Università di Pisa

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

Sviluppo di un'applicazione per la localizzazione indoor

Candidato

Michele Agostini

Relatori

dott. Antonino Crivello

prof. Dino Pedreschi

Anno Accademico 2015/2016

Indice

Capitolo 1

Introduzione

1.1 Il problema dell'Indoor Localization

Da oltre dieci anni i servizi basati sulla localizzazione diventano sempre più importanti e necessari per migliorare la qualità della vita delle persone e dei servizi offerti agli utenti che, spostandosi, possono essere raggiunti da servizi mirati di advertising o sfruttare la navigazione per il raggiungimento di determinate aree. Questi obiettivi sono stati ben raggiunti per quanto riguarda gli spazi esterni: è sufficiente riflettere sulla quantità di automobili e dispositivi mobili esistenti muniti di navigatore satellitare e dei servizi correlati, sfruttando il ben noto standard della tecnologia Global Positioning System (GPS). Considerando tuttavia che, durante le proprie giornate, le persone trascorrono molto del loro tempo in spazi chiusi (indoor)[1], la comunità scientifica sta cercando di trovare delle soluzioni che possano efficacemente inserirsi in questo contento. La maggior parte delle attività avviene infatti in casa, a lavoro o in posti per il tempo libero come i centri commerciali. Purtroppo il GPS tipicamente utilizzato per il posizionamento in

ambienti aperti (outdoor) non ha un segnale sufficientemente forte per localizzare con precisione un device, quindi un utente, all'interno degli ambienti chiusi. Nonostante questa mancanza, le possibili applicazioni di una funzionalità del genere sono molteplici: dal software per settori come il social networking e la pubblicità, agli scenari più critici come il tracciamento dei medici all'interno di un ospedale durante un'emergenza o dei pompieri durante il recupero delle persone da un edificio in caso di calamità o situazioni di pericolo.[2, 3, 4, 5, 6]

1.2 Stato dell'arte

Come già precedentemente accennato, per gli obiettivi oggetto di questo lavoro, va considerato che nello stato dell'arte non è ancora definibile una soluzione standard che possa ovviare e risolvere in maniera efficace questa il problema della localizzazione indoor. Negli anni sono state introdotte varie soluzioni che cercano di affrontare il problema considerando diverse variabili: costo, facilità di implementazione, accuratezza, dispositivi indossabili o non indossabili, facilità d'uso da parte dell'utente e altro ancora. In termini di accuratezza assoluta, ad esempio, le soluzioni fino ad oggi proposte per ambienti indoor e che consentono di ottenere una precisione dell'ordine del centimetro prevedono l'installazione di costose infrastrutture dedicate che sfruttano infrarossi, ultrasuoni e/o onde radio a banda ultra-larga [7][8][9][10][11][12]. La necessità di un grado di precisione relativamente alto si ha a causa della tipica presenza di aree eterogenee molto vicine tra loro come corridoi, stanze e ascensori.

Non è sempre possibile però affidarsi a un approccio del genere. Oltre che per una mera questione economica, la modifica strutturale e l'installazione di nuovi dispositivi potrebbe non essere praticabile, ad esempio per ragioni di sicurezza.

Un approccio decisamente meno costoso, ma ad oggi meno preciso in termini di accuratezza, prevede l'uso di elementi che sono già presenti nell'edificio come il campo magnetico e i segnali provenienti dalle reti Wi-Fi [13][14][15][16]. In termini di propagazione del segnale Wi-Fi, un edificio è generalmente un sistema complesso a causa dei materiali di cui è composto e della presenza di diversi elementi di mobilia e di arredo. Per queste ragioni, in alcune aree, il segnale Wi-Fi potrebbe non essere presente anche considerando l'eventualità di una grande concetrazione di Access Point.

Infine, si può pensare di fondere tutte le informazioni precedentemente citate con quelle dei sensori inerziali (accelerometro, giroscopio) e ambientali (magnetometro e barometro) tipicamente presenti all'interno di uno smartphone commerciale.

1.3 L'attività di tirocinio

L'attività di tirocinio si è svolta nell'arco di 300 ore presso il Wireless Networks Laboratory dell'Istituto di Scienze e Tecnologie dell'Informazione (ISTI) del Consiglio Nazionale delle Ricerche (CNR) di Pisa. L'obiettivo dell'attività era quello di ottenere un'architettura modulare ed estendibile per il software dedicato alla localizzazione indoor e la realizzazione di un'applicazione smartphone che ne facesse uso. Il tirocinio è stato caratterizzato da una fase introduttiva al problema e da un ampio studio dello stato dell'arte [17] di questa specifica problematica. Successivamente sono stati definiti i requisiti funzionali e l'architettura dell'applicazione. Infine, al termine dello sviluppo dell'applicazione è stata anche effettuata una vasta campagna di misurazione per la validazione e il test dell'applicazione creata. Questa relazione è strutturata secondo la sequenza appena descritta.

La finalità di questo software è il fornire un'architettura software modulare, quindi estendibile, costituente una base di partenza comoda per lo sviluppo e il test degli approcci alla localizzazione indoor presenti e futuri.

Capitolo 2

Requisiti e scelte effettuate

2.1 Requisiti dell'applicazione

L'obiettivo principale dell'applicazione oggetto di questo lavoro è di avere, ad ogni passo dell'utente, un output in termini di posizione spaziale a seguito dell'elaborazione dei dati provenienti dai sensori presenti all'interno di uno smartphone di uso comune.

Per il raggiungimento degli obiettivi prefissati e per la realizzazione dell'applicazione per la localizzazione indoor.

2.1.1 Requisiti funzionali

- Deve essere possibile selezionare singolarmente quali tipi di sensori devono essere usati per la localizzazione. Questo requisito è stato formalizzato per permettere un'estensivo uso dell'applicazione anche all'interno di dispositivi dotati di una tecnologia limitata.

- Deve essere possibile cambiare la modalità di strategia di data fusion. Questo requisito ha due obiettivi: permettere l'implementazione di nuove strategie e valutarne le prestazioni in confronto alle stategie precedentemente sviluppate.

- Deve essere possibile avviare e fermare esplicitamente il servizio di localizzazione.

2.1.2 Requisiti non funzionali

Considerando i requisiti funzionali sovraesposti, sono stati definiti ulteriori requisiti in merito all'usabilità e alla facilità di uso dell'applicazione da parte dell'utente. È necessario premettere che il rendering della posizione all'interno di una mappa, o l'aggiornamento della stessa in termini di navigazione, non è un tema oggetto di questo lavoro. L'output grafico per l'utente quindi è da considerarsi il susseguirsi di coordinate $\langle x, y \rangle$ che andranno eventualmente usate all'interno di una mappa o sfruttate per ulteriori servizi di navigazione.

- La strategia di localizzazione deve prevedere soltanto l'uso di sensori disponibili in smartphone commerciali, evitando quindi l'installazione di hardware aggiunto rispetto a quello presente in un classico ambiente di lavoro (comuni Access Point per reti Wi-Fi).

- Facilmente estendibile e modificabile. Considerando la precedente discussa immaturità attuale delle tecniche e dei sistemi di posizionamento e di navigazione indoor, questo aspetto è stato ritenuto un obiettivo fondamentale nel corso di questo tirocinio. Il raggiungimento di questa specifica, attraverso un accurato studio dell'architettura, permetterà l'integrazione di soluzioni presenti e future di data fusion e di modelli accurati (ad esempio in meri-

to alla mobilità delle persone) per migliorare l'accuratezza e la precisione dell'applicazione.

- Indipendente da Android quando possibile. Esistono infatti anche soluzioni per la localizzazione che non elaborano le informazioni sul device, ma seguendo un paradigma client-server.

2.2 Tecnologie utilizzate

È stata lasciata libertà nella decisione del linguaggio di programmazione e di eventuali framework da utilizzare. La scelta è ricaduta sul linguaggio Java (escludendo altre possibilità come Python[1] e C[2]) per familiarità del tirocinante e per l'ampia offerta di adeguati strumenti per la strutturazione dell'architettura. Considerata l'assenza di requisiti in termini di performance e velocità di computazione, non sono stati scritti moduli software con codice nativo ma è facile l'introduzione di componenti CPU-intensive sviluppati con tali modalità. Naturale conseguenza della scelta del linguaggio è stato l'impiego dell'Android Framework[3], già noto al tirocinante, con compatibilità minima alla versione 19 delle API per garantire la compatibilità con il device fornito dalla struttura, uno Xiaomi Mi 3W dotato di Android 4.4 KitKat. Questa ragionevole scelta ha costretto l'utilizzo delle API Java 7[4], rendendo incompatibile l'impiego di Java 8[5][6] a causa di limiti tecnologici.

[1] QPYthon, Script engine per l'esecuzione di codice Python su Android. http://qpython.com/

[2] Android NDK, API per C/C++. https://developer.android.com/ndk/index.html

[3] Android Framework, SDK. https://developer.android.com

[4] Java 7, Distribuzione Oracle. http://docs.oracle.com/javase/7/docs/api/

[5] JACK, Toolchain per Java8 su Android. https://developer.android.com/guide/platform/j8-jack.html

[6] Il toolchain è stato successivamente deprecato dalle API ufficiali in data 14 Marzo 2017.

Sono state solo marginalmente utilizzate le librerie Apache Math 3.6.1[7] e Guava 20[8], scelte dal tirocinante in fase di sviluppo per agevolare la scrittura del codice.

2.3 Design patterns

I seguenti design pattern[18] sono stati proposti dal tirocinante e validati dal tutore e dai ricercatori interessati a questa applicazione. È stato scelto di utilizzare i pattern perchè si è ritenuto agevolassero il raggiungimento dell'estensibilità e della manutenibilità richieste.

