Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

PEDOMETRI PER SMARTPHONE: ANALISI, IMPLEMENTAZIONE E CONFRONTO DEI MODELLI PROPOSTI IN LETTERATURA

Relatore:
Dott.
FEDERICO MONTORI

Presentata da:
GIACOMO NERI

III Sessione
Anno Accademico 2019/2020

Sommario

Il presente elaborato è stato realizzato per fornire un'analisi dettagliata degli studi relativi all'implementazione di pedometri realizzati tramite l'utilizzo dei sensori presenti all'interno di smartphone Android.
La ricerca ha avuto lo scopo di fornire un riassunto di tutte le possibili scelte implementative proposte all'interno di questi studi, confrontandole tra loro ed evidenziandone le ridondanze, e fornire un'analisi sull'effettiva efficacia di ognuna di esse.
Per ottenere questo tipo di informazioni è stato eseguito un lavoro diviso in quattro differenti fasi.
Durante la prima fase è stata realizzata un'approfondita ricerca dei principali studi relativi all'argomento appena descritto, ottenendo un importante quantitativo di informazioni relative all'attuale stato dell'arte.
Durante la seconda fase è stata invece realizzata una dettagliata Tassonomia, ovvero uno schema contenente tutti i principali step proposti dalle diverse implementazioni, fornendo in questo modo una chiara visualizzazione delle diverse opzioni di utilizzo degli strumenti proposti.
Durante la terza fase è stata quindi sviluppata una specifica applicazione Android attraverso la quale è possibile replicare tutti i diversi strumenti proposti, con la possibilità di combinarli tra loro in ogni possibile combinazione.
Durante l'ultima fase è stato infine possibile testare, attraverso l'applicazione appena descritta, tutte le diverse implementazioni proposte all'interno dei differenti studi.
Tramite l'utilizzo di specifici test è stato infatti possibile raccogliere un quantitativo di dati sufficiente a trarre conclusioni sulla reale efficacia delle varie implementazioni.
Questi dati hanno quindi permesso di identificare i punti di forza ed i limiti di ciascuna implementazione, e di determinare quale di esse fornisca effettivamente i risultati migliori.

Introduzione

In letteratura è possibile trovare una grossa quantità di studi relativi all'implementazione di pedometri realizzati tramite l'utilizzo dei sensori presenti all'interno di dispositivi mobile.
Analizzando questi studi si può notare come nella maggior parte di essi vengano proposte delle scelte implementative spesso ricorrenti, che differiscono le une dalle altre per il solo valore di una determinata variabile, o per l'ordine di utilizzo di specifici calcoli.
Inoltre, nella quasi totalità degli studi viene dichiarata una assoluta precisione dell'implementazione proposta, rendendo quindi difficile comprendere gli effettivi punti di forza ed i limiti di ciascuna di esse.
Il presente elaborato è stato quindi realizzato per fornire un'analisi dettagliata della maggior parte di questi studi, in particolare quelli relativi all'utilizzo di dispositivi Android, con lo scopo di riassumere insieme tutte le possibili scelte implementative proposte, confrontandole tra loro ed evidenziandone le principali differenze e somiglianze riscontrate.
La ricerca ha avuto come obiettivo anche l'analisi dell'effettiva efficacia di ognuna di queste implementazioni.

Il lavoro è stato quindi suddiviso in quattro differenti fasi, ognuna delle quali è stata descritta all'interno di uno specifico capitolo.
La prima fase, descritta all'interno del primo capitolo, ha previsto un'approfondita ricerca dei principali studi relativi all'argomento appena descritto, in modo da raccogliere il maggior quantitativo di informazioni possibile.
All'interno di questo capitolo viene quindi fornita un'approfondita analisi dell'attuale stato dell'arte, attraverso la quale è possibile ottenere una prima idea generale della

