Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Studio di sistemi di posizionamento inerziale tramite sensori su smartphone

Relatore:
Chiar.mo Prof.
Luciano Bononi

Correlatore:
Dott.
Luca Bedogni

Presentata da:
Marco Rondelli

Sessione III
Anno Accademico 2015/2016

Introduzione

Nel presente documento si descrive la progettazione e implementazione di Movement Control, un'applicazione android che sfruttando il sensore di accelerazione calcola lo spazio percorso.

I sistemi di localizzazione indoor wireless sono diventati molto popolari negli ultimi anni. Le applicazioni nel mondo reale di questa tecnologia sono molteplici, per elencarne alcune: localizzazione di personale medico in un'ospedale, localizzazione di un vigile del fuoco in un edificio incendiato. I progressi iniziali nella localizzazione indoor sono stati compiuti negli ultimi venti anni. Siccome é una tecnologia piuttosto nuova vi sono tante persone coinvolte attualmente. Nel campo della localizzazione la tecnologia maggiormente precisa é il GPS, l'unico problema derivante da questo sistema é quello che all'interno di un edificio non funziona perché il segnale non arriva. Per questo motivo sono presenti molteplici tecnologie nell'ambito della localizzazione indoor. I sistemi che sfruttano queste tecnologie sono innumerevoli, con requisiti e scopi differenti. Molti sistemi di localizzazione indoor infatti si basano su dispositivi da installare nell'ambiente, oppure su software particolari. É stato scelto di implementare un sistema con l'accelerometro perché é uno dei pochi dispositivi giá presenti in tutti gli smartphone e quindi non c'é il bisogno di investire soldi e neanche di installare trasmettitori in casa. In questa maniera il sistema é sfruttabile da chiunque lo voglia utilizzare.

Figura 1: Mercato localizzazione indoor [12]

L'idea alla base del sistema é che lo smartphone tenuto in mano dall'utente registra i dati dell'accelerometro e calcola lo spazio percorso durante la rilevazione. Nel documento verranno spiegate anche alcune migliorie attuate grazie allo studio del comportamento dell'algoritmo basato sulle formule fisiche.

Struttura del documento

Nel primo capitolo viene fatto un cenno al mondo della localizzazione, enunciando e spiegando i vari metodi esistenti per la creazione di un sistema di localizzazione. Nel particolare, é presente anche una spiegazione delle tecnologie che servono per l'implementazione dei sistemi. Infine, per ogni metodologia é stato citato almeno un sistema funzionante pubblicato.

Nel secondo capitolo si descrive lo studio generale che ha portato alla creazione dell'applicazione. Nel capitolo viene descritta l'implementazione del Web Service, creato per effettuare le analisi dei dati, e dell'algoritmo. Vengono inoltre analizzati i dati di rilevazione e vengono spiegate le misure adot-

tate per migliorare l'algoritmo. Per ultimo viene effettuata un'analisi del funzionamento dell'algoritmo e della sua precisione attraverso la creazione e discussione di grafici che analizzano l'errore ottenuto su molteplici rilevazioni.

Nel terzo capitolo infine c'é la descrizione dell'applicazione e della sua implementazione. Si forniscono screenshot dell'interfaccia e ci si sofferma sulla struttura interna.

Indice

Introduzione **i**

1 Stato dell'arte **1**

1.1 Dead-reckoning 1

1.2 Direct Sensing 3

1.3 Triangolazione 6

1.4 Riconoscimento Pattern 8

2 Studio Generale **11**

2.1 Web Service 11

2.1.1 Flask 12

2.1.2 Implementazione 13

2.2 Algoritmo 18

2.2.1 Formule Fisiche 18

2.2.2 Analisi Dati 19

2.3 Implementazione Algoritmo 22

2.3.1 Struttura 22

2.3.2 Analisi Risultati 24

3 Applicazione Android **29**

3.1 Descrizione Generale 29

3.1.1 Sensori 30

3.2 Implementazione 32

3.2.1 Strumenti Utilizzati 32

3.2.2 Struttura 33

4 Conclusioni **39**

Conclusioni **39**

4.1 Lavori Futuri 39

4.1.1 Introduzione del Sensore di Orientamento 39

4.1.2 Aggiunta di una mappa 40

4.1.3 Algoritmo step-length 40

4.1.4 Estensione funzionalitá del Web Service 40

Bibliografia **41**

Capitolo 1

Stato dell'arte

Esistono diverse tecnologie che permettono il rilevamento della posizione tramite smartphone. Generalmente si possono suddividere in quattro differenti tecniche: dead-reckoning, direct-sensing, triangolazione, riconoscimento pattern.

1.1 Dead-reckoning

É una tecnica basata sul rilevamento della posizione attraverso l'accumulo di dati conseguenti allo spostamento effettuato. Questi dati possono essere acquisiti tramite sensori come: accelerometro, magnetometro, bussola, giroscopio. Molto spesso questa tecnica viene utilizzata insieme alle altre perché altrimenti risulta poco accurata. In particolare, il dead reckoning é preferibile utilizzarlo negli spazi chiusi perché risulta piú efficace di altri metodi come il gps che a fatica prende in un edificio. Questa é anche la tecnica utilizzata dal mio progetto.

Per spiegare meglio questa tecnica descriveró l'implementazione del Pedestrian Dead Reckoning (PDR) [1] che si basa appunto sul rilevamento dello spostamento a piedi. L'implementazione del PDR é caratterizzata da quattro operazioni: riconoscimento dell'orientamento, filtro del rumore, rilevamento passi, stima della lunghezza del passo. Il primo avviene tramite un algoritmo che analizza le rilevazioni dell'accelerometro combinate con i sensori di magnetismo in modo tale da avere per ogni coordinata la sua accelerazione e capire cosí in quale inclinazione e direzione stiamo procedendo. Il filtro del rumore é neccessario perché i dati grezzi hanno all'interno accelerazione di gravitá e altri rumori non necessari e dannosi alla rilevazione effettiva. Il rilevamento passi é il corpo dell'algoritmo per il rilevamento, infatti si occupa di rilevare quanti passi compie la persona deducendolo dai dati ottenuti con l'accelerometro. Solo con l'ultima operazione peró si arriva a capire la distanza percorsa, infatti con l'algoritmo di stima della lunghezza del passo si possono calcolare i metri percorsi nella rilevazione. Con l'unione di queste tecniche si ottiene un valido sistema di dead-reckoning.

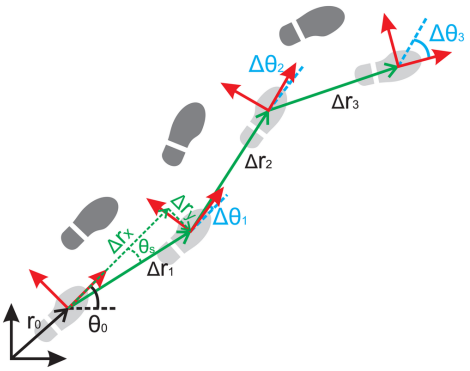


Figura 1.1: Step-length. [9]

[7] descrive un sistema di dead-reckoning basato su un dispositivo chiamato NavMote. L'articolo inizialmente descrive nel generale tutte le tecnniche possibili per la localizzazione, quasi tutti i metodi li che ho descritto anche io

nelle pagine a seguire, giungendo alla conclusione che il metodo migliore per un'accurato sistema di localizzazione é quello di utilizzare piú tecniche contemporaneamente, creando cosí il NavMote. Il NavMote ottiene dati mentre la persona sta passeggiando, i dati sono ottenuti grazie a due sensori presenti al suo interno, la bussola e l'accelerometro. Una volta ottenuti i dati si connette alla rete per inviarli a un server al cui interno c'é l'algoritmo di step-detection e di step-length estimation. Calcolata la distanza percorsa la si compara con un algoritmo di map matching per ottenere una precisione piú accurata.

