POLITECNICO DI TORINO

Ingegneria Elettrica

Tesi di laurea magistrale

Applicativo di realtà aumentata per il calcolo e la visualizzazione dei potenziali di terra

Relatori

Prof. Paolo DI LEO

Prof. Pietro COLELLA

Candidato

Alessandro BASILE

19 Marzo 2025

Ringraziamenti

Desidero aprire questi ringraziamenti con un pensiero profondo e sentito rivolto alla mia famiglia, che mi ha accompagnato instancabilmente lungo questo percorso accademico e di vita che mi ha condotto a questo traguardo. In modo particolare, vorrei ringraziare i miei genitori, che mi hanno trasmesso valori fondamentali come l'impegno, l'onestà e la determinazione, e che mi hanno sempre permesso di seguire le mie passioni e i miei interessi. Il loro incoraggiamento mi ha sostenuto nei momenti di sconforto, aiutandomi a raggiungere quest'obiettivo che più di una volta ho creduto troppo lontano e a tratti irragiungibile.
Un ringraziamento particolare desidero rivolgerlo anche a mia sorella, che fin dall'inizio mi ha supportato, interessandosi ai miei progressi e offrendomi quell'affetto fraterno che alleggerisce il carico di stress ricordandomi che ci sono obiettivi nella vita che vanno ben oltre i voti o i riconoscimenti.
Rivolgo un pensiero di gratitudine anche ai miei zii, che, in ogni occasione, si sono dimostrati presenti e sempre pronti a offrire il loro supporto. So di poter contare su di loro, non solo come figura familiare ma anche come punto di riferimento.
È grazie alla vicinanza della mia famiglia, che i risultati di questo lungo percorso di studi trovano un significato ancor più profondo.
Un pensiero di riconoscenza va anche ai miei compagni di università, con i quali ho condiviso lezioni, esami, fatiche, progetti e non poche risate. È stato anche grazie a loro che ho potuto vivere a pieno l'esperienza universitaria, fatta di collaborazione, stimoli reciproci e confronti costruttivi.
Un sincero ringraziamento è dovuto, infine, ai professori che mi hanno accompagnato lungo il cammino accademico. La loro competenza, la passione per l'insegnamento e la disponibilità a chiarire dubbi hanno alimentato la mia curiosità e consolidato la base di conoscenze necessaria per giungere a questo risultato. In particolare, esprimo la mia più grande riconoscenza ai miei relatori, il Prof. Paolo Di Leo e il Prof. Pietro Colella i quali, con gentilezza e comprensione, sono venuti incontro alle mie esigenze lavorative, guidandomi con attenzione, offrendomi mi spunti di riflessione e permettendomi di arrivare alla conclusione di questo lavoro.

A tutti voi, grazie di cuore.

"Alla mia famiglia"

Indice

Elenco delle figure

Elenco degli acronimi

- **AR**: Augmented Reality
- **IEC**: International Electrotechnical Commission
- **IEEE**: Institute of Electrical and Electronics Engineers
- **CEI**: Comitato Elettrotecnico Italiano
- **MaSM**: Maxwell Sub-area Method
- **API**: Application Programming Interface
- **FEM**: Finite Element Method
- **CAD**: Computer-Aided Design
- **GUI**: Graphical User Interface
- **VR**: Virtual Reality

Capitolo 1

Introduzione e contesto

1.1 Scenario e motivazioni

Negli ultimi anni la realtà aumentata (AR) ha conosciuto una notevole diffusione in svariati ambiti professionali e tecnici. Alla base della disseminazione così generalizzata di tale tecnologia vi è la possibilità di poter sovrapporre in tempo reale oggetti virtuali allo spazio fisico, percepito tramite la fotocamera di uno smartphone, di un tablet o di un visore. Tale potenzialità apre dunque nuove prospettive per la visualizzazione interattiva di dati e modelli matematici direttamente nell'ambiente reale. Parallelamente, in ambiti come l'ingegneria elettrica, è spesso necessario calcolare campi di potenziale elettrico e interpretare i risultati registrati, ottenuti a seguito di rilievi sperimentali condotti in campagne di misure ben coordinate. Operazione seguente è quindi l'esportazione e la successiva aggregazione di tali dati in formato grafico o tabellare. Tuttavia, queste rappresentazioni statiche, seppur rigorose, possono risultare di lettura meno immediata, specialmente quando si vogliono visualizzare i risultati in relazione diretta con la posizione fisica o con elementi del mondo reale.
Il lavoro di questa tesi si colloca pertanto nell'ottica di unire diverse componenti: l'analisi teorica e numerica di un impianto di terra, la validazione del metodo analitico per via sperimentale mediante prove in campo e la realizzazione di un applicativo che consenta di calcolare e mostrare il potenziale di terra in maniera intuitiva e interattiva. Lo sviluppo di tale applicativo, con particolare attenzione all'aspetto visuale e alla possibilità di impiegare la realtà aumentata, mira ad arricchire sia le potenzialità didattiche che quelle professionali di un modello

analitico, fornendo uno strumento che, in fase di progettazione o di esercitazione, possa immediatamente rendere comprensibili i risultati matematici.
Sono dunque emersi due obiettivi principali:

1. Verificare l'affidabilità del modello di calcolo, basato sul metodo delle sottoaree di Maxwell, confrontandolo con simulazioni numeriche più complete e con prove sperimentali, valutando scostamenti e limiti d'applicazione;

2. Sviluppare un'applicazione interattiva che integri la parte di calcolo e la visualizzazione immersiva, consentendo di configurare le caratteristiche del sistema (geometria, parametri fisici, correnti) e di generare dinamicamente mappe di potenziale in modalità 2D o 3D, con particolare apertura alla possibilità di impiegare la realtà aumentata come strumento didattico e di supporto operativo;

Il primo obiettivo riflette l'esigenza di sapere in quali condizioni il modello risulta coerente con la realtà fisica, verificando se e quanto le ipotesi fondanti possano influire. Il secondo obiettivo, invece, sottolinea l'intento di creare un ambiente software moderno, dotato di un'interfaccia che non si limiti a restituire dati numerici, ma che favorisca la comprensione di un fenomeno fisico spesso ritenuto astratto, mediante rappresentazioni grafiche convincenti, ancorate al mondo reale grazie alle tecniche di realtà aumentata.

1.2 Importanza degli impianti di terra

Gli impianti di terra svolgono un ruolo fondamentale nella sicurezza e nell'affidabilità di qualunque installazione elettrica, al fine di garantire una gestione controllata delle correnti di guasto che possono insorgere durante il lungo periodo d'esercizio. Essi infatti sono preposti a convogliare tali sovracorrenti nel terreno, con l'obiettivo di ridurre eventuali rischi per persone ed apparecchiature garantendo, al tempo stesso, limitazioni a possibili interruzioni di servizio.
Nell'ambito dell'ingegneria elettrica, l'analisi degli impianti di terra non si limita semplicemente alla valutazione numerica della resistenza di terra, ma coinvolge lo studio approfondito della distribuzione dei potenziali superficiali e sub-superficiali che si instaurano quando l'impianto è percorso da correnti di guasto. Questo

aspetto risulta cruciale, dato che il potenziale elettrico nel suolo determina, in ultima istanza, la tensione a cui possono trovarsi persone o strutture metalliche in contatto con il terreno; ne consegue che, se la curva o la mappa dei potenziali dovesse assumere valori troppo elevati in prossimità di aree accessibili, si costituirebbe un serio pericolo per l'incolumità di operatori e utenti. L'importanza di un buon sistema di terra è pertanto riconosciuta a livello normativo, attraverso standard internazionali (IEC, IEEE) e nazionali (CEI) che stabiliscono metodologie e limiti di accettabilità. In particolare la normativa pone l'attezione sulla determinazione dei valori di tensione di passo e contatto, prescrivendo valori massimi oltre i quali si ritiene che il rischio di elettrocuzione diventi eccessivo.

- La **tensione di passo** è la differenza di potenziale che può presentarsi tra i due piedi di un individuo che cammina o si ferma nelle vicinanze del dispersore, quando il suolo è attraversato da una corrente di guasto. Questa tensione si misura fra due punti a distanza mediamente pari al passo umano, fissata tipicamente ad 1 metro;

- La **tensione di contatto** è invece la differenza di potenziale fra una struttura metallica (ad esempio il telaio di una macchina elettrica) e il terreno. Se una persona tocca tale struttura mentre i piedi poggiano a terra, la tensione di contatto è determinante per valutare il rischio di shock elettrico. In condizioni di guasto infatti, o con un impianto di terra inadeguato, la struttura può assumere potenziali sensibili rispetto al terreno, determinando un concreto pericolo di elettrocuzione;

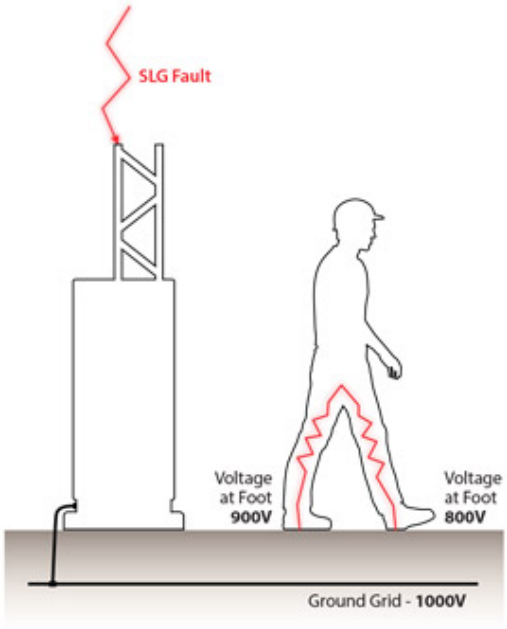

Figura 1.1: Tensione di passo

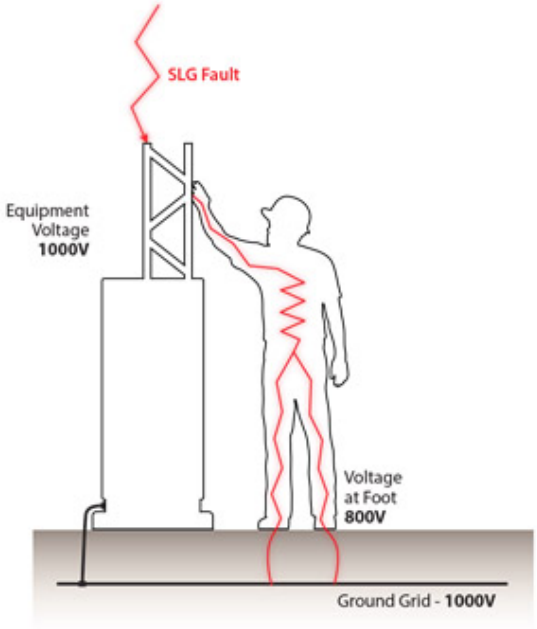

Figura 1.2: Tensione di contatto

Alla luce di tali considerazioni diviene pertanto evidente come la distribuzione di potenziale, intesa come mappatura dell'area di interesse e dei relativi potenziali in gioco, assuma un rilievo operativo imprescindibile nella progettazione e verifica di un impianto di terra rispondente allo stato dell'arte. Ogni miglioramento degli strumenti analitici a disposizione può quindi tradursi nella capacità di poter individuare potenziali situazioni di pericolo, consentendo di intervenire al fine di garantire il livello di sicurezza più cautelativo possibile.

Capitolo 2

Fondamenti teorici

2.1 Il metodo delle sottoaree di Maxwell

L'analisi del potenziale elettrico in un terreno conduttore rappresenta un tema che ha accompagnato sin dall'origine lo sviluppo dell'elettrotecnica. James Clerk Maxwell, nella seconda metà dell'800, sviluppò le basi teoriche dell'elettromagnetismo, introducendo, per la prima volta, metodi analitici per il calcolo di campi elettrici e magnetici in sistemi di geometria elementare. Da questi studi ha avuto origine il **metodo delle sottoaree** (MaSM), che si pone l'obiettivo di studiare sistemi in conduzione stazionaria, valutando la distribuzione di corrente in conduttori immersi in un mezzo di resistività nota. Il risultato è una tecnica sufficientemente accurata per geometrie composte da segmenti di conduttori (ad es. picchetti cilindrici), che conserva il vantaggio dell'approccio circuitale e che ben si presta all'implementazione di una risoluzione affidata al calcolatore.

2.1.1 Calcolo della matrice delle resistenze

Il metodo parte dal principio di suddividere il conduttore in un numero n di segmenti lineari, detti appunto "sottoaree" o "sottoelementi." Ogni segmento, di lunghezza l, viene considerato come un'unità elementare che può iniettare una quota di corrente nel terreno, in maniera indipendente dagli altri segmenti.

Ciascun elemento è caratterizzato da:

- Una propria **auto-resistenza** R_{ii}, che rappresenta la resistenza di un segmento verso l'ambiente (il suolo);
- Un insieme di **mutue resistenze** R_{ij}, ovvero la "resistenza di influenza" tra il segmento R_{ii} e il segmento R_{ij};

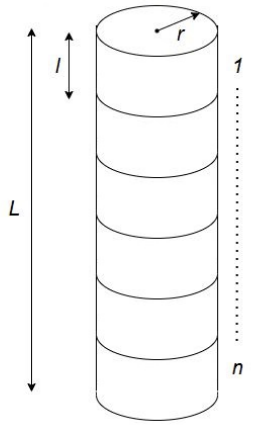

Figura 2.1: Picchetto suddiviso in sottoaree

Una volta ottenuta la discretizzazione dell'elemento disperdente in sottoaree e individuato il numero di segmenti, è possibile procedere con la determinazione dei coefficienti di auto e mutua resistenza, a partire dalla seguente formulazione:

$$V_P = \int_{-\frac{L}{2}}^{\frac{L}{2}} \frac{\rho I \, d\xi}{4\pi\sqrt{(x-\xi)^2 + y^2 + z^2}} \tag{2.1}$$

dove il potenziale nel punto $P(x, y, z)$ è calcolato per sovrapposizione degli effetti assumendo un elettrodo cilindrico equipotenziale immerso in un mezzo conduttivo. In ultima istanza, è possibile calcolare i suddetti coefficienti resistivi dividendo la 2.1 per la corrente di guasto iniettata nel terreno:

$$R(x,y,z) = \frac{\rho}{4\pi L} \ln \left[\frac{x + \frac{L}{2} + \sqrt{z^2 + y^2 + \left(x + \frac{L}{2}\right)^2}}{x - \frac{L}{2} + \sqrt{z^2 + y^2 + \left(x - \frac{L}{2}\right)^2}} \right] \tag{2.2}$$

Una volta determinati i contributi di resistenza si forma la matrice:

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1n} \\ R_{21} & R_{22} & \cdots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ R_{n1} & R_{n2} & \cdots & R_{nn} \end{bmatrix}.$$

dove gli elementi sulla diagonale R_{ii} sono rappresentativi delle auto-resistenze e quelli fuori diagonale R_{ij} individuano i contributi mutui. In un caso fisicamente coerente la matrice risulta simmetrica ($R_{ii} = R_{ij}$), poichè la resistenza reciproca tra due segmenti è identica in entrambe le direzioni.

2.1.2 Calcolo del potenziale

Un aspetto particolarmente potente del metodo delle sottoaree di Maxwell risiede nell'adozione di un'interpretazione circuitale, che permette la traduzione del problema dalla sua complessità fisica ad una forma circuitale più tradizionale, costituita da **nodi** e **rami**. Di seguito il circuito equivalente di ciascuna sottoarea:

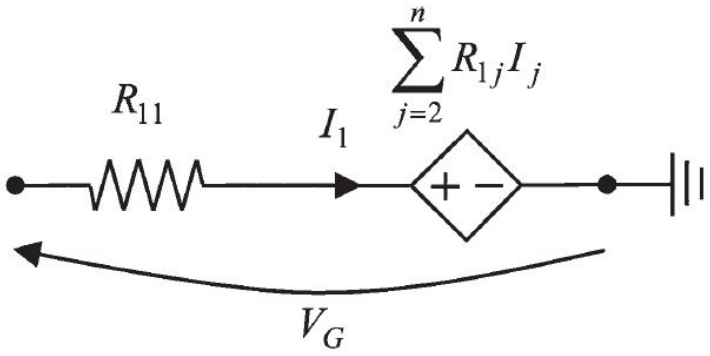

Figura 2.2: Circuito equivalente di una sottoarea

Operazione successiva alla costruzione della suddetta matrice di dimensioni nxn, interpretata come una matrice di impedenza, è pertanto l'assemblaggio del sistema lineare volto alla determinazione dei valori puntuali di potenziale, calcolati su una griglia di punti costituente l'area di interesse. Si può quindi scrivere l'equazione in forma matriciale compatta:

$$[V] = [R][I] \tag{2.3}$$

in cui:

- $[V]$ è il vettore colonna dei potenziali assunti verso l'infinito da ciascuna sottoarea;
- $[R]$ è la matrice quadrata dei coefficienti di potenziale proprio e mutuo;
- $[I]$ è il vettore colonna delle correnti scambiate con il terreno da ciascuna sottoarea;

Esplicitando i termini matriciali si giunge alla seguente formulazione:

$$\begin{bmatrix} V_G \\ V_G \\ V_G \\ \vdots \\ V_G \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1n} \\ R_{21} & R_{22} & R_{23} & \cdots & R_{2n} \\ R_{31} & R_{32} & R_{33} & \cdots & R_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{n1} & R_{n2} & R_{n3} & \cdots & R_{nn} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_n \end{bmatrix} \tag{2.4}$$

Si ottengono quindi le n equazioni nelle n incognite costituite dalle n correnti parziali I_i e dai n potenziali incogniti V_i degli elementi in cui sono stati suddivisi gli elettrodi. Il sistema diviene determinato aggiungendo le seguenti equazioni ausiliarie che descrivono le condizioni fisiche del sistema:

- bilancio delle correnti:
$$\sum_{i=1}^{n} I_i = I_G; \tag{2.5}$$
- equipotenzialità degli elettrodi:
$$V_i = V_G; \tag{2.6}$$

Risolto il sistema, sono noti i singoli valori di corrente in ogni elemento dei dispersori e i loro potenziali rispettivamente assunti. A partire da queste grandezze, si può ricavare il potenziale indotto in un qualsiasi punto della superficie del terreno, mediante applicazione del principio di sovrapposizione degli effetti e per mezzo della formulazione:

$$V_P = \frac{I\rho}{4\pi L} \ln \left[\frac{x + \frac{L}{2} + \sqrt{z^2 + y^2 + \left(x + \frac{L}{2}\right)^2}}{x - \frac{L}{2} + \sqrt{z^2 + y^2 + \left(x - \frac{L}{2}\right)^2}} \right] \tag{2.7}$$

Capitolo 3

Validazione del metodo

La procedura di validazione di un metodo numerico, come il metodo delle sottoaree di Maxwell, rappresenta un passaggio di imprescindibile necessità quando si intende proporre un modello che introduce semplificazioni e idealità nella rappresentazione di un fenomeno fisico. In casi come questo diviene quindi di fondamentale importanza l'accertamento della rispondenza scientifica del modello alla realtà.
Questo capitolo si sofferma proprio sulle forme e le modalità adottate per la verifica dei risultati, evidenziando la configurazione del setup di simulazione, le differenze di risultati e, in ultima analisi, la bontà del metodo in condizioni note, prevedendo il confronto con un modello agli elementi finiti prima e, in seguito, con risultati strumentali ottenuti per via sperimentale di uno scenario realmente implementato.