principale struttura algoritmica utilizzata dalla diverse implementazioni e dei principali strumenti utilizzati all'interno di esse.
La seconda fase, descritta all'interno del secondo capitolo, ha previsto invece lo sviluppo di una dettagliata *Tassonomia*.
All'interno di essa sono stati inseriti tutti i principali *step* proposti dalle diverse implementazioni, accorpando insieme i diversi strumenti ridondanti e fornendo una chiara visualizzazione diramata delle diverse opzioni di utilizzo degli stessi.
La terza fase, descritta all'interno del terzo capitolo, ha previsto invece lo sviluppo di un'applicazione Android attraverso la quale è possibile testare tutte le diverse implementazioni proposte all'interno dei differenti studi.
L'implementazione dell'applicazione è stata possibile grazie all'utilizzo della Tassonomia realizzata durante la precedente fase, e permette quindi di replicare i diversi strumenti proposti all'interno di essa, con la possibilità di utilizzarli in ogni possibile combinazione.
L'ultima fase, descritta all'interno del quarto ed ultimo capitolo, ha previsto infine l'utilizzo dell'applicazione appena descritta per testare le diverse implementazioni.
Tramite l'utilizzo di specifici test è stato possibile raccogliere un quantitativo di dati sufficiente a trarre conclusioni sulla reale efficacia delle varie implementazioni.
All'interno di questo capitolo vengono quindi descritti i diversi test eseguiti, e vengono fatte considerazioni sui risultati ottenuti.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

I Pedometri

1.1 Storia e Definizione dei Pedometri

Una generica definizione di pedometro, anche chiamato contapassi, può essere data come *"Strumento in grado di misurare il numero di passi compiuti da un individuo tramite la rilevazione di particolari movimenti"*.
Lo scopo di questo strumento è quindi quello di fornire un'indicazione quanto più precisa possibile del numero di passi compiuti da una persona durante una sua camminata.
L'esigenza di avere questo tipo di informazione risale a diversi secoli fa, quando si diffusero le prime idee di sviluppo di contapassi meccanici da poter utilizzare in ambito militare.

Con l'avanzamento tecnologico anche lo sviluppo dei pedometri si è modernizzato, passando da implementazioni completamente meccaniche a una più complessa combinazione di sensori meccanici e componenti software, in grado di fornire risultati molto più accurati.
La notevole riduzione della dimensione di questi sensori ha inoltre permesso nell'ultimo decennio di poterli inserire agevolmente all'interno di dispositivi mobili di uso comune, quali smartphone e tablet.
La potenza di calcolo di questi dispositivi è notevolmente cresciuta negli ultimi anni,

tanto da diventare ampiamente sufficiente a poter garantire una corretta esecuzione degli algoritmi utilizzati per implementare i pedometri.
I pedometri implementati all'interno di questi dispositivi si basano quindi principalmente su due fattori: i sensori e l'implementazione software.

I sensori principalmente utilizzati nei pedometri sono tre: l'accelerometro, il giroscopio e il magnetometro.
L'*accelerometro* è uno strumento in grado di misurare l'accelerazione che un determinato oggetto sta subendo in un preciso momento. Il funzionamento di questo tipo di sensori avviene tramite l'utilizzo di una piccola massa circondata da un insieme di molle. Lo spostamento di questa massa, dovuta ad un'accelerazione esterna, provoca una forza su queste molle, che viene registrata per ottenere il valore dell'accelerazione stessa.
Utilizzando queste informazioni è possibile capire l'accelerazione che sta agendo sull'oggetto in un determinato momento e, di conseguenza, è possibile determinarne lo spostamento.
Nell'utilizzo di questo sensore bisogna sempre considerare la presenza dall'accelerazione gravitazionale, che viene costantemente registrata, e che va quindi rimossa per poter ottenere la reale accelerazione del dispositivo.
Gli accelerometri maggiormente montati sugli smartphone sono della tipologia definita *"a tre assi"*. Questi tipi di accelerometri funzionano dando un'indicazione dell'accelerazione che il dispositivo sta subendo sui suoi tre assi, ovvero l'asse X, l'asse Y e l'asse Z.
Il *giroscopio* è invece uno strumento in grado di misurare il moto rotazionale di un oggetto, ovvero la velocità angolare con cui sta ruotando.
Anche il questo caso la maggior parte degli smartphone utilizza dei giroscopi a tre assi. In particolare questi giroscopi forniscono indicazioni definite *"Pitch"*, *"Roll"*e *"Yaw"*, indicanti rispettivamente la velocità angolare registrata attorno all'asse X, Y e Z.
Infine, il *magnetometro* è lo strumento utilizzato per misurare l'intensità di un campo magnetico. In assenza di calamite nelle vicinanze è quindi utile a rilevare il campo geomagnetico, ovvero il campo magnetico generato dal pianeta Terra. Utilizzando questo sensore è quindi possibile identificare i poli terrestri e, di conseguenza, il modo in cui è