Strategy pattern

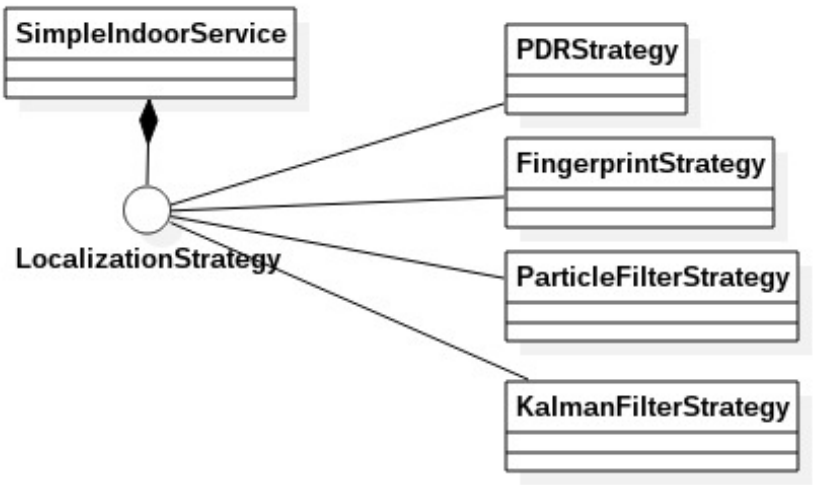


Figura 2.1: Caso d'uso dello Strategy Pattern nell'applicazione realizzata.

[7]Apache Math, Libreria. http://commons.apache.org/proper/commons-math/javadocs/api-3.6.1

[8]Guava, Libreria. https://github.com/google/guava

Definisce una famiglia di algoritmi tramite un'interfaccia, li incapsula e li rende intercambiabili. Questo pattern è stato utilizzato esplicitamente per astrarre l'algoritmo di localizzazione: questo rende più facile la scelta dello stesso in fase di avvio, rende maggiormente manutenibile il codice e può essere utile cambiare strategia dipendentemente dalla situazione (i.e. Wi-Fi non disponibile, device in tasca, device in mano[2]).

Observer pattern

Definisce una dipendenza uno a molti tra oggetti, i registranti vengono notificati in caso di aggiornamenti delle informazioni ai quali sono interessati. È stato ampiamente utilizzato nell'interazione tra i moduli per favorirne l'indipendenza.

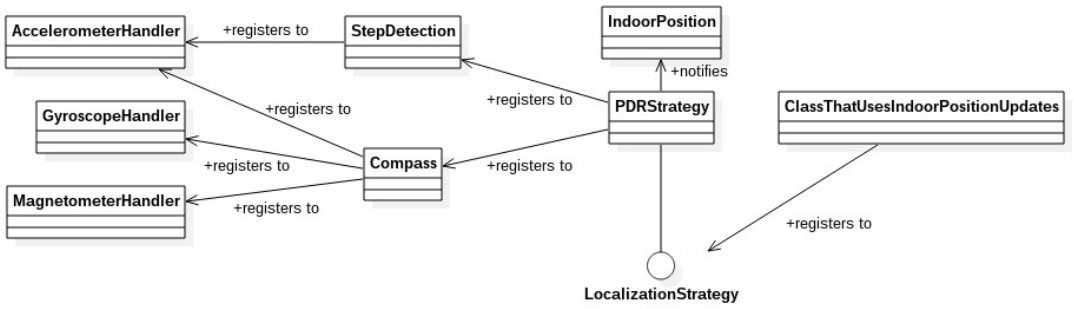


Figura 2.2: Uso dell'Observer Pattern nel caso dell'algoritmo di localizzazione che necessita solamente di bussola e step detection.

Nell'applicazione ogni componente (i.e. bussola) ha la responsabilità di registrarsi agli oggetti dai quali dipende (i.e. handlers di accelerometro, giroscopio e magnetometro) nel momento in cui ha almeno un observer nella sua lista (i.e. algoritmo di localizzazione unicamente su sensori inerziali). Questo comportamento *lazy* non è previsto dalla definizione ufficiale del pattern ed è stato scelto di

realizzare l'architettura in modo da non dover essere forzatamente rispettato, per quanto utile all'efficienza del sistema generale.

Adapter

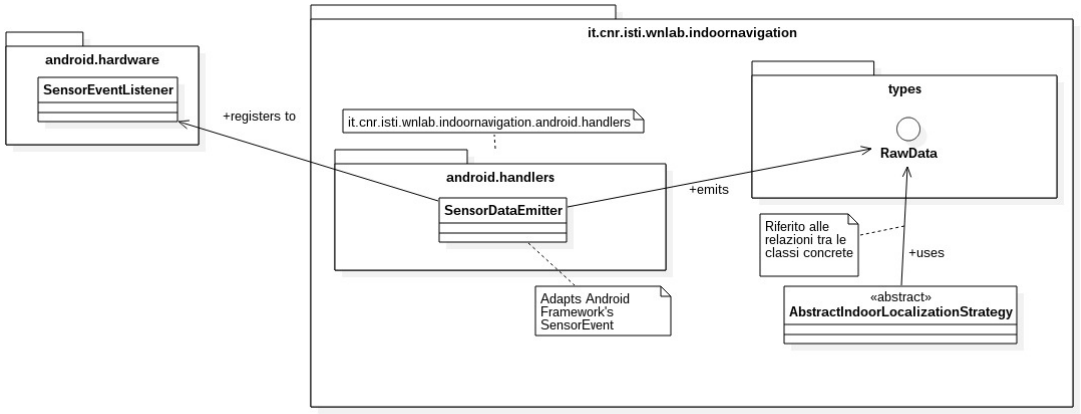


Figura 2.3: Uso dell'Adapter Pattern per interfacciare l'algoritmo di localizzazione, indipendente dalla piattaforma, con le strutture dati di Android.

Converte l'interfaccia di una classe in quella che un client si aspetta. È stato usato per rendere gli oggetti SensorEvent[9] ricevuti dai SensorListener compatibili con gli observer dell'aggiornamento dei dati. Questo rende di fatto indipendente il codice che elabora i dati dal formato utilizzato dal sistema che li fornisce.

Dependency Injection

Consiste nell'avere un'entità separata che ha la responsabilità di assemblare un campo di un oggetto con un'implementazione adeguata. In particolare è stata utilizzata la Constructor Injection in ogni componente che necessita di informazioni

[9]SensorEvent, Classe delle API di Android. https://developer.android.com/reference/android/hardware/SensorEvent.html

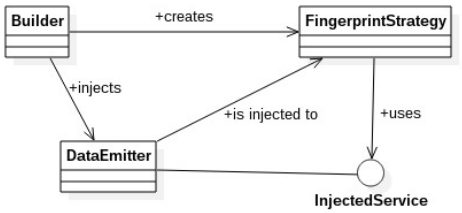


Figura 2.4: Esempio di Dependency Injection nel caso dell'algoritmo di localizzazione basato su fingerprinting.

output di altri moduli: questo concorre a favorire l'indipendenza, quindi la manutenibilità[19], tra componenti. Per fare un esempio, la bussola non crea al suo interno gli handler per i sensori che utilizza ma li riceve tramite il costruttore.

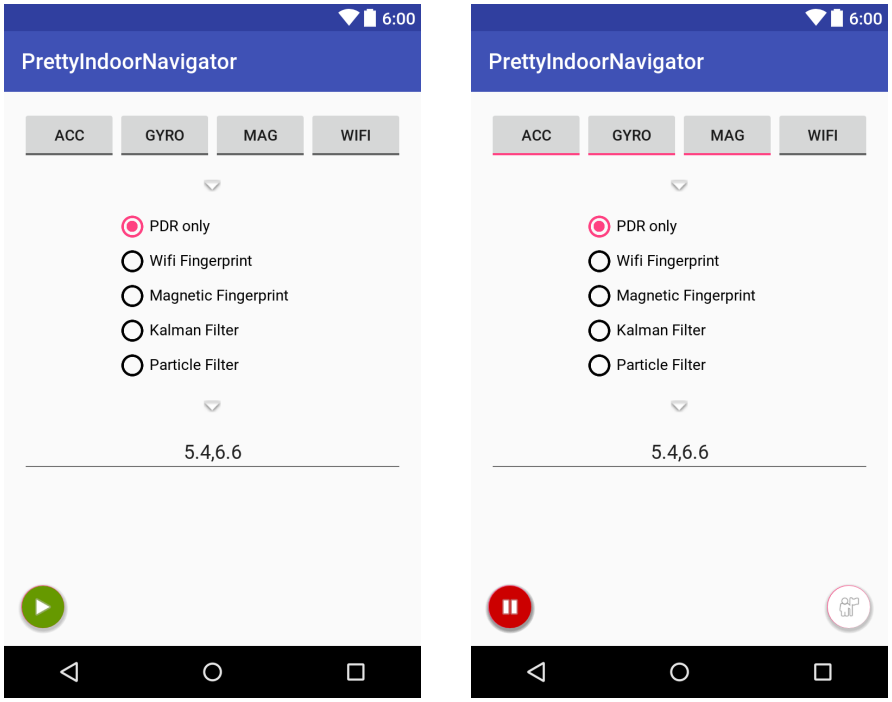
2.4 L'applicazione per la localizzazione indoor

Il risultato dell'attività di tirocinio è un'applicazione per la localizzazione indoor che permette inizialmente di configurare l'algoritmo di localizzazione da utilizzare e, una volta avviato, segue in tempo reale gli spostamenti del device anche in background. È presente una semplice interfaccia grafica, mostrata in 2.5, per scegliere le opzioni del servizio.

Quando l'utente preme il bottone dedicato, l'applicazione registra su un file CSV (Comma Separated Values) le ultime coordinate rilevate nel formato:

```
timestamp,x,y,z
```

Dove `x` e `y` sono coordinate in un formato scelto (i.e. coordinate cartesiane, latitudine e longitudine) e `z` è un intero indicante il piano (0 per il piano terra). Il rilevamento del piano è stato previsto nell'architettura ma non implementato.



