulazioni, suggerendo potenziali futuri miglioramenti nelle tecnologie di simulazione e modellazione.

1 Introduzione

Il primo capitolo della tesi verge sugli obiettivi e la struttura della stessa.

1.1 Obiettivi della tesi

Il progetto mira alla realizzazione di un Digital Twin per la simulazione della propagazione del segnale radio telefonico in un ambiente urbano. Pertanto, è stato sviluppato un software che in modo automatico consente di modellare, analizzare e prevedere le aree con copertura wireless insufficiente, permettendo di adattare in modo intelligente l'infrastruttura di rete. La creazione del Digital Twin, una replica digitale dell'ambiente reale, permette di simulare e analizzare con precisione la propagazione del segnale, rendendo più efficiente il processo di studio e ottimizzazione. Tale tecnologia si avvale del machine learning per condurre simulazioni accurate, tenendo conto degli ostacoli rappresentati dagli edifici e delle proprietà dei materiali circostanti.

Per conseguire tutte le analisi, è stato fondamentale l'utilizzo di Nvidia Sionna, una libreria open-source basata su TensorFlow e ottimizzata per GPU, pensata per la simulazione di reti wireless in contesti urbani complessi. Grazie al ray tracing differenziabile, Nvidia Sionna offre un'accurata modellazione dei percorsi delle onde radio rispetto all'ambiente circostante, permettendo di calcolare in dettaglio la propagazione del segnale.

Sionna consente inoltre di definire i parametri delle antenne con grande precisione, adattandoli alle esigenze specifiche, rendendola una tecnologia fondamentale per lo sviluppo delle reti di comunicazione di prossima generazione. Grazie all'integrazione con Python e TensorFlow, Nvidia Sionna gestisce l'elaborazione dei dati per simulazioni avanzate. L'intervento dell'utente è ridotto al minimo: basta fornire un file OSM come parametro iniziale, dopodiché il sistema elabora i dati restituendo una visualizzazione a 360 gradi dei percorsi del segnale e della copertura nelle aree attorno alle antenne. [4]

1.2 Struttura della tesi

La struttura della tesi è articolata in cinque capitoli, ognuno dei quali dedicato ad un aspetto cruciale del progetto e della sua valutazione finale, basata sulla comparazione dei risultati con misurazioni reali. Ecco una descrizione della sua composizione:

1. **Introduzione**: In questo primo capitolo viene contestualizzato il progetto e definiti gli obiettivi principali dello stesso. Si affrontano i concetti di base del Digital Twin e le aspettative sui risultati finali.

2. **Strumenti utilizzati per la creazione del Digital Twin**: Il secondo capitolo esamina nel dettaglio gli strumenti impiegati per creare il Digital Twin e simulare la propagazione del segnale. Si esplorano le capacità di Blender nella modellazione tridimensionale, compreso il settaggio dei materiali, ed il ruolo di Nvidia Sionna nella simulazione del comportamento delle onde elettromagnetiche in un ambiente urbano.

3. **Descrizione del lavoro**: Il terzo capitolo descrive lo sviluppo di un software per il calcolo della copertura wireless in contesti urbani, facendo uso di Nvidia Sionna. Il software impiega il ray tracing differenziabile per simulazioni precise, tenendo conto di ostacoli e proprietà materiali degli edifici, e si avvale di Python e TensorFlow per l'elaborazione dei dati e l'ottimizzazione dei parametri dell'antenna, essenziali per le reti di nuova generazione.

4. **Sperimentazione**: In questa fase vengono presentati i risultati delle simulazioni e la loro validazione attraverso il confronto con i dati raccolti sul campo. Si analizzano i metodi utilizzati per verificare l'accuratezza del modello sviluppato e se ne discute il potenziale applicativo.

5. **Conclusioni**: Nell'ultimo capitolo si riassumono i principali risultati ottenuti, riflettendo sulle sfide affrontate e sui limiti del progetto.

2 Strumenti per la creazione del Digital Twin

Il secondo capitolo della tesi esplora le principali tecnologie impiegate per la creazione di un Digital Twin urbano: Blender, Nvidia Sionna, Python e TensorFlow. Ciascuno di questi strumenti offre funzionalità fondamentali che, quando integrate con altri software, permettono di ottenere simulazioni dettagliate della copertura telefonica in ambienti urbani. L'uso combinato di Blender per la modellazione 3D, Nvidia Sionna per la simulazione delle onde elettromagnetiche, e le capacità computazionali di Python e TensorFlow, consente di ottenere risultati accurati e rappresentativi della realtà fisica. Relativamente al codice, è stato scritto ed eseguito su Jupyter Notebook.

2.1 Jupyter Notebook

Logo di Jupyter Notebook

2.1.1 Spiegazione

Jupyter Notebook è un'applicazione web che consente di creare e condividere documenti che integrano codice eseguibile, formule matematiche, testo descrittivo e visualizzazioni grafiche. Questa piattaforma è ampiamente utilizzata nel campo della data science, apprendimento automatico, e computing scientifico grazie alla sua interfaccia interattiva che supporta diversi linguaggi di programmazione, come ad esempio Python, R, Julia, e molti altri. [8]

2.1.2 Utilizzo nel Progetto

Jupyter Notebook è stato essenziale per la riuscita di questo progetto, in quanto ha permesso la scrittura, la divisione in celle e l'esecuzione ottimale del codice Python, integrandosi con il sistema operativo.

2.2 Blender 4.1

Logo di Blender

2.2.1 Spiegazione

Blender 4.1 è un software open source molto apprezzato per la sua versatilità nella creazione di contenuti 3D. Utilizzato in ambiti come grafica, animazione, realtà virtuale, simulazione e design, offre un ampio ventaglio di strumenti e funzionalità. Tra esse, spiccano la modellazione, il texturing, il rendering, l'animazione e la scultura digitale. La compatibilità con diversi formati di file e la possibilità di estendere le sue capacità tramite script Python, rende Blender facilmente integrabile nelle pipeline di produzione, rendendolo uno strumento ideale per professionisti e creativi. [1]

2.2.2 Utilizzo nel Progetto

Nel progetto, il primo passo per interpretare un contesto urbano partendo da un file OpenStreetMap (OSM) è l'utilizzo di Blender: i dati OSM vengono importati in Blender in modo tale da produrre una rappresentazione tridimensionale dell'area urbana o metropolitana. Tale processo permette di esportare un file Mitsuba XML.

Il file Mitsuba XML esportato da Blender viene impiegato come input per Nvidia Sionna, poiché include una descrizione dettagliata delle mesh degli edifici e dell'ambiente circostante, indispensabile per le simulazioni di propagazione del segnale. Prima di esportare è fondamentale che il modello 3D venga allineato al sistema di coordinate di Nvidia Sionna, garantendo così la coerenza delle simulazioni con le condizioni reali dell'area urbana.

Per facilitare la gestione di vari formati di file e l'esportazione, Blender dispone di diversi add-on tra cui Mitsuba Renderer [2] e Blosm [3]. Il primo consente l'esportazione di scene in formato Mitsuba XML, cruciale per l'integrazione con Nvidia Sionna, fornendo descrizioni precise delle mesh e dell'ambiente urbano; invece, l'add-on Blosm permette di importare dati da OpenStreetMap (OSM), creando modelli 3D urbani dettagliati, particolarmente utili per simulazioni di copertura del segnale.

Questa configurazione di strumenti e add-on ottimizza il flusso di lavoro per la modellazione e simulazione di scenari urbani complessi, assicurando risultati precisi e realistici.

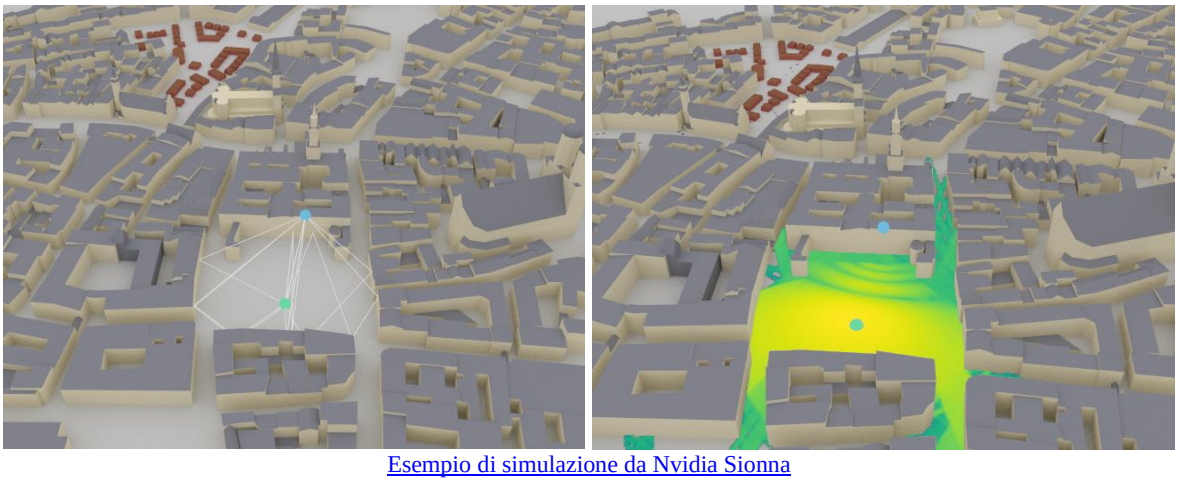
2.3 Nvidia Sionna

Logo Nvidia

2.3.1 Spiegazione

Nvidia Sionna è una libreria open source progettata per la simulazione e l'analisi delle reti wireless, costruita su TensorFlow e ottimizzata per l'uso su GPU. È uno strumento potente pensato per affrontare le sfide delle moderne reti di comunicazione, come 5G e 6G. Questa libreria consente simulazioni dettagliate della propagazione del segnale in ambienti complessi, fornendo agli ingegneri e ricercatori strumenti per progettare, pianificare e ottimizzare le reti wireless. Uno dei punti di forza di Sionna è la sua capacità di simulare scenari articolati grazie all'integrazione con tecnologie avanzate come il ray tracing e l'utilizzo di GPU, garantendo sia precisione che scalabilità nelle simulazioni:

1. **Ray Tracing**: Una delle tecnologie centrali di Sionna è il ray tracing, il quale consente di tracciare i percorsi delle onde radio attraverso l'ambiente, tenendo conto di ostacoli come edifici e altre strutture fisiche. Questo approccio non si limita alla simulazione del percorso diretto tra trasmettitori e ricevitori, ma include anche riflessioni, diffrazioni e rifrazioni delle onde radio, elementi fondamentali per ottenere simulazioni accurate, specialmente in contesti urbani complessi. [4]



Esempio di simulazione da Nvidia Sionna

2. **Integrazione con TensorFlow**: Grazie alla stretta integrazione con TensorFlow, Sionna può sfruttare la potenza del calcolo parallelo su GPU, essenziale per eseguire simulazioni su larga scala. Ciò consente di addestrare modelli di apprendimento automatico su set di dati di grandi dimensioni, rendendo Sionna un potente strumento per sviluppare soluzioni avanzate per l'ottimizzazione delle reti wireless. [4]

3. **Simulazione di scenari complessi**: Sionna è in grado di simulare una vasta gamma di scenari che

spaziano da ambienti urbani densi a spazi interni, fino ad aree rurali. La libreria consente la configurazione precisa dei parametri di antenne e ricevitori, inclusa la posizione, l'orientamento e il pattern di radiazione, ottenendo una modellazione realistica delle prestazioni di rete in diversi contesti. [4]

2.3.2 Utilizzo nel Progetto

Nel contesto del progetto, Nvidia Sionna viene utilizzata per simulare la propagazione del segnale in un ambiente urbano ricreato tramite un Digital Twin di un quartiere specifico. Dopo aver esportato la scena XML da Blender, le antenne trasmittenti (TX) e riceventi (RX) vengono posizionate nella scena utilizzando le coordinate fornite da un database [9], le quali vengono convertite per adattarsi al sistema di riferimento di Nvidia Sionna.

Importazione e Configurazione della Scena:
Il processo inizia con l'importazione della scena in formato XML all'interno di Nvidia Sionna. Questo file contiene non solo informazioni dettagliate sull'ambiente circostante, ma anche i mesh degli edifici. Una volta importata, la scena viene configurata aggiungendo e personalizzando i trasmettitori e i ricevitori, adattandoli ai requisiti specifici del modello di simulazione.

```
scene= load_scene(xml_path)
# Configure antenna array for all transmitters
scene.tx_array = PlanarArray(num_rows=8,
                         num_cols=2,
                         vertical_spacing=0.7,
                         horizontal_spacing=0.5,
                         pattern="tr38901",
                         polarization="VH")

# Configure antenna array for all receivers
scene.rx_array = PlanarArray(num_rows=1,
                         num_cols=1,
                         vertical_spacing=0.5,
                         horizontal_spacing=0.5,
                         pattern="dipole",
                         polarization="cross")

# Create transmitter
tx = Transmitter(name="tx",
              position=[8.5,21,27],
              orientation=[0,0,0])
scene.add(tx)

# Create a receiver
rx = Receiver(name="rx",
           position=[45,90,1.5],
           orientation=[0,0,0])
scene.add(rx)

# TX points towards RX
tx.look_at(rx)
```

Calcolo del percorso del segnale:
Una volta configurata la scena, il passo successivo prevede il calcolo dei percorsi del segnale, realizzato utilizzando il metodo *compute_paths*, il quale considera gli ostacoli fisici e le proprietà dei materiali presenti nella scena per determinare la modalità di propagazione delle onde radio.

```
# Compute propagation paths
paths = scene.compute_paths(max_depth=1,
                            num_samples=1e6)
```

Generazione della mappa di copertura:
Successivamente, viene generata una mappa di copertura che visualizza l'intensità del segnale, espressa in decibel, per ogni area della scena. La risoluzione della mappa dipende dalla complessità della simulazione e dalla dimensione dell'area definita, calcolata in metri quadrati. I dati ottenuti vengono quindi convertiti in formato CSV e utilizzati per generare un'immagine ad alta risoluzione della mappa di copertura.

```
cm = scene.coverage_map(cm_cell_size=[2.,2.],
                        num_samples=int(10e6))
# Salva il DataFrame come file CSV
df.to_csv(csv_path, index=False)
# Salva l'immagine in alta risoluzione senza la barra della legenda
plt.savefig('Coverage_map.png', dpi=600, bbox_inches='tight', pad_inches=0)
```

Descrizione del codice:
Il codice Python utilizzato nel progetto esegue diverse operazioni, dalla configurazione di TensorFlow alla creazione della scena con i trasmettitori e ricevitori, passando per il calcolo dei percorsi di propagazione e la generazione della mappa di copertura. TensorFlow viene configurato in modo da sfruttare la GPU per ottimizzare la memoria dinamicamente, mentre le coordinate dei trasmettitori e ricevitori sono importate dalla scena creata.
Le librerie Python come Matplotlib e Pandas sono utilizzate rispettivamente per la visualizzazione dei risultati e la gestione dei dati. Questo permette di produrre una rappresentazione dettagliata della copertura del segnale, utile per la pianificazione e l'ottimizzazione delle reti wireless. [10]

2.4 Python e Tensorflow


Logo Python e Logo Tensorflow

2.4.1 Spiegazione

Python è un linguaggio di programmazione di alto livello, noto per la sua semplicità e chiarezza. Queste qualità lo rendono ideale sia per principianti che per sviluppatori esperti, consentendo lo sviluppo di codice efficiente e facilmente comprensibile. Grazie alla sua sintassi intuitiva e alla vasta libreria standard, Python è ampiamente utilizzato in numerosi settori, come lo sviluppo web, l'analisi dei dati, l'automazione e l'intelligenza artificiale. Il supporto di Python per molte librerie esterne consente una rapida implementazione di soluzioni flessibili e complesse. [5] [10]
TensorFlow, una libreria open source di machine learning sviluppata da Google, è progettata per semplificare la creazione e l'addestramento di modelli di apprendimento automatico. Con la sua capacità di eseguire operazioni parallele su CPU e GPU, TensorFlow è ideale per gestire grandi quantità di dati e calcoli intensivi, rendendolo un componente essenziale per applicazioni che richiedono capacità predittive e analitiche avanzate. [6] [10]

2.4.2 Utilizzo nel Progetto

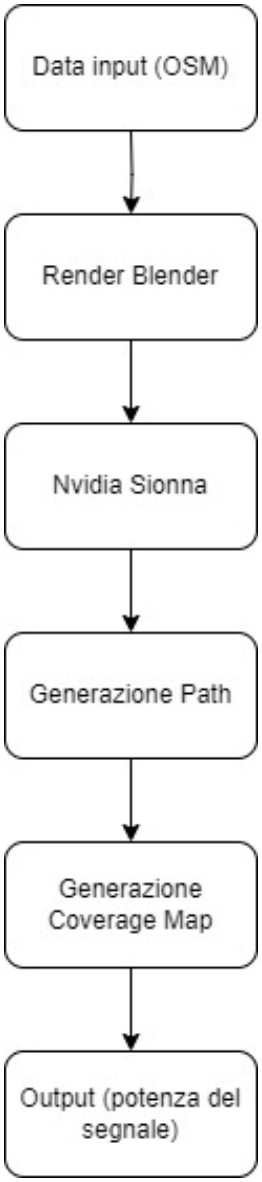
Nel contesto di questo progetto, Python e TensorFlow rivestono un ruolo cruciale poiché automatizzano e gestiscono processi articolati che altrimenti sarebbero difficili da eseguire manualmente. La combinazione della flessibilità di scripting di Python e delle funzionalità di apprendimento automatico di TensorFlow, sfruttata da Nvidia Sionna, consente di coordinare ed eseguire in modo efficiente le diverse fasi del progetto, dal calcolo alla simulazione della propagazione del segnale.
Python ha giocato un ruolo fondamentale nell'automazione del processo di integrazione tra Blender e Nvidia Sionna, semplificando operazioni che altrimenti richiederebbero molteplici passaggi complessi. Grazie alla sua flessibilità, Python ha permesso di collegare efficacemente i diversi componenti del progetto, facilitando il flusso di dati tra Blender e Sionna. Tramite script Python separati da celle, è stato possibile definire l'intero processo: dall'importazione dei dati della scena esportati da Blender in formato XML, alla creazione di antenne trasmittenti (TX) e riceventi (RX) per la simulazione del segnale, fino all'esportazione dei risultati per la successiva visualizzazione.
Questa suddivisione ha consentito una maggiore flessibilità, permettendo modifiche rapide alle configurazioni e adattamenti dinamici alle diverse esigenze del progetto. Grazie alla combinazione con TensorFlow, utilizzato per l'ottimizzazione del calcolo parallelo, l'integrazione di Python con Nvidia Sionna ha permesso di creare un ambiente di simulazione capace di gestire scenari complessi con risultati dettagliati e accurati in tempi relativamente brevi.