3.1 Confronto con modello agli elementi finiti

Per condurre la simulazione agli elementi finiti (FEM) si è optato per **COMSOL Multiphysics**. La simulazione in esame mira a valutare la distribuzione di potenziale elettrico generata da picchetti interrati sulla superficie di un terreno omogeneo di resistività assegnata. All'interno del software si è impostata un'analisi di conduzione in regime stazionario. L'obiettivo finale è stato quello di calcolare i valori puntuali di potenziale sulla superficie del terreno simulato per poi confrontarli con il metodo delle sottoaree di Maxwell.

3.1.1 Creazione del modello geometrico

Operazione preliminare per l'analisi del sistema è stata la definizione del problema dal punto di vista geometrico, per la quale si è resa necessaria la modellazione dei due dispersori, nonchè la porzione di suolo in cui essi risultano "annegati". L'intera modellazione 3D delle parti del sistema è avvenuta all'interno di COMSOL, con l'ausilio dell'interfaccia nativa di stampo CAD, messa a disposizione dallo stesso software. Punto di partenza è stato la modellazione degli elettrodi, assimilati a cilindri, realizzati con i parametri geometrici di seguito riportati:

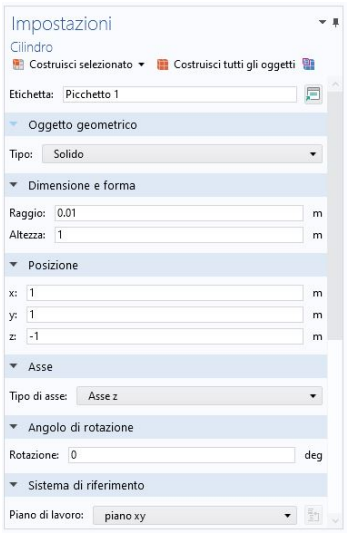

Figura 3.1: Picchetto 1

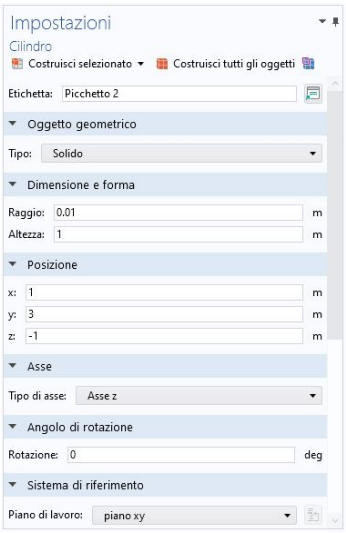

Figura 3.2: Picchetto 2

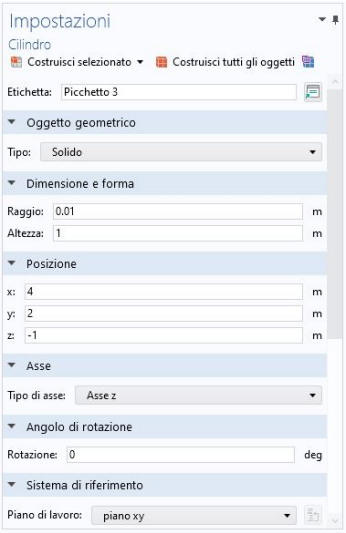

Figura 3.3: Picchetto 3

Successivamente è stato modellato il dominio del suolo ospitante gli elementi disperdenti, approssimato da una semisfera di raggio pari a 50 m, in modo che a questa distanza il potenziale possa considerarsi pressochè nullo. All'esterno di tale solido è stato previsto un bordo di spessore unitario, necessario per la definizione delle condizioni al contorno e indispensabili alla simulazione.

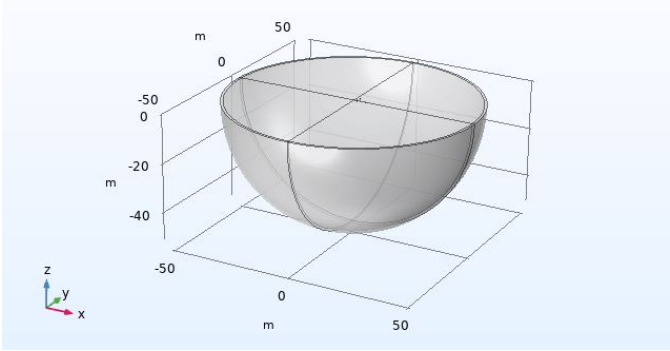

Figura 3.4: Dominio del suolo

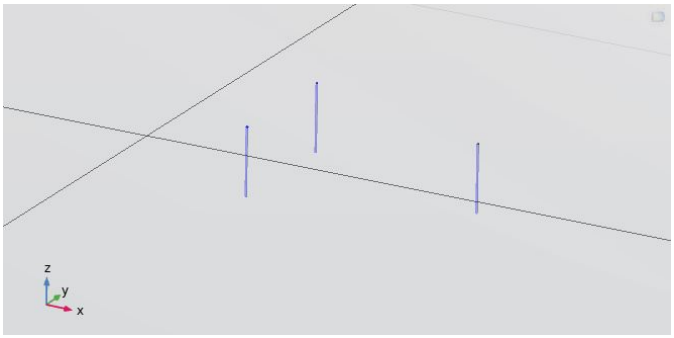

Figura 3.5: Disposizione picchetti

3.1.2 Impostazione della fisica

Delineata la geometria del sistema ed i volumi relativi a picchetti e suolo rispettivamente è stata la volta dell'impostazione dei parametri fisici e delle condizioni al contorno, fondamentali per la risoluzione del problema di conduzione stazionaria in esame. Di primaria necessità è stata l'assegnazione dei materiali più opportuni ai domini di calcolo, scegliendo il rame per i picchetti ed un materiale ad hoc per il terreno, caratterizzato dal seguente valore di conducibilità elettrica:

$$\sigma = 0.01 S/m \quad ovvero \quad \rho = 100 \Omega \cdot m$$

Successivamente sono state imposte le condizioni al contorno, imponendo dei potenziali fluttuanti in corrispondenza delle teste dei picchetti e imponendo la

condizione di potenziale nullo sul bordo esterno del dominio emisferico.
Nello specifico è stato impostato un valore di corrente dispersa pari a 0.5 A per ciascun picchetto attivo e una corrente di -1 A per il picchetto del percorso di ritorno. Di seguito i constraints di simulazione assegnati:

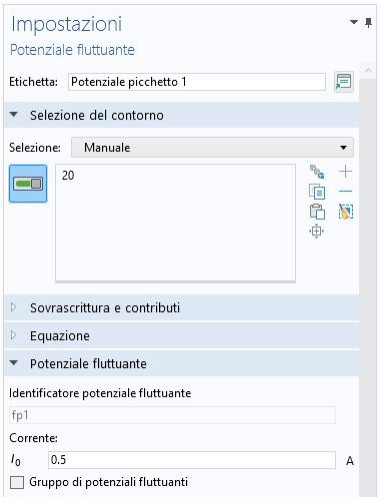

Figura 3.6: Condizione picchetto 1

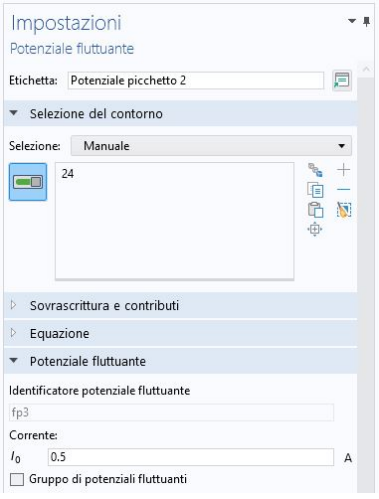

Figura 3.7: Condizione picchetto 2

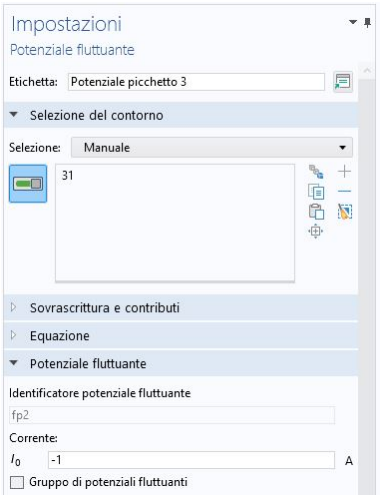

Figura 3.8: Condizione picchetto 3

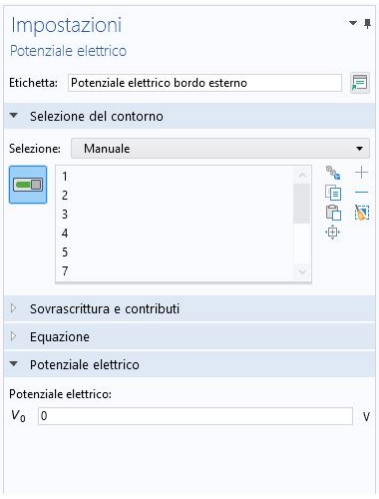

Figura 3.9: Condizione bordo esterno

3.1.3 Generazione della mesh

Alla base dei meccanismi adottati dai solver dei software FEM vi è la generazione della cosiddetta "mesh", ovvero l'insieme delle geometrie primitive che consentono la valutazione di un dominio nel passaggio da un modello continuo ad un modello discretizzato (elementi finiti), descrivendone il comportamento fisico-matematico mediante sistemi di equazioni differenziali. Per la generazione della mesh COMSOL adotta una discretizzazione poligonale, provvedendo automaticamente ad eventuali infittimenti con poligoni a superficie più ridotta in corrispondenza di punti in cui si prevede un maggior gradiente di variazione della grandezza analizzata. Di seguito la rappresentazione della mesh utilizzata:

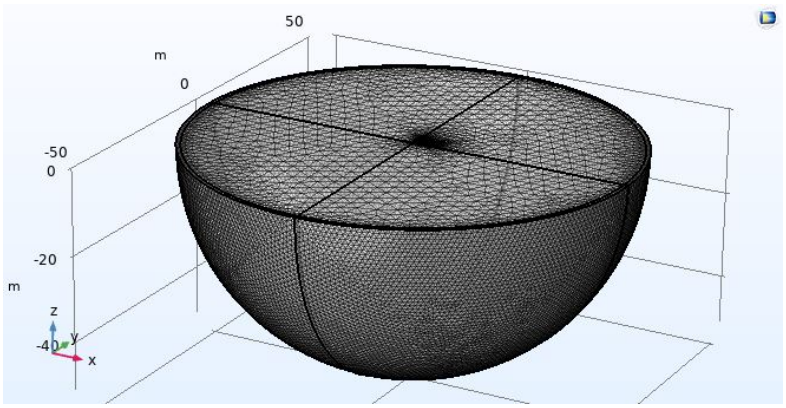

Figura 3.10: Mesh di calcolo

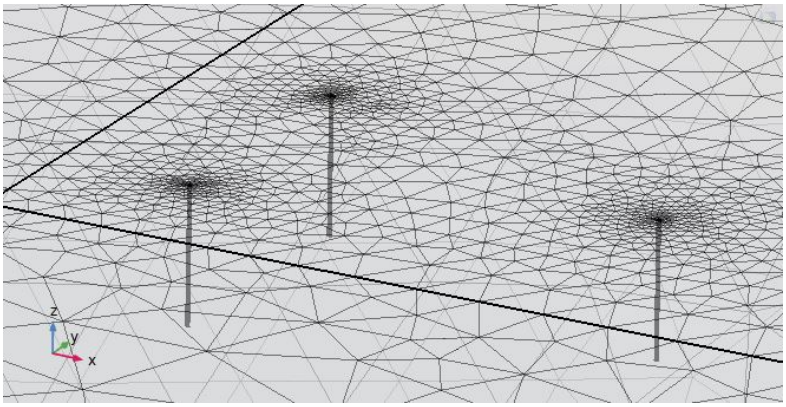

Figura 3.11: Dettaglio della mesh vicino ai picchetti

Dopo aver controllato che non vi fossero squilibri nelle dimensioni degli elementi costituenti la mesh, si è proceduto con l'analisi di uno studio in regime stazionario. Il solver ha risolto l'equazione di Laplace per il potenziale elettrico con le condizioni al contorno scelte: iniezione di una data corrente dai picchetti e potenziale nullo sul bordo del volume di dominio. In questa fase il software produce una matrice di equazioni lineari da risolvere.

3.1.4 Analisi dei risultati

Conclusa la fase di risoluzione, si è passati all'analisi dei risultati ottenuti. Il primo risultato estratto è stato relativo alla distribuzione di potenziale su tutto il dominio superficiale. In COMSOL si è fatto ricorso al plot di superficie, al fine di visualizzare un'immagine a colori che, adottando una mappa di colore per l'indicazione dei diversi livelli di tensione, mostra valori di potenziale decrescenti in funzione della distanza dagli elementi disperdenti. Di seguito i risultati registrati:

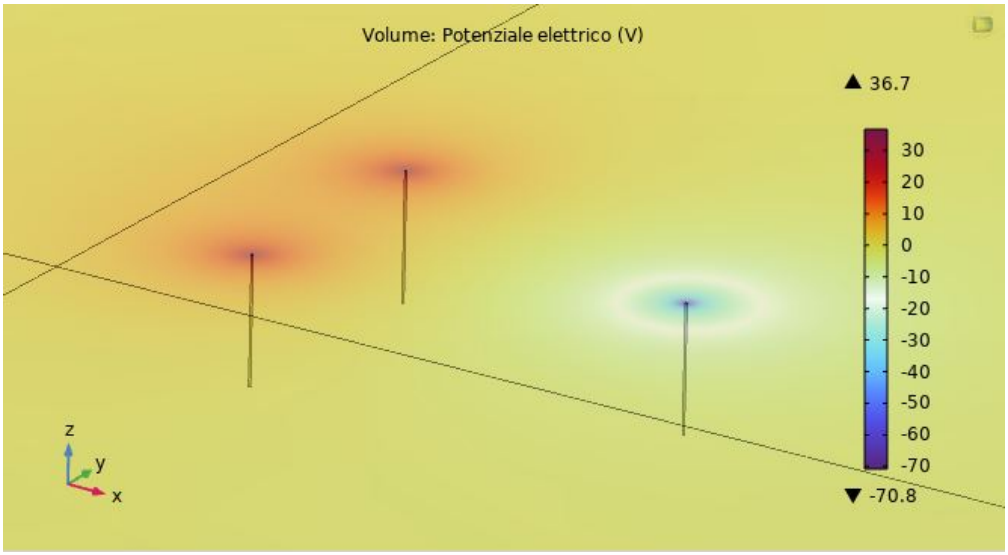

Figura 3.12: Potenziale superficiale in trasparenza

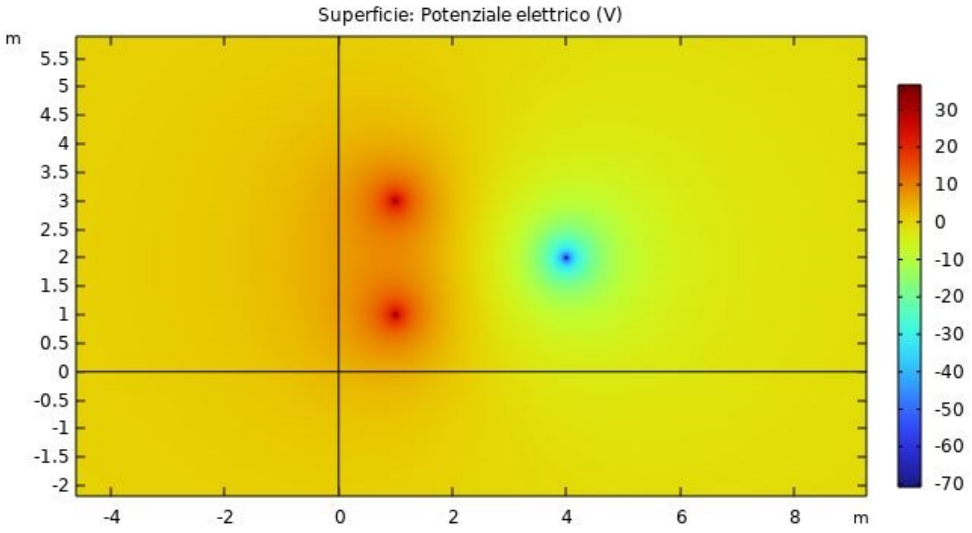

Figura 3.13: Potenziale superficiale in pianta

3.1.5 Confronto con metodo dello sottoaree di Maxwell

Al termine di questa procedura, si ottiene un quadro dettagliato di come la corrente si distribuisce nel terreno e di come varia il potenziale in ogni suo punto.
Il FEM rappresenta così un riferimento numerico accurato che, in molti casi, viene considerato il benchmark di simulazione. Da qui nasce l'idea di utilizzarlo come strumento per validare e tarare i metodi analitici, compreso quello delle sottoaree di Maxwell. Come spiegato in precedenza, seguendo tale metodologia si calcolano le correnti e i potenziali dei segmenti e poi si usa la sovrapposizione degli effetti per ricostruire il valore di V in superficie. Per il confronto è stato applicato il suddetto metodo al medesimo scenario modellato in MATLAB. Questo confronto, effettuato su uno stesso insieme di punti, permette di quantificare lo scostamento numerico fra il modello FEM e quello MATLAB. Si riportano di seguito i risultati ottenuti dalla comparazione delle due tecniche:

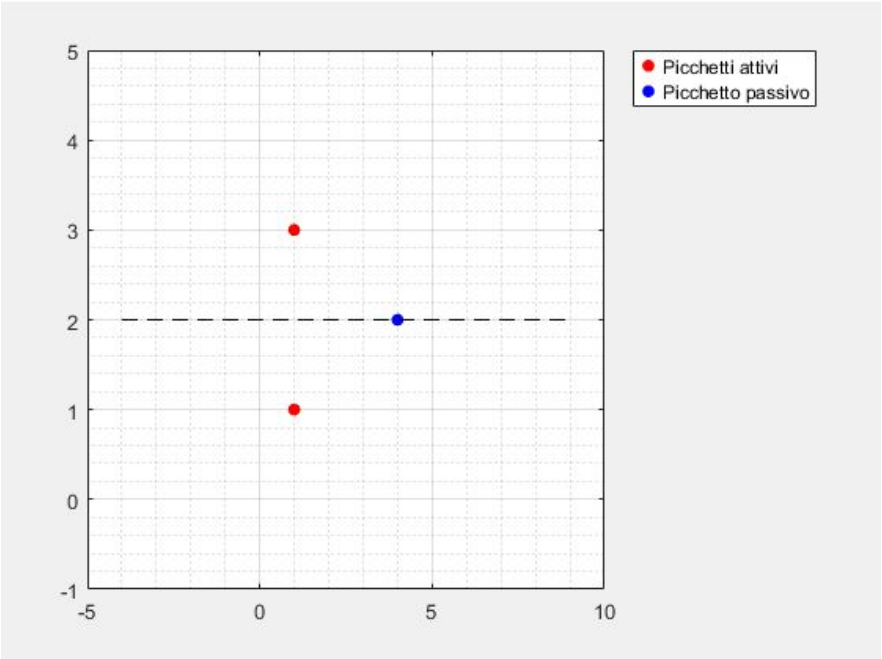

Figura 3.14: Linea di calcolo in pianta

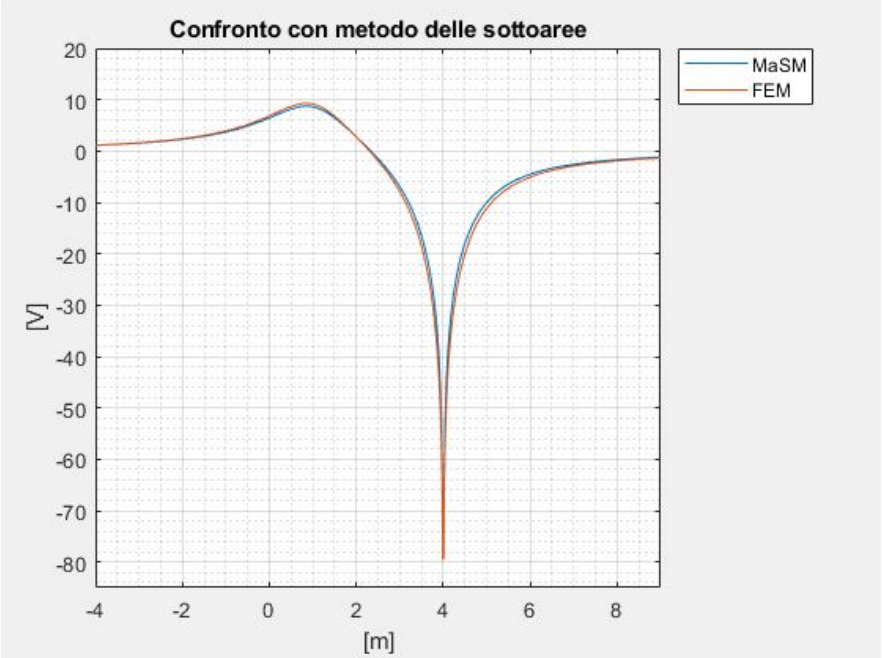

Figura 3.15: Risultato del confronto

Per attuare la comparazione effettuata sono stati estratti dal calcolo FEM i valori di potenziale appartenenti al set di punti specificato. Dall'analisi dei risultati si mostra come il metodo delle sottoaree di Maxwell segua in maniera fedele l'andamento del potenziale registrato dal modello FEM.
Alla luce di tale considerazione è pertanto possibile affermare che, almeno sotto l'ipotesi di terreno omogeneo e di resistività costante, il metodo numerico delle sottoarre rappresenti una valida tecnica per l'obiettivo perseguito.

3.2 Confronto con metodo sperimentale

Accertata la validità del metodo e l'adesione al corrispettivo fenomeno modellato agli elementi finiti è stata la volta della comparazione con un caso studio realmente implementato, effettuando il confronto con risultati provenienti da rilievi strumentali.

3.2.1 Setup sperimentale

Nel seguente scenario si è voluto anche indagare sull'influenza di elementi flottanti in prossimità del dispersore. La casistica rappresentata si rivela piuttosto frequente nella realtà applicativa del fenomeno, data la sovente compresenza di elementi metallici interrati, spesso ignoti, nelle vicinanze degli impianti di terra, in grado di influenzare l'andamento locale del potenziale. Il caso studio di riferimento è stato prelevato da un articolo scientifico che documenta la campagna di misure condotta per il rilevamento dei valori di potenziale superficiale del campo di prova, secondo la configurazione 3D di seguito rappresentata, corredata di informazione sui punti di misura:

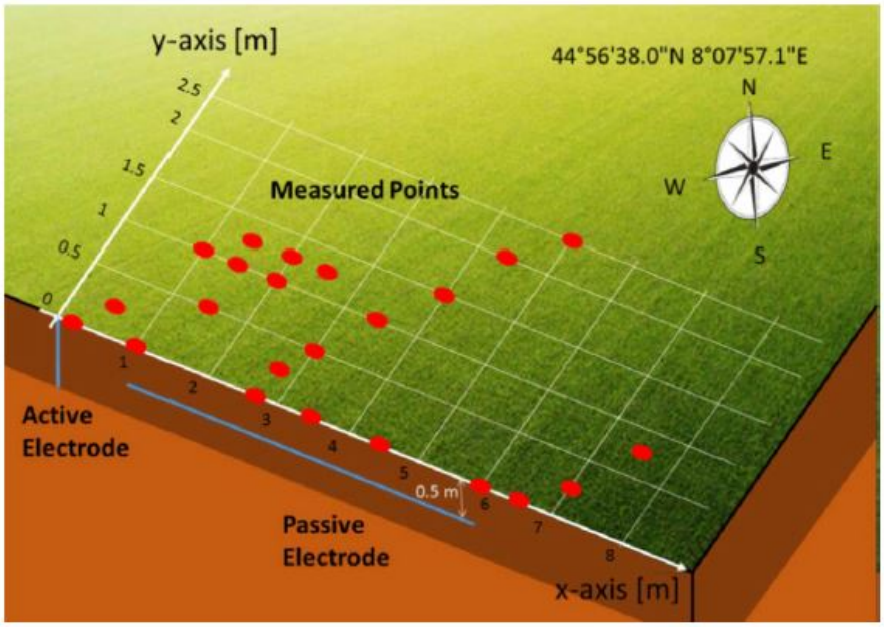

Figura 3.16: Modellazione del sito di misura

L'elettrodo attivo è un picchetto di terra inserito verticalmente nel terreno, mentre l'elettrodo passivo è un conduttore nudo orizzontale, interrato a una profondità di 0,5 m, assimilabile ad una conduttura idrica.
A conclusione dell'impostazione della prova sperimentale è stata iniettata una corrente di 1 A con l'impiego di un trasformatore di isolamento per mezzo dell'elettrodo attivo. Il valore della corrente dispersa nel terreno è stata opportunamente scelta al fine di ridurre il rischio associato a tensioni di contatto pericolose. L'origine delle coordinate cartesiane coincide con il centro dell'elettrodo attivo, mentre l'asse x è allineato con l'asse dell'elettrodo passivo.

3.2.2 Analisi e confronto con metodo delle sottoaree di Maxwell

La figura seguente rappresenta la vista in pianta della superficie del suolo; sono stati evidenziati con un cerchio rosso i punti in cui è stato misurato il potenziale elettrico. Accanto a ciascun punto, sono riportati i valori della tensione misurata. Nella scelta dei risultati più rappresentativi è stato deciso di aggregare due set di punti sulla superficie del terreno: il primo parallelo all'elemento flottante, in diretto allineamento, ed il secondo in maniera trasversale a quest'ultimo. I risultati raccolti sono evidenziati nella figura che segue:

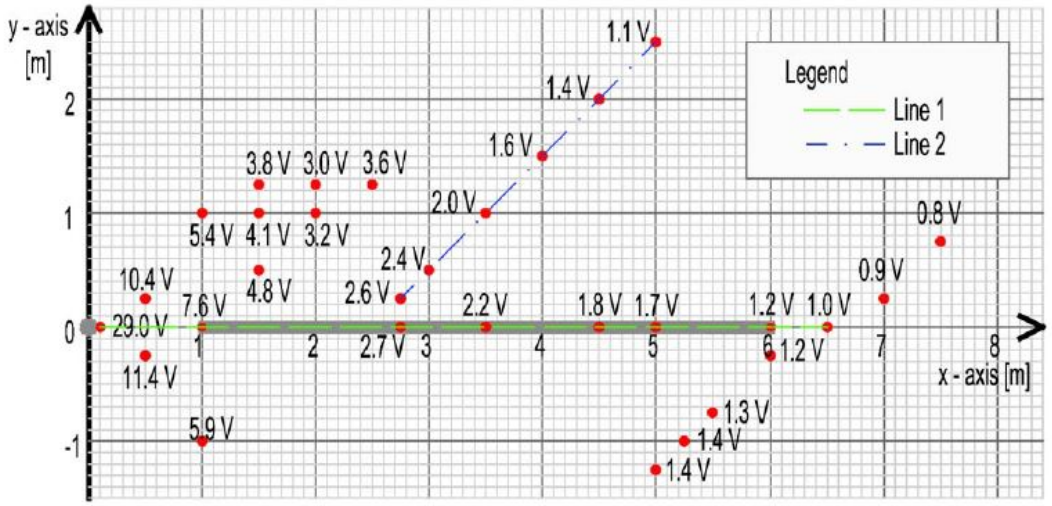

Figura 3.17: Valori di potenziale e linee di comparazione

Registrati i valori derivanti dai rilievi sperimentali è stato possibile, in ultima analisi, effettuare la comparazione con lo stesso scenario implementato in MATLAB adottando il metodo delle sottoaree di Maxwell, ottenendo i risultati di seguito illustrati:

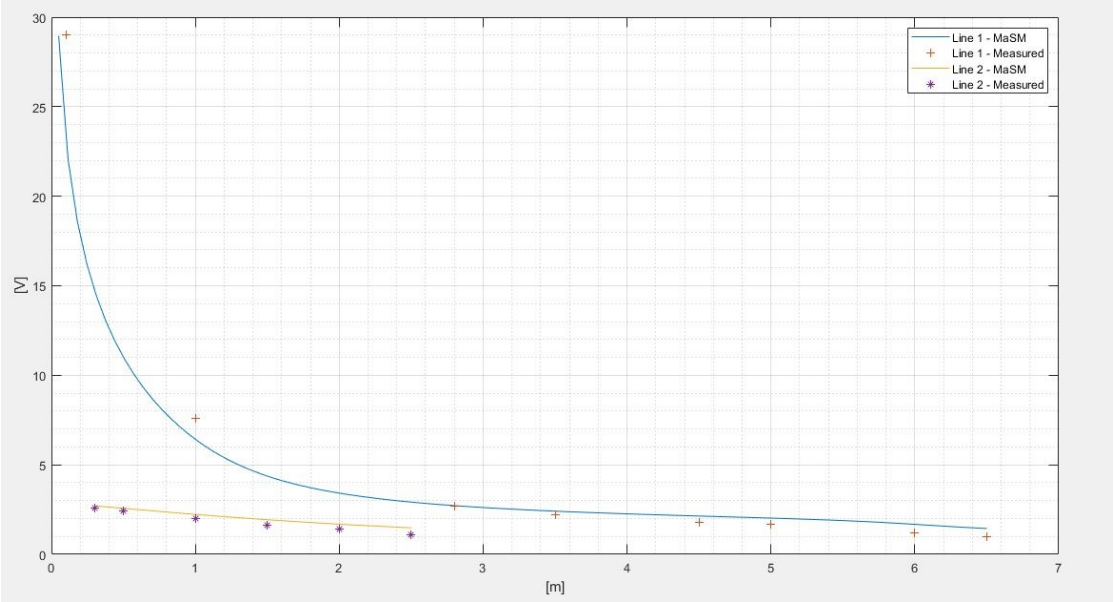

Figura 3.18: Comparazione tra metodo delle sottoaree e misure

Per confrontare i valori di potenziale sul terreno, sono state ricostruite le due linee di calcolo su cui valutare la grandezza di interesse. Da come si può notare i risultati mostrano una buona concordanza tra le tensioni misurate e quelle calcolate. In conclusione si tratta di un risultato positivo che consente di determinare, con buona dose di fiducia, la veridicità del modello analitico studiato.

Capitolo 4

Sviluppo dell'applicativo

Nel presente capitolo si espone il percorso di sviluppo di un'applicazione di realtà aumentata (AR) per dispositivi mobili che implementi il metodo delle sottoaree di Maxwell. Tale applicativo si pone l'obiettivo di calcolare e visualizzare, in tempo reale, l'andamento del potenziale elettrico prodotto da una corrente nota iniettata da picchetti virtualmente interrati in un terreno omogeneo. Lo scopo principale del progetto è combinare la potenza del modello matematico precedentemente discusso con la flessibilità e immediatezza di un sistema di realtà aumentata.
Grazie a questa integrazione, l'utente ottiene un'immagine diretta di come il potenziale si distribuisce e di come varia in punti diversi rispetto alla sorgente di corrente.
Nei paragrafi seguenti si traccerà un percorso esaustivo che parte dall'architettura generale dell'applicativo e prosegue descrivendo le singole componenti: l'interfaccia utente progettata per l'acquisizione dei parametri, il posizionamento fisico dei picchetti, la fase di calcolo vera e propria e la traduzione grafica dei risultati in una mesh interattiva. Verranno infine discusse problematiche di ottimizzazione e prestazioni, nonché le prospettive di sviluppo futuro.

4.1 Tecnologie e strumenti utilizzati

Per lo sviluppo del progetto è stato fondamentale adottare una serie di strumenti e tecnologie per garantire un'esperienza fluida ed efficiente. Il seguente capitolo intende presentare con maggiore dettaglio gli aspetti tecnologici che hanno reso possibile la realizzazione dell'applicativo.