(a) Localizzazione inattiva (b) Localizzazione attiva

Figura 2.5: GUI dell'applicazione di localizzazione indoor

Una componente per la gestione dei fingerprint

All'interno della sezione riguardo lo stato dell'arte sono stati più volte citati lavori che si basano, o includono, l'uso di tecniche di fingerprint, sia per quanto riguarda l'uso dei segnali Wi-Fi sia per i dati provenienti dal campo geomagnetico. Anche in questo lavoro, la trattazione di questi due tipi di segnali è stata affrontata implementando questa tecnica. Preliminarmente, si è quindi resa necessaria una campagna di acquisizione di fingerprint Wi-Fi e campo magnetico. Per acquisire un numero di record sufficiente ad analizzare i segnali presenti all'interno dello scenario in oggetto, è stato scelto di effettuare le rilevazioni in punti distanti 60cm. Questa granularità permette di avere una sufficiente copertura di tutti gli spazi

raggiungibili da un utente e, con buona approssimazione, vengono memorizzati i vettori di fingerprint per entrambi i sensori in ogni direzione possibile raggiungibile con un passo, considerato di 60cm. Successivamente è stata realizzata una componente dedicata, la cui interfaccia è visibile in 2.6, per questo scopo. Il software permette di registrare i valori rilevati per alcuni secondi e fonderli direttamente sul dispositivo facendo una media aritmetica, scelta per semplicità, dei valori.

Il risultato di queste operazioni è memorizzato in file di tipo CSV. Nel caso dell'acquisizione del campo magnetico il formato è:

```
x1,y1
Mx1,My1,Mz1
Mx2,My2,Mz2
...
x2,y2
...
```

Dove `Mx`, `My` e `Mz` sono le componenti del vettore campo magnetico. La fingerprint map viene registrata come:

```
x1,y1,Mx1,My1,Mz1
```

Dove `Mx`, `My` e `Mz` sono le medie delle componenti del campo magnetico registrate in quel punto per il periodo prescelto. Per quanto concerne il Wi-Fi, il formato è speculare a eccezione del valore, che in questo caso consiste in un insieme di coppie $\langle BSSID, RSSI \rangle$. I concetti di fingerprint singolo e in una mappa verranno approfonditi in seguito.

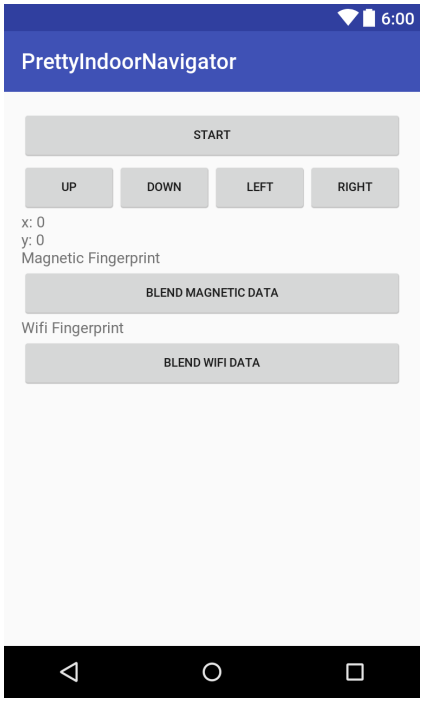


Figura 2.6: GUI per l'acquisizione e la realizzazione di mappe di fingerprint Wi-Fi e magnetici.

Capitolo 3

Strategie di localizzazione

Una strategia di localizzazione consiste in generale in uno o più algoritmi combinati che raffinano i dati raw provenienti dai sensori o da altri componenti per ricavarne un'informazione. In figura 3.1 è mostrato un generico workflow per una stategia di localizzazione.

Non avendo il tirocinante esperienze nel settore sono state, indicate dal tutore, studiate e implementate durante il tirocinio alcune soluzioni già esistenti che fanno uso di Kalman Filter[20, 17] e Particle Filter[21, 2]. Entrambi sono filtri per la state estimation[22], tuttavia nel primo caso la localizzazione avviene tramite Pedestrian Dead Reckoning[23], il cui errore viene corretto fondendo le posizioni rilevate tramite posizionamento con fingerprint Wi-Fi e magnetico[17]; nel secondo caso il filtro stesso viene dedicato al posizionamento sfruttando la sua intriseca proprietà di predizione dello stato successivo.

È uso comune delle strategie di localizzazione indoor elaborare informazioni risultanti da diverse ed eterogenee tecniche al fine di correggere l'errore, migliorarlo quindi in termini di accuratezza, rispetto al risultato che si otterrebbe utilizzando

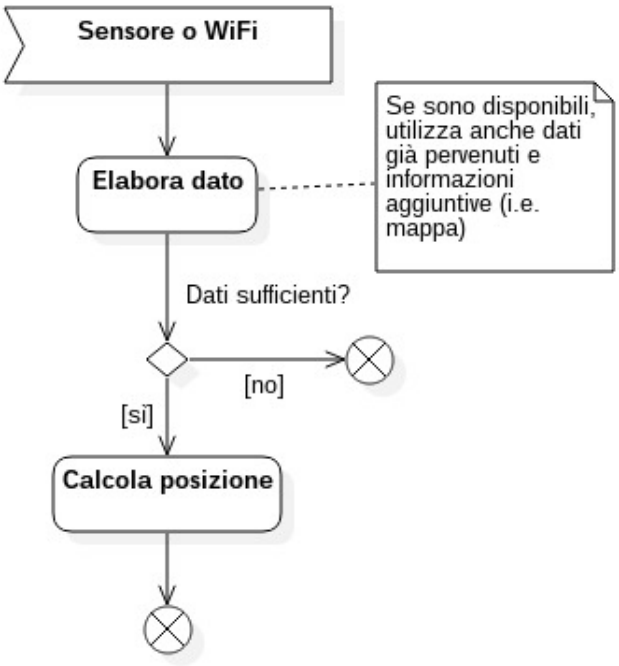


Figura 3.1: Diagramma di attività di un generico algoritmo per la localizzazione indoor.

soltanto una determinata tecnica o un determinato tipo di sensore.

3.1 Pedestrian Dead Reckoning

Il concetto alla base del Pedestrian Dead Reckoning (PDR)[23] è quello di avanzare di una certa distanza quando viene avvertito un passo. L'output di questa famiglia di algoritmi è una variazione di coordinate `dx` (o `dEst`) e `dy` (o `dNord`) ricavata dai dati di accelerometro, giroscopio e magnetometro combinati e dal modello di deambulazione dell'utente utilizzato.

3.1.1 Stima della lunghezza del passo

La lunghezza dello spostamento può essere considerata fissa o variabile, ma fortunatamente è possibile determinare un limite superiore attenendosi alla distanza che un essere umano percorre ragionevolmente in un singolo passo.

Nel lavoro realizzato sono state prese in considerazione entrambe le opzioni: nel primo caso, utilizzando un'idea implementata in una soluzione già esistente [17], è stata considerata una lunghezza del passo pari a 0,6m; nel secondo caso questa lunghezza è stata perturbata a ogni passo secondo una distribuzione normale avente σ=0,15m[21], allo scopo di considerare una componente variabile tra un passo e il successivo.

3.1.2 Step Detection

La step detection consiste nel monitorare nel tempo l'accelerazione avvertita dal sensore, filtrata dall'accelerazione della gravità. Se questa, considerata una finestra di eventi, presenta un picco massimo e un picco minimo si considera un passo. A causa dell'orientamento variabile del telefono, è buona norma, in termini di precisione[23], considerare tutti e tre gli assi. Il PDR si ottiene da: calcolo della lunghezza del passo, calcolo dell'orientamento, rilevamento del passo avvenuto.

Le API di Android implementano la step detection[1], ma la reattività nei rilevamenti è risultata troppo bassa per utilizzarle nell'applicazione.

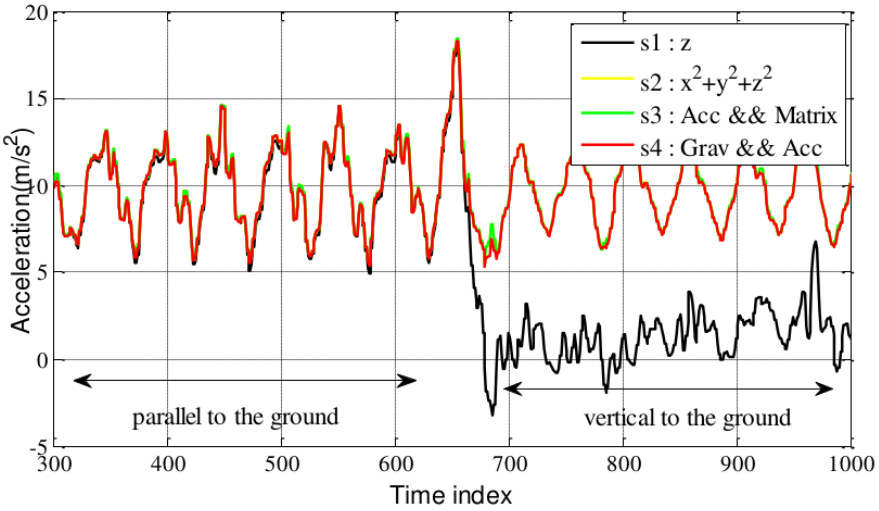


Figura 3.2: La variazione dell'accelerazione percepita durante una camminata[23].

3.1.3 Orientamento

L'orientamento è fondamentale per il corretto funzionamento del PDR. Inizialmente si è provato a fare affidamento soltanto sul dato fornito dal giroscopio, ma ciò ha condotto a scarsi risultati. Quindi, è stato introdotto un algoritmo[24] usato per: fondere accelerometro e magnetometro tramite un filtro passa basso, filtrare il giroscopio con un passa alto, correggere l'errore di quest'ultimo fondendo le altre informazioni disponibili tramite un Complementary Filter[25].