orientato il dispositivo.
Anche questo sensore, come per i sensori precedenti, è solitamente implementato sui tre assi X, Y, e Z.
L'utilizzo combinato di questi tre sensori permette di avere un'idea molto precisa dei movimenti che sta subendo il dispositivo, in particolare di identificare in ogni momento in che modo è girato, se sta subendo delle rotazioni e sta subendo degli spostamenti.

La parte di implementazione software dei pedometri è sicuramente la parte che più identifica il corretto funzionamento dello stesso.
I sensori che si possono trovare all'interno degli smartphone sono molto simili tra loro, e forniscono dati quasi sempre molto affidabili. La principale differenza riscontrabile è nella frequenza di campionamento dei dati, che varia dai 100 Hz negli smartphone più economici ai 500 Hz negli smartphone più costosi, che però non incide direttamente sulla qualità dei dati registrati, che rimangono sufficientemente accurati anche per i sensori montati sui dispositivi di fascia più bassa.
Si può quindi dire che la principale sfida nello sviluppo di un pedometro altamente preciso si concentra sulla ricerca di un algoritmo in grado di saper gestire ed elaborare al meglio i dati fornitigli dai vari sensori.
Lo scopo della seguente trattazione è fornire un confronto dettagliato dei principali algoritmi utilizzati per lo sviluppo di pedometri presenti in letteratura, e fornire dei risultati concreti su quali di questi siano i più efficaci.
Le principali metodologie di implementazione utilizzate per sviluppare gli algoritmi di riconoscimento dei passi attraverso l'utilizzo dei sensori presenti su dispositivi mobili verranno descritte nella sezione seguente.

1.2 Stato dell'Arte

La ricerca di un'implementazione software di un pedometro completamente affidabile, in grado di fornire risultati corretti in ogni condizione di utilizzo, è argomento di studio ormai da diversi anni, ed è stata trattata in diversi paper e pubblicazioni scientifiche.
Per la stesura della presente Tesi è stata fatta una ricerca approfondita relativa ai principali articoli presenti sui più affidabili siti contenenti pubblicazioni relative a informazioni di ingegneria, informatica e tecnologia tra i quali, principalmente, *"IEEE Xplore"* [1].
É di seguito fornito un elenco completo dei più rilevanti articoli trovati relativi allo sviluppo di pedometri tramite l'utilizzo dei sensori presenti negli smartphone. Questo elenco ha lo scopo di fornire un'analisi completa ed esaustiva delle principali implementazioni e scelte algoritmiche presenti in letteratura.

1.2.1 Analisi di uno studio completo

Il primo articolo preso in considerazione [9] utilizza tutti e tre i sensori descritti nel precedente capitolo, combinandoli per ottenere le informazioni necessarie a identificare i passi eseguiti dall'utente.
L'algoritmo esposto in questo articolo ha la particolarità di essere suddiviso in diversi step, da eseguire uno di seguito all'altro, ognuno dei quali si occupa di un determinato compito.
La struttura di questo algoritmo può essere identificata nella quasi totalità dei paper analizzati.
I principali step che si trovano in questa struttura sono:

1. Raccolta dei dati dei sensori.

2. Applicazione di specifici filtri ai dati registrati, in modo da identificare e rimuovere i dati non necessari, e diminuire l'errore nei calcoli successivi.

3. Elaborazione dei dati raccolti ed identificazioni di informazioni rilevanti da utilizzare successivamente per il riconoscimento dei passi.

4. Applicazione di un algoritmo di decisione in grado di definire se i dati presi in input corrispondono al verificarsi di un passo.