4.1.1 Il motore grafico Unity

Il software principale utilizzato è stato Unity, un motore di sviluppo estremamente versatile, ampiamente adottato nell'ambito della creazione di videogiochi e applicazioni interattive. Dal punto di vista tecnico, Unity utilizza un'architettura basata su componenti. Gli oggetti di gioco (GameObject) costituiscono gli elementi fondamentali della scena e possono essere arricchiti con componenti specifici per modificarne il comportamento. Ad esempio, un GameObject può avere un componente che ne definisce l'aspetto grafico (come un renderer), un componente fisico che ne regola la collisione con altri oggetti, e uno script che determina la sua logica di funzionamento.
La programmazione in Unity avviene principalmente attraverso il linguaggio C#, un linguaggio moderno e potente che consente un elevato controllo sulle dinamiche dell'applicazione. Gli script C# interagiscono con il motore di gioco tramite le API di Unity, permettendo la gestione degli eventi, delle interazioni tra oggetti e della fisica della simulazione.
Un aspetto chiave di Unity è la sua interfaccia utente intuitiva, che facilita l'organizzazione dei progetti e la gestione delle risorse. L'editor di Unity offre strumenti avanzati per la manipolazione della scena, la configurazione delle impostazioni di rendering, il debugging e l'ottimizzazione delle prestazioni. Gli sviluppatori possono anche sfruttare il sistema di prefab, che consente di creare e riutilizzare oggetti predefiniti, migliorando così la modularità e l'efficienza nello sviluppo.

Figura 4.1: Logo di Unity

4.1.2 Il framework AR Foundation

AR Foundation rappresenta un componente fondamentale nell'ecosistema di Unity per lo sviluppo di applicazioni in realtà aumentata. Questo framework offre un'astrazione che consente agli sviluppatori di creare esperienze AR in maniera multipiattaforma, sfruttando le potenzialità di ARKit per i dispositivi iOS e ARCore per quelli Android senza dover scrivere codice specifico per ciascuna piattaforma. La capacità di lavorare con un'unica API semplifica notevolmente il processo di sviluppo, permettendo di concentrarsi maggiormente sulla logica applicativa e sull'esperienza utente.
Uno degli aspetti più innovativi di AR Foundation è la sua capacità di acquisire e interpretare in tempo reale i dati provenienti dai sensori dei dispositivi mobili. Le fotocamere e gli accelerometri collaborano per mappare l'ambiente circostante, individuando superfici orizzontali e verticali e rilevando la profondità e l'orientamento degli oggetti presenti. Grazie a queste funzionalità, è possibile posizionare in modo realistico oggetti virtuali che interagiscono con il mondo reale: ad esempio, un modello 3D può essere ancorato a una superficie rilevata, permettendo all'utente di osservarlo da diverse angolazioni come se fosse fisicamente presente.
Il tracciamento spaziale è un altro elemento cruciale offerto da AR Foundation. Attraverso algoritmi sofisticati, il framework è in grado di mantenere un costante aggiornamento della posizione e dell'orientamento del dispositivo, garantendo che gli oggetti virtuali rimangano allineati con l'ambiente fisico anche quando l'utente si muove. Questa stabilità è essenziale per creare esperienze immersive e credibili, in modo da minimizzare la distanza tra il digitale e il reale.
Un ulteriore vantaggio di AR Foundation è la gestione dinamica della luce ambientale. Il framework sfrutta le informazioni raccolte dalla fotocamera per stimare l'illuminazione presente nell'ambiente, consentendo agli sviluppatori di adattare automaticamente la resa grafica degli oggetti virtuali. Questo significa che, indipendentemente dalle condizioni di luce esterne, gli oggetti digitali possono essere illuminati in modo coerente con l'ambiente circostante, migliorando notevolmente il realismo della scena. In applicazioni AR, dove la percezione dell'integrazione tra oggetti reali e virtuali è fondamentale, questo tipo di adattamento dinamico dell'illuminazione rappresenta un grande vantaggio.

Un ulteriore aspetto interessante è la possibilità di sfruttare il debugging e la profilazione direttamente all'interno dell'editor di Unity. Quando si sviluppa con AR Foundation, è possibile testare le applicazioni in modalità simulata, verificando come gli oggetti virtuali interagiscono con le superfici rilevate e come l'illuminazione si adatta alle condizioni ambientali. Questa capacità di iterazione rapida è particolarmente preziosa nel campo della realtà aumentata, dove piccoli dettagli possono fare una grande differenza nell'esperienza complessiva dell'utente.

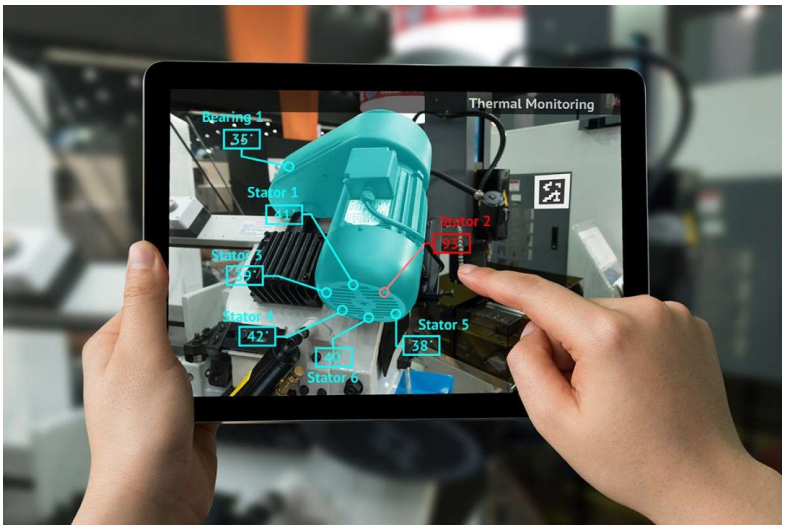

Figura 4.2: Esempio di applicativo AR di Unity in ambito industriale

4.2 Struttura generale del sistema

Nel corso dello sviluppo dell'applicativo, si è posta la necessità di concepire un'architettura software in grado di gestire in maniera discretizzata l'insieme di funzionalità richieste. Per raggiungere tale scopo, si è scelto di seguire una struttura modulare tipica dei progetti realizzati in Unity, con una partizione dei compiti ben definita. Si è definito un primo blocco per la gestione dell'interfaccia utente e la raccolta degli input, il cui script si lega a pulsanti e campi di testo. Tale parte di codice ha il compito di validare i dati inseriti (ad esempio controllare che i campi per la resistività e le correnti contengano numeri validi) e di passarli al modulo di calcolo. Il secondo blocco è quello della logica di calcolo, che contiene le funzioni responsabili di costruire la matrice delle resistenze e di risolvere il sistema lineare associato. Tale blocco non si occupa di alcun aspetto grafico, limitandosi a fornire funzioni per calcolare i valori di potenziale su una griglia di punti e restituendo strutture dati. Il terzo ed ultimo blocco è quello preposto alla funzione di rendering, con il compito di tradurre i risultati numerici in una rappresentazione visiva, come una mesh poligonale o una mappa planare. Se l'utente seleziona un'opzione di visualizzazione 2D, la componente di rendering procede a creare una texture colorata in base al valore di potenziale normalizzato e a mapparla su un piano 2D. Invece, se si preferisce la vista 3D con "rilievo," la stessa componente calcola le coordinate verticali dei vertici in base ai valori di potenziale. Questa parte del sistema deve anche gestire la texture e il colore per ogni vertice, così da visualizzare la variazione di potenziale in maniera cromaticamente efficace.

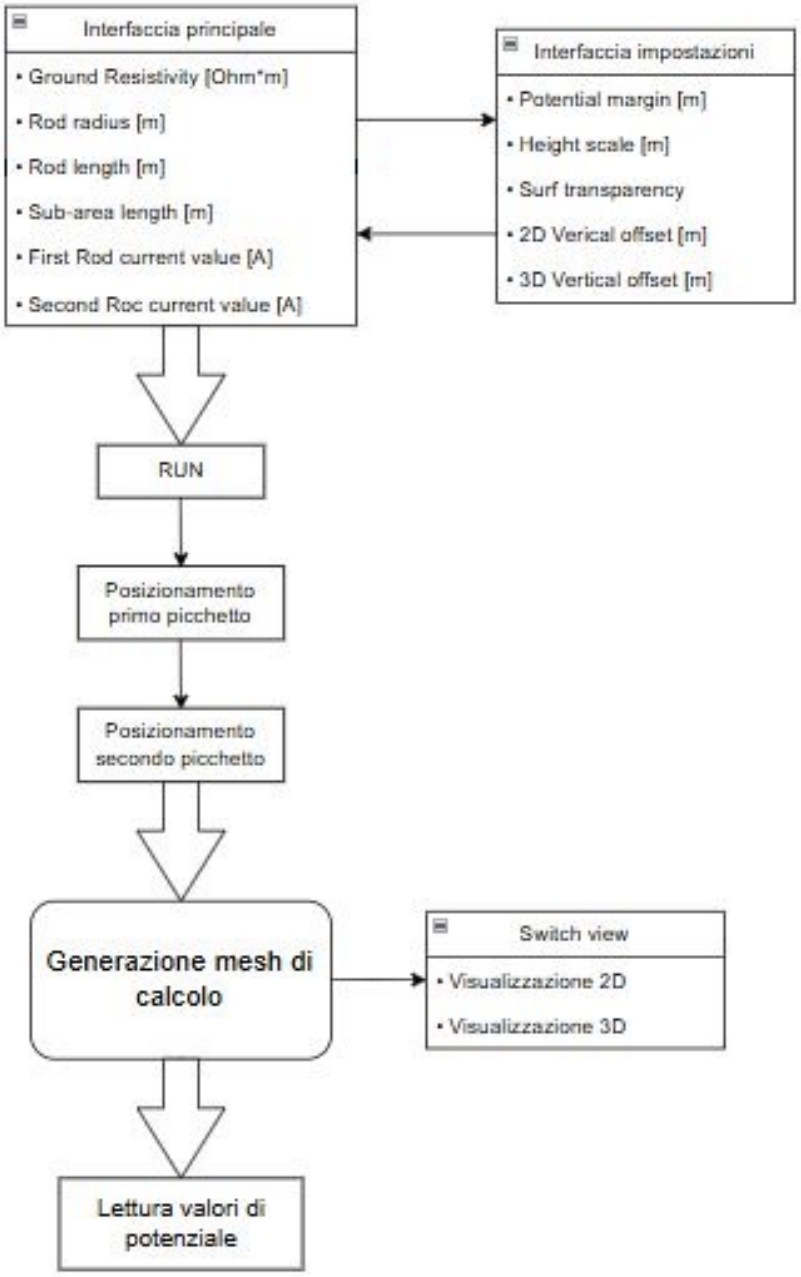

Figura 4.3: Flowchart utilizzo dell'applicazione

4.2.1 Gestione dell'interfaccia grafica e parametri di input

L'interfaccia grafica (GUI) gioca un ruolo decisivo nel favorire un utilizzo fluido e intuitivo del sistema. All'avvio dell'applicativo l'utente visualizza una schermata che consente di immettere i valori fisici fondamentali. Tali valori sono raggruppati in campi di testo, consentendo di inserire la resistività del terreno e parametri geometrici come il raggio dell'elettrodo, la sua lunghezza complessiva e la sua suddivisione in sottoaree. Questi ultimi ricoprono un'importanza cruciale nel calcolo, poiché determinano il livello di dettaglio con cui il conduttore è discretizzato. A questi dati di natura fisica si aggiungono le impostazioni riguardanti le correnti iniettate, ossia l'intensità di corrente associata a ciascun picchetto. L'utente può specificare due distinti valori di corrente, rendendo così possibile simulare situazioni in cui le correnti non siano identiche. Una volta che l'utente ha completato l'inserimento, è prevista una breve validazione del testo, che permette di gestire possibili errori di digitazione o l'uso involontario di virgole al posto dei punti decimali. In caso di input non valido, il sistema adotta valori predefiniti, minimizzando il rischio di blocchi o crash.
In seguito alla conferma di questi dati essenziali, l'utente può accedere a un pannello di impostazioni aggiuntive. Questa sezione avanzata offre la possibilità di perfezionare l'esperienza intervenendo su variabili di calcolo e di rendering. Ad esempio, si può introdurre un margine per allargare la zona di interesse attorno ai picchetti, in modo che la griglia di calcolo copra una porzione maggiore di spazio e non solo la stretta area tra i due marker. Si possono inoltre impostare valori di trasparenza, determinando quanto opaca o semitrasparente sarà la superficie del potenziale, e offset verticali distinti per la modalità 2D e 3D. Questi offset risultano utili per allineare correttamente la mesh con il piano reale, o per farla "galleggiare" a una certa altezza, rendendo più visibile il fenomeno.

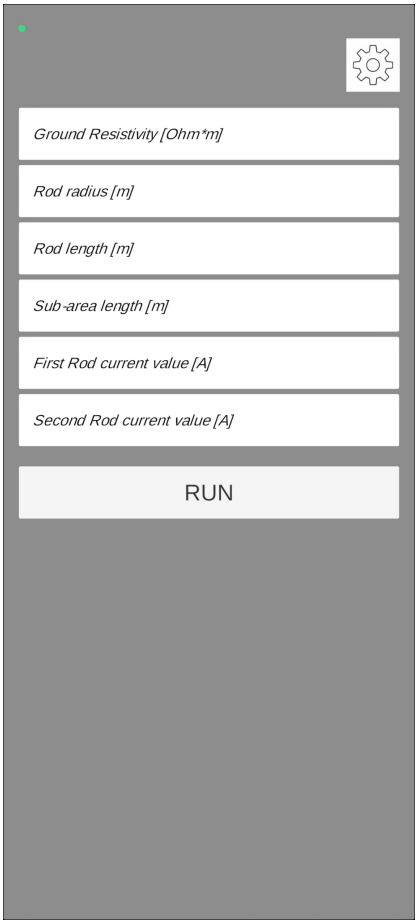

Figura 4.4: Interfaccia principale

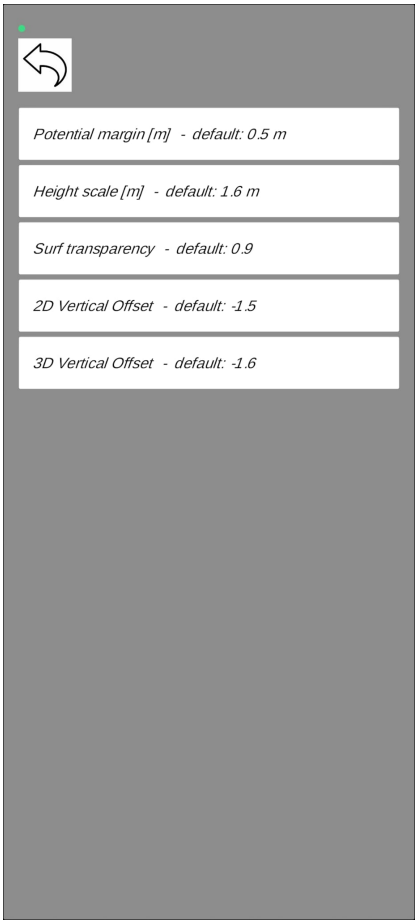

Figura 4.5: Interfaccia impostazioni

Quando l'utente conclude la configurazione, le due interfacce, quella principale e quella di impostazioni, cedono il passo alla modalità di posizionamento. Questo avviene premendo il pulsante RUN. In tal modo, la schermata di configurazione scompare e lascia spazio alla visuale in realtà aumentata, pronta a recepire i tocchi dell'utente per il posizionamento dei picchetti.

4.2.2 Posizionamento dei picchetti in AR

Il passaggio dal piano puramente configurativo a quello della realtà aumentata rappresenta uno snodo cruciale. In questa fase, l'utente si muove fisicamente nell'ambiente, inquadrando con la fotocamera la zona in cui desidera collocare i picchetti. Il sistema di riconoscimento dei piani analizza le immagini in tempo reale e, non appena individua superfici potenzialmente adatte, è pronto a rispondere ai tocchi. L'utente, toccando lo schermo in un punto appartenente ad una superficie planare, fornisce al sistema l'indicazione di posizionare in quello specifico luogo il primo picchetto.
La procedura prevede che venga generato un "prefab", ossia un oggetto 3D preconfigurato che appare nella posizione prescelta. È importante che l'ancoraggio avvenga correttamente, così da rispecchiare la rotazione e la posizione del piano. Subito dopo il primo tocco riuscito, l'utente può procedere a piazzare il secondo picchetto con la medesima modalità. Una volta collocati entrambi, la logica implementata blocca l'inserimento di ulteriori picchetti, dato che il modello di calcolo è concepito per gestire esattamente due punti di iniezione.
L'adozione di un meccanismo a stati garantisce che l'utente non possa accidentalmente piazzare troppi picchetti o attivare funzioni di calcolo in un momento in cui i dati non siano ancora completi. Nel momento in cui il secondo picchetto viene piazzato, il sistema registra entrambe le posizioni. Queste ultime costituiscono un'informazione fondamentale per la parte di calcolo, poiché determinano i riferimenti spaziali in cui verrà creata la griglia di punti per il potenziale e la successiva visualizzazione grafica. La scelta di abilitare la realtà aumentata solo dopo l'inserimento dei parametri garantisce un flusso lineare: prima si stabiliscono i valori fisici, poi si passa alla definizione interattiva nello spazio reale, infine si procede alla generazione della superficie.