3.2 Fingerprinting

Un fingerprint è una coppia $\langle posizione, valore \rangle$. Nel caso di un fingerprint Wi-Fi si ha una coppia $\langle posizione, vettoreRSSI \rangle$ nel caso di un fingerprint magnetico si

[1]Step Detection, API Android. https://developer.android.com/reference/android/hardware/Sensor.html#$TYPE\_STEP\_DETECTOR$

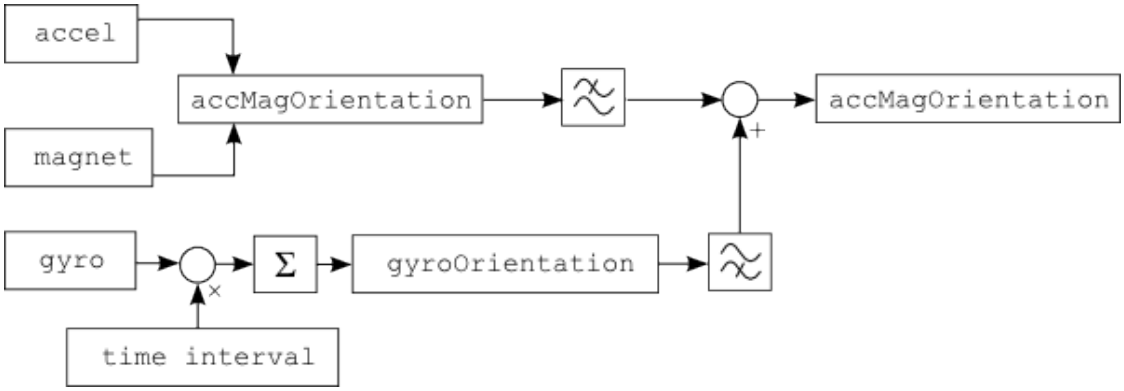


Figura 3.3: L'orientamento viene ricavato correggendo il giroscopio con accelerometro e magnetometro.

ha una coppia $\langle posizione, vettorecampomagnetico \rangle$.

Una mappa o database di fingerprint è una collezione di queste coppie, identificate univocamente dalla posizione unica in cui questi record sono stati raccolti.

La tecnica di fingerprinting, generalmente, consiste in due fasi[17]:

1. Campagna di acquisizione

2. Localizzazione

La prima fase consiste nel posizionarsi in posizioni prestabilite e acquisire per un certo periodo di tempo i valori interessanti. Durante il tirocinio si è scelto di stazione su un determinato punto per 5 secondi contemporaneamente così da considerare eventuali picciole fluttuazioni dei segnali ricevuti. La localizzazione avviene effettuando un rilevamento dei dati di interesse e, successivamente, confrontandoli con i valori precedentemente registrati e che compongono la mappa, con l'obiettivo di invidividuare la posizione più *vicina*. Introducendo il concetto di confronto, è necessario introdurre il concetto di distanza (tra due vettori in questo caso). Il tirocinante, a seguito dello studio dell'arte, ha scelto di implementare una distanza

euclidea tra i due vettori, garantendo allo stesso tempo buone prestazioni e facilità di implementazione e di calcolo.

3.2.1 K-Nearest Neighbours

La sola scelta di una metrica per stabilire quanto due vettori sia vicini tra di loro non è sufficiente implementare all'interno dell'algoritmo, una scelta tra quale o quali vettori debbano essere presi in considerazione. Scegliendo un intero K arbitrario si può operare una media pesata sulle K posizioni più vicine al rilevamento per ottenere una posizione. Questa tecnica viene denominata *K-Nearest Neighbours* o *K-NN*[26].

3.3 State estimation problem

È interessante osservare che la posizione corrente può essere vista come uno stato dipendente dai precedenti[2]. In quest'ottica risultano particolarmente utili filtri che risolvono il problema della stima dello stato attuale.

3.3.1 Kalman Filter

Il *Kalman Filter*[20] è un metodo che può essere applicato quando i modelli osservati sono lineari e il rumore dei suddetti è rappresentato da una distribuzione gaussiana con media e varianza note.

La stima del nuovo stato è il risultato di due fasi: una di *prediction* e una di *update*.

Sia x_k lo stato attuale del modello e P_k una matrice di covarianza.

1. *Prediction*

Stato predetto $x_p = \mathrm{A}x_k + \mathrm{B}u_{k+1}$

Covarianza predetta $P_p = A_{k+1}P_kA_{k+1}^T + \mathrm{Q}$

Dove A è la matrice di transizione di stato (per far avanzare lo stato secondo il modello), B è la matrice di controllo (per definire le equazioni dei fattori di controllo), u è il vettore di controllo ricevuto in input e Q è la covarianza dell'errore stimato del processo.

2. *Update*

Innovazione $\mathrm{y} = z_{k+1} - \mathrm{H}x_p$

Innovazione della covarianza $\mathrm{S} = \mathrm{H}P_pH^T + \mathrm{R}$

Kalman Gain $\mathrm{K} = P_p + H^TS^{-1}$

Stato aggiornato $x_{k+1} = x_p + \mathrm{Ky}$

Covarianza aggiornata $P_{k+1} = (\mathrm{I} - \mathrm{KH})P_p$

Dove H è la matrice di osservazione che serve a convertire lo stato in un vettore di misure e la matrice R contiene la covarianza dell'errore stimato nella misura,

3. Ora l'output è lo stato corrente. Quando arriva un nuovo vettore di controllo in input, torna a (1).

Nell'applicazione, il Kalman Filter è utilizzato per ridurre l'errore del PDR ed è presente con un'implementazione molto semplice. Nonostante questo, il fatto di essere astratto con un'interfaccia rende le sue implementazioni facilmente intercambiabili. Le matrici P, Q e R sono state previste nell'algoritmo ma, attualmente,

non hanno influenza nel risultato finale. Sono state settate come matrici identità, poichè la formulazione esplicita esula dagli obiettivi di questo tirocinio.

Come effettuare l'implementazione è stato lasciato al tirocinante ed è stato scelto di creare una classe con un metodo astratto per ogni passo di cui sopra. Questo permette a chi implementa la propria versione di avere una visione d'insieme sul formato delle strutture dati che sta utilizzando. È stata considerata anche l'implementazione del Kalman Filter già presente nella libreria Apache Math[2], ma è stata considerata non necessaria per via della semplicità richiesta dall'implementazione effettuata.

3.3.2 Particle Filter

Il *Particle Filter* è un metodo Monte Carlo che, dato un insieme di N particelle caratterizzate da una posizione e un peso, si esplicita ciclicamente in tre passi:

1. *Aggiornamento particelle*

   Le particelle vengono mosse nella direzione segnalata dal PDR, ognuna varia di un errore casuale. Questo dipende dal modello adottato: nel nostro caso, l'errore nella lunghezza del passo viene estratto da una distribuzione normale a media zero e $\sigma = 0{,}15$m mentre l'errore dell'orientamento viene estratto da una distribuzione normale a media zero con $\sigma = \frac{\pi}{2}$[21].

2. *Filtraggio particelle*

   Le particelle vengono filtrate secondo alcuni vincoli. Nell'applicazione ne vengono applicati tre in sequenza:

[2]Kalman Filter su Math, Libreria. https://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math4/filter/KalmanFilter.html

(a) Se una particella è in una posizione non valida secondo una mappa (i.e. fuori dall'edificio), questa viene eliminata.

(b) Per ognuna delle posizioni nello spazio rappresentate dalle particelle, viene interpolata una distanza euclidea dall'ultimo fingerprint magnetico ricevuto con $d_{particle} = \sum_{i=1}^{n} \frac{d_i}{r_{i,particle}}$ per ogni $posizione_i$ nella mappa di fingerprint. In pratica, si simula che la posizione della particella appartenga alla mappa di fingerprint. A questo punto viene estratto un numero casuale $p_{particle} \in [0, limit]$ dove *limit* è la distanza del punto più lontano dal fingerprint presente sulla mappa. Se $p_{particle} \in [0, d_{particle}]$, la particella viene tolta dall'insieme, altrimenti sopravvive. Così facendo, maggiore è $d_{particle}$, maggiore è la probabilità di essere eliminata.

(c) Si ripete il passo precedente con il fingerprint Wi-Fi.

3. *Rigenerazione particelle*

   Le particelle vengono riportate al numero iniziale duplicando quelle esistenti. Questa è solo una delle molte tecniche possibili[27].

Questa strategia di data fusion è concettualmente diversa dal Kalman Fiter[22] precedentemente esposto. All'interno dell'applicazione oggetto di questo tirocinio sono state implementate entrambe queste tecniche di data fusion. Nel caso di uso di KF lo stato è l'errore di posizione al passo stimato, nel caso di strategia con PF invece lo stato rappresenta direttamente la posizione scelta.

Architetturalmente, è stata qui preferita la *composition over inheritance*, scomponendo con lo Strategy Pattern gli algoritmi dei singoli passi: questo permette di mantenere i codici indipendenti nonostante operino sulla stessa collezione di particelle. La responsabilità di "leggere" l'insieme risultante di queste e ricavarne una posizione indoor è delegata a un quarto componente.

3.4 Utilizzo delle strategie di localizzazione

Sono state implementate in totale cinque strategie di localizzazione con cui poi sono stati eseguiti i test: solo PDR, solo fingerprint Wi-Fi (e K-NN), solo fingerprint magnetico (e K-NN), PDR con correzione dell'errore tramite Kalman Filter e fingerprint, posizionamento con Particle Filter. Tutte le strategie partono da una posizione nota, inserita dall'utente dopo aver avviato l'applicazione. Al momento, l'applicazione non permette il cambio di algoritmo a tempo di esecuzione. Questo sarebbe facilmente implementabile avviando la nuova strategia e segnalandole come posizione di partenza le ultime coordinate rilevate dalla precedente strategia.

Capitolo 4

Architettura software

4.1 Modello concettuale

Il modello concettuale, rappresentato nella figura 4.1, è stato proposto dal relatore e discusso con il tirocinante. L'obiettivo dell'architettura realizzata consiste nell'astrazione del comportamento usuale di un'applicazione per la localizzazione indoor (3.1) applicando un notevole grado di disaccoppiamento. La soluzione consiste concettualmente in un'organizzazione a due livelli principali: un insieme di handlers per i dati raw provenienti dai sensori, generalmente sensori o Wi-Fi, e una logica in grado di elaborarli. Il modulo di quest'ultima è idealmente diviso a sua volta in un sublivello di pre-processing (per moduli intermedi come bussola e step detection) e uno contenente la strategia di localizzazione vera e propria che combina le informazioni disponibili e invia in output una posizione valida.