Le caratteristiche principali dell'algoritmo esposto in questo studio sono l'utilizzo di un filtro passa-basso e il calcolo della matrice di rotazione.

Il *filtro passa-basso* è un filtro da applicare ai dati registrati dai sensori con lo scopo di ridurre i dati in ingresso ai soli corrispondenti a frequenze minori di una certa soglia. Questo processo serve principalmente a ridurre i dati in ingresso e far sì che vengano passati solo quelli più facilmente riconducibili ai movementi dell'utente, rimuovendo invece i dati probabilmente generati da vibrazioni non riconducibili a una cammiata.

La *matrice di rotazione* è invece uno strumento matematico utilizzato per convertire i valori registrati dai sensori in valori relativi a un sistema di riferimento fisso.
I valori registrati da un sensore, come ad esempio l'accelerometro, sono infatti relativi al sistema di riferimento del dispositivo, e quindi variano in base a come esso è orientato. Utilizzando questa matrice è invece possibile ottenere dei valori che si riferiscono a un sistema di riferimento indipendente dall'orientamento del dispositivo.

L'applicazione di questi strumenti ai dati ricevuti in ingresso permette a questo algoritmo di estrapolarne i più significativi.
In particolare, viene utilizzato un algoritmo definito *"Algoritmo dei Picchi"*, anch'esso presente nella quasi totalità degli studi che verranno successivamente analizzati. Come è facile intuire dal nome, questo algoritmo si occupa di estrapolare i valori di *massimo locale* e di *minimo locale* presenti in un insieme di dati.

Una delle differenza maggiori trovate all'interno degli studi analizzati riguarda l'analisi dei dati filtrati.
La maggior parte di essi propone uno sviluppo non in real-time. Viene infatti utilizzato

un algoritmo in grado di registrare i dati per un determinato lasso di tempo, identificato in base a specifici parametri, ed avvia l'algoritmo di decisione direttamente su quell'intero segmento di dati.
I restanti studi propongono invece uno sviluppo del tutto in real-time, in quanto l'algoritmo viene lanciato direttamente sui dati che vengono letti in input.
L'articolo che si sta ora analizzando fa parte del primo raggruppamento, in quanto i segmenti vengono identificati in base ai valori registrati dal giroscopio, e da essi vengono successivamente estrapolati i dati rilevanti, ovvero i *picchi*.
Successivamente questi dati vengono passati a un algoritmo di decisione in grado di definire, in base a certi valori soglia, se il segmento preso in considerazione corrisponde o meno ad un passo.

1.2.2 Altri studi basati su algoritmi a step

Nonostante non sia sempre esplicitato, la maggior parte degli articoli analizzati propone uno schema algoritmico suddiviso in step sequenziali del tutto simile a quello precedentemente riportato.
Tutti gli studi analizzati in questa sezione condividono questa caratteristica, ma differiscono in altre scelte implementative.
Tra questi, gli articoli che descrivono l'utilizzo di un filtro passa-basso sono [6], [15], [17], [13] e [18]. La principale differenza nell'utilizzo di questo filtro si può riscontrare nella frequenza di taglio, che varia dai 2Hz ai 10Hz.
Di questi articoli, solo [17] e [13] descrivono un algoritmo di riconoscimento completamente real-time, mentre tutti gli altri considerano un'implementazione basata sull'identificazione dei segmenti.
In particolare, mentre [15] e [18] riconoscono i segmenti in un modo molto simile a quello espresso nel primo articolo [9], l'articolo [6] propone un diverso algoritmo di suddivisione in segmenti.
Questo studio propone infatti un algoritmo definito *"Algoritmo di Intersezione con l'Asse delle Ascisse'* che, come è facilmente intuibile dal nome, si propone di individuare i

segmenti in base ai punti in cui il grafico relativo ai dati registrati dai sensori interseca l'asse X. Come verrà descritto successivamente, questo algoritmo verrà proposto anche in molti altri studi.
Per quanto riguarda l'utilizzo dei sensori, [6] è l'unico a proporre il solo utilizzo del giroscopio. Gli articoli [13] e [18] fanno uso del solo accelerometro, mentre [17] dichiara la necessità di utilizzare tutti e tre i sensori prima elencati, ovvero giroscopio, accelerometro e magnetometro.
Quest'ultimo articolo differisce infatti dagli altri quattro appena citati poiché include nel suo algoritmo il calcolo della matrice di rotazione con lo scopo di ottenere una conversione dei dati registrati da un sistema di riferimento mobile, e cioè relativo al dispositivo, a un sistema di riferimento fisso, cioè indipendente dal dispositivo.