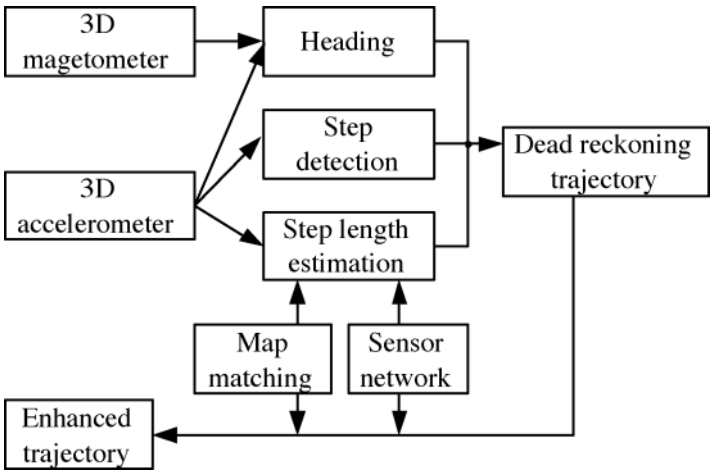


Figura 1.2: Algoritmo localizzazione. [7]

1.2 Direct Sensing

Si basa sulla presenza di "tag" preinstallati nell'ambiente, infatti i tag una volta rilevati e interrogati ci possono rispondere con la loro posizione e quindi deriviamo la nostra di conseguenza in base al calcolo della distanza coperta dal segnale ricevuto. Ci sono diverse tecnologie che possono essere utilizzate come tag, a scopo di esempio ne spiegheró alcune:

- *Radio Frequency Identifier Description (RFID)*

Il piú comune tag usato nel campo della localizzazione. Sono dei piccoli trasmettitori dotati anche di batteria autonoma a volte. Utilizzano il segnale radio per trasmettere e i piú avanzati possono anche incamerare dati. Importante é da considerarsi le diverse versioni di RFID che si possono utilizzare, infatti ci sono generalmente due tipologie: gli active RFID e i passive RFID. Gli active RFID possono rispondere alle query poste dall'utente e possono anche richiedere i dati dell'utente e salvarli in un database interno. I passive RFID possono solo rispondere alle query poste dal'utente. Naturalmente la differenza di prezzo fra i due dispositivi incide molto su quale tecnologia utilizzare.

Nell'articolo [4] vengono analizzati una serie di sistemi che sfruttano RFID per la localizzazione. Nello specifico viene descritto che per avere un sistema di localizzazione in generale é sufficiente avere dei RFID installati nell'ambiente e un lettore di RFID alla mano. L'articolo si sofferma sui problemi principali della tecnologia RFID come la propagazione del segnale, poi nella seconda parte propone delle possibili soluzioni ai problemi constatati. Infine dopo aver presentato un possibile algoritmo per il posizionamento che sfrutta il Kalman Filter hanno fatto un esperimento con passive RFID e l'algoritmo e hanno ottenuto una precisione di 0.055 metri mettendo peró un tag ogni 1.2 metri. Un buon risultato anche se i tag utilizzati devono essere tanti il costo sará contenuto considerato che sono passive RFID.

- *Infrarossi*

  I trasmettitori a infrarossi sono simili ai RFID ma utilizzano appunto gli infrarossi per comunicare. Un problema di questa tecnologia peró sta nella propagazione del segnale, infatti gli infrarossi sono facilmente schermabili.

  [8] analizza i problemi e le sfide di utilizzare un sistema di localizzazio-

ne basato unicamente su infrarossi. Il sistema descritto rileva il calore della persona all'interno dell'edificio e ne attua la localizzazione attraverso un database creato in precedenza con salvati la posizione dei sensori. Il maggior vantaggio di utilizzare questo sistema viene dato dal fatto che non c'é bisogno di indossare nessun dispositivo e neanche di utilizzare lo smartphone, infatti é tutto basato sui sensori installati all'interno dell'ambiente. Le principali problematiche sono risultate la riflessione delle onde e le radiazioni dinamiche del background. Il sistema proposto é semiautonomo, infatti é necessario un intervento umano di calibrazione in base al posto in cui si mettono i sensori visto che non é solo il corpo umano a emettere calore in una stanza.

- *Ultrasound Identification (USID)*

  Consiste nel mettere emettitori di onde a ultrasuoni nell'ambiente e indossare un ricevitore. É quindi possibile calcolare la posizione controllando il tempo di volo delle onde sonore ricevute. Un problema degli ultrasuoni peró sono sicuramente i muri che fanno rimbalzare l'onda sonora o la bloccano rendendo difficoltoso il calcolo della posizione.

- *Bluetooth beacons*

  Si utilizzano brevi messaggi trasportati attraverso il bluetooth per riuscire a decifrare la posizione in cui ci si ritrova. É necessario anche qui posizionare degli strumenti che mantengono la linea bluetooth sul percorso.

  [5] é un'analisi di come puó funzionare un sistema sfruttando dei bluetooth beacons. Il sistema descritto vuole ricavare la posizione in unsa stanza posizionando tre bluetooth beacons negli angoli e con un ricevitore catturarne il segnale. Il segnale catturato viene inviato a un servizio al computer che analizza il segnale dei tre emettitori e effet-

tuando una triangolazione riesce a dedurre le coordinate nella stanza. Naturalmente tutto questo é valido solo all'interno di una stanza unica altrimenti per ampliarlo a piú stanze bisognerebbe sfruttare anche gli ID unici dei bluetooth beacon per riuscire prima di tutto in che stanza ci si trova e poi calolarne la posizione. Il sistema presentato ha un margine di errore di due metri circa, attraverso analisi offline dei dati il margine di errore sono riusciti ad assottigliarlo fino a un metro.

- *Barcode*

  L'utente interessato alla localizzazione puó utilizzare anche i barcode, basta infatti portarsi dietro un lettore barcode. Leggendo i barcode, che devono essere giá presenti nell'area, il sistema puó localizzare l'utente grazie all'unicitá degli ID dei barcode. É una tecnica molto semplice che ha come problema il piazzamento precedente dei barcode e che l'utente deve andare alla ricerca di dove sono posizionati per poterli leggere.

1.3 Triangolazione

É la tecnica piú diffusa e anche la piú precisa. Infatti si basa sul rilevamento di tre coordinate per determinare la posizione attuale. Uno dei sistemi di triangolazione che sfrutta la triangolazione é il GPS che si serve della latitudine, longitudine e altitudine per triangolare la nostra posizione. Un altro metodo é quello di utilizzare il wifi fingerprint, sfrutta il calcolo della potenza dei wifi presenti in zona per riuscire a calcolarne la distanza da ognuno e triangolare la posizione attuale avendo almeno tre wifi nei pressi. La triangolazione é facilmente attuabile anche usando i RFID spiegati precedentemente. Un ulteriore metodo é quello di sfruttare il segnale delle torri cellulari per calcolarne la distanza da essa e dedurne la posizione, peró questo metodo risulta molto meno preciso del GPS e il suo utilizzo sarebbe sempre nell'ambiente esterno dove quindi il GPS non ha problemi di rendimento.

Il problema di ricezione del GPS é stato studiato da [2] riuscendo a trovare un algoritmo che anche senza segnale riesce ad essere preciso per qualche minuto. Nel dettaglio il problema analizzato dai ricercatori é stato quello della perdita di segnale dovuta al paesaggio cittadino mutevole mentre si viaggia in macchina. Utilizzando solo le informazioni ricevute dal GPS vengono fatte delle assunzioni che portano alla buona riuscita dell'algoritmo: l'altezza del veicolo non varierá significativamente, come anche la velocitá del veicolo, infatti si presuppone che la mancanza di segnale sia di qualche minuto e non di piú considerato che il veicolo é in movimento. Normalmente si utilizzano quattro satelliti per avere la rilevazione GPS, sfruttando le assunzioni fatte precedentemente si puó ottenere un valido algoritmo anche se non si ha nemmeno un satellite in ricezione. L'autore dell'articolo ha specificato e sperimentato l'utilizzo di una tecnica differente per tutti i casi in cui ci si puó ritrovare, sia dal non avere nessuna ricezione ad aver tre satelliti.