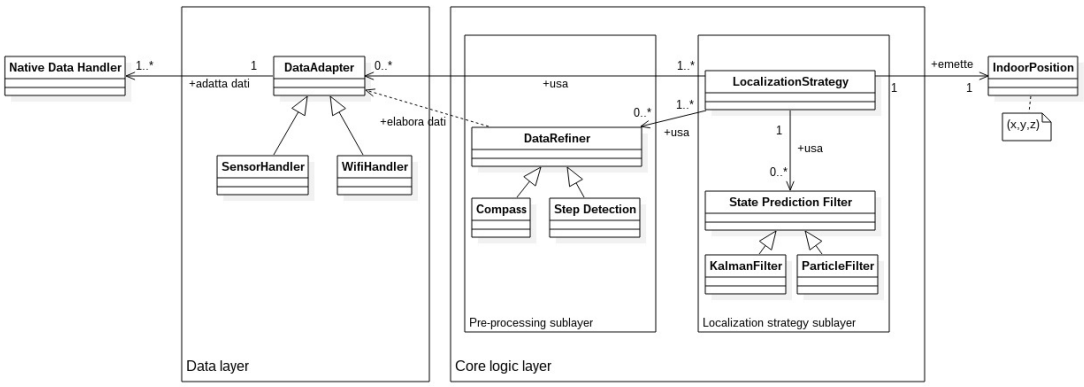


Figura 4.1: Modello concettuale

4.2 Schema dell'applicazione

L'applicazione ha, essenzialmente, due componenti[1]: una Activity con la GUI e un Service che incapsula il codice necessario alla localizzazione. È presente inoltre una classe dedicata al logging su file di testo. In figura 4.2 è rappresentato uno schema delle classi descritte.

4.3 Riusabilità: organizzazione a livelli

Al fine di permettere, quando possibile, un riutilizzo del codice anche su piattaforme non Android, è stato deciso di organizzare il codice scritto a livelli:

La libreria Java contiene interfacce e classi astratte che realizzano il modello concettuale e alcune implementazioni indipendenti da Android. La libreria Android contiene implementazioni specifiche della piattaforma (i.e. bussola, handler dei

[1]Componenti di un'applicazione, Android Framework. https://developer.android.com/guide/components

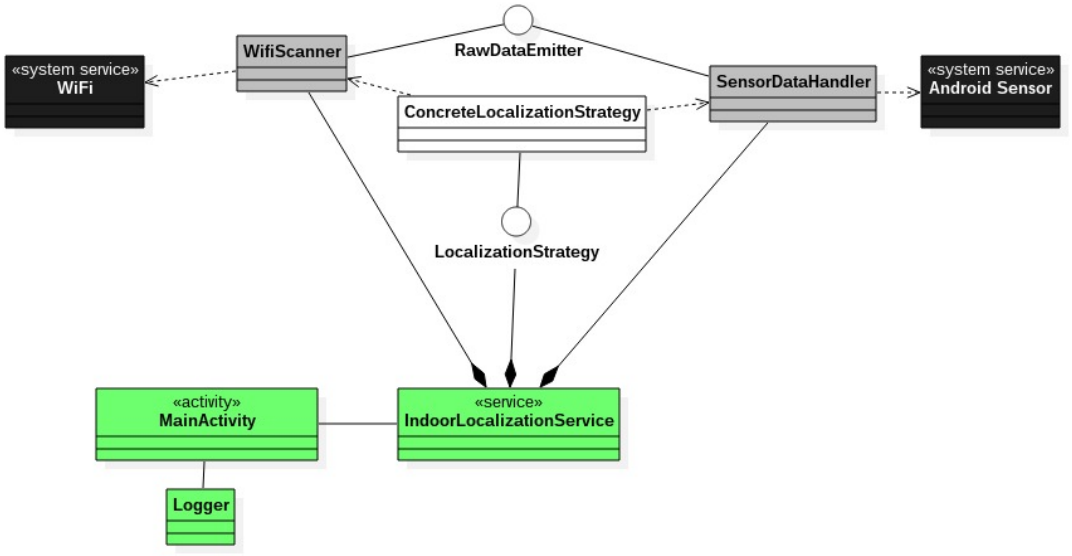


Figura 4.2: L'architettura dell'applicazione. In rosso le classi dell'applicazione, in azzurro le implementazioni per Android realizzate nella libreria specifica, in bianco le astrazioni della libreria scritta solamente in Java e in verde i componenti del sistema Android.

sensori). L'applicazione di localizzaione sfrutta i livelli sottostanti per offrire il servizio all'utente.

4.4 Tipi di dato

Essendo il dominio di utilizzo del lavoro non necessariamente ristretto ad ambienti Android, è stato scelto di incapsulare i dati dei sensori in oggetti dedicati per renderli indipendenti dal formato utilizzato dal sistema (ad esempio, Android struttura le informazioni di ogni singolo evento in un array di lunghezza variabile di numeri in virgola mobile).

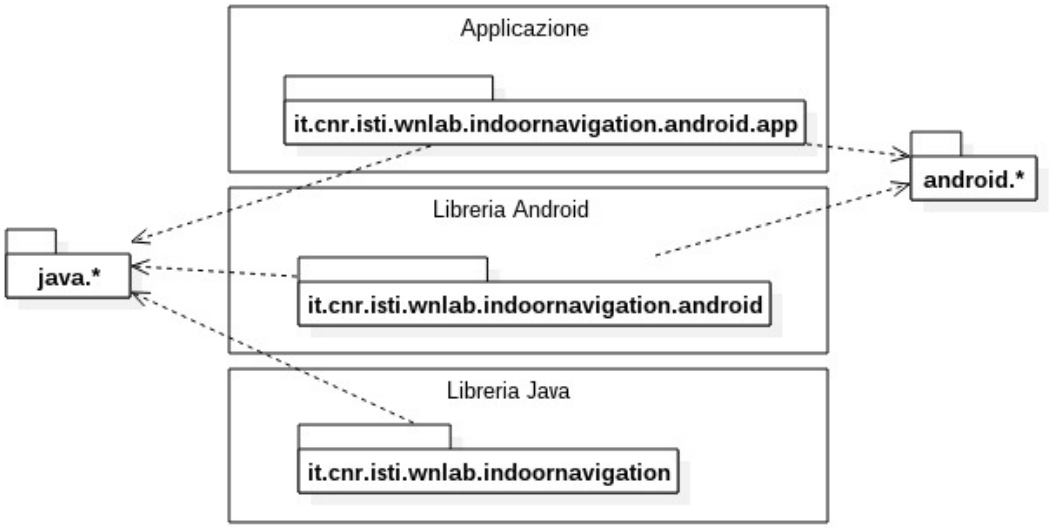


Figura 4.3: La strutturazione a livelli dell'applicazione permette l'indipendenza dalla piattaforma e la riusabilità del codice. Ogni livello fornisce interfacce e implementazioni (anche parziali) ai livelli superiori.

Le classi FingerprintMap e DistanceMap

Nell'applicazione è stata implementata una classe che rappresenta la mappa di Fingerprint. Inoltre è stata creata una classe utilitaria DistancesMap che rappresenta la distanza euclidea di ogni punto della mappa da una misurazione rilevata. È stato scelto di implementare il comportamento della classe con una politica *lazy*, che memorizza l'ultima misurazione rilevata e aggiorna le distanze solamente alla prima richiesta da parte di un componente client.

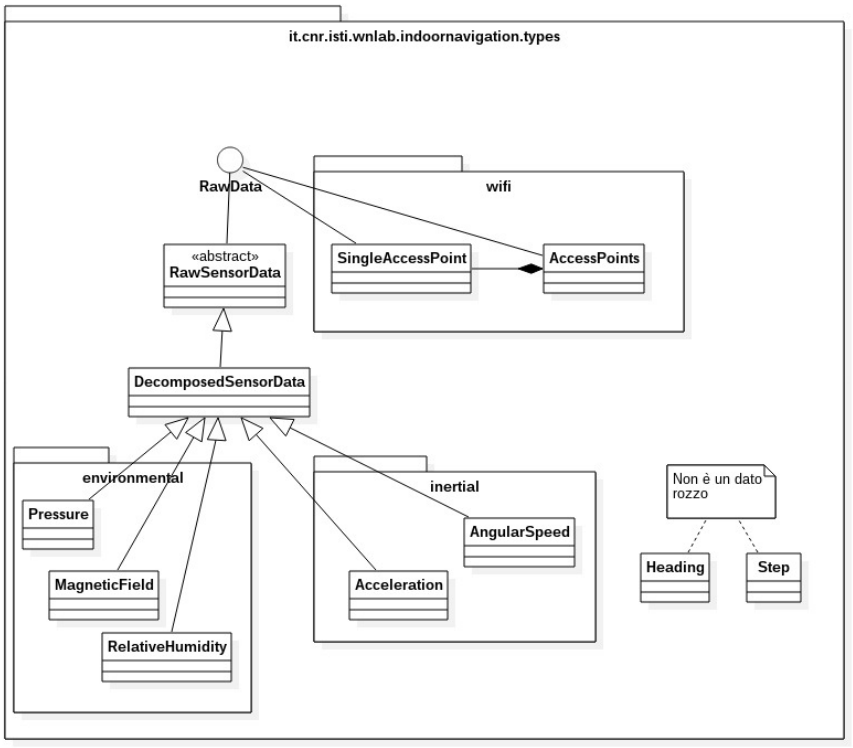


Figura 4.4: I tipi che incapsulano i dati che vengono utilizzati per localizzare. Quest'astrazione aggiunge overhead ma permette una maggiore indipendenza dal formato utilizzato dal sistema sottostante.