Altri articoli che propongono questa conversione sono [17], [16], [20], [5] e [12]. In tutti questi casi viene esplicitato come sia necessario avere i dati relativi a tutti e tre i sensori, ovvero accelerometro, giroscopio e magnetometro. La conversione viene però descritta in due modalità differenti.
Per quanto riguarda i primi tre articoli, viene proposto l'utilizzo della matrice di rotazione che, come descritto in precedenza, attraverso l'utilizzo combinato dei dati registrati, è in grado di fornire uno strumento comodo per eseguire la conversione.
Negli altri due casi vengono invece proposte specifiche formule di calcolo dei valori nel sistema di riferimento fisso, che non necessitano dell'utilizzo della matrice di rotazione.
La conversione dei valori nel sistema di riferimento fisso viene spesso utilizzata per poter successivamente concentrare i calcoli sui soli valori di accelerazione relativi all'asse Z, ovvero quello verticale rispetto all'utente, che corrisponde all'asse più soggetto a variazioni costanti durante una camminata, come si può vedere in [9] e [16].
La conversione nel sistema di riferimento fisso dà quindi la possibilità di risolvere il problema relativo alla presenza dell'accelerazione gravitazionale all'interno delle misurazioni fornite dall'accelerometro. Con l'identificazione del sistema fisso si ha infatti la certezza che questo valore venga segnato sempre nell'asse Z, in modo che possa essere facilmente rimossa.
In alternativa a questa tecnica, gli altri studi utilizzano i dati registrati dall'accelero-

metro per calcolarne la risultante, ovvero un valore unico in grado di descrivere la loro somma vettoriale. In questo modo è possibile ottenere un'indicazione sull'accelerazione totale che sta agendo sul dispositivo, comprensiva di quella gravitazionale, che però può essere facilmente rimossa.

Per quanto riguarda gli studi descritti in questo paragrafo, alcuni di essi propongono un sistema di riconoscimento dei passi basato sull'identificazione di segmenti temporali (in particolare [16] e [5]), mentre altri propongono un sistema di riconoscimento dei passi completamente real-time ([20] e [12]).

Anche molti altri articoli propongono una suddivisione dell'algoritmo in step, soffermandosi però principalmente sull'identificazione dei valori rilevanti più che sull'utilizzo dei filtri.
A esempio gli articoli [8], [10], [7], [11] e [4] forniscono un'analisi molto simile tra loro. Tutti questi fanno utilizzo del solo sensore di accelerometro e non presentano alcun approfondimento relativo al filtraggio dati ricevuti in input, a parte l'articolo [8] che ne fornisce un piccolo accenno senza però scendere nello specifico.
Un'altra caratteristica che accomuna tutti questi articoli è l'utilizzo dell'*Algoritmo dei Picchi* per l'Identificazione dei valori chiave potenzialmente riconducibili al verificarsi di un passo.
Una parte di questi articoli, [8] e [7], utilizza questo algoritmo all'interno di un'implementazione completamente real-time, mentre gli altri, [10], [11] e [4], lo utilizzano per riconoscere i picchi presenti in determinati segmenti temporali.

1.2.3 Altri studi

Altri studi si sono rivelati invece poco adatti alla ricerca presentata in questa Tesi. Ad esempio, lo studio [14] fornisce dettagliate informazioni sull'utilizzo di complessi filtri

da applicare ai valori registrati dall'accelerometro, senza però approfondire in che modo utilizzare questi dati all'interno di un algoritmo di riconoscimento dei passi.
Lo studio [19] è invece probabilmente il meno esaustivo, in quanto fornisce solo una spiegazione veramente poco approfondita dell'algoritmo implementato, e si sofferma maggiormente sull'analisi dei componenti Android utilizzati per sviluppare l'applicazione utilizzata per eseguire i propri test.