3 Descrizione dello svolgimento del progetto

Per il terzo capitolo della tesi è utile partire da uno schema a blocchi, il quale può efficacemente illustrare le diverse fasi del progetto, partendo dalla creazione di un modello 3D dell'ambiente urbano fino alla generazione e visualizzazione della mappa di copertura.

3.1 Schema a blocchi del progetto

Il progetto si articola in sei fasi:



3.2 Importazione dei dati da OpenStreetMap

Per importare dati da OpenStreetMap (OSM) in Blender 4.1, il primo passo consiste nella creazione di un modello 3D. OSM fornisce una vasta gamma di dati georeferenziati, come edifici, strade e altri elementi presenti sul territorio del comune di interesse. Utilizzando l'estensione Blosm [3], è possibile trasformare i dati forniti in modelli 3D che rappresentano edifici e altre strutture all'interno di Blender. Tale processo rende possibile la creazione di una mappa dettagliata dell'ambiente urbano desiderato, partendo dal formato OSM.

Per ottenere questo risultato, è necessario innanzitutto selezionare l'area di interesse, per dopo copiare le sue coordinate tramite il comando *Copy*.

Schermata per selezionare l'area di interesse

Un file OSM contiene informazioni geometriche come la disposizione degli edifici e la posizione delle strade, insieme a dei tag chiamati "Nodes", i quali sono punti georeferenziati definiti da coppie di coordinate (latitudine e longitudine).

3.3 Esportazione dei modelli con Blender

Il file OSM può essere importato in Blender facendo uso dell'estensione Blosm [3]. Attraverso la sezione "import", è possibile inserire la mappa direttamente nella scena e visualizzare in modo realistico le geometrie degli edifici. Durante questa fase, è importante configurare i materiali degli edifici al fine di assicurare la compatibilità con il modello di propagazione del segnale di Nvidia Sionna. Ciò richiede una precisa definizione delle proprietà fisiche dei materiali, come la riflessione e l'assorbimento delle onde radio. Al fine di realizzare questa configurazione, si fa riferimento alla documentazione Nvidia Sionna, in particolare alla sezione Radio Materials [7].

Material name	Real part of relative permittivity		Conductivity [S/m]		Frequency range (GHz)
	a	b	c	d	
vacuum	1	0	0	0	0.001 - 100
itu_concrete	5.24	0	0.0462	0.7822	1 - 100
itu_brick	3.91	0	0.0238	0.16	1 - 40
itu_plasterboard	2.73	0	0.0085	0.9395	1 - 100
itu_wood	1.99	0	0.0047	1.0718	0.001 - 100
itu_glass	6.31	0	0.0036	1.3394	0.1 - 100
	5.79	0	0.0004	1.658	220 - 450
itu_ceiling_board	1.48	0	0.0011	1.0750	1 - 100
	1.52	0	0.0029	1.029	220 - 450
itu_chipboard	2.58	0	0.0217	0.7800	1 - 100
itu_plywood	2.71	0	0.33	0	1 - 40
itu_marble	7.074	0	0.0055	0.9262	1 - 60
itu_floorboard	3.66	0	0.0044	1.3515	50 - 100
itu_metal	1	0	10^7	0	1 - 100
itu_very_dry_ground	3	0	0.00015	2.52	1 - 10
itu_medium_dry_ground	15	-0.1	0.035	1.63	1 - 10
itu_wet_ground	30	-0.4	0.15	1.30	1 - 10

Radio Materials

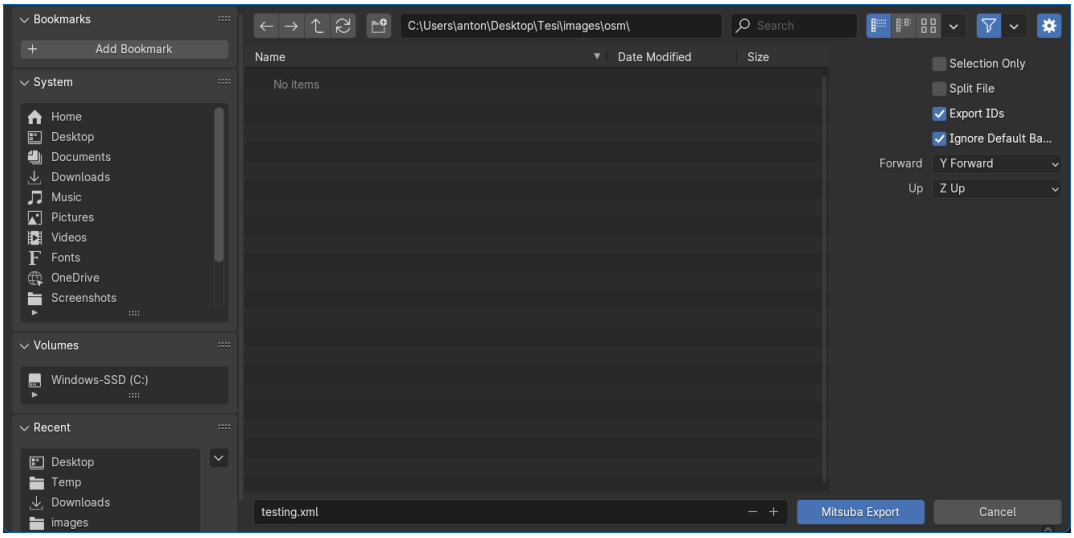
Ogni materiale è caratterizzato da un nome e da una serie di parametri che descrivono le sue proprietà fisiche, come la permettività, la conducibilità elettrica (espressa in Siemens per metro) e l'intervallo di frequenze a cui si applica tale conducibilità. Questi parametri sono essenziali per simulare il comportamento dei segnali elettromagnetici nel momento in cui attraversano diversi materiali, e risultano fondamentali per la progettazione e l'ottimizzazione di reti wireless. Nel contesto urbano di questa tesi, sono stati utilizzati tre materiali principali:

- *itu_concrete* per rappresentare il pavimento (Plane) della simulazione;
- *itu_marble* e *itu_metal* per definire rispettivamente le pareti e i tetti degli edifici.

Per il pavimento (Plane), è stata impiegata la funzione BSDF diffusa (Bidirectional Scattering Distribution Function) per simulare in modo realistico l'effetto della luce e delle riflessioni sulla superficie, rendendola opaca e più vicina alla realtà.
Per semplificare l'intero processo, è stato fondamentale l'impiego di vari componenti aggiuntivi per Blender, tra cui Blosm [3] per importare i dati OSM e Mitsuba Renderer [2] per esportare scene in formato Mitsuba XML, necessario per le simulazioni con Nvidia Sionna. Grazie a tali strumenti, è stato sviluppato un flusso di lavoro efficiente che ha consentito la modellazione, l'ottimizzazione e la simulazione accurata di scenari urbani complessi.
Dopo la fase di modellazione, la scena 3D viene esportata in un file Mitsuba XML, in seguito preparato per essere elaborato con Nvidia Sionna. Il file contiene una rappresentazione dettagliata delle mesh degli edifici e dell'ambiente circostante, indispensabile per le simulazioni di propagazione del segnale.
Durante l'esportazione, è stato essenziale assicurarsi che il modello fosse correttamente allineato e scalato rispettando il sistema di coordinate utilizzato da Nvidia Sionna, garantendo la compatibilità con le simulazioni successive.

3.4 Adattamento contesto e variabili per Nvidia Sionna

Nel primo passo della conversione, è necessario esportare il modello 3D da Blender in formato Mitsuba XML per l'utilizzo con Nvidia Sionna. Durante l'esportazione, è importante garantire che gli assi siano correttamente proporzionati. In Blender, l'asse Y rappresenta l'asse anteriore e Z l'asse in alto, quindi bisogna mantenere questa coerenza anche nel file Mitsuba XML per evitare discrepanze tra i sistemi di coordinate.