4.5 Ottimizzazione della localizzazione con la mappa

È stata espressamente richiesta l'implementazione della mappa. L'interfaccia della classe utilizzata nell'implementazione, in particolare, fornisce due metodi: uno per verificare se la posizione è valida e uno per trovare la posizione valida più vicina a quel punto. Essendo la porzione di mappa considerata una composizione di quadrati, non è stato difficile implementare anche quest'ultima funzionalità.

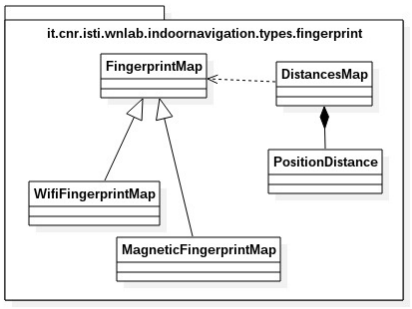


Figura 4.5: La FingerprintMap e la DistanceMap implementate nell'applicazione.

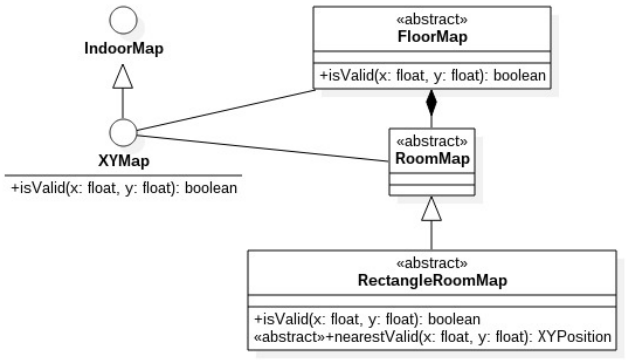


Figura 4.6: La gerarchia delle mappe.

Capitolo 5

Test e misure effettuate

5.1 Competizione EvAAL

EvAAL (Evaluating Ambient Assisted Living Systems through Competitive Benchmarking) è una competizione internazionale annuale che si pone l'obiettivo di proporre delle metriche e di valutare software riguardanti la tematica dell'Ambient Assisted Living (AAL), un programma di ricerca multidisciplinare dell'Unione Europea per contribuire alla vita indipendente di anziani e disabili[28]. La competizione è formata da quattro tracce:

1. Smartphone-based
2. Pedestrian Dead Reckoning Positioning
3. Smartphone-based (off-site)
4. PDR for warehouse picking (off-site)

Di particolare interesse ai fini del tirocinio è la prima traccia, in quanto fornisce delle metodologie rigorose per l'analisi dell'output.

Risultati degli ultimi anni

Di seguito sono presentati i risultati delle ultime due edizioni. Per ogni riga delle due tabelle è presente, in ordine, persona di riferimento, affiliazione, errore al terzo quartile.

- *2015*[1]

You Li	Univ. of Calgary (CA)	6.6m
Pawel Wilk	Samsung R&D (PL)	10.0m
Matteo Tomasi	Zetesis (IT)	-
J.C. Aguilar Herrera	Navix Indoor Navigation (MX)	-

- *2016*[2]

Frank Ebner	Univ. of Würzburg-Schweinfurt (GE)	5.4m
Pawel Wilk	Samsung, R&D Center (PL)	8.2m
Haiyong Luo	ICT & Beijing University (CN)	18.7m
Brian Bai	RMIT Univ. (AU)	44.1m
Lingxiang Zheng	Xiamen Univ. (CN)	55.7m
Zhongliang Zhao	Univ. of Bern (CH)	58.2m

Questi dati rappresentano una buona dimostrazione di ciò che si è affermato nell'introduzione: le strategie low-cost attuali per la localizzazione indoor non rispondono ancora alle esigenze del problema, in termini di accuratezza e precisione. Inoltre, sono anche una dimostrazione che la comunità scientifica sta facendo negli ultimi anni notevoli sforzi per rispondere a questa problematica.

[1] EvAAL 2015, Risultati. http://evaal.aaloa.org/2015/results
[2] EvAAL 2016, Risultati. http://evaal.aaloa.org/2016/competition-results

5.2 Metodo di testing

Le verifiche sono state effettuate presso l'Istituto di Scienze e Tecnologie Informatiche, al primo piano del CNR di Pisa. L'area considerata come test bed si compone di tre corridoi, due dei quali intermezzati da una stanza in cui sono presenti distributori automatici e ascensori. Nelle figure 5.1 e 5.2 quest'area è indicata con il colore grigio, in verde sono i punti indicanti i segnalini sopra i quali è stata acquisita la posizione rilevata dall'applicazione e il tracciato del percorso effettuato dall'attore, la cui direzione è indicata dalla freccia. Il dettaglio dei segnalini posti sul pavimento durante l'esecuzione del test è mostrato nella figura 5.3

Sono stati considerati due percorsi, come mostrato nelle figure 5.1 e 5.2. Nel primo, il più lungo, l'attore percorre la mappa da un estremo all'altro senza cambiare direzione. Questo permette di verificare l'applicazione coprendo tutta l'area disponibile. Il secondo itinerario, più corto, inizia a metà di un corridoio (appena fuori di uno studio), arriva ai distributori automatici, attende 30 secondi e torna indietro. Così facendo viene controllata anche la responsività dell'applicazione all'inversione della direzione, cambiamento nella direzione più drastico rispetto a una semplice curva. Inoltre, il percorso è pensato per enfatizzare il comportamente dell'applicazione in aree critiche, cioè i cambi di direzione tra un corridoio all'altra e, sopratutto, aree più vaste del corridorio, come appunto la piccola hall contenente il vano ascensore e i distributori di bevande. Infine, proprio per validare la robustezza delle strategie implementate, è stato scelto di far effettuare una pausa di 30 secondi all'attore che ha usato l'applicazione durante lo svolgimento del path.

I percorsi comprendono una *groundtruth*, cioè un insieme di posizioni note contrassegnate sul pavimento durante il test. Quando l'attore le calpesta, tocca contemporaneamente il pulsante dedicato al salvataggio delle coordinate in un file di

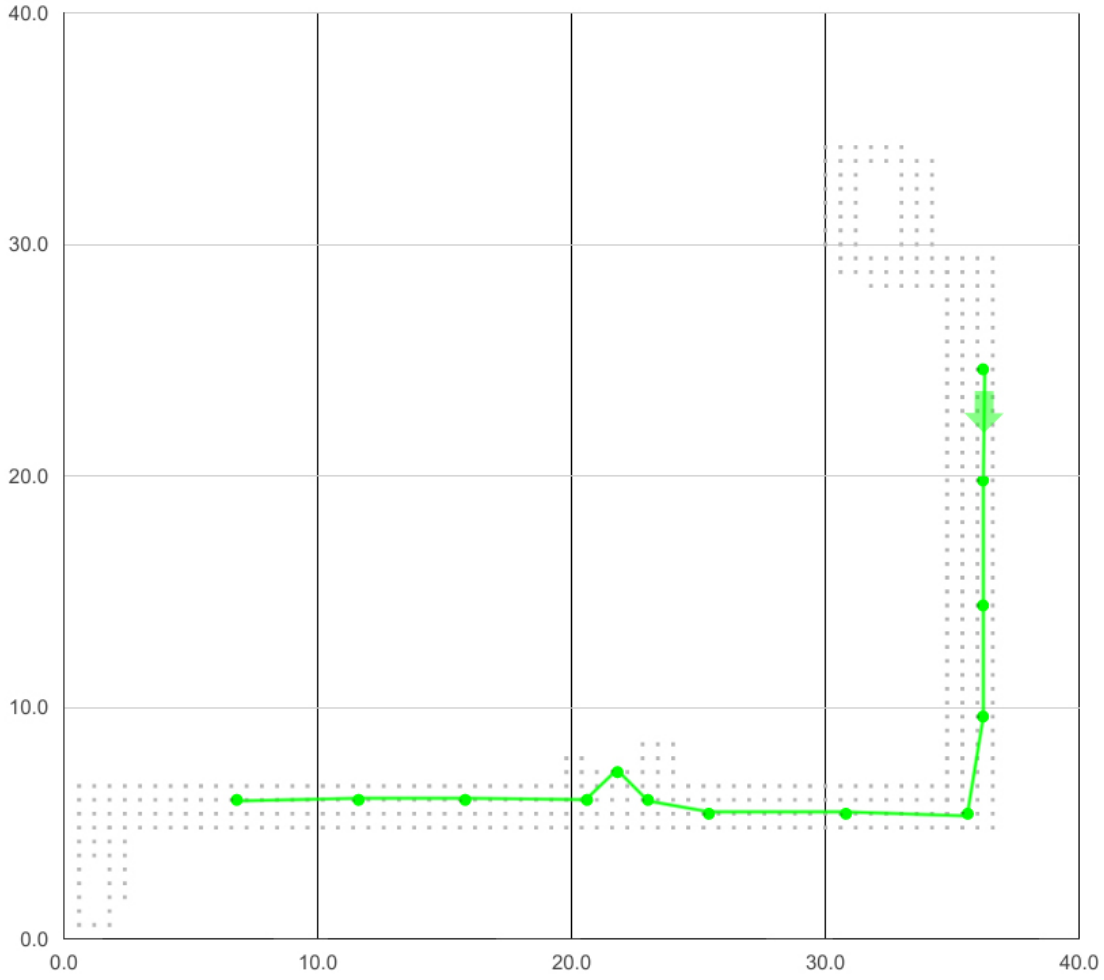


Figura 5.1: Il primo percorso utilizzato come caso di test consiste nell'attraversare tutto l'ambiente considerato, da un estremo all'altro, senza cambiare direzione.