1.2.4 Implementazioni presenti in GitHub

Oltre a un'analisi relativa agli articoli e agli studi presenti in letteratura, è stata fatta un'approfondita ricerca anche all'interno dei principali servizi di hosting per progetti software tra cui, principalmente, *"GitHub"* [2].
All'interno di questo servizio sono stati trovati una grossa quantità di progetti, molto simili tra loro, di implementazione di pedometri.
Tutti questi progetti sono stati direttamente implementati per verificarne l'effettiva efficacia ed il corretto funzionamento. Purtroppo la maggior parte di questi progetti si sono rivelati completamente deludenti, in quanto fornivano risultati del tutto incoerenti con i test fatti, segnalando il verificarsi di nuovi passi ad ogni minimo movimento del dispositivo.
L'unico progetto degno di nota, che si è scelto quindi di prendere in considerazione, è denominato *Algoritmo di Bagilevi* [3].
Purtroppo questa implementazione non è associata a nessuno articolo scientifico, rendendo quindi impossibile avere una reale idea di ciò che l'autore ha voluto implementare, ma lasciando solo la possibilità di intuirlo analizzandone il codice. Non è stato neanche possibile ottenere informazioni relative a studi sulla sua efficacia. Nonostante ciò, è sembrato essere il più affidabile tra tutte i progetti trovati.
Come verrà fatto notare successivamente, anche questa implementazione si è però dimostrata del tutto deludente, riportando risultati decisamente sottostimati rispetto ai test fatti, e completamente incoerenti con tutte le altre implementazioni sviluppate partendo dagli algoritmi esposti negli articoli precedentemente descritti.

1.2.5 Informazioni aggiuntive

Nella quasi totalità degli articoli studiati viene indicata la modalità in cui l'utente deve tenere il dispositivo affinché l'algoritmo sia in grado di riconoscere al meglio i passi fatti.
Negli studi [9], [6], [10] e [14] viene indicato che per far lavorare l'algoritmo al meglio l'utente deve tenere il telefono in tasca o alternativamente, nel caso dell'ultimo di questi, posizionato intorno alla vita.
La maggior parte degli articoli richiede invece l'utilizzo del dispositivo in modalità *texting*, ovvero nella posizione in cui viene tenuto un telefono quando viene utilizzato per scrivere.
Questa modalità è anche quella scelta per eseguire i test di comparazione che verranno esaustivamente descritti nei capitoli successivi.
Gli articoli che richiedono questo tipo di modalità sono [7], [16], [18], [15], [13], [12] e [8]. Alcuni di questi affermano però di poter lavorare anche con il dispositivo in altre posizioni, quali ad esempio vicino all'orecchio, come quando si effettua una chiamata, oppure in mano facendola oscillare durante la camminata, o anche in tasca, dentro ad una borsa o posizionata intorno a un braccio.
Ci sono poi articoli, [20], [11] e [4], che affermano di non avere bisogno di indicare il posizionamento del dispositivo, in quanto l'algoritmo da loro proposto è in grado di lavorare al meglio indipendentemente dal modo in cui esso viene tenuto.
Solamente nei due restanti casi, [17] e [5], questa informazione non viene specificata.

Lo studio di tutti gli articoli fino ad ora descritti ha anche fatto emergere che la maggior parte di essi non ha come unico scopo la ricerca di un algoritmo di riconoscimento passi, ma include anche la ricerca di altre informazioni.
Ad esempio gli articoli [9], [8], [6] e [15] forniscono un algoritmo in grado di riconoscere se l'utente sta camminando o se sta salendo o scendendo le scale. Il terzo di questi ar-