Schermata di esportazione del file Mitsuba XML

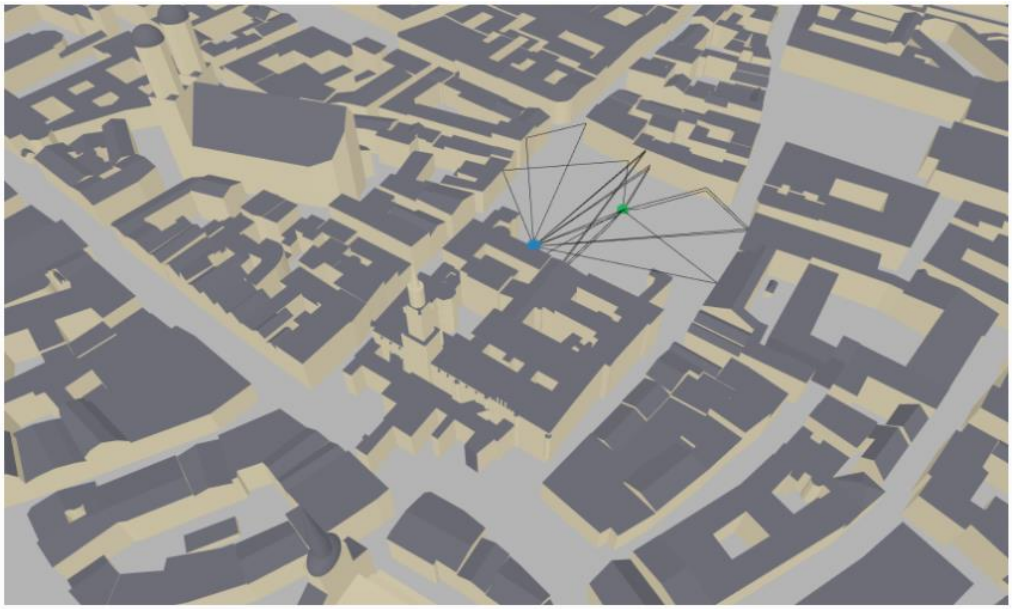
Blender utilizza i metri come unità di misura, e questo deve essere considerato durante l'esportazione per assicurare che la scala sia correttamente applicata nel modello. Inoltre, è cruciale abilitare l'opzione per l'esportazione degli ID degli edifici, in quanto di default è disabilitata. Questi ID sono essenziali per identificare e gestire le diverse strutture durante le simulazioni, permettendo un'elaborazione accurata nella propagazione del segnale all'interno dell'ambiente urbano modellato.

3.5 Generazione dei Path

Nel contesto di questo progetto il calcolo per la propagazione delle onde è fatto da Nvidia Sionna con l'uso di script Python dedicato. Le configurazioni di Tensorflow gestiscono la definizione della scena e simulano i percorsi di propagazione del segnale. Prima di eseguire i calcoli, Tensorflow viene impostato per utilizzare la GPU se disponibile, ottimizzando così le prestazioni della simulazione. Successivamente, viene richiamato il file XML generato da Blender per elaborare i calcoli. Ad inizio simulazione sono definite le posizioni per antenna trasmittente (TX) e ricevitore (RX), ricavate da un set di antenne già esistenti [9] realmente nella zona considerata.

```
scene.frequency = 2.14e9 # in Hz; implicitly updates RadioMaterials

scene.synthetic_array = True # If set to False, ray tracing will be done per antenna element (slower for large arrays)
```

Inoltre, le impostazioni descritte nello script sono fondamentali per configurare correttamente la simulazione di propagazione del segnale: impostare la frequenza permette di ottenere dati realistici e significativi in base alle peculiarità del segnale in esame, mentre l'uso dell'array sintetico ottimizza le prestazioni della simulazione, facilitando l'analisi dei dati relativi alla copertura e alla qualità del segnale in un ambiente urbano complesso.



Esempio visivo della generazione dei path

Il processo per la generazione del file CSV ricava i dati delle celle (coordinate) e i relativi valori di copertura, filtra i valori non validi (zero), sincronizza la lunghezza degli array e converte i valori di copertura in dB per una rappresentazione più standard. Successivamente, i dati vengono strutturati in un DataFrame per poi essere salvati in un file CSV, in cui sono contenute le coordinate 3D di ciascuna cella e i relativi valori di copertura, che possono essere utilizzati a loro volta per l'analisi della propagazione del segnale e la visualizzazione delle aree coperte dalla rete.

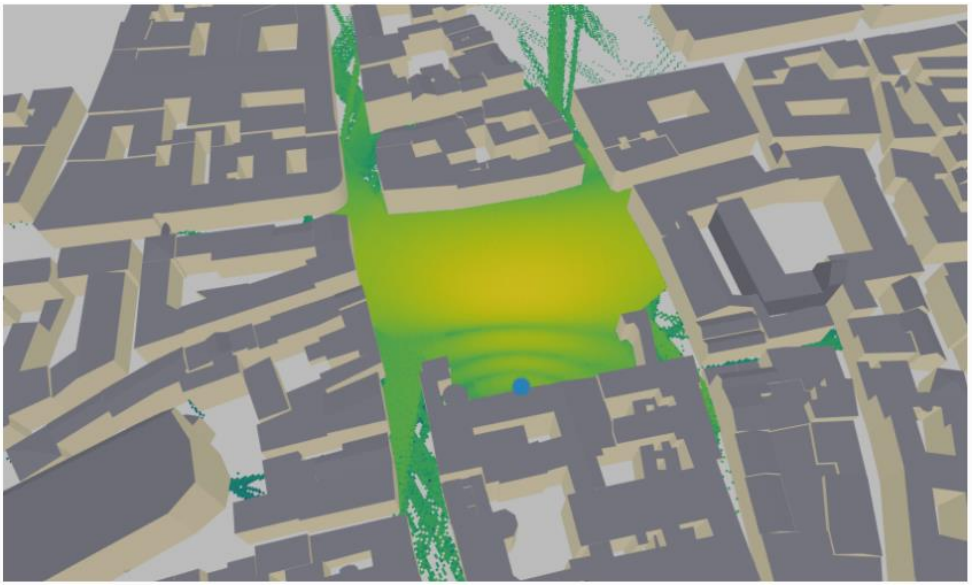
	A	B	C	D	E
1	index	x	y	z	coverage_db
2	0	-399.0	299.0	1.5	-95.6
3	1	-397.0	299.0	1.5	-105.1
4	2	-395.0	299.0	1.5	-116.7
5	3	-393.0	299.0	1.5	-98.9
6	4	-391.0	299.0	1.5	-98.9
7	5	-389.0	299.0	1.5	-98.9
8	6	-387.0	299.0	1.5	-99.0
9	7	-385.0	299.0	1.5	-95.8
10	8	-383.0	299.0	1.5	-148.2
11	9	-381.0	299.0	1.5	-97.5
12	10	-379.0	299.0	1.5	-99.6
13	11	-377.0	299.0	1.5	-99.7
14	12	-375.0	299.0	1.5	-99.6
15	13	-373.0	299.0	1.5	-96.9
16	14	-371.0	299.0	1.5	-137.5

Esempio di un file CSV estrapolato dal progetto

3.6 Generazione Coverage map e lettura dei dati

Grazie all'utilizzo di Nvidia Sionna, è possibile creare delle mappe di copertura raffiguranti la distribuzione della potenza del segnale in una determinata area, come in ambienti urbani. Queste coverage_map visualizzano la potenza del segnale espressa in decibel (dB), e utilizzano una scala cromatica per evidenziare zone con diversi livelli di copertura: ottima, discreta o scarsa. Tali mappe sono essenziali per valutare e ottimizzare le reti di telecomunicazione, poiché forniscono un quadro dettagliato delle prestazioni della rete nelle varie aree esaminate.

Sionna sfrutta la tecnica del ray tracing per simulare come le onde radio si propagano all'interno di un ambiente urbano complesso. Durante la simulazione, viene generata una mappa di copertura che rappresenta la potenza del segnale in dB per ogni punto dell'area analizzata. Al termine, la mappa viene esportata come immagine ad alta risoluzione (formato PNG), permettendo una facile analisi visiva dei risultati.



Esempio di coverage_map

Le coverage_map sono strumenti visivi che rappresentano la distribuzione della potenza del segnale in una determinata area, utilizzando colori diversi per indicare livelli di copertura che vanno da buoni a scarsi. La simulazione può essere visualizzata dall'alto per offrire una panoramica generale della copertura nella zona selezionata. Ciò viene realizzato posizionando una telecamera a un'altitudine di mille metri, in modo tale da permettere all'utente di osservare la distribuzione del segnale nell'area. Attraverso uno script dedicato, la visualizzazione dall'alto offre un'idea complessiva della copertura senza il bisogno di dover analizzare dettagli specifici punto per punto, ma fornendo una chiara comprensione del comportamento del segnale nell'ambiente urbano considerato. Di seguito viene riportato un esempio di coverage_map, tratto da Nvidia Sionna, dove:

- X rossa: posizione del ricevitore
- Giallo: Segnale forte (valori di path gain più alti, ad es. -70 dB)
- Verde: Segnale moderato (ad es. -80 dB)
- Blu: Segnale debole (ad es. -100 dB o inferiore)

```
bird_cam = Camera("birds_view", position=bird_pos, look_at=tx_pos)
scene.add(bird_cam)
scene.render(camera="birds_view", coverage_map=cm, num_samples=512, resolution=[1920,1080], show_paths=True)
```

Esempio di coverage_map generata da Nvidia Sionna

Con Nvidia Sionna, è possibile creare mappe ad alta risoluzione che illustrano come la potenza del segnale si propaga sul territorio, espressa in decibel (dB) per ciascun punto. Oltre all'immagine della mappa di copertura, viene generato un file CSV contenente i valori di potenza in decibel (dB) per ogni area, suddivisa in celle definite dall'utente basandosi sulla complessità della simulazione e alla risoluzione della griglia scelta. Questo file CSV fornisce una rappresentazione dettagliata della potenza del segnale per ogni cella, permettendo di consultare i valori corrispondenti a specifiche coordinate. Infatti, per interpretare l'output basta aprire il file CSV, individuare il valore di potenza alle coordinate desiderate e confrontarlo con i dati di misurazione reali.