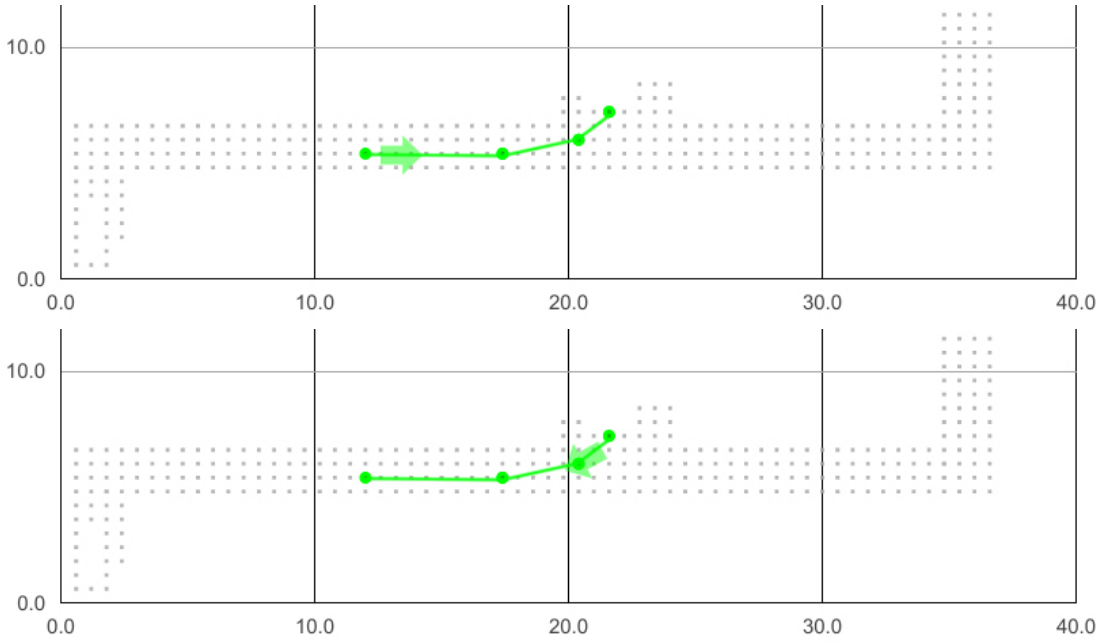


Figura 5.2: Il secondo percorso utilizzato come caso di test copre solo una porzione dell'area, ma in entrambi i versi.

Figura 5.3: Dei segnalini sono stati posti alle coordinate interessate dal test per agevolare l'esecuzione dello stesso segnalando quando l'attore doveva salvare la posizione sul file di log.

log. Terminata la prova le posizioni risultanti dalla localizzazione vengono confrontate con quelle reali, ottenendo così una stima dell'errore del sistema. Possono esserci dei trascurabili errori di posizionamento tra l'attore, il momento in cui interagisce con l'app e il segnale avente funzione di ground truth. Ciononostante, questo errore può essere considerato trascurabile, visto l'ordine di grandezza degli errori in gioco nelle due competizioni internazionali di riferimento in questo settore usando approcci di tipo smartphone-based.

Sono stati effettuati test con tutte le strategie di localizzazione disponibili nell'applicazione: data fusion con Particle Filter, data fusion con Kalman Filter, solo PDR, solo fingerprint Wi-Fi, solo fingerprint magnetico. Questo insieme di prove ha una duplice funzione: validare il sistema e verificarne il comportamento in tutti i suoi aspetti e, inoltre, cercando di isolare eventuali sensori che possono compromettere i risultati finali. La strategia può, ad esempio, essere correttamente implementata e perfettamente funzionante. Però, se in seguito a degli errori di calibrazione dei sensori o di errori umani durante la campagna di misurazione e costruzione della mappa di fingerprint, il risultato potrebbe essere compromesso da singole parti dell'applicazione finale. Questo metodo cerca di evidenziare eventuali problemi di questo tipo.

Metriche utilizzate

Ogni test effettuato ha fornito come risultato una lista di posizioni parallela a quella della groundtruth. L'errore ϵ nel posizionamento consiste nella distanza euclidea tra il punto noto e quello ricavato dalla localizzazione.

Il risultato del metodo di testing precedentemente illustrato è una serie di errori per ogni strategia, per entrambi i percorsi. Questi errori possono restituire mag-

giore informazione isolando massimo, minimo, media e terzo quartile per ogni caso di test. Queste metriche sono ampiamente usate in EvAAL. Il tirocinante, insieme al gruppo di lavoro, le hanno considerate valide anche per gli obiettivi di questo tirocinio, che tratta, del resto, di tematiche speculari a quelle della competizione.

5.3 Analisi dei risultati

Risultati percorso 1

	Particle F	Kalman F	PDR	FP Wi-Fi	FP mag
ϵ_{min}	3.78	6.71	4.82	0.97	1.10
ϵ_{max}	31.93	41.09	36.52	24.10	20.98
ϵ_{medio}	20.46	26.44	23.24	11.87	10.41
$\epsilon_{terzoquartile}$	22.91	36.09	27.12	20.50	16.18

Risultati percorso 2

	Particle F	Kalman F	PDR	FP Wi-Fi	FP mag
ϵ_{min}	0.44	16.66	0.6	1.41	5.3
ϵ_{max}	7.3	34.32	9.85	7.92	13.95
ϵ_{medio}	5.05	21.74	6.9	4.03	7.84
$\epsilon_{terzoquartile}$	6.68	23.61	9.4	5.07	9.05

Per favorire l'analisi dei risultati, i dati sono stati elaborati nei grafici delle figure 5.4, 5.5, 5.6 e 5.7. Da questi si può vedere come la tecnica del fingerprinting abbia avuto generalmente i risultati migliori.

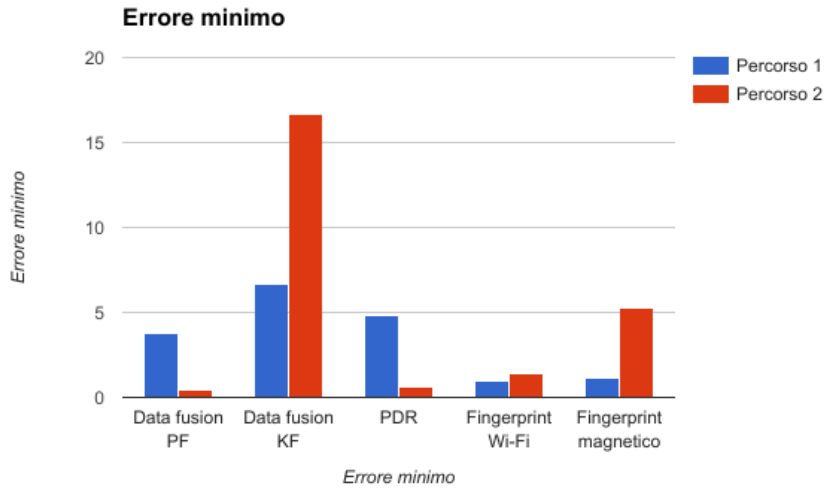


Figura 5.4: Confronto degli errori minimi effettuati utilizzando le differenti strategie nei singoli percorsi.

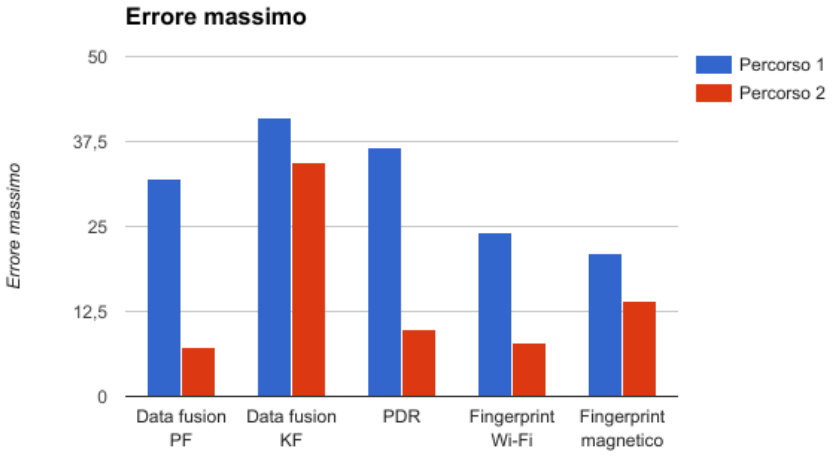


Figura 5.5: Confronto degli errori massimi effettuati utilizzando le differenti strategie nei singoli percorsi.

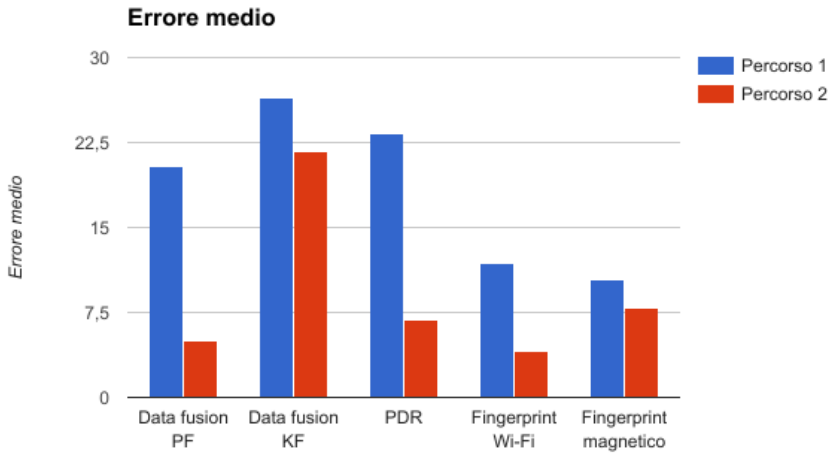


Figura 5.6: Confronto degli errori medi effettuati utilizzando le differenti strategie nei singoli percorsi.