4.2.3 Calcolo del potenziale e generazione della superficie

I parametri di resistività e geometria, inseriti dall'utente in precedenza, giocano un ruolo fondamentale nella definizione, come visto, dell'approssimazione del comportamento fisico desiderato. Una volta posizionati a schermo i due picchetti, il modello definisce un sistema lineare che, una volta risolto, rende noti i flussi di corrente in ogni sotto-segmento. Conoscere i valori di corrente nei segmenti è il

prerequisito per calcolare il potenziale in qualsiasi punto dello spazio circostante. A questo scopo, il programma genera una griglia di punti orizzontali che si estende da un minimo a un massimo di coordinate, comprendendo l'area dei picchetti e aggiungendo il margine stabilito. Su ciascun punto della griglia, il potenziale viene calcolato sommando il contributo di tutti i segmenti, sulla base del principio di sovrapposizione degli effetti.
Conclusa la fase di calcolo numerico, si ottiene una matrice 2D in cui ad ogni cella è associato un valore di potenziale. Si passa quindi alla costruzione della mesh grafica. I singoli elementi della matrice vengono convertiti in altrettanti vertici, posizionati nello spazio con una coordinata orizzontale che coincide con i valori della griglia e, in caso di visualizzazione 3D, con una coordinata verticale pari al potenziale scalato in base a fattori di rilievo selezionati dall'utente. Si ottiene così una superficie continua, formata dall'insieme di poligoni che collegano i vertici vicini. Ogni vertice può essere colorato in funzione del proprio valore di potenziale, andando dal blu più intenso per i punti a potenziale minimo al rosso più acceso per quelli a potenziale massimo. È inoltre possibile regolare l'opacità, così che la superficie risulti semitrasparente e consenta di intravedere il pavimento o altre entità reali al di sotto.
A vantaggio di una visualizzazione più chiara sono stati previsti ulteriori passaggi di interpolazione e smoothing. L'interpolazione aumenta la densità di punti, rendendo la superficie più uniforme e riducendo la percezione di scalini, specialmente quando la griglia di base non è molto fitta. Lo smoothing, invece, applica tecniche e algoritmi volti ad eliminare picchi o discontinuità troppo marcate. L'effetto complessivo è una mesh più liscia e graduale, anche se comporta un costo computazionale aggiuntivo. L'utente può controllare l'entità di questi filtraggi, in modo da decidere se privilegiare la rapidità di calcolo o la qualità estetica della visualizzazione.

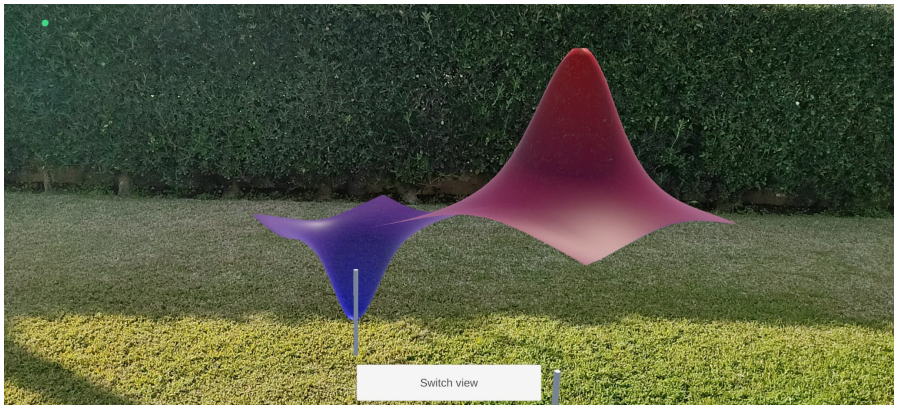

Figura 4.6: Visualizzazione mesh 3D

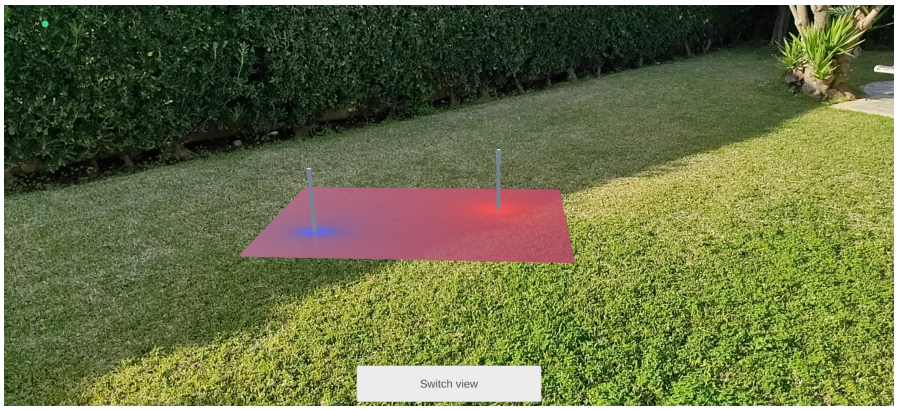

Figura 4.7: Visualizzazione mesh 2D

4.2.4 Interazione con la superficie e lettura del potenziale

Una volta generata e proiettata nello spazio, la superficie del potenziale diviene un elemento virtuale con cui l'utente può interagire. L'applicazione rende possibile toccare un punto qualunque di questa mesh, in modo da ottenere informazioni numeriche puntuali. L'azione di toccare lo schermo, mentre si inquadra la superficie in AR, produce un raycast, ovvero una procedura matematica per determinare collisioni con oggetti in ambienti virtuali, che il sistema sfrutta per determinare dove esattamente l'utente tocca la mesh. Al contatto l'applicazione recupera il valore di potenziale corrispondente al vertice più vicino al punto selezionato. Il feedback viene mostrato mediante un piccolo indicatore o "callout" che appare nello spazio virtuale, leggermente sopra la mesh, a una quota predefinita chiamata offset,

impedendo così che il callout si fonda con la superficie o che risulti in parte nascosto e garantendo una buona leggibilità del testo mostrato. A supporto della leggibilità dell'informazione, il callout si orienta verso la fotocamera. Se l'utente tocca un punto che non appartiene alla mesh, oppure se non ci sono collisioni, l'etichetta non appare o, se era già attiva, scompare. In tal modo, il sistema risulta reattivo e pulito, senza dispersioni di elementi grafici. Questa interazione tattile enfatizza la dimensione esplorativa dell'applicativo; l'utente può avvicinarsi ai picchetti, spostarsi lateralmente, osservare l'andamento del potenziale da diverse prospettive e verificare differenze tra diverse zone. Il callout fornisce un riscontro numerico immediato e coerente, trasformando l'esperienza in una sorta di laboratorio virtuale in cui il potenziale elettrico diventa visibile e tangibile in modo immersivo.

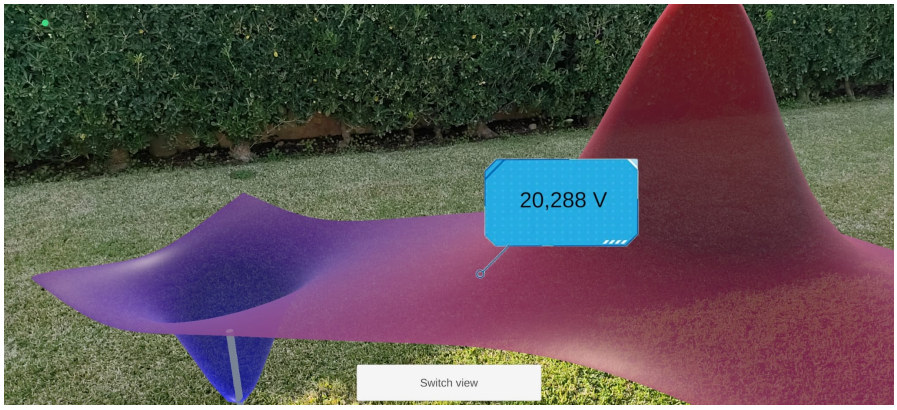

Figura 4.8: Callout su mesh 3D

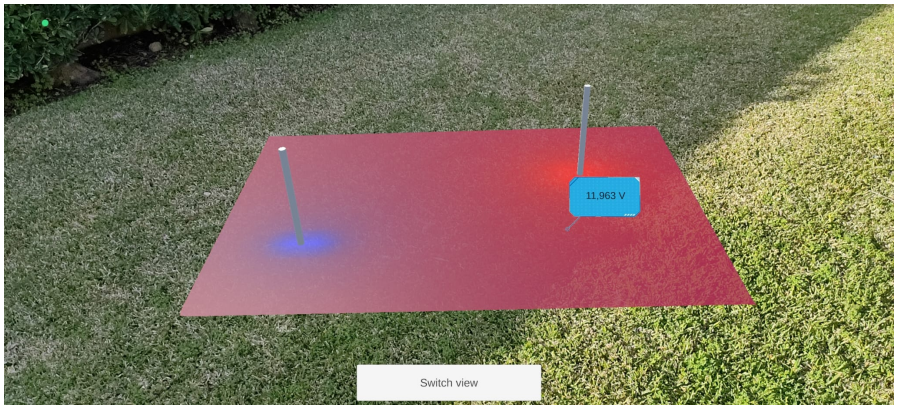

Figura 4.9: Callout su mesh 2D

4.2.5 Considerazioni sulle performance e conclusioni

Un progetto che unisce calcolo matematico, generazione di mesh e realtà aumentata deve necessariamente considerare aspetti di performance. Quando la griglia di calcolo è molto densa e il numero di sotto-segmenti del conduttore è elevato, la risoluzione del sistema lineare può comportare un onere computazionale non trascurabile, specialmente se il dispositivo mobile non è dotato di un'ampia potenza di calcolo. Analogamente, la creazione di migliaia di vertici e poligoni per la mesh può influire sui frame al secondo e quindi sulla fluidità dell'esperienza.
In conclusione, l'applicazione illustrata in questo capitolo costituisce una sintesi di diverse competenze, che spaziano dalla modellazione matematica alla programmazione grafica, fino alla progettazione di interfacce utente.
Sin dall'avvio, l'utente è posto al centro dell'esperienza, in quanto può impostare i valori di resistività, geometria e intensità di corrente e quindi procedere al posizionamento fisico dei picchetti in uno spazio reale grazie alla realtà aumentata. L'algoritmo di calcolo si occupa di elaborare i dati e generare una mappa di potenziale, la quale viene immediatamente tradotta in una superficie colorata che si fonde con l'ambiente circostante. La possibilità di toccare la superficie per leggere un valore locale di potenziale aggiunge all'esperienza una dimensione esplorativa, rendendo il sistema particolarmente adatto a finalità didattiche. Nel complesso, il lavoro qui presentato testimonia la forza e la versatilità della realtà aumentata unita a un motore grafico come quello adoperato. L'integrazione di queste componenti consente di esplorare fenomeni fisici in maniera intuitiva, spingendo la didattica e la ricerca verso forme di comunicazione più interattive.

4.3 Esempio di use case

Per illustrare in modo dettagliato come l'utente può interagire con l'app, si propone di seguito uno "use case" completo relativo al seguente scenario:
Scenario: Un utente vuole simulare la configurazione data da due picchetti infissi in un terreno con resistività 100 $\Omega \cdot$ m, ciscuno lungo 1 metro, da cui viene iniettata una corrente nota (1 A sul primo picchetto e -1 A sul secondo) e valutare la distribuzione di potenziale in 2D e 3D.

1. **Avvio dell'app**: l'utente apre l'applicazione su uno smartphone o un tablet. Compare la finestra principale con i campi di input: "Ground resistivity ($\Omega \cdot$ m)", "Rod length (m)", "Subarea length (m)", "First rod current value (A)" e "Second rod current value (A)";

2. **Inserimento dei dati**: l'utente imposta i valori numerici scelti nelle apposite sezioni;

3. **Posizionamento dei picchetti**: l'utente preme "RUN". Inquadrando il suolo con la fotocamera il sistema recepisce automaticamente la superficie orizzantale. Si può quindi procedere al posizionamento dei due picchetti nella posizione desiderata tramite tocco su schermo;

4. **Calcolo**: A questo punto lo script di calcolo costruisce la matrice di resistenze e risolve il sistema. Quindi genera la mappa di potenziale su una griglia NxN;

5. **Visualizzazione**: l'app mostra una mesh poligonale colorata in 3D, su cui il colore va dal blu (basso potenziale) al rosso (alto potenziale);

6. **Cambio di rappresentazione**: se l'utente vuole passare a una rappresentazione 2D planare, clicca sul pulsante "Switch view". L'app ricostruisce la mesh con l'asse verticale all'altezza del suolo, visualizzando il potenziale come un "tappeto";

7. **Lettura dei valori**: cliccando su un punto della mesh appare un'etichetta riportante il potenziale numerico. L'utente può esplorare diverse zone per capire come varia il potenziale in corrispondenza di distanze differenti dai picchetti. In questo modo, l'utente può muoversi fisicamente in uno scenario AR e vedere la label fluttuare in corrispondenza del punto selezionato.

Questo use case mostra come, nella pratica, uno studente o un docente possano configurare velocemente una determinata situazione fisica ottenendo un riscontro grafico immediato.

Capitolo 5

Utilizzo dell'applicativo a supporto della prova sperimentale

Quando si affronta il tema del potenziale di terra e della distribuzione delle tensioni attorno a un conduttore interrato, uno dei modi più diretti e convincenti per verificare l'accuratezza delle soluzioni calcolate consiste nel recarsi sul campo e svolgere misure mirate. Questa procedura permette di acquisire dati in merito all'effettivo comportamento del sistema suolo-conduttore, mettendo in evidenza non solo i valori di resistenza o di potenziale, ma anche eventuali effetti legati all'eterogeneità del terreno e a condizioni ambientali variabili.
Questo capitolo discuterà le basi operative della prova, descrivendo le attrezzature fondamentali, lo schema di posizionamento dei picchetti di prova in questione, la modalità di iniezione della corrente e la registrazione dei valori di tensione.

5.1 Descrizione della prova

Per introdurre la prova pratica, è utile richiamare le motivazioni di fondo. Nel corso dello sviluppo del modello teorico basato sul metodo delle sottoaree di Maxwell, si è presupposto una serie di condizioni ideali, come l'omogeneità della resistività del terreno, la linearità geometrica dei picchetti, l'assenza di elementi metallici estranei nelle vicinanze, e un certo regime stazionario dell'iniezione di corrente. Queste ipotesi, sebbene ragionevoli in un contesto accademico, rischiano di scontrarsi

con la complessità del suolo reale. Stratificazioni, variazioni di umidità, presenza di pietrisco o materiali diversi, tubazioni o reti metalliche possono distorcere notevolmente la distribuzione prevista. Di conseguenza, una misurazione sul campo consente di osservare e quantificare tali possibili scostamenti, e di capire se il modello adottato regge anche in condizioni concrete.
L'intera prova si articola nelle seguenti fasi:

1. Misura di resistività del terreno con il metodo di Wenner;
2. Misura di resistenza di terra di un dispersore;
3. Misure di potenziale sulla superficie del terreno;

5.1.1 Misura di resistività del terreno con il metodo di Wenner

La conoscenza della resistività del terreno risulta di fondamentale importanza nell'ottica della progettazione di un impianto di terra, dal momento che costituisce uno dei maggiori parametri di influenza del valore di resistenza di terra che si intende misurare. Fra i metodi di misura più comuni per la determinazione della resistività del terreno vi è il metodo di Wenner.

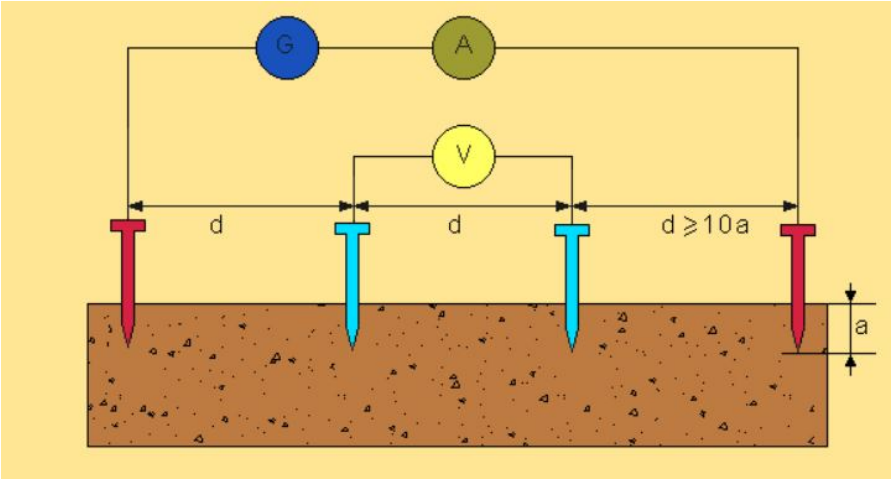

Figura 5.1: Circuito di misura con metodo di Wenner

Come illustrato nella figura precedente, il metodo prevede il posizionamento nel terreno di quattro sonde, poste in maniera allineata ed equidistanti le une dalle altre, infisse alla medesima profondità, da considerarsi trascurabile rispetto alla distanza reciproca d. Si procede a questo punto con il collegamento delle sonde all'apposito strumento di misura connettendo due picchetti ad un generatore di corrente e gli altri altri due ad un voltmetro, a formare un circuito voltmetrico. Dai rilievi del dispositivo di misura è possibile ricavare la resistività del terreno per mezzo della relazione:

$$\rho = 2\pi d \frac{U}{I} \tag{5.1}$$

dove U è la tensione misurata dal voltmetro tra la coppia di elettrodi intermedi e I la corrente di prova iniettata nel terreno attraverso la coppia di elettrodi estremi. Poiché normalmente il terreno non è omogeneo la resistività del terreno non risulta uniforme. La resistività varia al variare della distanza fra le sonde e può essere utile effettuare una serie di misure variando la distanza d mantenendo costante la profondità di infissione degli elettrodi.