4 Sperimentazione

All’interno della quarta fase del progetto, analizzeremo il funzionamento delle procedure sviluppate, illustrandone la finalità e il loro meccanismo. Successivamente, testeremo queste funzionalità in cinque differenti punti per valutarne l'efficacia.

4.1 Scenari di simulazione

Gli scenari scelti corrispondono a diversi punti chiave di una mappa, che in tale progetto si concentra sul centro di Trento. I luoghi presi in esame includono *Piazza Pasi, Residenza Galileo, Santa Maria Maggiore, Palazzo Quetta e Palazzo Tabarelli.* Si tratta di una zona urbana che presenta un’alta densità abitativa, caratterizzata da numerose stradine e edifici residenziali di media altezza. Le cinque simulazioni verranno eseguite una alla volta, illustrando i vari passaggi. Al termine delle simulazioni, secondo i parametri descritti nei capitoli precedenti, procederemo a valutare l'accuratezza del modello confrontando i dati di copertura ottenuti con le simulazioni e i dati reali rilevati negli stessi punti georeferenziati. L’obiettivo è analizzare le differenze tra le misurazioni reali e quelle simulate per ciascun punto.

Il rilevamento del segnale è stato compiuto utilizzando uno smartphone Android, nello specifico uno Xiaomi Redmi Note 11 Pro 5G. Infatti, tale dispositivo ha reso possibile monitorare l'intensità del segnale per mezzo di uno strumento integrato. L'impiego di questo stesso dispositivo, insieme alle antenne esistenti, ha consentito di raccogliere dati precisi che possono essere messi a confronto con quelli delle simulazioni, garantendo una convalida attendibile del modello.

Immagine tratta da Blender della porzione di mappa desiderata

Per tutte le simulazioni è stata estratta da OpenStreetMap la traccia OSM contenente i dati della zona di mappa selezionata, comprendente edifici e strade. La mappa ricopre il centro di Trento, più specificatamente è possibile notare a sinistra Via Rosmini, il Duomo in basso, in alto via Torre verde e Piazza Dante, ed a destra Piazza Venezia. Di seguito viene presentata l’immagine della zona desiderata con l’aggiunta della posizione delle antenne, raffigurate mediante dei puntini colorati.

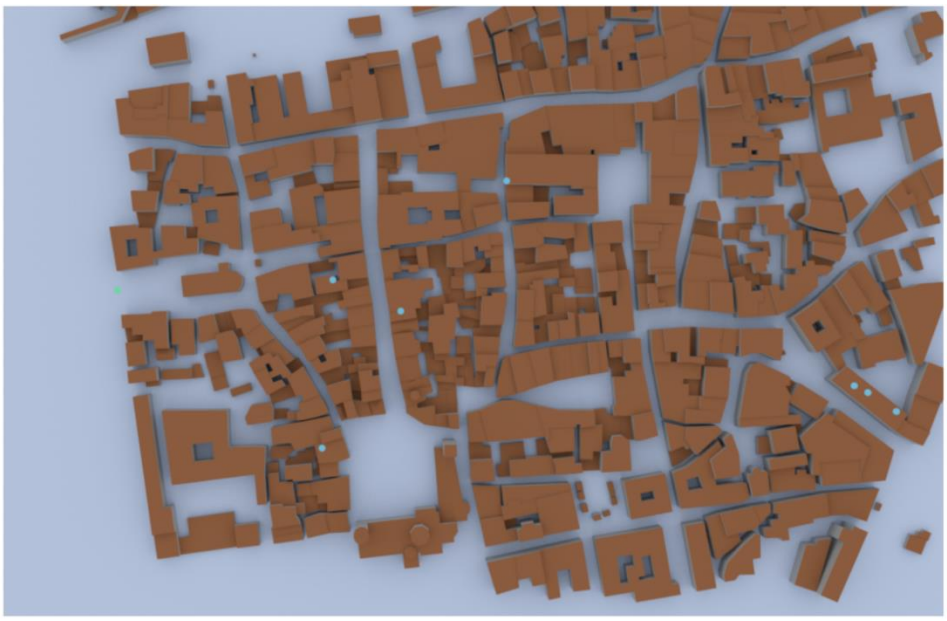


Immagine tratta da Sionna con le posizioni delle antenne

- *Punto 1: Piazza Pasi*

Come primo passo si seleziona la posizione del ricevitore, individuabile da un cerchio:

Immagine tratta da Blender, contenente la posizione del ricevitore

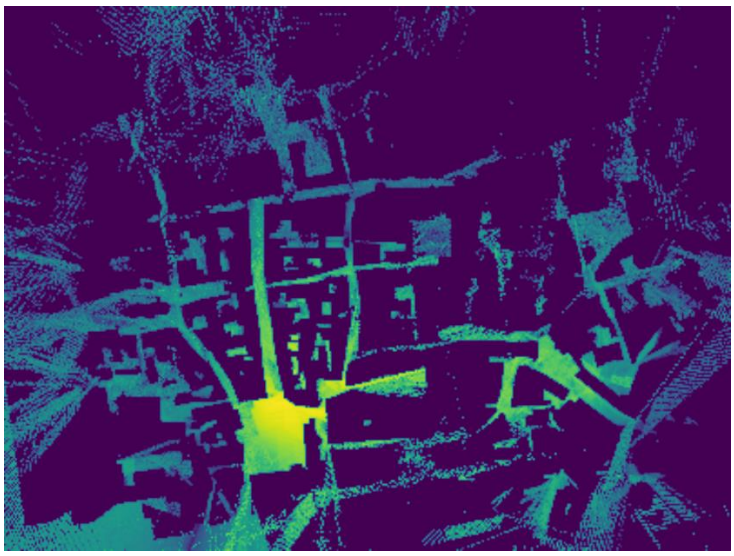
Una volta posizionato il ricevitore, è possibile proseguire con la compilazione dello script. Successivamente, tale comando genera il seguente output:

```
scene.render(camera="birds_view", num_samples=512, resolution=[1920,1080], show_paths=True)
```

Coverage map generate da Nvidia Sionna

Il file PNG viene invece generato con il comando:

```
# Salva l'immagine della mappa di copertura
coverage_data = cm._value.numpy()[0, :, :]  # Ottieni i dati della mappa di copertura

# Verifica i dati della mappa di copertura
print(f"Coverage data shape: {coverage_data.shape}")
print(f"Coverage data min: {coverage_data.min()}")
print(f"Coverage data max: {coverage_data.max()}")
print("Coverage data sample values:", coverage_data[0:5, 0:5])  # Stampa alcuni valori della matrice per verifica

# Configura i limiti della mappa di colori usando una scala logaritmica
vmin = np.log10(coverage_data[coverage_data > 0].min() + 1e-15)  # Usa il valore minimo effettivo
vmax = np.log10(coverage_data.max() + 1e-15)  # Usa il valore massimo effettivo

# Applicare la scala logaritmica
log_coverage_data = np.log10(coverage_data + 1e-15)  # Aggiungi un piccolo valore per evitare log(0)

# Salva l'immagine senza la barra della legenda
fig, ax = plt.subplots(figsize=(40, 40))  # Dimensione della figura aumentata per maggiore risoluzione
cax = ax.imshow(log_coverage_data, cmap='viridis', origin='lower', vmin=vmin, vmax=vmax, interpolation='bilinear')
ax.axis('off')  # Rimuovi gli assi

# Assicurati che la barra della legenda non influenzi l'immagine salvata
cbar = fig.colorbar(cax)
cbar.remove()

# Salva l'immagine in alta risoluzione senza la barra della legenda
plt.savefig('coverage_map.png', dpi=600, bbox_inches='tight', pad_inches=0)
plt.close()
```



Coverage map (PNG file) generate da Nvidia Sionna

La simulazione genera anche un file CSV, attraverso il quale è possibile recuperare il valore di intensità del segnale cercato.