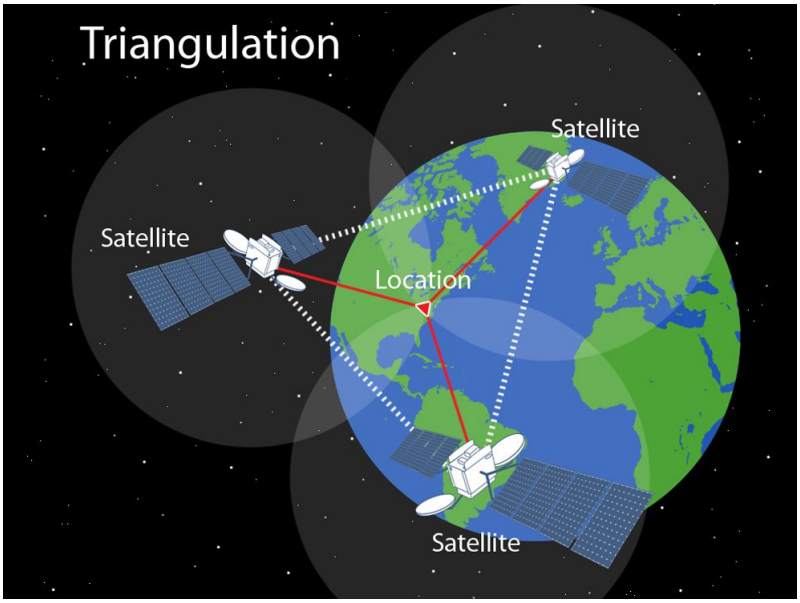


Figura 1.3: Triangolazione GPS. [10]

1.4 Riconoscimento Pattern

Metodo che sfrutta altre tecnologie come quella del posizionamento di sensori per rilevare la posizione, ma in piú si basa su dei dati creati precedentemente e salvati in memoria. Ci sono diversi dati che possono essere utili da salvare, come una mappa dove sono posizionati i sensori oppure un database con delle immagini dell'ambiente in modo tale da riconoscere attraverso le immagini in che posto ci si trova. Un problema di questa tecnica é sicuramente il grande spazio di archiviazione necessario per le immagini o i diversi dati necessari da mantenere in memoria. Nel caso delle immagini é anche necessario fornirsi di un software in grado di fare il riconoscimento e quindi c'é bisogno di una tecnologia abbastanza particolare. Un altro metodo si basa sul wifi fingerprinting, che consiste in due fasi, offline training e online operating. Nella fase offline l'utente misura il segnale Wifi di diversi access point, i record di queste misurazioni vengono salvati all'interno di un database assieme alla posizione dei dispositivi. Durante la fase online l'utente si localizzerá attraverso un'applicazione che misura la potenza dei segnali wifi, l'applicazione poi interroga il server su cui si trova l'algoritmo che riconosce la posizione attraverso il controllo incrociato di dati inviati dall'app e dei dati presenti nel database.

[3] propone un metodo per il riconoscimento della posizione attraverso la fotocamera del cellulare. Il sistema é molto semplice e funzionale, infatti necessita solo di un cellulare e di un database di immagini creato precedentemente. Il databasae necessita di almeno dieci immagini per ogni angolo della struttura. Durante la rilevazione bisogna tenere il cellulare legato al collo comeuna collana perché le immagini vengono prese periodicamente e inviate al database che risponderá dopo averle comparate con quelle giá presenti. L'algoritmo si basa su tre analisi per riconoscere l'immagine: analisi dei colori, analisi del negativo e analisi delle ombre, ogni analisi conferisce dei punti all'immagine che si sta comparando. Le sperimentazioni effettuate dal team nel riconoscimento delle stanze sono risultate accurate fino al 90% dei casi. É un dato molto interessante visto che questa tecnologia non necessita

di installazioni di sensori o di hardware particolarmente avanzato.

L'articolo [6] analizza i problemi legati alla privacy che possono scaturire utilizzando il Wifi fingerprinting. I problemi rilevati possono essere di due tipi, quelli per il cliente e quelli per il service provider. Il cliente naturalmente perde la privacy nel momento in cui effettua le query sul server e quindi tutto questo lo puó esporre a problemi come spam, pubblicitá, ritorsioni economiche. Il service provider invece é interessato a tenere il suo database al sicuro visto il costo dell'infrastruttura. Privacy Preserving WiFi Fingerprint-based Localization scheme (PriWFL) é quello che propongono gli autori di questo articolo, si basa sul criptare i dati che vengono trasmessi attraverso il Pailler cryptosystem, un sistema di criptazione scelto fra quelli esistenti. L'algoritmo in generale consiste di due fasi, la fase di preparazione e la fase di localizzazione. Nella fase di preparazione il client crea la chiave pubblica e privata utilizzando Pailler e la invia al server con i dati criptati ottenuti dal Wifi, nello stesso momento il client scarica dei metadati che il server invia per effettuare la localizzazione. La seconda fase consiste nel ricevere i dati computati e criptati dal server e una volta decriptati utilizzando la chiave segreta é possibile fare la comparazione con i metadati ricevuti inizialmente e cosí ottenere la localizzazione. L'algoritmo viene descritto con precisione nell'articolo e mostra anche delle prove effettuate con successo nell'ambiente reale. Infine hanno proposto anche un algoritmo piú efficiente per abbassare il costo computazione che deve effettuare il client sul suo dispositivo.

Triangolazione	GPS	ambienti esterni
	Torri-cellulare	ambienti esterni, scarsa precisione
	WLAN	ambienti interni
	RFID	installazione disp. nell'ambiente
Riconoscimento Pattern	Computer Vision	algoritmo complicato
	Signal Distrib	richiede memorizzazione dati
Direct Sensing	RFID	installazione disp. nell'ambiente
	Infrarossi	
	Ultrasuoni	
	Bluetooth	
	Barcode	
Dead-Reckoning	Sensori	disponibilitá dei sensori

Tabella 1.1: Riassunto delle tecnologie, dei metodi di localizzazione e dei problemi possibili

Ricapitolando quindi nel caso bisogni fare delle rilevazioni in ambienti esterni la tecnologia piú precisa attualmente é il GPS, e anche quella preferibile considerato il basso costo e la disponibilitá in tutto il mondo. Invece dovendo fare misurazioni all'interno di edifici non si ha un metodo di molto migliore come per l'esterno, infatti sviluppando adeguatamente e anche tenendo conto delle possibilitá é preferibile attuare una tecnica o un altra. Bisogna infatti ricordare che sensori come RFID, ultrasuoni e altri bisogna installarli all'interno dell'ambiente di rilevazione, sebbene quindi siano piú precisi non sempre é possibile utilizzarli. Optare invece per l'utilizzo di sensori piú comuni presenti negli smartphone puó essere la soluzione piú adatta anche se quella che si risulterá meno precisa.

Capitolo 2

Studio Generale

In questo capitolo verranno descritti i processi che hanno permesso alla creazione di due algoritmi di localizzazione basati sull'accelerometro di uno smartphone comune. Uno sfrutta le formule di fisica e l'altro sfrutta la step-detection. Gli algoritmi sono stati sviluppati grazie all'analisi di dati provenienti da molteplici rilevazioni effettuate da un'applicazione creata appositamente per la raccolta dati su android. Le rilevazioni vengono inviate poi a un web service per essere analizzate. Infine l'applicazione per smartphone é stata aggiornata ed é stato incluso l'algoritmo sviluppato sul computer. A seguire sará spiegato nel dettaglio l'implementazione del web service, l'analisi che é stata fatta sui dati e la descrizione degli algoritmi sviluppati al computer.

2.1 Web Service

É stato implementato un web service per poter controllare i dati raccolti dall'applicazione su smartphone. Il web service mostra un grafico per ogni rilevazione effettuata. Il grafico ha sull'asse delle x il tempo e su quello delle y il valore di accelerazione. Facendo un'analisi approfondita di questi grafici é stato possibile arrivare a sviluppare un algoritmo per la localizzazione.

2.1.1 Flask

Il web service é stato implementato utilizzando come base Flask, un microframework open source per Python. Flask é un tool molto semplice e intuitivo sfruttato per creare un server con un sito web che mostra i dati raccolti dall'applicazione per android. Il tool é disponibile per qualunque sistema operativo e ció rende il web service multi piattaforma. Il framework risulta molto semplice da installare e sul sito ufficiale é possibile trovare documentazione e tutorial. Essendo scritto per python vi sono anche tutti i vantaggi che derivano dall'utilizzo di questo linguaggio, fra cui l'utilizzo di tutte le librerie semplicemente importandole. La visualizzazione di pagine html é gestita con una sola chiamata di funzione e quindi anche questo rimane molto semplice. Sotto é riportato un segmento di codice Flask per rendere l'idea della semplicitá del codice.

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

Listing 2.1: "semplice esempio di codice in flask che renderizza un template html"

2.1.2 Implementazione

L'implementazione del web service grazie a Flask rimane piuttosto leggera e con poche righe di codice; infatti per tutte le operazioni necessarie é bastato solo un file in python ed un file html.