5.1.2 Misura della resistenza di terra

Determinato sperimentalmente il valore di resistività del terreno è la volta del calcolo della resistenza di terra di un dispersore, ottenuta mediante l'adozione del protocollo di misura previsto dal metodo voltamperometrico.
Il metodo prevede l'iniezione di una corrente alternata attraverso i dispersori in misura, in modo da permetterne la richiusura attraverso un dispersore ausiliario. La correttezza della misura dipende dalla posizione che assumono il dispersore ausiliario e l'elettrodo di tensione fra di loro in relazione al dispersore in misura. Il dispersore ausiliario deve essere posto in un punto del terreno sufficientemente lontano rispetto a quello in prova in modo che la misura non sia viziata dall'influenza reciproca. Si può ritenere con buona approssimazione che ad una distanza di circa cinque volte la lunghezza del dispersore possa ritenersi trascurabile qualsiasi contributo di reciproca influenza tra i dispersori.

Di seguito il circuito di misura previsto dal metodo:

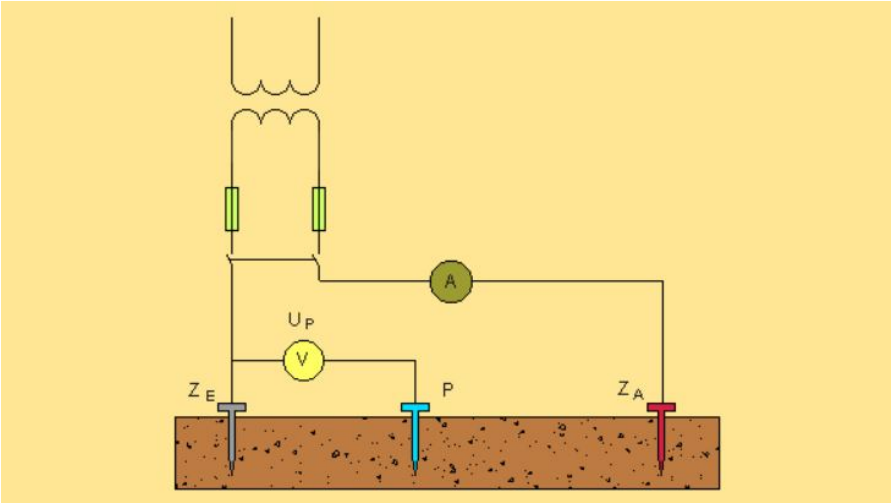

Figura 5.2: Circuito di misura metodo voltamperometrico

Lo schema di collegamento rappresentato nella figura appena illustrata presuppone un'alimentazione dalla rete tramite trasformatore di sicurezza oltre all'impiego di un voltmetro e di un amperometro. Si alimenta quindi il circuito di prova facendo circolare una corrente tra il dispersore in misura e il dispersore ausiliario. Se si indica con I la corrente che circola nel circuito e con U_E la tensione indicata dal voltmetro, applicando la legge di ohm si può calcolare la resistenza di terra:

$$R_E = \frac{U_E}{I} \tag{5.2}$$

La misura può essere considerata attendibile e tale rapporto rappresenta l'effettiva resistenza di terra solo se la sonda di tensione è infissa nella zona di non influenza del dispersore in misura e del dispersore ausiliario di corrente. Per verificarne l'esatta posizione, disposta la sonda di corrente alla maggior distanza possibile dal dispersore in prova, si sposta di alcuni metri la sonda di tensione da e verso il dispersore in prova finché le letture del voltmetro, non subendo più variazioni significative, possono essere considerate affidabili.

5.1.3 Misura di potenziale sulla superficie del terreno

Operazione conclusiva e successiva alla determinazione del valore di resistenza di terra è la caratterizzazione del suolo circostante il sistema, attraverso letture del valore di potenziale, rilevate a distanze progressivamente maggiori del dispersore, al fine di "mappare" l'intera area di interesse con valori di tensione da organizzare in tabella.
Con questi presupposti, la prova pratica si configura come un elemento essenziale di confronto, grazie alla sua intrinseca capacità di consentire, seppur con le dovute incertezze strumentali e fisiche del sistema, il riscontro più fedele con la reale natura del fenomeno studiato.

5.2 Impiego dell'applicativo

Con il seguente sottocapitolo si intende passare in rassegna una serie di scenari in cui l'utilizzo dell'app può fungere da supporto pratico al contesto sperimentale, dalla fase di previsione dei risultati, al confronto in tempo reale con i dati rilevati e fino alla possibilità di rivedere i parametri del sistema.
L'applicazione può essere infatti impiegata in maniera proficua durante la prova pratica su campo per calcolare e visualizzare in modo dinamico il potenziale di terra, integrando i risultati teorici e i dati sperimentali in un unico flusso di lavoro. Prima di svolgere la prova, è possibile usare il software per prevedere in via preliminare i valori di tensione attesi a determinate distanze dal picchetto sulla base di una stima di resistività del suolo e dei parametri di configurazione del sistema, consentendo agli studenti di avvicinarsi alla prova pratica già in possesso di stime e grafici che illustrano l'andamento del potenziale.
Una volta sul campo, l'applicazione offre la possibilità di confrontare in tempo reale i valori sperimentali con quelli simulati. In questo modo, se i valori misurati mostrano scostamenti significativi rispetto a quelli previsti, è possibile comprendere se ci sono fenomeni aggiuntivi in gioco, come ad esempio stratificazioni nel suolo, presenza di conduttori vicini o umidità diversa dal previsto.
Oltre al controllo immediato della coerenza tra teoria e pratica, il software consente di esplorare rapidamente l'effetto di modifiche ai parametri del sistema. Se si modifica la resistività ipotizzata o si varia la posizione di un elettrodo, la mappa

di potenziale viene aggiornata e si può immediatamente rendersi conto di come cambierebbero i valori. Questo si rivela utile sia a fini didattici sia per un'analisi più approfondita del comportamento dell'impianto di terra. Lo studente, ritrovandosi a regolare un valore e a vedere in modo diretto le differenze, può tentare eventuali variazioni laddove i dati reali mostrino divergenze, verificando se queste correzioni riducono lo scostamento. Così facendo, la verifica pratica si trasforma in un ciclo di confronto e aggiustamento, in cui la teoria e l'esperimento si arricchiscono reciprocamente.
In termini didattici quindi il risultato finale è un approccio integrato, in cui teoria, misurazione e visualizzazione interagiscono per fornire un quadro più completo e robusto del fenomeno elettrico studiato.

Capitolo 6

Impiego della realtà aumentata a fini didattici

Il seguente capitolo intende approfondire i risvolti, dal punto di vista didattico, derivanti dall'adozione in ambito universitario dell'applicativo precedentemente esposto, pensato come strumento di affiancamento durante lo svolgimento della prova sperimentale volta al rilievo dei potenziali di terra per via strumentale.
L'impiego della realtà aumentata in ambito educativo rappresenta un passo avanti significativo rispetto alle modalità tradizionali di erogazione di lezioni e contenuti didattici. Mentre i metodi convenzionali, come le lezioni frontali o le simulazioni bidimensionali, rimangono validi e continuano a fornire solide basi teoriche, la realtà aumentata (AR) offre un livello di coinvolgimento e di interattività che consente di trasformare gli ambienti di apprendimento in vere e proprie esperienze esplorative. Gli studenti non sono più semplici fruitori passivi di una spiegazione, ma si ritrovano a interagire con elementi virtuali che si integrano nell'ambiente fisico, spostandosi e manipolando gli oggetti sovrapposti per scoprirne le caratteristiche. Numerosi studi hanno messo in evidenza il potenziale di tali approcci: per esempio, ricerche di Azuma e colleghi hanno definito i principi della realtà aumentata, e in anni successivi vari lavori (tra cui quelli di Kaufmann e Schmalstieg e altri autori impegnati nell'ambito educativo) hanno sottolineato l'aumento di motivazione e di comprensione concettuale negli studenti che impiegano ambienti AR.
La ragione principale di questa efficacia risiede nel modo in cui l'AR potenzia la capacità di visualizzazione tridimensionale e rende le nozioni astratte o complesse più concrete e tangibili. Se si pensa alle lezioni frontali tradizionali, esse si basano

in larga parte sull'ascolto e sulla visione di materiali 2D (come diapositive o libri di testo), dove lo studente deve fare un passaggio cognitivo di interpretazione per ricostruire mentalmente la dimensionalità e la dinamica dei fenomeni. Con la realtà aumentata, invece, lo studente può vedere apparire di fronte a sé un modello 3D del fenomeno in esame, toccarne virtualmente i punti chiave. Questo livello di esplorazione diretta, secondo diverse ricerche nel campo delle scienze cognitive effettuate da Wu o da Chiang, risulta correlato a un incremento della motivazione intrinseca e a una migliore memorizzazione degli schemi spaziali. Le nozioni di base, a volte difficili da visualizzare, si tramutano in oggetti e scene integrate con il mondo fisico, riducendo i tempi di comprensione.
Nell'ambito dell'elettrotecnica e delle discipline STEM in generale, questo vantaggio appare ancor più evidente, considerando tutti i casi in cui i fenomeni rispondono a logiche complesse o multidimensionali. La possibilità di studiare linee di campo, superfici equipotenziali, configurazioni di circuiti in uno spazio virtuale, sovrapposto a un contesto reale, sblocca modalità di apprendimento potenziate. Le lezioni frontali, d'altro canto, continuano a servire da fondamento teorico e di trasmissione lineare dei contenuti, ma i risultati di ricerche (ad esempio quelli pubblicati da Santos e altri) evidenziano un incremento del coinvolgimento e della memorizzazione dei concetti quando si passa a tecniche di apprendimento esperienziali e interattive come quelle offerte dall'AR.
Nonostante i vantaggi e l'entusiasmo che questa tecnologia ha suscitato, è opportuno richiamare anche i limiti e le sfide dell'AR in contesto didattico. Un primo aspetto è quello relativo ai costi e all'accessibilità dei dispositivi. Se in alcuni casi basta un comune smartphone o tablet dotato di fotocamera e di un processore ragionevolmente potente, in altri si ricorre a visori dedicati che possono risultare costosi e non sempre facilmente reperibili in istituti scolastici o universitari con budget limitati. Di conseguenza, la diffusione dell'AR come pratica curricolare richiede una disponibilità di strumenti o un programma di condivisione dei device che non è sempre garantito.
In aggiunta, occorre considerare l'eventualità che alcuni ambienti reali non si prestino bene alle procedure di tracking e di riconoscimento dei piani, soprattutto se le infrastrutture di illuminazione o i riferimenti visivi sono scarsi. Gli studenti potrebbero sperimentare difficoltà o frustrazioni nel posizionare correttamente gli oggetti virtuali se il device fatica a rilevare superfici o marker, specie in contesti

affollati o con scarsa illuminazione.
Nel complesso, però, le ricerche portate avanti negli ultimi anni convergono sul fatto che, nonostante le sfide e le barriere ancora presenti, la realtà aumentata incrementa l'impatto e la forza didattica delle lezioni, specialmente nelle discipline scientifiche e tecnologiche. Quando ci si riferisce a un tema come il calcolo del potenziale di terra, dove il fenomeno è invisibile, vederlo proiettato in uno spazio a misura umana risulta enormemente chiarificatore. L'utente non deve soltanto immaginare la curva di potenziale in un grafico 2D, ma può vedere un vero e proprio tappeto colorato o un rilievo tridimensionale.
In conclusione, l'uso dell'AR a fini didattici rappresenta uno sviluppo naturale degli sforzi di innovazione nell'istruzione. Il fatto che gran parte delle ricerche citi un aumento del coinvolgimento, della motivazione e una maggiore propensione a ricordare e comprendere i concetti di base sembra confermare il valore pedagogico di questi sistemi. Al tempo stesso, rimane cruciale il contesto di adozione: la necessità di dispositivi adeguati e la consapevolezza che l'AR dev'essere utilizzata come un mezzo e non come un fine. Trovare il giusto equilibrio tra teoria, pratica sperimentale e visualizzazione virtuale permette di ottenere risultati davvero notevoli, spostando in avanti i confini di come l'elettrotecnica, la fisica e le altre discipline possono essere insegnate in modo coinvolgente e proficuo.

Capitolo 7

Conclusione e sviluppi futuri

Nel capitolo conclusivo di questo lavoro di tesi si desidera tracciare un bilancio complessivo di quanto è stato svolto e porre in evidenza le possibili vie per estendere e migliorare i risultati ottenuti.
Sin dalle prime fasi del lavoro è emerso un doppio obiettivo: verificare l'affidabilità del modello analitico e, contemporaneamente, offrire uno strumento software facile da usare e da integrare in contesti di misurazione sul campo e di apprendimento.
Nel dettaglio, sono stati raggiunti diversi risultati significativi. Da un punto di vista teorico, si è preso in esame il metodo delle sottoaree di Maxwell, confrontandone le previsioni con simulazioni numeriche più articolate tramite il software FEM COMSOL Multiphysics; i risultati hanno mostrato una buona convergenza, a patto di accettare le ipotesi di suolo omogeneo e di conduttori lineari. A rinforzo di queste conclusioni, è stato effettuato il riscontro diretto con uno scenario realmente implementato che ha permesso, anche qui, di verificare l'accuratezza del metodo nel fornire un ordine di grandezza e una distribuzione di potenziale coerenti.
Punto cardine dell'intero lavoro è stato, come detto, lo sviluppo di un applicativo con un'architettura software ben definita, dotato di un motore di calcolo in grado di connettersi ad una componente grafica in realtà aumentata. Questo permette di costruire o aggiornare una mesh 3D o una mappa 2D del potenziale, consentendo all'utente di gestire i parametri in tempo reale e di passare da una visualizzazione all'altra. Obiettivo fondante della tesi è quindi l'integrazione fra calcolo e visualizzazione AR, concepita non soltanto come strumento di calcolo, ma come un supporto didattico e operativo nelle fasi di verifica pratica. L'utente può configurare una determinata situazione di impianto di terra e confrontare istantaneamente

tali stime con le misure sul campo, aggiornando i parametri qualora emergano differenze sistematiche.

Le implicazioni principali di questi risultati riguardano tanto la didattica quanto la pratica professionale. La verifica sul campo dimostra che un software di questo tipo può davvero fungere da ponte tra teoria e pratica, purché si resti consapevoli di alcune limitazioni. In particolare, se il suolo è fortemente eterogeneo o stratificato o se la configurazione dell'impianto è molto complessa (con anelli, maglie o conduttori di forma irregolare), il metodo analizzato può non riflettere in maniera fedele la realtà. Inoltre, l'integrazione con la realtà aumentata richiede la disponiibilità di dispositivi adeguati, il che può costituire un ostacolo in alcuni contesti formativi o lavorativi con limitate risorse.

Gli sviluppi futuri di questo lavoro possono seguire più direzioni, sia sul piano pedagogico che operativo. Un primo miglioramento potrebbe riguardare l'estensione a terreni stratificati, inserendo funzioni di calcolo che tengano conto di uno o più strati con resistività differenti, in modo da consentire stime più accurate quando i dati geologici indicano discontinuità nel suolo. Ciò implicherebbe rivedere in parte il modulo di calcolo, aumentando la complessità ma ampliando notevolmente il campo di applicabilità. In secondo luogo si potrebbe estendere il sistema a geometrie di cavi orizzontali, reti di terra più elaborate e non soltanto al picchetto lineare, tenendo quindi conto di collegamenti multipli, sistemi magliati o sezioni di anelli interrati. In ultimo si potrebbe sviluppare un'estensione in realtà virtuale (VR), consentendo la fruizione immersiva tramite visore dedicato, dove si potrebbe "camminare" all'interno della mappa di potenziale, osservandola non più soltanto sovrapposta al mondo reale ma in un ambiente virtuale liberamente esplorabile. Ciò garantirebbe un livello di immersione ulteriore in contesti di ricerca o divulgazione avanzata.

Tutte queste prospettive sono da intendersi come rappresentazione di una naturale evoluzione del lavoro svolto fino ad ora; il prototipo realizzato dimostra che un approccio unificato tra teoria, misura e visualizzazione interattiva funziona e può essere ampliato e specializzato a seconda delle esigenze, siano esse didattiche o professionali. In ultimo si è dimostrato come la piattaforma di sviluppo Unity, congiuntamente ai pacchetti AR, possa offrire un ambiente in cui è relativamente agevole introdurre nuove funzionalità e consentire una pianificazione accuratamente dei passaggi tra calcolo numerico e resa grafica.

In definitiva, l'esperienza svolta in questa tesi dimostra l'efficacia di un percorso che va dalla teoria alla pratica. Ciò getta le basi per ulteriori evoluzioni verso prodotti software più ricchi, integrati e rispondenti alle molteplici variabili che si incontrano nella realtà ingegneristica, con prospettive di maggiore interazione e accuratezza.