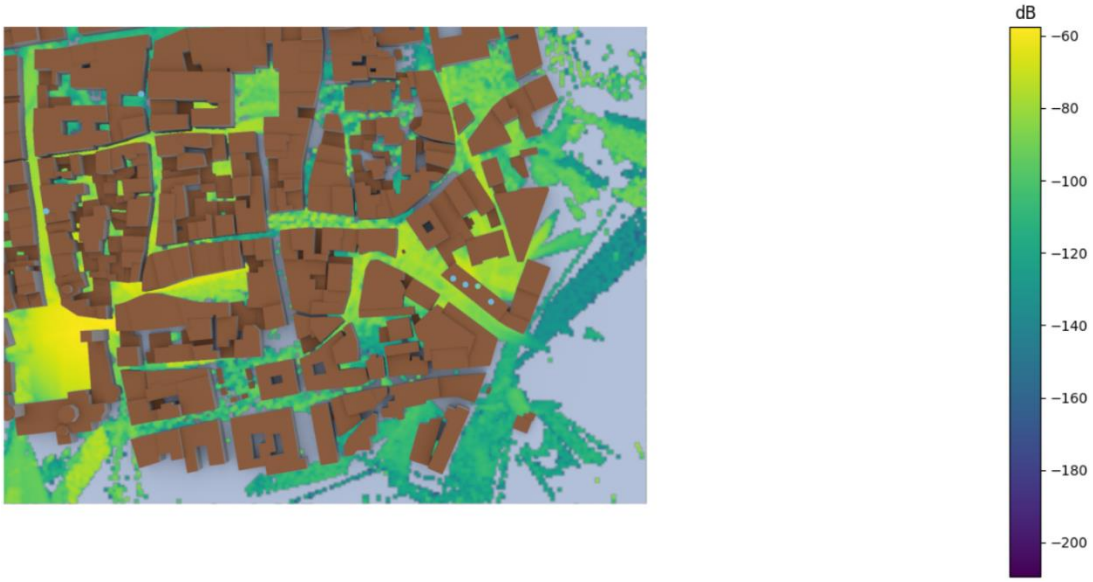
- *Punto 2: Residenza Galileo*

Come primo passo si seleziona la posizione del ricevitore, individuabile da un cerchio:

Immagine tratta da Blender, contenente la posizione del ricevitore

Una volta posizionato il ricevitore, è possibile proseguire con la compilazione dello script. Successivamente, tale comando genera il seguente output:

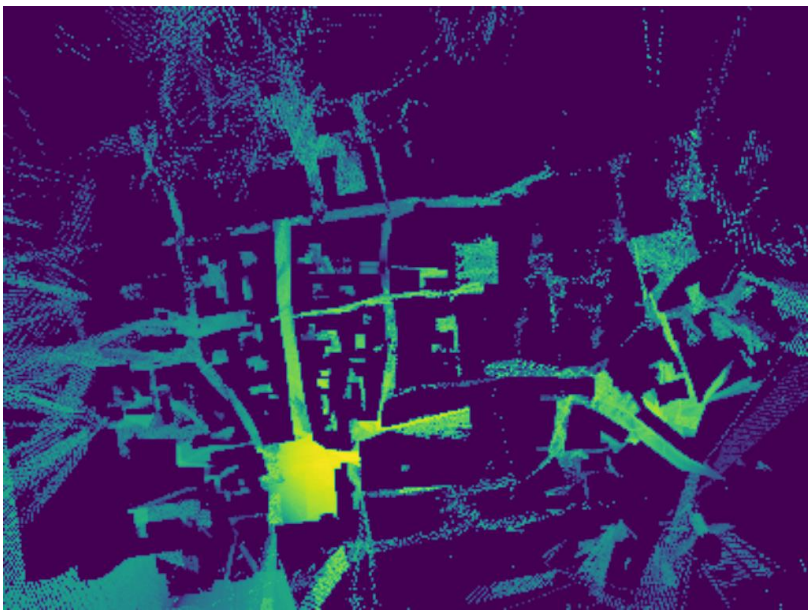
```
scene.render(camera="birds_view", num_samples=512, resolution=[1920,1080], show_paths=True)
```



Coverage map generate da Nvidia Sionna

Il file PNG viene invece generato con il comando:

```
# Salva l'immagine della mappa di copertura
coverage_data = cm._value.numpy()[0, :, :]  # Ottieni i dati della mappa di copertura

# Verifica i dati della mappa di copertura
print(f"Coverage data shape: {coverage_data.shape}")
print(f"Coverage data min: {coverage_data.min()}")
print(f"Coverage data max: {coverage_data.max()}")
print("Coverage data sample values:", coverage_data[0:5, 0:5])  # Stampa alcuni valori della matrice per verifica

# Configura i limiti della mappa di colori usando una scala logaritmica
vmin = np.log10(coverage_data[coverage_data > 0].min() + 1e-15)  # Usa il valore minimo effettivo
vmax = np.log10(coverage_data.max() + 1e-15)  # Usa il valore massimo effettivo

# Applicare la scala logaritmica
log_coverage_data = np.log10(coverage_data + 1e-15)  # Aggiungi un piccolo valore per evitare log(0)

# Salva l'immagine senza la barra della legenda
fig, ax = plt.subplots(figsize=(40, 40))  # Dimensione della figura aumentata per maggiore risoluzione
cax = ax.imshow(log_coverage_data, cmap='viridis', origin='lower', vmin=vmin, vmax=vmax, interpolation='bilinear')
ax.axis('off')  # Rimuovi gli assi

# Assicurati che la barra della legenda non influenzi l'immagine salvata
cbar = fig.colorbar(cax)
cbar.remove()

# Salva l'immagine in alta risoluzione senza la barra della legenda
plt.savefig('coverage_map.png', dpi=600, bbox_inches='tight', pad_inches=0)
plt.close()
```



Coverage map (PNG file) generate da Nvidia Sionna

La simulazione genera anche un file CSV, attraverso il quale è possibile recuperare il valore di intensità del segnale cercato.

- *Punto 3: Santa Maria Maggiore*

Come primo passo si seleziona la posizione del ricevitore, individuabile da un cerchio:

Immagine tratta da Blender, contenente la posizione del ricevitore

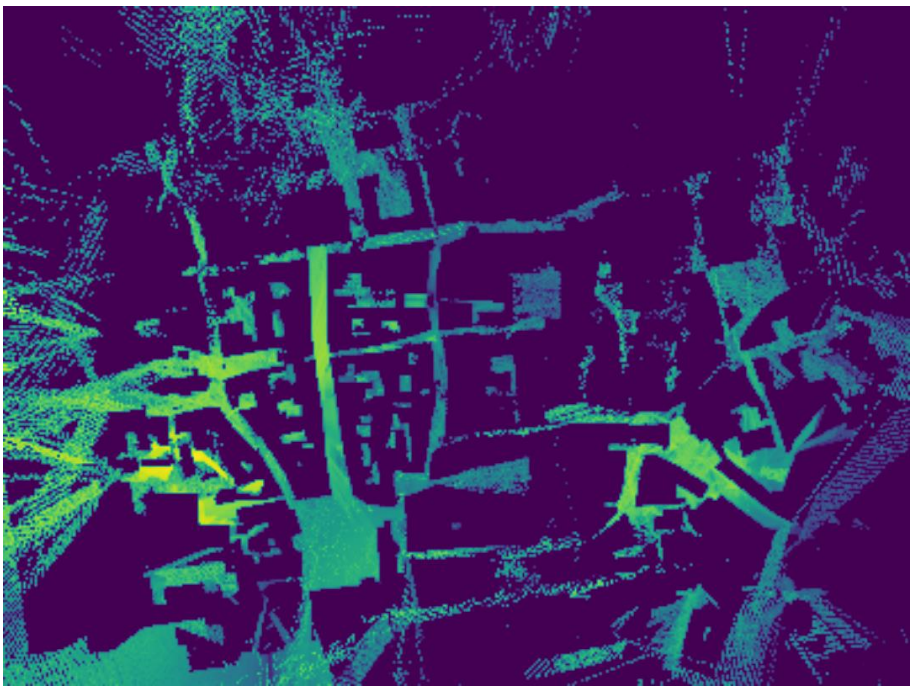
Una volta posizionato il ricevitore, è possibile proseguire con la compilazione dello script. Successivamente, tale comando genera il seguente output:

```
scene.render(camera="birds_view", num_samples=512, resolution=[1920,1080], show_paths=True)
```

Coverage map generate da Nvidia Sionna

Il file PNG viene invece generato con il comando:

```
# Salva l'immagine della mappa di copertura
coverage_data = cm._value.numpy()[0, :, :]  # Ottieni i dati della mappa di copertura

# Verifica i dati della mappa di copertura
print(f"Coverage data shape: {coverage_data.shape}")
print(f"Coverage data min: {coverage_data.min()}")
print(f"Coverage data max: {coverage_data.max()}")
print("Coverage data sample values:", coverage_data[0:5, 0:5])  # Stampa alcuni valori della matrice per verifica

# Configura i limiti della mappa di colori usando una scala logaritmica
vmin = np.log10(coverage_data[coverage_data > 0].min() + 1e-15)  # Usa il valore minimo effettivo
vmax = np.log10(coverage_data.max() + 1e-15)  # Usa il valore massimo effettivo

# Applicare la scala logaritmica
log_coverage_data = np.log10(coverage_data + 1e-15)  # Aggiungi un piccolo valore per evitare log(0)

# Salva l'immagine senza la barra della legenda
fig, ax = plt.subplots(figsize=(40, 40))  # Dimensione della figura aumentata per maggiore risoluzione
cax = ax.imshow(log_coverage_data, cmap='viridis', origin='lower', vmin=vmin, vmax=vmax, interpolation='bilinear')
ax.axis('off')  # Rimuovi gli assi

# Assicurati che la barra della legenda non influenzi l'immagine salvata
cbar = fig.colorbar(cax)
cbar.remove()

# Salva l'immagine in alta risoluzione senza la barra della legenda
plt.savefig('coverage_map.png', dpi=600, bbox_inches='tight', pad_inches=0)
plt.close()
```



Coverage map (PNG file) generate da Nvidia Sionna

La simulazione genera anche un file CSV, attraverso il quale è possibile recuperare il valore di intensità del segnale cercato.