ticoli indica anche di essere specifico per le camminate lente, mentre il quarto di essere in grado di identificare in che modo l'utente sta tenendo il dispositivo e di conseguenza adattare l'algoritmo di riconoscimento, caratteristica riscontrata anche in [18].
Gli studi [7], [16] e [17] approfondiscono invece l'utilizzo del pedometro applicato allo studio del *Dead Reckoning*, ovvero l'insieme di calcoli effettuati per determinare la posizione di un dispositivo rispetto a una sua determinata posizione di partenza tramite l'utilizzo dei sensori presenti su un dispositivo mobile.
Lo studio [14] fornisce un algoritmo in grado di identificare se l'utente sta compiendo una camminata o una corsa, mentre [4] si concentra sull'individuazione dei passi *falsi positivi*, cioè un insieme di dati non corrispondente a un passo che viene però spesso individuato erroneamente come tale.
Tutti questi aspetti particolari non sono stati presi in considerazione successivamente poiché non in linea con il principale interesse della ricerca.

Capitolo 2

Tassonomia dei Pedometri per Smartphone

2.1 Definizione di Tassonomia

Partendo dalle informazioni raccolte con l'analisi di tutti gli studi precedentemente descritti si è potuto generare una Tassonomia completa ed esaustiva comprendente tutte le possibili combinazioni di implementazioni dei pedometri.
Con *Tassonomia* si intende una classificazione, generalmente rappresentata tramite una struttura ad albero, di argomenti o concetti ordinata secondo determinati criteri.
Per il presente studio questo tipo di struttura è stata utilizzata per poter proporre un grafico riassuntivo di tutte le implementazioni precedentemente analizzate, in grado di accorpare i concetti ridondanti ed esplorare tutti gli iter algoritmici definiti all'interno dei vari studi.
La creazione di questa tassonomia ha inoltre implicitamente creato nuove possibili combinazioni di implementazione, derivate da quelle già esistenti.
La tassonomia qui illustrata è stata poi utilizzata come punto di partenza per generare un dettagliato *Diagramma di Flusso* utile al successivo sviluppo di un'applicazione Android in grado di implementare tutte le possibili combinazioni descritte all'interno della stessa, come verrà ampiamente descritto nel successivo capitolo.