Lato Server

Andando maggiormente nel dettaglio a lato server, il file python ha quattro funzioni:

- **home()** : é la funzione base che renderizza il template html e che mostra nel browser la pagina.
- **receive()** : si tratta della funzione che si occupa di attendere una POST all'indirizzo `http://indirizzoserver:5000/receive` con all'interno i file dati; la funzione poi salva i dati in file creati all'interno della cartella `filerilevazioni`.
- **cercatuttifile()** : caricando la pagina dal browser automaticamente parte una richiesta AJAX che viene gestita da questa funzione. Alla chiamata viene fatta una ricerca all'interno della cartella `filerilevazioni`; se all'interno ci sono file .txt sono considerati file di dati ed il loro titolo é inviato all'interno di una lista come risposta alla richiesta AJAX. Il titolo dei file andrá a riempire una select della pagina html.
- **cercafile()** : viene chiamata quando arriva una POST all'indirizzo `http://indirizzoserver:5000/cercafile`. Si tratta di una richiesta AJAX inviata dal client con il nome del file che vogliamo sia visualizzato. Il file viene letto e salvato in una stringa che viene inviata come risposta al client.

```
@app.route('/cercafile' , methods = ['POST'])
def cercafile():
    if request.method == 'POST':
        text = request.form["nomefile"]
        if text in os.listdir(os.getcwd()):
            str = open(text, 'r').read()
            return str
        else :
            return "non esiste il file"
```

Listing 2.2: "funzione cercafile() lato server"

Sopra si puó leggere il codice della funzione cercafile. Nel dettaglio, la prima riga é il codice Flask che specifica per quale indirizzo e per quale richiesta HTTP la funzione verrá chiamata, il resto é Python e si comporta come é stato descritto sopra.

Lato Client

Per l'implementazione dell'interfaccia web é stato utilizzato HTML come linguaggio di markup e javascript per effettuare le chiamate AJAX e creare il grafico. La pagina HTML é bianca con una select che viene riempita dinamicamente dal javascript durante il caricamento della pagina. La select viene riempita con i nomi dei file presenti all'interno della cartella `filerilevazioni`. É possibile selezionare questi file e quando questo avviene comparirá il grafico creato per l'analisi dei dati.

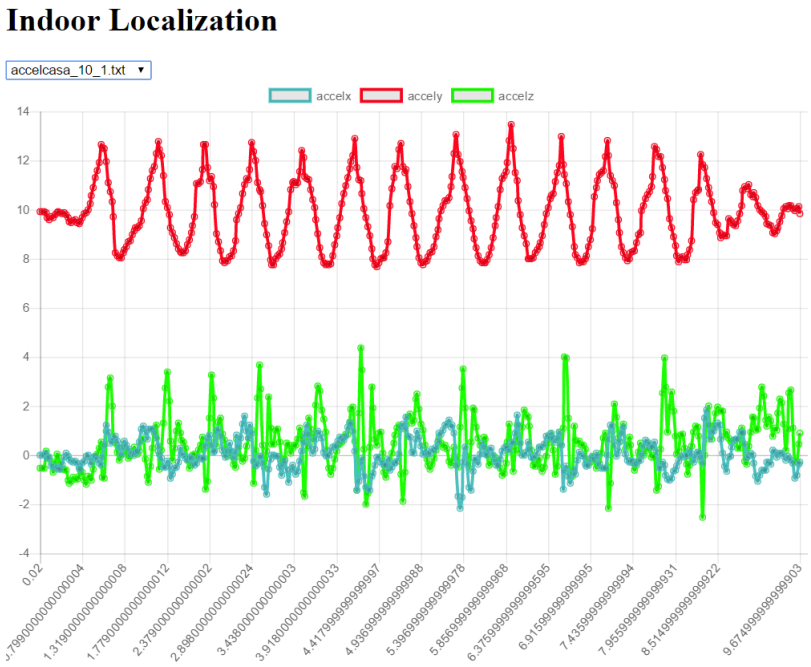


Figura 2.1: Vista lato client

La figura mostra un grafico dove si puó analizzare una rilevazione effettuata dal sensore dell'accelerometro. I dati vengono mostrati nel grafico che ha sulle asse delle x il tempo e sull'asse y i valori di accelerazione. L'accelerometro per ogni spostamento rileva i dati rispettivi all'asse x, y e z derivanti dallo spostamento dello smartphone nello spazio 3D. I dati vengono quindi mostrati nel grafico che ha sulle asse delle x il tempo e sull'asse y i valori di accelerazione.

La realizzazione della vista 2.1 é stata implementata attraverso il javascript. Nel dettaglio le comunicazioni client-server sono state gestite attraverso **AJAX** e la realizzazione del grafico é avvenuta sfruttando l'API open-source basata su javascript **Chart.js**.

- **AJAX** : acronimo di Asynchronous JavaScript and XML. É una tecnica di sviluppo software per la realizzazione di applicazioni web interattive. E grazie ad AJAX non c'é il bisogno di ricaricare la pagina

per aggiornarla perché la comunicazione avviene in maniera asincrona e interamente in background. La libreria jQuery implementa un'interfaccia semplice per eseguire richieste AJAX. Nel web service é stato utilizzato per effettuare richieste HTTP al server Flask. Il codice delle funzioni che sfrutta le chiamate AJAX per fare richieste al server é molto semplice e chiaro. I nomi delle funzioni che effettuano le richieste HTTP sono uguali a quelle che prendono carico della loro domanda nel server e quindi ci sono due funzioni che si chiamano rispettivamente `cercafile()` e `cercatuttifile()`.

```
function cercafile(){
        $.ajax({
                type: "POST" ,
                url: $SCRIPT_ROOT+ "/cercafile" ,
                data: {
                   "nomefile": document.
                        .getElementById("select")
                        .value} ,
                success: function(data) {
                         puliscitutto()
                         processData(data)
                },
                dataType : "text"
        })
}
```

Listing 2.3: "funzione cercafile() lato client"

Per completezza sopra é stato inserito il codice della funzione cercafile lato client, in modo tale da comprendere meglio come vengono effettuate le richieste AJAX. In caso di successo é possibile notare che vengono chiamate due funzioni `puliscitutto()` e `processData()`. Molto semplicemente la funzione `puliscitutto()` si occupa di pulire gli array di

dati nel caso non fosse il primo file che viene scelto; invece la funzione `processData()` analizza i dati arrivati dal server e riempie gli array vuoti effettuando uno scrape del file .txt.

- **Chart.js** : é l'API open-source sfruttata per creare il grafico all'interno della pagina web. Grazie a questa libreria é stato possibile creare il grafico dinamicamente tramite Javascript. La funzione denominata `creategraph()` si occupa di prendere i dati giá divisi nei rispettivi array e chiamare la funzione `Chart()` per renderizzare il grafico. La creazione del grafico é molto semplice; infatti avendo giá i dati separati per array la funzione `Chart()` permette di riempire il grafico con dei dataset cosí composti :

```
{
        label:"accely",
        borderColor: "rgba(255,5,30,1)",
        data: accely
    }
```

Listing 2.4: "esempio di dataset"

In **data** viene passato l'array con i dati, in questo caso i dati dell'accelerazione sull'asse y, poi ci viene permesso di scegliere il colore della retta nel grafo attraverso il sistema rgba e di aggiungere una label per creare una legenda.

2.2 Algoritmo

In questa sezione verrá spiegato l'algoritmo e piú nel dettaglio anche come si é arrivati a questo livello di sviluppo. Innanzitutto vi é da precisare che la base del progetto era la creazione di un algoritmo basato sulle formule fisiche che si possono utilizzare per derivare la distanza percorsa con l'accelerometro. Al fine di una comparazione é stato sviluppato anche un altro algoritmo basato sulla step-detection. Grazie allo sviluppo del secondo algoritmo é stato possibile effettuare delle analisi piú approfondite sul grado di precisione di entrambi. Di seguito verrá spiegato anche l'algoritmo dello step-detection.

2.2.1 Formule Fisiche

Per iniziare é stato necessario informarsi su quali formule di fisica era giusto utilizzare per poter calcolare lo spazio percorso in un determinato lasso di tempo. I dati rilevati dal sensore di accelerazione da sfruttare sono due: accelerazione istantanea e timestamp:

- **Accelerazione istantanea** – rilevata quando avviene un cambio nella velocitá di spostamento dello smartphone. L'accelerometro rileva quanto é stato forte e viene suddiviso nei tre assi x, y e z.
- **Timestamp** – é l'istante in cui avviene il cambio di velocitá. Non é quanto dura l'accelerazione ma l'istante in cui viene rilevata.