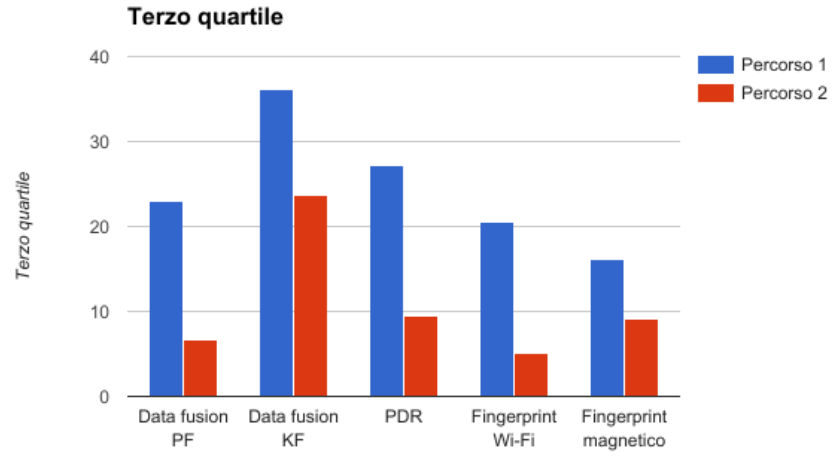


Figura 5.7: Confronto degli errori al terzo quartile effettuati utilizzando le differenti strategie nei singoli percorsi.

È inoltre interessante notare come il Particle Filter riesca a mitigare, grazie alle informazioni provenienti dalla localizzazione tramite Wi-Fi e campo magnetico, l'errore del PDR causato dai sensori relativamente di basso costo di cui è munito lo smartphone e dai modelli non troppo raffinati che rappresentano il movimento. Si osserva che questo difetto è mitigato nel percorso 2, dove è presente un'inversione di direzione.

5.4 Possibili lavori futuri

Le potenzialità e le features del lavoro presentato sono molteplici. Da un punto di vista algoritmico, può essere implementata la funzionalità del rilevamento dinamico del piano nel quale si trova l'utente (esistono soluzioni che usano il barometro per questo scopo[2]). Anche il modello di movimento può essere raffinato e le strategie di localizzazione rese migliori, implementando alcune tecniche note in letteratura, per cercare di ottenere una precisione più accurata. Si potrebbero anche implementare alcune ottimizzazioni, come l'utilizzo di una griglia per rappresentare lo spazio e quindi di coordinate meno granulari.

Da un punto di vista di user experience si potrebbe sviluppare un'interfaccia grafica che visualizzi in real-time su una mappa la posizione attuale e altri elementi interessanti all'utente. Le astrazioni che riguardano la mappa possono essere ampliate per rappresentare meglio le località indoor e fornire eventuali metodi utilitari.

5.5 Conclusioni

Sin dal momento della progettazione il software, oggetto di questo tirocinio, ha avuto come requisito fondamentale di poter essere facilmente estendibile con l'implementazione di nuove strategie e di nuovi modelli per il raffinamento dei dati provenienti dai sensori. Grazie alla modularità dell'architettura finale l'obiettivo è stato raggiunto nei tempi prestabiliti per farlo. Ciò permette la rapida integrazione di soluzioni diverse e di poterle valutare in maniera efficace. Inoltre, è anche possibile per eventuali applicazioni terze utilizzare il servizio di localizzazione indoor realizzato.

L'architettura è stata molto apprezzata e verrà utilizzata in futuri lavori del WNLab.

La precisione raggiunta per il posizionamento è soddisfacente in quanto, pur considerando alcune implementazioni basilari per determinate tecniche, l'output finale restituisce sempre posizioni valide e avviene con un errore in linea con i risultati dello stato dell'arte.

Il tirocinio è stato un'occasione per approfondire l'utilizzo dei sensori con l'Android Framework. Inoltre sono state acquisite diverse nozioni teoriche riguardo la localizzazione indoor e i metodi di stima degli stati quali Kalman Filter e Particle Filter. L'ottima opportunità fornita da questo tirocinio è stata quella di poter passare da un piano puramente teorico ad un piano implementativo. Inoltre i due metodi sono di ampio utilizzo in vari settori (i.e. computer vision, biologia molecolare).

Dal punto di vista della crescita professionale è importante sottolineare l'importanza di aver svolto un lavoro in team con spazio dedicato alle proposte e al ragionamento riguardo le idee degli altri membri, con il fine di trovare la soluzio-

ne che potesse soddisfare nel miglior modo possibile gli obiettivi prefissati. Ad arricchire quest'esperienza è stata la presenza di persone aventi background professionali diversi che ha permesso, nei ripetuti meeting di aggiornamento fatti nel periodo del tirocinio, di acquisire nuove competenze e punti di vista riguardo i problemi affrontati. Molto importante è stata anche la metodologia raggiunta nella rappresentazione dei risultati, nella quale il tirocinante non aveva esperienza.

Bibliografia

[1] K. A. Rahim, "Heading drift mitigation for low-cost inertial pedestrian navigation," 2012.

[2] F. Ebner, T. Fetzer, F. Deinzer, L. Köping, and M. Grzegorzek, "Multi sensor 3d indoor localisation," in *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*. IEEE, 2015, pp. 1–11.

[3] M. Hazas, J. Scott, and J. Krumm, "Location-aware computing comes of age," *Computer*, vol. 37, no. 2, pp. 95–97, 2004.

[4] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[5] N. Hughes, J. Pinchin, M. Brown, and D. Shaw, "Navigating in large hospitals," in *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*. IEEE, 2015, pp. 1–9.

[6] P. Hafner, T. Moder, M. Wieser, and T. Bernoulli, "Evaluation of smartphone-based indoor positioning using different bayes filters," in *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*. IEEE, 2013, pp. 1–10.

[7] X. Guo, W. Shao, F. Zhao, Q. Wang, D. Li, and H. Luo, "Wimag: Multimode fusion localization system based on magnetic/wifi/pdr," in *Indoor Positioning and Indoor Navigation (IPIN), 2016 International Conference on*. IEEE, 2016, pp. 1–8.

[8] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems (TOIS)*, vol. 10, no. 1, pp. 91–102, 1992.

[9] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: indoor location sensing using active rfid," *Wireless networks*, vol. 10, no. 6, pp. 701–710, 2004.

[10] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM, 2001, pp. 166–179.

[11] G. Deak, K. Curran, and J. Condell, "A survey of active and passive indoor localisation systems," *Computer Communications*, vol. 35, no. 16, pp. 1939–1954, 2012.

[12] F. Palumbo, P. Barsocchi, S. Chessa, and J. C. Augusto, "A stigmergic approach to indoor localization using bluetooth low energy beacons," in *Advanced Video and Signal Based Surveillance (AVSS), 2015 12th IEEE International Conference on*. IEEE, 2015, pp. 1–6.

[13] F. Potortì, A. Crivello, M. Girolami, E. Traficante, and P. Barsocchi, "Wi-fi probes as digital crumbs for crowd localisation," in *Indoor Positioning and Indoor Navigation (IPIN), 2016 International Conference on*. IEEE, 2016, pp. 1–8.

[14] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 2010, pp. 173–184.

[15] V. C. Ta, D. Vaufreydaz, T. K. Dao, and E. Castelli, "Smartphone-based user location tracking in indoor environment," in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Oct 2016, pp. 1–8.

[16] A. Moreira, M. J. Nicolau, A. Costa, and F. Meneses, "Indoor tracking from multidimensional sensor data," in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Oct 2016, pp. 1–8.

[17] Y. Li, P. Zhang, X. Niu, Y. Zhuang, H. Lan, and N. El-Sheimy, "Real-time indoor navigation using smartphone sensors," in *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*. IEEE, 2015, pp. 1–10.

[18] E. Gamma, *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[19] E. Razina and D. S. Janzen, "Effects of dependency injection on maintainability," 2007.

[20] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[21] A. Masiero, A. Guarnieri, F. Pirotti, and A. Vettore, "A particle filter for smartphone-based indoor pedestrian navigation," *Micromachines*, vol. 5, no. 4, pp. 1012–1033, 2014.

[22] J. Yim, J. Kim, G. Lee, and K. Shim, "Kalman filter vs. particle filter in improving k-nn indoor positioning," *Knowlege-Based and Intelligent Information and Engineering Systems*, pp. 203–213, 2011.

[23] Q. Zeng, B. Zhou, C. Jing, N. Kim, and Y. Kim, "A novel step counting algorithm based on acceleration and gravity sensors of a smart-phone," *International Journal of Smart Home*, vol. 9, no. 4, pp. 211–224, 2015.

[24] P. Wozniak and K. Mehner-Heindl, "Orientation determination for android smartphones."

[25] S. Colton and F. Mentor, "The balance filter," *Presentation, Massachusetts Institute of Technology*, 2007.

[26] X. Luo, W. J. O'Brien, and C. L. Julien, "Comparative evaluation of received signal-strength index (rssi) based indoor localization techniques for construction jobsites," *Advanced Engineering Informatics*, vol. 25, no. 2, pp. 355–363, 2011.

[27] R. Douc and O. Cappé, "Comparison of resampling schemes for particle filtering," in *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*. IEEE, 2005, pp. 64–69.

[28] G. van den Broek, F. Cavallo, and C. Wehrmann, *AALIANCE ambient assisted living roadmap*. IOS press, 2010, vol. 6.

Ringraziamenti

Grazie alla mia famiglia (mamma, papà, sorella, cugini, zii e nonne) per esserci stati dall'inizio alla fine anche quando non si capiva quello che stavo facendo o non era chiaro il nome degli esami che sostenevo. Grazie ai *bimbi* di I.P.A. (Info Proposte Alcoliche), amici prima di essere colleghi. Grazie al mio relatore Antonino Crivello che ha creduto in me, mi ha dedicato il suo tempo e, a volte, la pazienza. Grazie al coro MdF (Mezzogiorno di Fuoco), seconda famiglia, per l'ascolto e le belle situazioni passate insieme. Grazie a quelli di Nocera e paesi più o meno limitrofi: gli amici di sempre e le persone vicine che hanno camminato e camminano con me. Grazie a me stesso, perchè ho tenuto duro. Grazie a tutti quelli che mi aspettano quando sono in ritardo.

Una dedica speciale va a papà, che quando ero bambino mi ha insegnato a non dare nomi a caso alle cartelle e a usare il mouse con la destra invece che con la sinistra, a nonna Mina per avermi introdotto alla multimedialità e al mondo del computer in generale, a zia Paola per avermi insegnato a battere la barra spaziatrice con il pollice.