Appendice

Si riporta di seguito il codice C# alla base dell'applicativo Unity sviluppato.

```
1
2  using System;
3  using System.Collections.Generic;
4  using System.Globalization;
5  using UnityEngine;
6  using UnityEngine.UI;
7  using UnityEngine.XR.ARFoundation;
8  using UnityEngine.XR.ARSubsystems;
9  using TMPro;
10
11 [RequireComponent(typeof(ARRaycastManager))]
12 public class MatlabARSurfaceManager : MonoBehaviour
13 {
14     // ============================================
15     //         BLOCCO 1: INTERFACCIA UTENTE (UI)
16     // ============================================
17
18     [Header("UI Principale")]
19     [SerializeField] private GameObject UIParentMain;
20     [SerializeField] private TMP_InputField rhoInputField;
21     [SerializeField] private TMP_InputField rInputField;
22     [SerializeField] private TMP_InputField LInputField;
23     [SerializeField] private TMP_InputField lInputField;
24     [SerializeField] private TMP_InputField I1InputField;
25     [SerializeField] private TMP_InputField I2InputField;
26     [SerializeField] private Button calcolaButton;
27     [SerializeField] private Button settingsButton;
28
29     [Header("UI Secondaria (Settings)")]
30     [SerializeField] private GameObject UIParentSettings;
31     [SerializeField] private TMP_InputField
   margineInputField;
```

```
32      [SerializeField] private TMP_InputField
    heightScaleInputField;
33      [SerializeField] private TMP_InputField
    surfAlphaInputField;
34      [SerializeField] private TMP_InputField
    verticalOffset2DInputField;
35      [SerializeField] private TMP_InputField
    verticalOffset3DInputField;
36      [SerializeField] private Button backButton;
37
38      [Header("UI Surf (Toggle 2D/3D)")]
39      [SerializeField] private GameObject UIParentSurf;
40      [SerializeField] private Button toggleViewButton;
41
42      [Header("Campo di testo per visualizzare il potenziale (
    testo generico)")]
43      [SerializeField] private TMP_Text potentialValueText;
44
45      // Altri parametri fisici e di calcolo (letti da UI
    Principale)
46      [Header("Parametri fisici (MATLAB) - da UI Principale")]
47      [SerializeField] private float rho = 60f;
48      [SerializeField] private float r = 0.01f;
49      [SerializeField] private float L = 1f;
50      [SerializeField] private float l = 0.25f;
51
52      [Header("Parametri di calcolo")]
53      [SerializeField] private float passo = 0.05f;
54      [Range(1, 10)]
55      [SerializeField] private int interpFactor = 1;
56
57      [Header("Correnti (da UI Principale)")]
58      [SerializeField] private float I1 = 10f;
59      [SerializeField] private float I2 = 8f;
60
61      // Parametri secondari (da UI Settings)
62      [Header("Parametri secondari (da UI Settings)")]
63      [SerializeField] private float margine = 0.5f;
64      [SerializeField] private float heightScale = 1.6f;
65      [Range(0f, 1f)]
66      [SerializeField] private float surfAlpha = 0.9f;
67
68      [Header("Offset verticali separati")]
69      [Tooltip("Quota in cui visualizzare il surf in modalità
    2D")]
```

```
70      [SerializeField] private float verticalOffset2D = -1.5f;
71      [Tooltip("Quota in cui visualizzare il surf in modalità
    3D")]
72      [SerializeField] private float verticalOffset3D = -1.6f;
73
74      [Header("Altri parametri di visualizzazione")]
75      [Range(0, 10)]
76      [SerializeField] private int meshSmoothing = 0;
77
78      [Header("Materiale per il surf (Fade/Transparent)")]
79      [SerializeField] private Material surfMaterial;
80
81      [Header("Balloon popup per mostrare il potenziale
    toccato")]
82      [SerializeField] private GameObject balloonPrefab;
83      [Tooltip("Offset verticale/posizionale del balloon
    rispetto al punto toccato")]
84      [SerializeField] private Vector3 balloonOffset = new
    Vector3(0f, 0.1f, 0f);
85
86      // Prefab AR (Picchetti)
87      [Header("Prefab AR (Picchetti)")]
88      [SerializeField] private GameObject markerPrefab;
89
90      // Gestione AR
91      private ARRaycastManager raycastManager;
92      private bool canPlaceMarkers = false;
93      private int markersPlaced = 0;
94      private GameObject marker1 = null;
95      private GameObject marker2 = null;
96      private Vector3 markerPos1, markerPos2;
97
98      // mesh di output
99      private GameObject surfObject = null;
100     private MeshCollider surfCollider = null;
101
102     // Toggle 2D/3D
103     private bool isPlanar = false;
104     private float userHeightScale = 1.6f;
105
106     // Dati un-normalized per potValue
107     private double[,] lastVgrid = null;
108     private List<double> lastXsurf = null;
109     private List<double> lastZsurf = null;
110     private int NxLast = 0, NzLast = 0;
```

```
111
112     // balloon runtime
113     private GameObject currentBalloon = null;
114     private TMP_Text balloonText;
115
116     private CultureInfo ci = CultureInfo.InvariantCulture;
117
118     void Awake()
119     {
120         raycastManager = GetComponent<ARRaycastManager>();
121
122         // UI Principale
123         if (calcolaButton != null)
124             calcolaButton.onClick.AddListener(
    OnCalcolaClicked);
125         if (settingsButton != null)
126             settingsButton.onClick.AddListener(
    OnSettingsClicked);
127
128         // UI Secondaria (Settings)
129         if (backButton != null)
130             backButton.onClick.AddListener(OnBackClicked);
131
132         // UI Surf
133         if (toggleViewButton != null)
134             toggleViewButton.onClick.AddListener(
    OnToggleViewClicked);
135
136         // Inizializziamo la UI di default
137         if (UIParentMain) UIParentMain.SetActive(true);
138         if (UIParentSettings) UIParentSettings.SetActive(
    false);
139         if (UIParentSurf) UIParentSurf.SetActive(false);
140
141         if (potentialValueText)
142             potentialValueText.text = "Electric Potential: 0
     V";
143
144         currentBalloon = null;
145         balloonText = null;
146     }
147
148     void Update()
149     {
150         if (!canPlaceMarkers) return;
```

```
151         if (markersPlaced >= 2) return;
152 
153         if (Input.touchCount > 0)
154         {
155             Touch t = Input.GetTouch(0);
156             if (t.phase == TouchPhase.Began)
157             {
158                 List<ARRaycastHit> hits = new List<
    ARRaycastHit>();
159                 if (raycastManager.Raycast(t.position, hits,
     TrackableType.PlaneWithinPolygon))
160                 {
161                     Pose pose = hits[0].pose;
162                     if (markerPrefab != null)
163                     {
164                         if (markersPlaced == 0)
165                         {
166                             marker1 = Instantiate(
    markerPrefab, pose.position, pose.rotation);
167                             markerPos1 = marker1.transform.
    position;
168                             markersPlaced++;
169                         }
170                         else if (markersPlaced == 1)
171                         {
172                             marker2 = Instantiate(
    markerPrefab, pose.position, pose.rotation);
173                             markerPos2 = marker2.transform.
    position;
174                             markersPlaced++;
175 
176                             OnBothMarkersPlaced();
177                         }
178                     }
179                 }
180             }
181         }
182     }
183 
184     void LateUpdate()
185     {
186         // Gestione tocco sulla mesh per mostrare potenziale
     e balloon
187         if (surfObject != null && surfCollider != null)
188         {
```

```
189             if (Input.touchCount > 0)
190             {
191                 Touch t = Input.GetTouch(0);
192                 if (t.phase == TouchPhase.Began)
193                 {
194                     Ray ray = Camera.main.ScreenPointToRay(t
    .position);
195                     if (Physics.Raycast(ray, out RaycastHit
    hitInfo))
196                     {
197                         if (hitInfo.collider == surfCollider
    )
198                         {
199                             Vector3 hitPos = hitInfo.point;
200                             double potVal =
    FindPotentialUnnormalized(hitPos.x, hitPos.z);
201 
202                             if (potentialValueText)
203                                 potentialValueText.text = $"
    Electric Potential: {potVal:F3} V";
204 
205                             ShowBalloon(hitPos, (float)
    potVal);
206                         }
207                         else
208                         {
209                             if (potentialValueText)
210                                 potentialValueText.text = "
    Electric Potential: 0 V";
211 
212                             HideBalloon();
213                         }
214                     }
215                     else
216                     {
217                         if (potentialValueText)
218                             potentialValueText.text = "
    Electric Potential: 0 V";
219 
220                         HideBalloon();
221                     }
222                 }
223             }
224         }
225         else
```

```
226         {
227             HideBalloon();
228         }
229     }
230 
231     private void ShowBalloon(Vector3 worldPos, float
    potValue)
232     {
233         if (balloonPrefab == null) return;
234 
235         if (currentBalloon == null)
236         {
237             currentBalloon = Instantiate(balloonPrefab);
238             balloonText = currentBalloon.
    GetComponentInChildren<TMP_Text>();
239         }
240 
241         currentBalloon.transform.position = worldPos +
    balloonOffset;
242         currentBalloon.transform.LookAt(Camera.main.
    transform);
243         currentBalloon.transform.Rotate(0f, 180f, 0f);
244 
245         if (balloonText != null)
246         {
247             balloonText.text = $"{potValue:F3} V";
248         }
249 
250         currentBalloon.SetActive(true);
251     }
252 
253     private void HideBalloon()
254     {
255         if (currentBalloon != null)
256         {
257             currentBalloon.SetActive(false);
258         }
259     }
260 
261     private void OnCalcolaClicked()
262     {
263         if (rhoInputField != null)
264             float.TryParse(rhoInputField.text.Replace(',',
    '.'), NumberStyles.Float, ci, out rho);
265         if (rInputField != null)
```

```
266             float.TryParse(rInputField.text.Replace(',',
    '.'), NumberStyles.Float, ci, out r);
267         if (LInputField != null)
268             float.TryParse(LInputField.text.Replace(',',
    '.'), NumberStyles.Float, ci, out L);
269         if (lInputField != null)
270             float.TryParse(lInputField.text.Replace(',',
    '.'), NumberStyles.Float, ci, out l);
271         if (I1InputField != null)
272             float.TryParse(I1InputField.text.Replace(',',
    '.'), NumberStyles.Float, ci, out I1);
273         if (I2InputField != null)
274             float.TryParse(I2InputField.text.Replace(',',
    '.'), NumberStyles.Float, ci, out I2);
275
276         if (UIParentMain) UIParentMain.SetActive(false);
277         canPlaceMarkers = true;
278     }
279
280     private void OnSettingsClicked()
281     {
282         if (UIParentMain) UIParentMain.SetActive(false);
283         if (UIParentSettings) UIParentSettings.SetActive(
    true);
284     }
285
286     private void OnBackClicked()
287     {
288         float tempVal;
289
290         // margine
291         if (margineInputField != null && !string.
    IsNullOrWhiteSpace(margineInputField.text))
292         {
293             if (float.TryParse(margineInputField.text.
    Replace(',', '.'), NumberStyles.Float, ci, out tempVal))
294                 margine = tempVal;
295             else
296                 margine = 0.5f;
297         }
298         else
299         {
300             margine = 0.5f;
301         }
302
```

```
303         // heightScale
304         if (heightScaleInputField != null && !string.
    IsNullOrWhiteSpace(heightScaleInputField.text))
305         {
306             if (float.TryParse(heightScaleInputField.text.
    Replace(',', '.'), NumberStyles.Float, ci, out tempVal))
307                 heightScale = tempVal;
308             else
309                 heightScale = 1.6f;
310         }
311         else
312         {
313             heightScale = 1.6f;
314         }
315         userHeightScale = heightScale;
316 
317         // surfAlpha
318         if (surfAlphaInputField != null && !string.
    IsNullOrWhiteSpace(surfAlphaInputField.text))
319         {
320             if (float.TryParse(surfAlphaInputField.text.
    Replace(',', '.'), NumberStyles.Float, ci, out tempVal))
321                 surfAlpha = Mathf.Clamp01(tempVal);
322             else
323                 surfAlpha = 0.9f;
324         }
325         else
326         {
327             surfAlpha = 0.9f;
328         }
329 
330         // verticalOffset2D
331         if (verticalOffset2DInputField != null && !string.
    IsNullOrWhiteSpace(verticalOffset2DInputField.text))
332         {
333             if (float.TryParse(verticalOffset2DInputField.
    text.Replace(',', '.'), NumberStyles.Float, ci, out
    tempVal))
334                 verticalOffset2D = tempVal;
335             else
336                 verticalOffset2D = -1.5f;
337         }
338         else
339         {
340             verticalOffset2D = -1.5f;
```

```
341         }
342
343         // verticalOffset3D
344         if (verticalOffset3DInputField != null && !string.
    IsNullOrWhiteSpace(verticalOffset3DInputField.text))
345         {
346             if (float.TryParse(verticalOffset3DInputField.
    text.Replace(',', '.'), NumberStyles.Float, ci, out
    tempVal))
347                 verticalOffset3D = tempVal;
348             else
349                 verticalOffset3D = -1.6f;
350         }
351         else
352         {
353             verticalOffset3D = -1.6f;
354         }
355
356         if (UIParentSettings) UIParentSettings.SetActive(
    false);
357         if (UIParentMain) UIParentMain.SetActive(true);
358     }
359
360     private void OnToggleViewClicked()
361     {
362         isPlanar = !isPlanar;
363         if (isPlanar)
364         {
365             heightScale = 0f;
366         }
367         else
368         {
369             heightScale = userHeightScale;
370         }
371
372         if (surfObject != null)
373             Destroy(surfObject);
374
375         GenerateSurface(markerPos1, markerPos2);
376
377         if (potentialValueText)
378             potentialValueText.text = "Electric Potential: 0
     V";
379
380         HideBalloon();
```

```
381     }
382
383     private void OnBothMarkersPlaced()
384     {
385         if (!marker1 || !marker2) return;
386         markerPos1 = marker1.transform.position;
387         markerPos2 = marker2.transform.position;
388
389         GenerateSurface(markerPos1, markerPos2);
390
391         if (UIParentSurf)
392             UIParentSurf.SetActive(true);
393     }
394
395     // ============================================
396     //         BLOCCO 2: LOGICA DI CALCOLO
397     // ============================================
398
399     private void GenerateSurface(Vector3 pos1, Vector3 pos2)
400     {
401         int n_sa = (int)Math.Ceiling(L / l) * 2;
402         int half = n_sa / 2;
403
404         double x1 = pos1.x; double z1 = pos1.z;
405         double x2 = pos2.x; double z2 = pos2.z;
406
407         double[] X_first = CreateConstantArray(x1, half);
408         double[] X_second = CreateConstantArray(x2, half);
409         double[] X = ConcatArrays(X_first, X_second);
410
411         double[] Y_first = LinSpace(0, -L, half);
412         double[] Y_second = LinSpace(0, -L, half);
413         double[] Y = ConcatArrays(Y_first, Y_second);
414
415         double[] Z_first = CreateConstantArray(z1, half);
416         double[] Z_second = CreateConstantArray(z2, half);
417         double[] Z_ = ConcatArrays(Z_first, Z_second);
418
419         double A = rho / (4.0 * Math.PI * l);
420         double dr = r * Math.Cos(Math.PI / 4.0);
421
422         double[,] R = new double[n_sa, n_sa];
423         for (int i = 0; i < n_sa; i++)
424         {
425             for (int j = i; j < n_sa; j++)
```

```
426             {
427                 if (i == j)
428                 {
429                     double R_s = SelfResistanceTerm(A, X, Y,
     Z_, i, dr, +1);
430                     double R_i = SelfResistanceTerm(A, X, Y,
     Z_, i, dr, -1);
431                     R[i, j] = R_s + R_i;
432                 }
433                 else
434                 {
435                     double R_s = MutualResistanceTerm(A, X,
    Y, Z_, i, j, +1);
436                     double R_i = MutualResistanceTerm(A, X,
    Y, Z_, i, j, -1);
437                     R[i, j] = R_s + R_i;
438                     R[j, i] = R[i, j];
439                 }
440             }
441         }
442 
443         double[,] A_current = new double[2, n_sa];
444         for (int c = 0; c < half; c++)
445             A_current[0, c] = 1f;
446         for (int c = half; c < n_sa; c++)
447             A_current[1, c] = 1f;
448 
449         double[,] A_voltage = new double[n_sa, 2];
450         for (int rr = 0; rr < n_sa; rr++)
451             for (int cc = 0; cc < 2; cc++)
452                 A_voltage[rr, cc] = -A_current[cc, rr];
453 
454         int bigSize = n_sa + 2;
455         double[,] A3 = new double[bigSize, bigSize];
456         for (int rr = 0; rr < n_sa; rr++)
457         {
458             for (int cc = 0; cc < n_sa; cc++)
459                 A3[rr, cc] = R[rr, cc];
460         }
461         for (int rr = 0; rr < n_sa; rr++)
462         {
463             for (int cc = 0; cc < 2; cc++)
464                 A3[rr, n_sa + cc] = A_voltage[rr, cc];
465         }
466         for (int rr = 0; rr < 2; rr++)
```

```
467         {
468             for (int cc = 0; cc < n_sa; cc++)
469                 A3[n_sa + rr, cc] = A_current[rr, cc];
470         }
471 
472         double[] b = new double[bigSize];
473         b[n_sa + 0] = I1;
474         b[n_sa + 1] = I2;
475 
476         double[] sol = SolveLinearSystem(A3, b);
477         double[] I_sottoaree = new double[n_sa];
478         for (int i = 0; i < n_sa; i++)
479             I_sottoaree[i] = sol[i];
480 
481         // Determino griglia NxC, NzC
482         double xmin = Math.Min(x1, x2) - margine;
483         double xmax = Math.Max(x1, x2) + margine;
484         double zmin = Math.Min(z1, z2) - margine;
485         double zmax = Math.Max(z1, z2) + margine;
486 
487         List<double> xSurf = new List<double>();
488         for (double xx = xmin + passo / 2; xx <= (xmax -
    passo / 2 + 1e-9); xx += passo)
489             xSurf.Add(xx);
490 
491         List<double> zSurf = new List<double>();
492         for (double zz = zmin + passo / 2; zz <= (zmax -
    passo / 2 + 1e-9); zz += passo)
493             zSurf.Add(zz);
494 
495         int NxC = xSurf.Count;
496         int NzC = zSurf.Count;
497 
498         double[] V_column = new double[NxC * NzC];
499         int idxCol = 0;
500         for (int iz = 0; iz < NzC; iz++)
501         {
502             for (int ix = 0; ix < NxC; ix++)
503             {
504                 double px = xSurf[ix];
505                 double pz = zSurf[iz];
506                 double py = 0.0;
507 
508                 double val = 0.0;
509                 for (int sa = 0; sa < n_sa; sa++)
```

```
510                 {
511                     val += PotentialContribution(A,
    I_sottoaree[sa], px, py, pz,
512                                                  X[sa], Y[sa
    ], Z_[sa], l);
513                 }
514                 V_column[idxCol++] = val;
515             }
516         }
517 
518         double[,] V_grid = new double[NzC, NxC];
519         idxCol = 0;
520         double minVal = double.MaxValue;
521         double maxVal = double.MinValue;
522         for (int j = 0; j < NzC; j++)
523         {
524             for (int i = 0; i < NxC; i++)
525             {
526                 double pot = V_column[idxCol++];
527                 V_grid[j, i] = pot;
528                 if (pot < minVal) minVal = pot;
529                 if (pot > maxVal) maxVal = pot;
530             }
531         }
532 
533         if (NxC < 2 || NzC < 2 || interpFactor <= 1)
534         {
535             lastVgrid = V_grid;
536             lastXsurf = xSurf;
537             lastZsurf = zSurf;
538             NxLast = NxC;
539             NzLast = NzC;
540 
541             CreateAndSmoothMesh(V_grid, xSurf, zSurf, minVal
    , maxVal, NxC, NzC);
542             return;
543         }
544 
545         int NxF = (NxC - 1) * interpFactor + 1;
546         int NzF = (NzC - 1) * interpFactor + 1;
547         double passof = passo / interpFactor;
548 
549         List<double> xSurfFine = new List<double>();
550         for (int ixf = 0; ixf < NxF; ixf++)
551         {
```

```
552             double xx = (xmin + passo / 2) + ixf * passof;
553             xSurfFine.Add(xx);
554         }
555 
556         List<double> zSurfFine = new List<double>();
557         for (int izf = 0; izf < NzF; izf++)
558         {
559             double zz = (zmin + passo / 2) + izf * passof;
560             zSurfFine.Add(zz);
561         }
562 
563         double[,] V_grid_fine = new double[NzF, NxF];
564         for (int izf = 0; izf < NzF; izf++)
565         {
566             int jC = izf / interpFactor;
567             float alpha = (izf % interpFactor) / (float)
    interpFactor;
568             if (jC >= NzC - 1) jC = NzC - 2;
569 
570             for (int ixf = 0; ixf < NxF; ixf++)
571             {
572                 int iC = ixf / interpFactor;
573                 float beta = (ixf % interpFactor) / (float)
    interpFactor;
574                 if (iC >= NxC - 1) iC = NxC - 2;
575 
576                 double V00 = V_grid[jC, iC];
577                 double V01 = V_grid[jC, iC + 1];
578                 double V10 = V_grid[jC + 1, iC];
579                 double V11 = V_grid[jC + 1, iC + 1];
580 
581                 double vInterp = (1 - alpha) * (1 - beta) *
    V00
582                                  + (1 - alpha) * beta * V01
583                                  + alpha * (1 - beta) * V10
584                                  + alpha * beta * V11;
585 
586                 V_grid_fine[izf, ixf] = vInterp;
587             }
588         }
589 
590         lastVgrid = V_grid_fine;
591         lastXsurf = xSurfFine;
592         lastZsurf = zSurfFine;
593         NxLast = NxF;
```

```
594         NzLast = NzF;
595 
596         CreateAndSmoothMesh(V_grid_fine, xSurfFine,
    zSurfFine, minVal, maxVal, NxF, NzF);
597      }
598 
599      // ============================================
600      //   BLOCCO 3: RENDERING E VISUALIZZAZIONE
601      // ============================================
602 
603      private void CreateAndSmoothMesh(double[,] V_grid, List<
    double> xSurf, List<double> zSurf,
604                                       double minVal, double
    maxVal, int Nx, int Nz)
605      {
606          if (surfObject != null)
607              Destroy(surfObject);
608 
609          surfObject = new GameObject("SurfMesh");
610          MeshFilter mf = surfObject.AddComponent<MeshFilter
    >();
611          MeshRenderer mr = surfObject.AddComponent<
    MeshRenderer>();
612          mr.material = surfMaterial;
613 
614          surfCollider = surfObject.AddComponent<MeshCollider
    >();
615 
616          double range = maxVal - minVal;
617          if (range < 1e-12) range = 1e-12;
618 
619          float usedOffset = isPlanar ? verticalOffset2D :
    verticalOffset3D;
620 
621          Texture2D potTexture = new Texture2D(Nx, Nz,
    TextureFormat.RGBA32, false);
622          Vector3[] vertices = new Vector3[Nx * Nz];
623          Vector2[] uvs = new Vector2[vertices.Length];
624 
625          int idx = 0;
626          for (int j = 0; j < Nz; j++)
627          {
628              for (int i = 0; i < Nx; i++)
629              {
630                  double pot = V_grid[j, i];
```

```
631                 double normVal = (pot - minVal) / range;
632                 Color c = Color.Lerp(Color.blue, Color.red,
    (float)normVal);
633                 c.a = surfAlpha;
634                 potTexture.SetPixel(i, j, c);
635 
636                 float py = usedOffset + (float)normVal * (
    heightScale);
637                 float px = (float)xSurf[i];
638                 float pz = (float)zSurf[j];
639 
640                 vertices[idx] = new Vector3(px, py, pz);
641 
642                 float u = (Nx > 1) ? (float)i / (Nx - 1) : 0
    f;
643                 float v = (Nz > 1) ? (float)j / (Nz - 1) : 0
    f;
644                 uvs[idx] = new Vector2(u, v);
645 
646                 idx++;
647             }
648         }
649         potTexture.Apply();
650         mr.material.mainTexture = potTexture;
651 
652         List<int> triangles = new List<int>();
653         for (int j = 0; j < Nz - 1; j++)
654         {
655             for (int i = 0; i < Nx - 1; i++)
656             {
657                 int i0 = j * Nx + i;
658                 int i1 = i0 + 1;
659                 int i2 = (j + 1) * Nx + i;
660                 int i3 = i2 + 1;
661 
662                 triangles.Add(i0);
663                 triangles.Add(i3);
664                 triangles.Add(i1);
665 
666                 triangles.Add(i0);
667                 triangles.Add(i2);
668                 triangles.Add(i3);
669             }
670         }
671 
```

```
672         Mesh mesh = new Mesh();
673         mesh.indexFormat = UnityEngine.Rendering.IndexFormat
    .UInt32;
674         mesh.vertices = vertices;
675         mesh.uv = uvs;
676         mesh.triangles = triangles.ToArray();
677 
678         for (int s = 0; s < meshSmoothing; s++)
679         {
680             LaplacianSmooth(mesh);
681         }
682         mesh.RecalculateNormals();
683 
684         mf.mesh = mesh;
685         surfCollider.sharedMesh = mesh;
686     }
687 
688     private void LaplacianSmooth(Mesh mesh)
689     {
690         Vector3[] verts = mesh.vertices;
691         int[] tris = mesh.triangles;
692 
693         List<HashSet<int>> neighbors = new List<HashSet<int
    >>(verts.Length);
694         for (int i = 0; i < verts.Length; i++)
695             neighbors.Add(new HashSet<int>());
696 
697         for (int i = 0; i < tris.Length; i += 3)
698         {
699             int i0 = tris[i];
700             int i1 = tris[i + 1];
701             int i2 = tris[i + 2];
702 
703             neighbors[i0].Add(i1);
704             neighbors[i0].Add(i2);
705             neighbors[i1].Add(i0);
706             neighbors[i1].Add(i2);
707             neighbors[i2].Add(i0);
708             neighbors[i2].Add(i1);
709         }
710 
711         Vector3[] newPos = new Vector3[verts.Length];
712         for (int v = 0; v < verts.Length; v++)
713         {
714             if (neighbors[v].Count == 0)
```

```
715             {
716                 newPos[v] = verts[v];
717             }
718             else
719             {
720                 Vector3 sum = Vector3.zero;
721                 foreach (int n in neighbors[v])
722                     sum += verts[n];
723                 sum /= neighbors[v].Count;
724                 newPos[v] = sum;
725             }
726         }
727 
728         mesh.vertices = newPos;
729     }
730 
731     private double FindPotentialUnnormalized(double x,
    double z)
732     {
733         if (lastVgrid == null || lastXsurf == null ||
    lastZsurf == null)
734             return 0.0;
735 
736         double bestDist = double.MaxValue;
737         int bestI = 0, bestJ = 0;
738         for (int j = 0; j < NzLast; j++)
739         {
740             double zz = lastZsurf[j];
741             for (int i = 0; i < NxLast; i++)
742             {
743                 double xx = lastXsurf[i];
744                 double dx = xx - x;
745                 double dz = zz - z;
746                 double dist = dx * dx + dz * dz;
747                 if (dist < bestDist)
748                 {
749                     bestDist = dist;
750                     bestI = i;
751                     bestJ = j;
752                 }
753             }
754         }
755         return lastVgrid[bestJ, bestI];
756     }
757 
```

```
758      private double[] CreateConstantArray(double val, int
    length)
759      {
760          double[] arr = new double[length];
761          for (int i = 0; i < length; i++)
762              arr[i] = val;
763          return arr;
764      }
765
766      private double[] ConcatArrays(double[] a, double[] b)
767      {
768          double[] c = new double[a.Length + b.Length];
769          Array.Copy(a, 0, c, 0, a.Length);
770          Array.Copy(b, 0, c, a.Length, b.Length);
771          return c;
772      }
773
774      private double[] LinSpace(double start, double end, int
    count)
775      {
776          double[] arr = new double[count];
777          if (count == 1)
778          {
779              arr[0] = start;
780              return arr;
781          }
782          double step = (end - start) / (count - 1);
783          for (int i = 0; i < count; i++)
784          {
785              arr[i] = start + step * i;
786          }
787          return arr;
788      }
789
790      private double MutualResistanceTerm(double A, double[] X
    , double[] Y, double[] Z,
791                                          int i, int j, int
    segno)
792      {
793          double xDiff = X[j] - X[i];
794          double yDiff = Y[j] - Y[i];
795          double zDiff = Z[j] - Z[i];
796          double halfL = l / 2.0;
797
798          double num = xDiff + halfL
```

```
799                     + Math.Sqrt(Math.Pow(segno * yDiff, 2)
800                                 + Math.Pow(zDiff, 2)
801                                 + Math.Pow((xDiff + halfL),
    2));
802         double den = xDiff - halfL
803                     + Math.Sqrt(Math.Pow(segno * yDiff, 2)
804                                 + Math.Pow(zDiff, 2)
805                                 + Math.Pow((xDiff - halfL),
    2));
806         return A * Math.Log(num / den);
807     }
808
809     private double SelfResistanceTerm(double A, double[] X,
    double[] Y, double[] Z,
810                                       int idx, double dr,
    int segno)
811     {
812         double halfL = l / 2.0;
813         double xDiff = 0;
814         double yDiff = 0;
815         double zDiff = 0;
816
817         double num = xDiff + dr + halfL
818                     + Math.Sqrt(Math.Pow(segno * yDiff, 2)
819                                 + Math.Pow(zDiff, 2)
820                                 + Math.Pow((xDiff + halfL),
    2));
821         double den = xDiff + dr - halfL
822                     + Math.Sqrt(Math.Pow(segno * yDiff, 2)
823                                 + Math.Pow(zDiff, 2)
824                                 + Math.Pow((xDiff - halfL),
    2));
825         return A * Math.Log(num / den);
826     }
827
828     private double PotentialContribution(double A, double
    I_i,
829                                          double px, double py
    , double pz,
830                                          double xi, double yi
    , double zi,
831                                          double lLocal)
832     {
833         double halfL = lLocal / 2.0;
834
```

```
835         double num_s = (px - xi) + halfL
836                        + Math.Sqrt(Math.Pow(py - yi, 2)
837                                    + Math.Pow(pz - zi, 2)
838                                    + Math.Pow((px - xi) +
    halfL, 2));
839         double den_s = (px - xi) - halfL
840                        + Math.Sqrt(Math.Pow(py - yi, 2)
841                                    + Math.Pow(pz - zi, 2)
842                                    + Math.Pow((px - xi) -
    halfL, 2));
843         double Vs = A * I_i * Math.Log(num_s / den_s);
844
845         double num_i = (px - xi) + halfL
846                        + Math.Sqrt(Math.Pow((py + yi), 2)
847                                    + Math.Pow(pz - zi, 2)
848                                    + Math.Pow((px - xi) +
    halfL, 2));
849         double den_i = (px - xi) - halfL
850                        + Math.Sqrt(Math.Pow((py + yi), 2)
851                                    + Math.Pow(pz - zi, 2)
852                                    + Math.Pow((px - xi) -
    halfL, 2));
853         double Vi = A * I_i * Math.Log(num_i / den_i);
854
855         return (Vs + Vi);
856     }
857
858     private double[] SolveLinearSystem(double[,] A, double[]
    b)
859     {
860         int n = b.Length;
861         double[,] M = (double[,])A.Clone();
862         double[] B = (double[])b.Clone();
863
864         for (int k = 0; k < n - 1; k++)
865         {
866             double max = Math.Abs(M[k, k]);
867             int pivot = k;
868             for (int i = k + 1; i < n; i++)
869             {
870                 double val = Math.Abs(M[i, k]);
871                 if (val > max)
872                 {
873                     max = val;
874                     pivot = i;
```

```
875                     }
876                 }
877                 if (pivot != k)
878                 {
879                     for (int col = 0; col < n; col++)
880                     {
881                         double tmp = M[k, col];
882                         M[k, col] = M[pivot, col];
883                         M[pivot, col] = tmp;
884                     }
885                     double tmpB = B[k];
886                     B[k] = B[pivot];
887                     B[pivot] = tmpB;
888                 }
889                 for (int i = k + 1; i < n; i++)
890                 {
891                     double factor = M[i, k] / M[k, k];
892                     for (int col = k; col < n; col++)
893                     {
894                         M[i, col] -= factor * M[k, col];
895                     }
896                     B[i] -= factor * B[k];
897                 }
898             }
899 
900             double[] x = new double[n];
901             for (int i = n - 1; i >= 0; i--)
902             {
903                 double sum = B[i];
904                 for (int j = i + 1; j < n; j++)
905                 {
906                     sum -= M[i, j] * x[j];
907                 }
908                 x[i] = sum / M[i, i];
909             }
910             return x;
911         }
912 }
```