Sfruttando solo questi due dati é possibile ricavare lo spazio percorso dal primo rilevamento all'ultimo. Considerato che il sensore rileva l'accelerazione sui tre assi, inizialmente bisogna unire le accelerazioni e ottenerne una sola generale. Questa operazione si effettua grazie a una formula chiamata magnitude:

$$magn = \sqrt{(accelerazionex + accelerazioney + accelerazionez - 9.81)^2}$$

Grazie a questa formula viene ricavata l'accelerazione generale. Qui sopra é stata presentata la magnitude necessaria per il sensore accelerometro che ha

al suo interno anche l'accelerazione di gravitá dovuta alla Terra. L'applicazione per smartphone, come verrá spiegato nel prossimo capitolo, avrá anche le rilevazioni dell'accelerometro lineare che si discosta dall'accelerometro comune perché grazie ad un'operazione di un software elimina l'accelerazione di gravitá. Allo scopo del nostro algoritmo é sufficiente togliere la sottrazione per 9.81 dovuta alla gravitá, durante la creazione dell'algoritmo finale molto semplicemente é stata effettuata la scelta di utilizzare i dati dell'accelerometro comune. Per ottenere la velocitá effettiva verrá utilizzata la formula :

$$Vel = accelerazione * tempo$$

dovrá essere applicata per ogni intervallo di tempo tra una rilevazione e l'altra. Per calcolare il tempo bisogna prendere il timestamp della rilevazione successiva e sottrarlo a quello della rilevazione interessata, cosí si ottiene il tempo per cui il valore di accelerazione non é mutato.
Una volta effettuata questa operazione, si avranno tante velocitá quante sono state le rilevazioni e per ricavarne lo spazio finale basta effettuare la somma di tutte:

$$Spazio = \sum V1 + V2 + ... + Vn$$

dove n sono tutte le Velocitá ottenute.

2.2.2 Analisi Dati

Al termine del primo concept di algoritmo basato unicamente sulle formule di fisica spiegate sopra, é apparso chiaro che la precisione non era sufficiente; infatti nelle distanze relativamente corte come tre o quattro metri l'errore rimaneva intorno al metro, invece quando si percorreva qualche metro in piú l'errore diventava di tre o quattro metri. É stata presa la decisione di effettuare rilevazioni mantenendo lo smartphone fisso in una posizione per rendere minimo il "rumore" di fondo. Inoltre sfruttando la visualizzazione dei grafici attraverso il web service, sono state effettuate diverse considerazioni al fine di migliorare l'algoritmo attraverso una scraping dei dati.

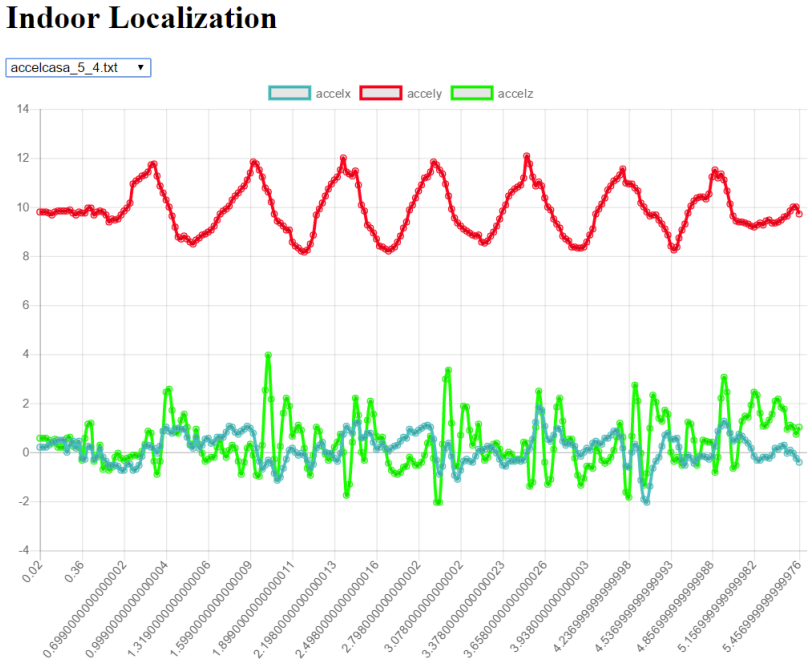


Figura 2.2: Esempio per confrontare le supposizioni effettuate

Eliminazione Asse Y

É stato assunto che durante le rilevazioni si mantenesse sempre lo smartphone in verticale per poter identificare l'asse su cui é presente l'accelerazione di gravitá ed eliminarlo dai dati. In questo caso controllando il grafico 2.2 risulta chiaro che sia l'accelerazione sull'asse y che verrá scartata.

Scarto Dati

Analizzando i grafici é stato possibile capire che all'inizio e alla fine di ogni rilevazione vengono individuati degli spostamenti minimi che sono di assestamento per iniziare e concludere lo spostamento. É quindi stato ritenuto opportuno scartare il primo mezzo secondo e l'ultimo di ogni file di rilevazione.

Step-detection

Analizzando i grafici ottenuti dal web service é stato possibile ottenere un algoritmo di step-detection. Infatti guardando l'accelerazione sull'asse y é stato possibile contare i passi chiaramente per ogni rilevazione effettuata. Per esempio nel caso della Figura 2.2 si possono contare sette passi. Grazie a queste analisi si é potuto poi provare anche la precisione di un algoritmo che sfrutta la step-detection effetttuando delle prove con lunghezza del passo variabile.

2.3 Implementazione Algoritmo

L'algoritmo é stato implementato interamente in python, l'infrastruttura generale é basata su piú moduli collegati. Durante la creazione dell'algoritmo é venuto naturale suddividere in moduli per fare in modo da rendere accessibili tutte le funzionalitá anche singolarmente.

2.3.1 Struttura

La struttura é stata divisa in tre moduli, contando entrambi gli algoritmi.

- **sistemacentrale.py** – come si puó dedurre dal nome é la base di tutto il sistema. Contiene al suo interno l'inizializzazione degli array che vengono riempiti dalla funzione `leggifile(path, nomefile)`. La funzione molto semplicemente apre il file passato come argomento efd effettua uno scrape per inserire i dati negli appositi array

- **algoritmoaccel.py** – é il modulo in cui avvengono i calcoli spiegati nella sezione precedente. Ha infatti al suo interno `calcolamagnaccel()` e `calcoladistanza(accelmagnitude)`, rispettivamente una calcola la magnitude e l'altra calcola lo spazio percorso. Nello specifico in questo modulo, chiamandolo da terminale, é possibile vedere quanta distanza sarebbe calcolata se tenessimo conto di tutte le combinazioni di ogni asse.

  É facilmente controllabile dai dati emersi dal calcolo che alcune misurazioni sono piú precise di altre. Partendo da un'analisi visiva come questa, si é potuto escludere la maggior parte delle rilevazioni tenendo solo quella degli assi xz come spiegato precedentemente.

Per rendere un'idea della realizzazione in python, di seguito vi é il codice della funzione che effettivamente calcola lo spazio:

```
def calcoladistanza(accelmagnitude):
    velocita = []
    y = 0
    for x in range(25,len(sistcent.tempo)-25):
        velox = float(accelmagnitude[y])*
                    *float(sistcent.tempo[x])
        velocita.append(velox)
        y = y+ 1
    spazio = 0
    for x in range(len(velocita)):
        spazio = spazio + velocita[x]
    return spazio
```

Listing 2.5: "funzione calcoladistanza(accelmagnitude)"

La funzione prende come argomento un array dove in precedenza devono essere state calcolate le magnitude delle accelerazioni. All'interno esegue le formule fisiche per ricavare lo spazio e poi lo ritorna.

- **algoritmopassi.py** – modulo in cui é presente una sola funzione che si occupa di contare quanti passi sono stati effettuati durante lo spostamento. La step-detection viene fatta sui valori dell'accelerazione di asse y; viene considerato un passo ogni volta che si sale sopra al valore 11 di accelerazione e fino a che non torna almeno una volta sotto a quel valore non si contano altri passi.

2.3.2 Analisi Risultati

Per effettuare un'analisi dei risultati ottenuti con l'algoritmo implementato, é stato creato un modulo in python che sfrutta quelli spiegati nella sezione precedente per creare due grafici. I due grafici sono stati creati per poter ossservare la precisione dei due algoritmi e per compararli.

mediaerr.py

É il modulo che viene chiamato per ottenere un'analisi di tutte le rilevazioni effettuate. Al termine di questo vengono creati due grafici che riuniscono le rilevazioni e mostrano la media di errore dei risultati. I grafici vengono creati sfruttando matplotlib e plotly, due librerie open-source per creare grafici. Le funzioni principali di questo modulo sono `calcolamediaerrore()` e `creategraph()`. `Calcolamediaerrore()` si occupa di prendere le distanze reali e confrontarle con lo spazio calcolato per poi ritornare la media per ogni distanza percorsa. Avendo effettuato molteplici rilevazioni della stessa distanza é possibile ottenere un valore medio di errore che rispecchia quanto si avvicina l'algoritmo alla distanza reale. `Creategraph()` sfrutta le medie calcolate prima e crea due grafici che possano dare l'idea della precisione degli algoritmi.