- *Punto 4: Palazzo Quetta*

Come primo passo si seleziona la posizione del ricevitore, individuabile da un cerchio:

Immagine tratta da Blender, contenente la posizione del ricevitore

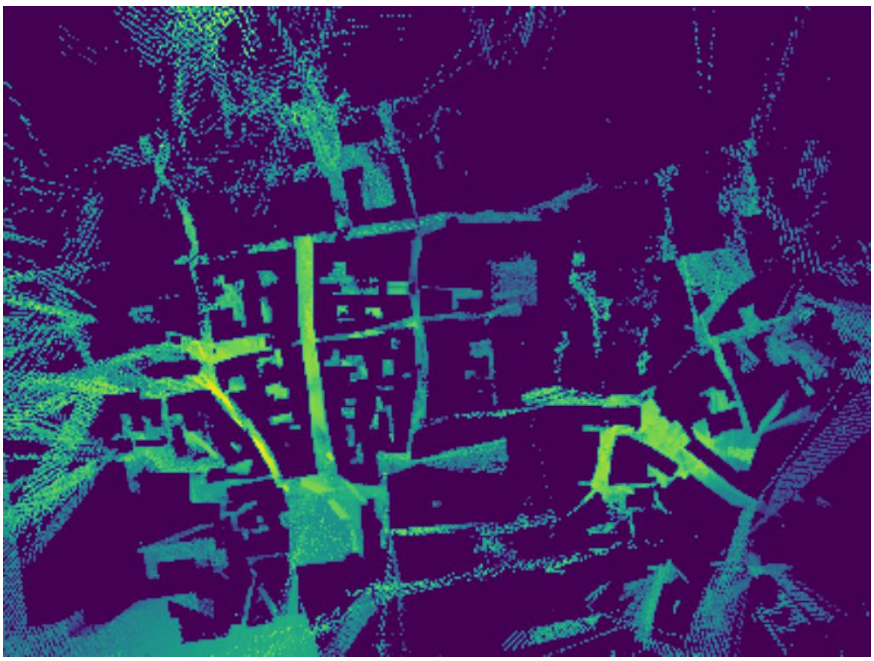
Una volta posizionato il ricevitore, è possibile proseguire con la compilazione dello script. Successivamente, tale comando genera il seguente output:

```
scene.render(camera="birds_view", num_samples=512, resolution=[1920,1080], show_paths=True)
```

Coverage map generate da Nvidia Sionna

Il file PNG viene invece generato con il comando:

```
# Salva l'immagine della mappa di copertura
coverage_data = cm._value.numpy()[0, :, :]  # Ottieni i dati della mappa di copertura

# Verifica i dati della mappa di copertura
print(f"Coverage data shape: {coverage_data.shape}")
print(f"Coverage data min: {coverage_data.min()}")
print(f"Coverage data max: {coverage_data.max()}")
print("Coverage data sample values:", coverage_data[0:5, 0:5])  # Stampa alcuni valori della matrice per verifica

# Configura i limiti della mappa di colori usando una scala logaritmica
vmin = np.log10(coverage_data[coverage_data > 0].min() + 1e-15)  # Usa il valore minimo effettivo
vmax = np.log10(coverage_data.max() + 1e-15)  # Usa il valore massimo effettivo

# Applicare la scala logaritmica
log_coverage_data = np.log10(coverage_data + 1e-15)  # Aggiungi un piccolo valore per evitare log(0)

# Salva l'immagine senza la barra della legenda
fig, ax = plt.subplots(figsize=(40, 40))  # Dimensione della figura aumentata per maggiore risoluzione
cax = ax.imshow(log_coverage_data, cmap='viridis', origin='lower', vmin=vmin, vmax=vmax, interpolation='bilinear')
ax.axis('off')  # Rimuovi gli assi

# Assicurati che la barra della legenda non influenzi l'immagine salvata
cbar = fig.colorbar(cax)
cbar.remove()

# Salva l'immagine in alta risoluzione senza la barra della legenda
plt.savefig('coverage_map.png', dpi=600, bbox_inches='tight', pad_inches=0)
plt.close()
```



Coverage map (PNG file) generate da Nvidia Sionna

La simulazione genera anche un file CSV, attraverso il quale è possibile recuperare il valore di intensità del segnale cercato.

- *Punto 5: Palazzo Tabarelli*

Come primo passo si seleziona la posizione del ricevitore, individuabile da un cerchio:

Immagine tratta da Blender, contenente la posizione del ricevitore

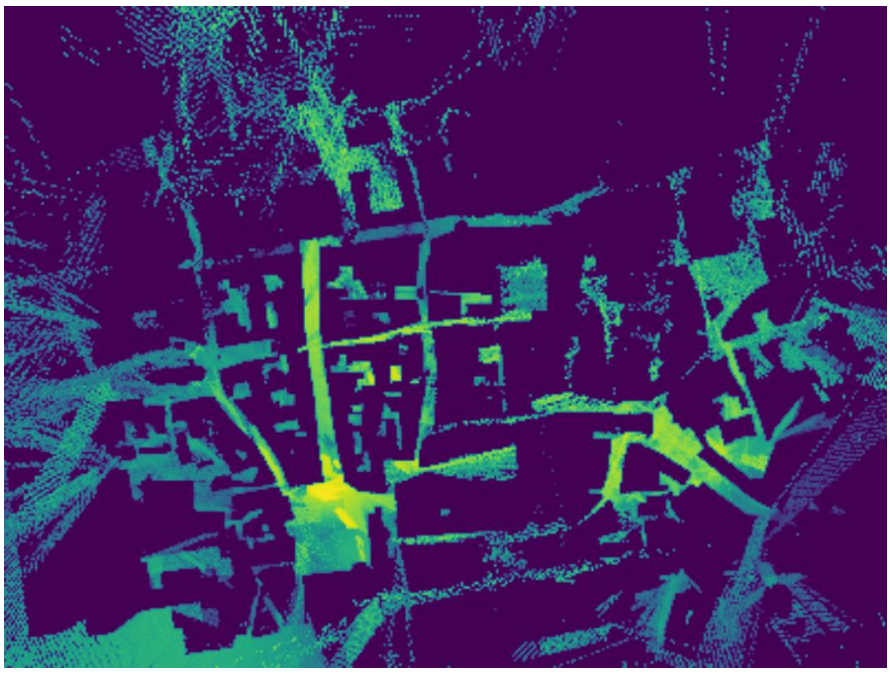
Una volta posizionato il ricevitore, è possibile proseguire con la compilazione dello script. Successivamente, tale comando genera il seguente output:

```
scene.render(camera="birds_view", num_samples=512, resolution=[1920,1080], show_paths=True)
```

Coverage map generate da Nvidia Sionna

Il file PNG viene invece generato con il comando:

```
# Salva l'immagine della mappa di copertura
coverage_data = cm._value.numpy()[0, :, :]  # Ottieni i dati della mappa di copertura

# Verifica i dati della mappa di copertura
print(f"Coverage data shape: {coverage_data.shape}")
print(f"Coverage data min: {coverage_data.min()}")
print(f"Coverage data max: {coverage_data.max()}")
print("Coverage data sample values:", coverage_data[0:5, 0:5])  # Stampa alcuni valori della matrice per verifica

# Configura i limiti della mappa di colori usando una scala logaritmica
vmin = np.log10(coverage_data[coverage_data > 0].min() + 1e-15)  # Usa il valore minimo effettivo
vmax = np.log10(coverage_data.max() + 1e-15)  # Usa il valore massimo effettivo

# Applicare la scala logaritmica
log_coverage_data = np.log10(coverage_data + 1e-15)  # Aggiungi un piccolo valore per evitare log(0)

# Salva l'immagine senza la barra della legenda
fig, ax = plt.subplots(figsize=(40, 40))  # Dimensione della figura aumentata per maggiore risoluzione
cax = ax.imshow(log_coverage_data, cmap='viridis', origin='lower', vmin=vmin, vmax=vmax, interpolation='bilinear')
ax.axis('off')  # Rimuovi gli assi

# Assicurati che la barra della legenda non influenzi l'immagine salvata
cbar = fig.colorbar(cax)
cbar.remove()

# Salva l'immagine in alta risoluzione senza la barra della legenda
plt.savefig('coverage_map.png', dpi=600, bbox_inches='tight', pad_inches=0)
plt.close()
```



Coverage map (PNG file) generate da Nvidia Sionna

La simulazione genera anche un file CSV, attraverso il quale è possibile recuperare il valore di intensità del segnale cercato.