Bibliografia

[1] G. Cafaro et al., "Influence of LV neutral grounding on global Earthing Systems," 2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC), Rome, Italy, 2015, pp. 389-394

[2] E. Pons et al., "Global Earthing System: Can Buried Metallic Structures Significantly Modify the Ground Potential Profile?," in IEEE Transactions on Industry Applications, vol. 51, no. 6, pp. 5237-5246, Nov.-Dec. 2015

[3] F. Freschi, M. Mitolo and M. Tartaglia, "An Effective Semianalytical Method for Simulating Grounding Grids," in IEEE Transactions on Industry Applications, vol. 49, no. 1, pp. 256-263, Jan.-Feb. 2013

[4] O. Gervasi, D. Perri and M. Simonetti, "Empowering Knowledge With Virtual and Augmented Reality," in IEEE Access, vol. 11, pp. 144649-144662, 2023

[5] S. Criollo-C et al., "Improving Higher Education With the Use of Mobile Augmented Reality (MAR): A Case Study," in IEEE Access, vol. 12, pp. 139003-139017, 2024

[6] WP2-LA2, Influenza delle masse estranee sulla distribuzione delle correnti di guasto

[7] Azuma, R. (1997). "A Survey of Augmented Reality". Presence: Teleoperators and Virtual Environments, 6, 355-385

[8] Kaufmann, H., Schmalstieg, D. (2003). "Mathematics and geometry education with collaborative augmented reality". Computers Graphics, 27(3), 339–345

[9] Wu, H.-K., Lee, S. W.-Y., Chang, H.-Y., Liang, J.-C. (2013). "Current status, opportunities and challenges of augmented reality in education". Computers Education, 62, 41–49

[10] Chiang, T. H. C., Yang, S. J. H., Hwang, G. J. (2014). "Students' online interactive patterns in augmented reality-based inquiry activities". Educational Technology Research and Development, 62(5), 671–686

[11] Santos, M. J., Chen, A. J., Taketomi, T. (2016). "A review of augmented reality in education: Opportunities and challenges". International Journal of Learning Technology, 11(2), 75–89