Il primo grafico mostra la media dell'errore rispetto alla distanza reale

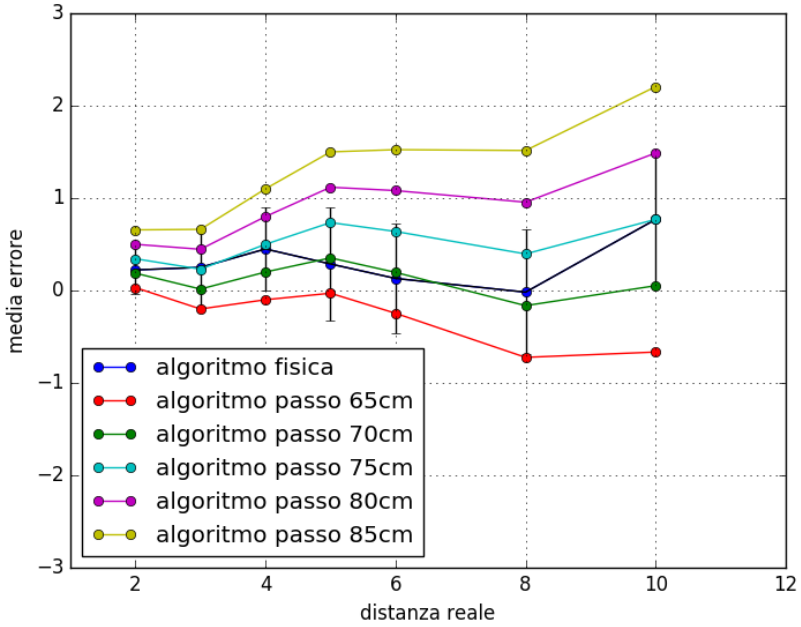


Figura 2.3: Grafico distanza reale - media errore

Nel grafico mostrato sopra sono stati inseriti i due algoritmi di calcolo dello spostamento. Seguendo la legenda, infatti, é possibile capire che la linea blu é l'algoritmo che sfrutta le formule fisiche, mentre le altre linee sono quelle che sfruttano la step-detection. L'algoritmo che sfrutta la step-detection non ha al suo interno un algoritmo di step-length e quindi sono state create diverse supposizioni sulla lunghezza del passo. Ogni riga mostra quanto la media dell'errore puó cambiare se si prendono in considerazione lunghezze di passi diverse. Nel grafico sono state mostrate le lunghezze da 65 cm a 85 cm con scarto di 5 cm per ogni riga. É facile concludere che mediamente l'algoritmo che sfrutta le formule fisiche e le supposizioni presentate nella sezione precedente, risulta piú preciso di quello che si basa sulla step-detection. Dal grafico peró si nota anche che la linea verde risulta maggiormente precisa in media. Questo significa che l'utente che ha effettuato le rilevazioni aveva un passo di circa 70 cm e che se l'algoritmo avesse avuto una sezione di

step-length efficente probabilmente sarebbe stato maggiormente preciso dell'algoritmo basato sulle formule fisiche. Per concludere é possibile osservare che l'algoritmo generale ha una precisione che in media rientra nel metro, quindi molto buona. Bisognerá effettuare rilevazioni su distanze maggiori per capire quanta precisione mantiene.

Il secondo grafico invece mostra un boxplot in cui per ogni rilevazione é stato calcolato l'errore preciso ed é stato raggruppato per avere un'idea di quanto variano le rilevazioni.

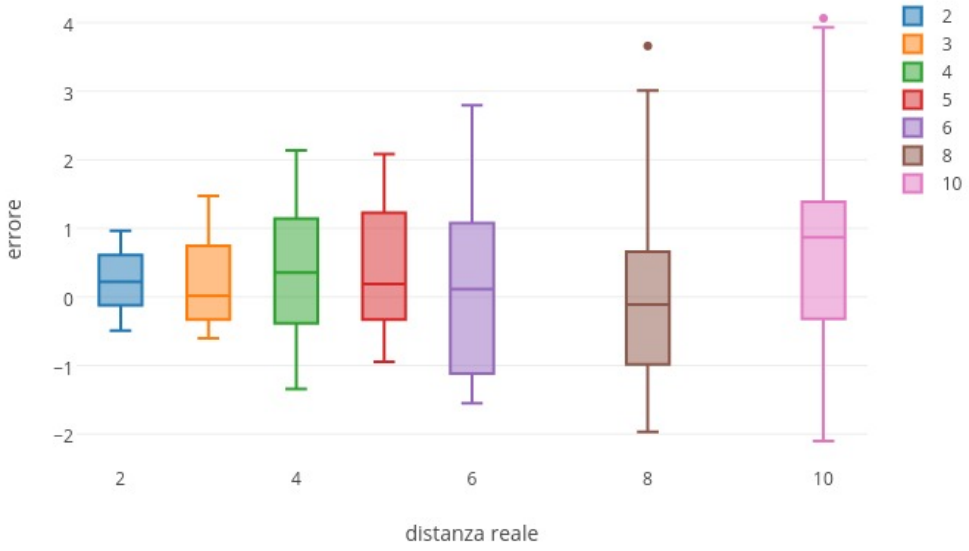


Figura 2.4: Boxplot distanza - errore

In questo grafico é possibile effettuare diverse supposizioni. La prima é che maggiore é la distanza, maggiore puó risultare l'errore. Considerato che il riquadro rimane all'interno del metro di precisione, significa che l'errore, in media anche nelle grandi distanze, rimane di un metro. Capita peró che ci siano alcune rilevazioni che diventano imprecise e questo lo si deduce dalla

errorbar molto piú lunga rispetto alle distanze corte. Sarebbe interessante vedere se effettuando molte piú rilevazioni a grandi distanze il box si restringe oppure rimane invariato.

É interessante chiarire che questi due grafici sono stati ricavati effettuando circa trenta rilevazioni per ogni distanza presente. Questo numero di rilevazioni é stato considerato sufficiente per avere un'idea generale dell'efficienza dell'algoritmo. Riassumendo, l'algoritmo basato sulle formule fisiche ha una precisione che si aggira intorno ad un metro; invece l'algoritmo basato sulla step-detection rimane sensibilmente variabile a causa della differenza che deriva dalla step-length. La precisione della step-detection quindi varia dal mezzo metro fino a raggiungere i due metri nel caso pessimo.

Capitolo 3

Applicazione Android

In questo capitolo verranno spiegati i processi dello sviluppo dell'applicazione per smartphone a partire dalla sua descrizione e dai requisiti richiesti.

3.1 Descrizione Generale

Movement Control é un'applicazione per smartphone implementata per dispositivi Android, le sue funzionalitá sono state aggiornate durante lo sviluppo del progetto e infine é stato aggiunto l'algoritmo basato sulla fisica per poter calcolare lo spazio percorso senza doversi appoggiare a servizi esterni. L'applicazione offre le seguenti funzionalitá:

- Rilevare gli spostamenti con l'accelerometro
- Calcolare la distanza percorsa sia con accelerometro che con accelerometro lineare
- Possibilitá di immettere un valore per cui i dati con valori inferiori vengano esclusi
- Salvare i dati ottenuti da una rilevazione in un file CSV .txt
- Inviare i file al web service spiegato nei capitoli precedenti

3.1.1 Sensori

I sensori che sono stati sfruttati per effettuare i rilevamenti degli spostamenti sono due: accelerometro e accelerometro lineare.

Accelerometro

É il sensore piú diffuso negli smartphone, viene sfruttato per capire la posizione in cui il cellulare si trova, é usato anche nei giochi, in particolare in quelli basati sulla simulazione di guida dove viene richiesto di inclinare lo smartphone. É possibile catturare i dati rilevati da questo sensore come é stato fatto con Movement Control. I dati che rileva il sensore sono suddivisi sui tre assi x, y e z; infatti se il cellulare viene tenuto in verticale avremo l'asse y che rileva gli spostamenti in altezza, l'asse x per gli spostamenti a destra e sinistra e l'asse z per la profonditá. Per capire meglio di seguito é stata inserita la figura della descrizione degli assi.