4.2 Confronto dei dati e convalida del modello

La convalida del modello è un passaggio cruciale per assicurare la precisione e l'affidabilità delle simulazioni relative alla copertura del segnale. In questa fase del progetto, è necessario recarsi fisicamente nei luoghi dove sono stati effettuati i rilevamenti e confrontare i dati raccolti sul campo con quelli derivanti dalle simulazioni. Questo processo serve a verificare la correttezza del modello sviluppato e a identificare possibili differenze o aree che necessitano di miglioramenti. Prima di procedere con i rilevamenti, bisogna predisporre l'attrezzatura per misurare la potenza del segnale in vari punti dell'area oggetto di studio. Tale attrezzatura comprende dispositivi per la misurazione del segnale, strumenti GPS per garantire una geolocalizzazione precisa e dispositivi per la registrazione dei dati. Successivamente, si visitano i siti di interesse individuati per la simulazione (Piazza Pasi, Residenza Galileo, Santa Maria Maggiore, Palazzo Quetta e Palazzo Tabarelli) e raccolgono le misurazioni della potenza del segnale nei punti strategici identificati durante la fase di simulazione.

Il passaggio successivo si focalizza nel confrontare i dati rilevati sul campo con quelli ottenuti dalle simulazioni. Il confronto viene effettuato tramite software di analisi che consente di sovrapporre le misurazioni reali alle mappe di copertura generate. Le discrepanze tra i dati misurati e quelli simulati vengono analizzate in modo tale da individuarne le possibili cause; ed in base ai risultati dell'analisi, si può stabilire l'accuratezza del modello: infatti, i risultati della convalida sono fondamentali, poiché confermano l'affidabilità delle simulazioni e la loro capacità di rappresentare fedelmente la propagazione del segnale nei diversi contesti esaminati.

Dopo aver individuato i punti su cui eseguire le misurazioni, si procede con l'avvio della simulazione estraendo dai file CSV i risultati di interesse. Dopodiché tali dati vengono confrontati con i valori effettivi rilevati tramite il dispositivo Android. Parallelamente, bisogna calcolare la differenza tra i valori simulati e quelli misurati sul campo, un'analisi che permette di trarre considerazioni sulla precisione della simulazione e sulla sua affidabilità complessiva. Questa differenza rende possibile fare inferenza circa l'accuratezza della simulazione e il valore reale, modificato dal contesto del mondo reale come la vegetazione, persone e ostacoli non presenti nel modello 3D.

Posizione	Valore simulazione	Valore reale	Differenza
Piazza Pasi	-107	-92	15
Residenza Galileo	-93	-75	18
Santa Maria Maggiore	-113	-100	13
Palazzo Quetta	-105	-87	18
Palazzo Tabarelli	-114	-91	23

Tabella contenente i valori della simulazione ed i relativi valori reali

La tabella mostra una correlazione significativa tra i valori ottenuti dalla simulazione e quelli rilevati, suggerendo una buona precisione complessiva dei modelli. Tuttavia, la differenza all'incirca costante tra i due, con un valore massimo ben definito, indica la possibile presenza di fattori locali che influenzano i risultati, come la vegetazione, le condizioni atmosferiche e le persone stesse. Ciò implica che il rumore ambientale, nel nostro contesto, genera un contributo favorevole migliorando la qualità delle misurazioni rispetto a quella delle simulazioni.

In conclusione, il modello di simulazione, come Nvidia Sionna, dimostra di essere generalmente preciso, ma potrebbe necessitare di ottimizzazioni specifiche per rappresentare in modo più accurato alcune condizioni locali.

5 Conclusioni

Nell'ultimo capitolo della tesi verranno mostrati i risultati ottenuti, con annessa una riflessione riguardante i limiti ed i possibili miglioramenti futuri del progetto stesso.

5.1 Risultati ottenuti

I risultati delle simulazioni e delle misurazioni hanno dimostrato che, pur mantenendo una media di accuratezza accettabile per il modello, il segnale stimato è risultato più debole di ciò che è stato realmente misurato. Tali discrepanze suggeriscono la presenza di elementi locali che influenzano i risultati, nonostante la validità complessiva del modello.

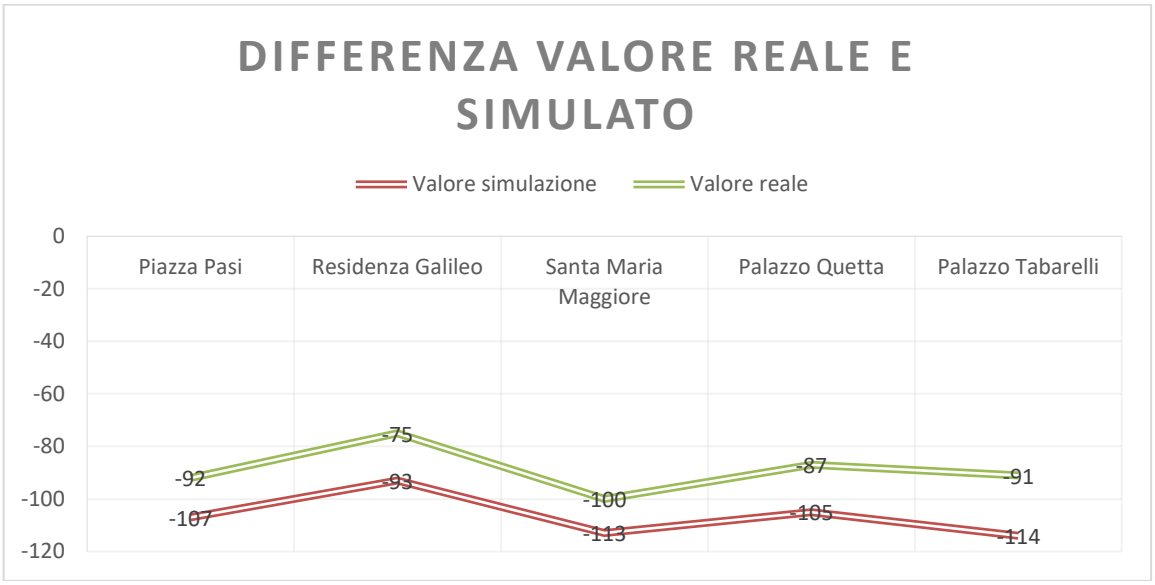


Grafico relativo l'andamento dei valori

Il grafico confronta i valori ottenuti dalla simulazione con quelli misurati in cinque punti diversi. Complessivamente si può osservare che sia i dati simulati che quelli misurati seguono lo stesso andamento, anche se sono presenti alcune discrepanze dovute a effetti locali non completamente catturati dai modelli di simulazione. Tali differenze, sebbene generalmente accettabili, indicano che le simulazioni tendono a sottostimare leggermente i livelli dell'intensità di segnale rispetto alle misurazioni reali. Al fine di aiutare a identificare ulteriori cause delle discrepanze e migliorare la precisione delle previsioni, una delle possibilità è aumentare il numero di misurazioni integrate nei modelli. Infatti, nonostante la buona accuratezza complessiva, ulteriori rilevazioni saranno necessarie per affinare le previsioni delle condizioni reali.

5.2 Limiti e possibili sviluppi futuri

Durante lo sviluppo del progetto sono apparsi dei limiti, i quali possono essere tramutati a loro volta in dei miglioramenti futuri per migliorare il progetto stesso. Il primo concerne la selezione e il posizionamento automatico delle antenne. Poiché non esiste un open data nazionale che includa le posizioni dettagliate delle antenne, attualmente rimane un'operazione manuale: una futura implementazione migliorerebbe la qualità del risultato, oltre a ridurre il tempo necessario alla sua importazione manuale. Inoltre, le simulazioni sono sempre avvenute in una casistica meteorologica neutrale, al contrario delle misurazioni reali che dipendono anche dalle condizioni atmosferiche: futuri sviluppi in tale ambito aumenterebbero la precisione dei risultati ed ottimizzerebbero la creazione del Digital Twin.

6 Bibliografia

[1] Wikipedia, "Blender", Wikipedia https://it.wikipedia.org/wiki/Blender_(programma) (ottobre 2024)

[2] Mitsuba Blender Plugin, GitHub Pages, https://github.com/mitsuba-renderer/mitsubablender?tab=readme-ov-file (ottobre 2024)

[3] Prochitecture, "Blosm Plugin for Blender", GitHub Pages, https://github.com/vvoovv/blosm (ottobre 2024)

[4] Nvidia, "Sionna: A GPU-accelerated library for link-level simulations of communication systems", GitHub Pages, https://nvlabs.github.io/sionna/ (ottobre 2024)

[5] Python, https://www.python.it/about/ (ottobre 2024)

[6] Databricks, “Tensorflow”, https://www.databricks.com/it/glossary/tensorflow-guide (ottobre 2024)

[7] Nvidia, "API Sionna: Radio Materials", Nvidia, https://nvlabs.github.io/sionna/api/rt.html?highlight=itu_marble#radio-materials

[8] Databricks, “Jupyter Notebook”, https://www.databricks.com/glossary/jupyter-notebook (ottobre 2024)

[9] Cellmapper, https://www.cellmapper.net (settembre 2024)

[10] https://www.ossolanews.it/2023/12/13/leggi-notizia/argomenti/digitale/articolo/linguaggio-python-perche-e-il-motore-dellintelligenza-artificiale-e-dellanalisi-dei-dati-1.html (ottobre 2024)