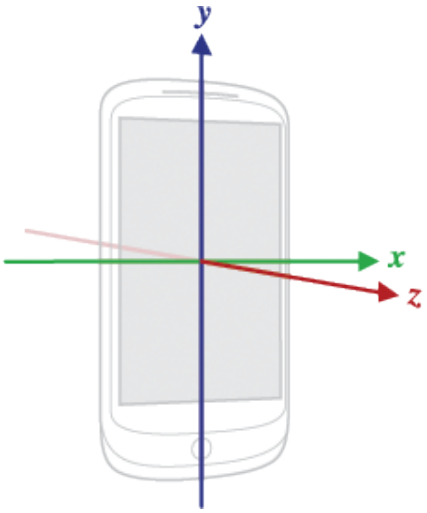


Figura 3.1: Descrizione visiva degli assi [11]

Accelerometro Lineare

Non si tratta di un vero e proprio sensore in piú perché non ha un hardware unico; infatti sfrutta i dati rilevati dall'accelerometro per analizzarli attraverso un software che scarta l'accelerazione di gravitá, sempre presente nell'accelerometro. I dati rilevati per il resto non cambiano rispetto all'accelerometro. Per il progetto inizialmente si é pensato di sfruttare questo sensore ma dopo aver effettuato diverse prove si é notato che attraverso l'eliminazione della gravitá sarebbe risultato piú difficile scartare altri dati e anche implementare l'algoritmo di step-detection.

3.2 Implementazione

In questa sezione sará spiegato quali strumenti sono stati utilizzati per creare l'applicazione, quali sono i requisiti e la sua struttura interna.

3.2.1 Strumenti Utilizzati

L'applicazione é stata sviluppata per Android in linguaggio Java sfruttando gli strumenti riportati di seguito.

Android SDK

Android SDK (Software Development Kit) é il pacchetto di sviluppo attraverso il quale é possibile sfruttare le API ufficiali Android. É grazie alla presenza di queste librerie che é stato possibile ottenere i valori dell'accelerometro.

Android Studio

É l'IDE ufficiale messo a disposizione da Android per permettere di scrivere codice e farlo eseguire sugli smartphone. La piattaforma mette a disposizione un emulatore per poter testare le applicazioni ma vista la natura dell'applicazione Movement Control non é stato possibile sfruttarlo a dovere e quindi tutti i test sono stati effettuati da un ONEPLUS 3 con all'interno Android 7.0.

Volley

La libreria HTTP sfruttata per le comunicazioni con il Web Service che risulta molto semplice e veloce.

3.2.2 Struttura

L'applicazione in generale ha una struttura come descritta nella Figura 3.2 con un'unica activity bianca gestita da un NavigationDrawer, con all'interno due fragment. I fragment sono le interfacce sfruttate per attivare la rilevazione dello spostamento e per l'invio dei dati. Infine c'é un adapter che gestisce lo scambio fra i due fragment attuabile tramite il NavigationDrawer.

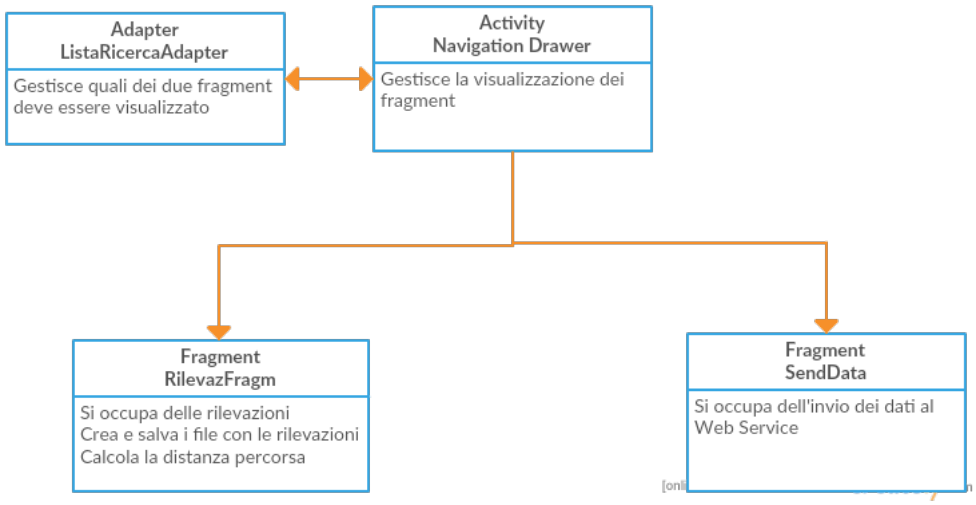


Figura 3.2: Grafico della struttura dell'applicazione

Vi é inoltre una classe creata appositamente per ottenere i permessi di esecuzione dell'applicazione chiamata **PermissionCheck**. Infatti la prima volta che verrá lanciata l'applicazione comparirá un dialog con la richiesta di poter usufruire di Internet e di accedere alla memori esterna del dispositivo in lettura e scrittura.

Activity

- **NavigationDrawer** – Come giá spiegato é l'unica activity presente nell'applicazione. Si occupa di gestire tramite un adapter il cambio dei fragment. All'avvio dell'applicazione automaticamente lancia il RilevazFrag che é il corpo principale dell'app.

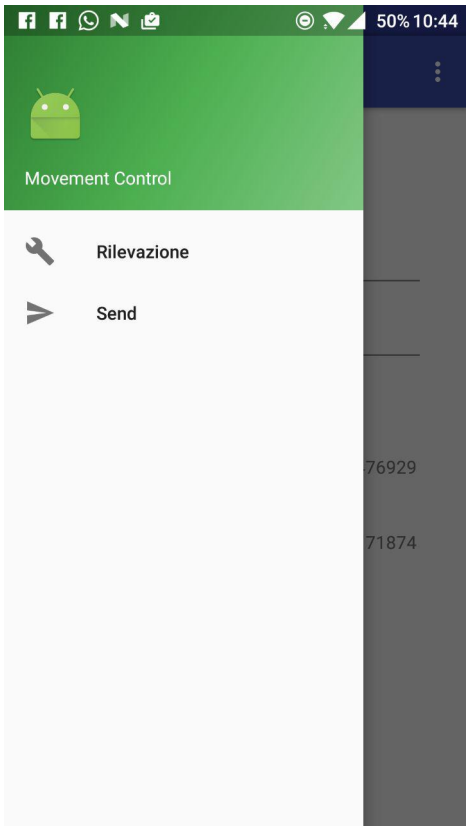


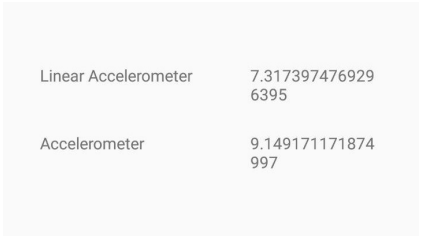
Figura 3.3: Interfaccia di navigazione

Per effettuare un cambio fra i fragment di rilevazione e di invio é sufficiente effettuare uno swipe da sinistra verso il centro dello scermo e comparirá il menu di navigazione.

Fragment

- **RilevazFrag** – É il fragment che si occupa della rilevazione, salvataggio dei dati e del calcolo della distanza percorsa.

(a) Interfaccia Rilevazione



(b) Risultato del calcolo della distanza

Figura 3.4: Interfaccia di rilevazione

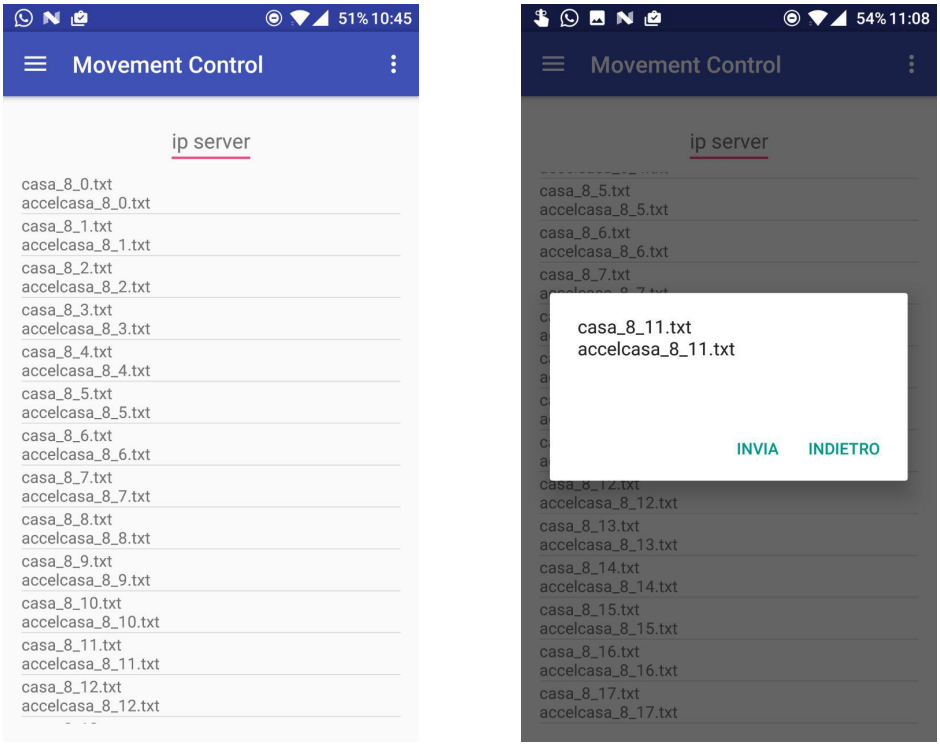
– *Rilevazione* – Sfruttando il SensorManager é possibile richiedere di ricevere un Sensor event ogni volta che l'accelerometro rileva un'accelerazione. Viene quindi richiesto di poter stare in attesa degli eventi dell'accelerometro e dell'accelerometro lineare.
Quando arriva un evento viene controllato a quale dei due appartiene e viene salvato il timestamp e i dati ricevuti dei tre assi in

una lista.

- *Calcolo* – Attraverso i dati salvati nelle liste adesso é possibile effetuare i calcoli. Il calcolo della distanza é ottenuto sfruttando le formule fisiche spiegate nei capitoli precedenti e gli accorgimenti a cui si é arrivati dopo le analisi.

- *Salvataggio* – Avviene tutto all'insaputa dell'utente, infatti nel momento in cui si ferma la rilevazione il sistema calcola la distanza percorsa e salva tutti i dati ottenuti in un file .txt. Questi dati sono organizzati seguendo le regole CSV in modo tale da poter essere poi sfruttati in maniera veloce per creare i grafici visti in precedenza. Ogni rilevazione crea due file, uno contenente i dati dell'accelerometro e l'altro contenente i dati dell'accelerometro lineare. Vi é inoltre un terzo file che viene aggiornato ad ogni rilevazione fatta che salva il nome della rilevazione e quanti metri si é risultato percorrere sia con l'accelerometro normale che con quello lineare. Nel caso sia la prima volta che si utilizza l'applicazione viene creata anche una cartella chiamata "rilevazioni" all'interno della home della scheda SD, dove si troveranno tutti i file che verranno creati.

- **SendData** – Cambiando fragment e entrando nell'interfaccia di invio dei dati si ha la possibilitá di scegliere a quale indirizzo inviare i file e attraverso una lista scrollabile é possibile scegliere quali file inviare. La lista viene aggiornata automaticamente ogni volta che si effettua una nuova rilevazione. Cliccando sui file si apre un dialog realizzato sfruttando la classe ProcessDialog messa a disposizione da android. Attraverso la libreria Volley come citato precedentemente possono essere inviati i file; nel caso che l'invio avvenga con successo ci sará un Toast che annuncerá la buona riuscita.



(a) Interfaccia Invio dati (b) Dialog per l'invio

Figura 3.5: Interfaccia di Invio dati

Capitolo 4

Conclusioni

Lo stato finale dell'applicazione soddisfa i requisiti prefissati: utilizzando unicamente l'accelerometro é possibile quindi ricavare un'algoritmo efficiente di localizzazione. L'algoritmo risulta mantenersi in media intorno al metro di errore. In futuro per avere una migliore idea di come si comporta l'algoritmo sará necessario ottenere piú dati su lunghe distanze.

4.1 Lavori Futuri

In questa sezione si elencano possibili funzionalitá includibili in futuro:

- **Introduzione del sensore di Orientamento**
- **Aggiunta di una mappa**
- **Algoritmo step-length**
- **Estensione funzionalitá del Web Service**

4.1.1 Introduzione del Sensore di Orientamento

Gli smartphone hanno al loro interno anche un sensore di orientamento, questo sensore ritorna l'inclinazione in gradi del cellulare e gli eventuali spostamenti nello spazio del dispositivo. Grazie a questo sensore sarebbe

possibile rilevare quando l'utente effettua una curva oppure entra in una stanza.

4.1.2 Aggiunta di una mappa

Aggiungendo una mappa dell'edificio in cui si effettua la rilevazione sarebbe possibile aggiornare l'algoritmo per renderlo in grado di dedurre in quale stanza ci si trova.

4.1.3 Algoritmo step-length

Implementando un algoritmo di step-length migliorerebbe l'algoritmo di step-detection a tal punto forse da preferirlo a quello delle formule fisiche. Non ci sarebbero piú problemi di lunghezza del passo che vorrebbe significare che un qualsiasi utente puó utilizzare il sistema e l'algoritmo rimane preciso, come giá avviene per quello basato sulla fisica.

4.1.4 Estensione funzionalitá del Web Service

Estendere le funzionalitá disponibili nel web service é sicuramente una feature che sarebbe bene implementare. Grazie a piú funzionalitá infatti si potrebbe migliorare ulteriormente l'algoritmo. Per esempio si potrebbe creare un tool che permetta l'analisi dei dati grezzi e la creazione di grafici coi dati immessi da computer.

Bibliografia

[1] A. R. Pratama, Widyawan and R. Hidayat, "Smartphone-based Pedestrian Dead Reckoning as an indoor positioning system," 2012 International Conference on System Engineering and Technology (ICSET), Bandung, 2012, pp. 1-6.

[2] T. H. Chang, L. S. Wang and F. R. Chang, "A Solution to the Ill-Conditioned GPS Positioning Problem in an Urban Environment," in IEEE Transactions on Intelligent Transportation Systems, vol. 10, no. 1, pp. 135-145, March 2009.

[3] N. Ravi, P. Shankar, A. Frankel, A. Elgammal and L. Iftode, "Indoor Localization Using Camera Phones," Seventh IEEE Workshop on Mobile Computing Systems & Applications (WMCSA'06 Supplement), Orcas Island, WA, 2006, pp. 1-7.

[4] S. S. Saab and Z. S. Nakad, "A Standalone RFID Indoor Positioning System Using Passive Tags," in IEEE Transactions on Industrial Electronics, vol. 58, no. 5, pp. 1961-1970, May 2011.

[5] Mateusz Korona, Matija Mandić, Vukoman Pejić and Daniel Siladi, "Bluetooth Indoor Positioning," in Summer School of Science, PoÅ$\frac{3}{4}$ega, Croatia, University of Primorska, Koper, Slovenia, August 24, 2015.

[6] H. Li, L. Sun, H. Zhu, X. Lu and X. Cheng, "Achieving privacy preservation in WiFi fingerprint-based localization," IEEE INFOCOM 2014

- IEEE Conference on Computer Communications, Toronto, ON, 2014, pp. 2337-2345.

[7] Lei Fang et al., "Design of a wireless assisted pedestrian dead reckoning system - the NavMote experience," in IEEE Transactions on Instrumentation and Measurement, vol. 54, no. 6, pp. 2342-2358, Dec. 2005.

[8] D. Hauschildt and N. Kirchhof, "Advances in thermal infrared localization: Challenges and solutions," 2010 International Conference on Indoor Positioning and Indoor Navigation, Zurich, 2010, pp. 1-8.

[9] JimÃ©nez, A.R.; Zampella, F.; Seco, F. Improving Inertial Pedestrian Dead-Reckoning by Detecting Unmodified Switched-on Lamps in Buildings. Sensors 2014, 14, 731-769.

[10] http://www.nationalgeographic.org/topics/gps/

[11] https://developer.android.com/guide/topics/sensors/sensors_overview.html

[12] https://www.slideshare.net/Cynapsys/prsentation-noura-baccar-innovation-on-indoor-geolocalization-applications-based-on-artificial-intelligence

[13] L. Bedogni, M. Di Felice, L. Bononi, "Context-Aware Android Applications through Transportation Mode Detection Techniques", Wiley's Wireless Communications and Mobile Computing Journal (WCMC), 2016

[14] L. Bedogni, M. Di Felice, L. Bononi, "By Train or By Car? Detecting the User's Motion Type through Smartphone Sensors Data" on proceedings of the 5th IFIP International Conference Wireless Days 2012 (WD 2012), November 21-23, 2012, Dublin, Ireland