2.2 Le Variabili Discriminanti

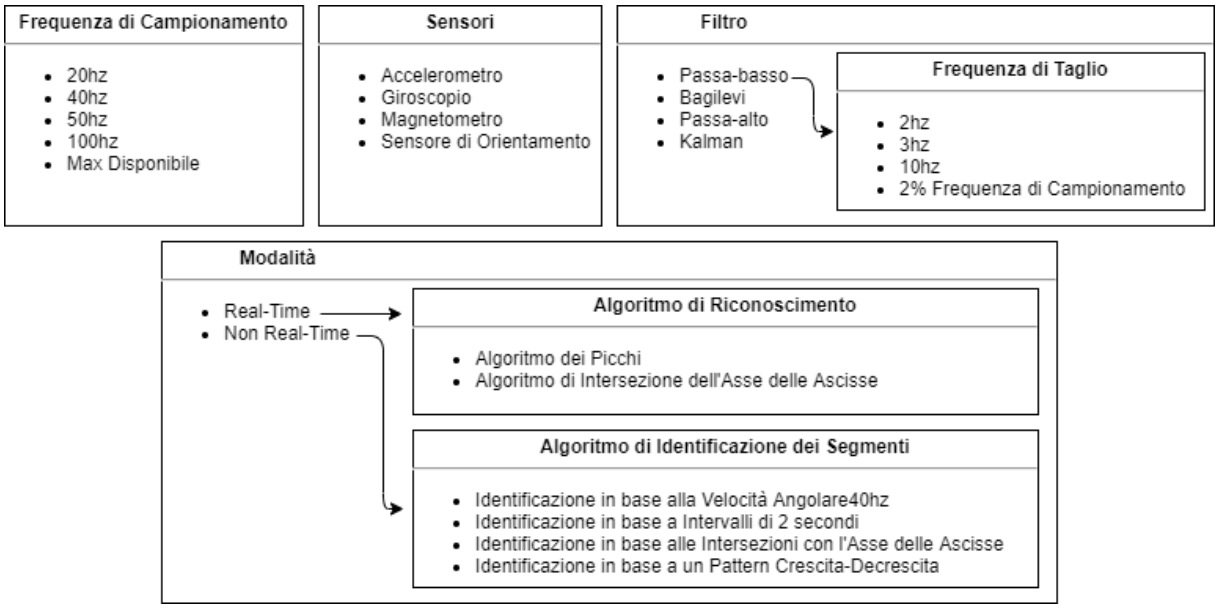

Figura 2.1: Variabili Discriminanti

La figura 2.1 fornisce una sintesi delle principali variabili discriminanti che caratterizzano i differenti algoritmi trovati negli studi precedentemente illustrati.All'interno dell'immagine viene proposto, per ogni variabile, l'insieme di valori che possono assumere, corrispondente a quelli identificati all'interno degli studi.
All'interno della tassonomia queste variabili rappresentano i punti principali in cui i diversi algoritmi si diversificano. La scelta dei valori da attribuire a ciascuna di esse permette di definire uno specifico percorso corrispondente a un algoritmo utilizzabile per l'implementazione di un pedometro.
Le principali variabili esposte in questa figura sono:

1. Frequenza di Campionamento.

2. Sensori.

3. Filtro.
4. Frequenza di Taglio.
5. Modalità di registrazione dei dati.
6. Algoritmo di Riconoscimento.
7. Algoritmo di Identificazione dei Segmenti.

La prima variabile, cioè la *Frequenza di Campionamento* riguarda la scelta della frequenza con cui registrare i dati che, come si può vedere, può variare dai 20Hz fino ad arrivare all'utilizzo della massima frequenza di registrazione disponibile nel dispositivo.
Altra discriminante particolarmente importante è la scelta dei *Sensori* da utilizzare, in quanto determina la possibilità di utilizzare determinati strumenti quali, ad esempio, la matrice di rotazione, descritta precedentemente.
Nell'immagine 2.1 si può notare come, oltre ai tre sensori già descritti in precedenza, sia presente anche il *Sensore di Orientamento*. Questo sensore si differenzia dagli altri tre poiché è un sensore software, e fornisce quindi dati sulla base di informazioni fornite da altri sensori.
In particolare questo sensore combina i valori registrati da accelerometro e magnetometro per fornire informazioni relative all'orientamento del dispositivo.
Per questo motivo questo sensore non è stato descritto in precedenza, ma è stato direttamente inserito in questa sezione.
Tra tutte le altre discriminanti, quelle che più identificano l'algoritmo da utilizzare sono le due successivi.
La prima di queste corrisponde al *Filtro*, il quale identifica in che modo si vuole filtrare i dati registrati dai sensori prima di utilizzarli per effettuare i calcoli successivi.
I filtri Kalman e passa-alto sono i meno utilizzati all'interno degli studi, mentre il filtro Bagilevi, come si può intuire, corrisponde a quello utilizzato all'interno della rispettiva implementazione trovata su GitHub.
Il filtro passa-basso è invece il filtro nettamente più utilizzato all'interno degli studi e, come si vedrà successivamente, anche quello in grado di fornire i risultati migliori.
La scelta di questo filtro necessita anche della scelta della *Frequenza di Taglio*, che come

si può vedere varia dai 2Hz ai 10Hz o, in alternativa, a un valor pari al 2% della frequenza di campionamento dei dati.
L'altra variabile discriminante particolarmente identificativa è invece la *Modalità* di registrazione dei dati, la quale identifica con quale modalità utilizzare i dati per determinare l'identificazione di un passo.
In particolare questa variabile permette di scegliere se utilizzare una modalità *Real-Time*, cioè effettuando un'analisi dei dati immediatamente sucessiva alla loro registrzione, o una modalità *Non Real-Time*, la quale richiede invece che i dati registrati vengano suddivisi in segmenti temporali prima di essere analizzati.
Il primo caso richiede la successiva scelta dell'*Algoritmo di Riconoscimento* da utilizzare, che può corrispondere all'*algoritmo dei picchi* o all'*algoritmo di intersezione con l'asse delle ascisse*.
Nel secondo caso invece l'algoritmo di riconoscimento utilizzato corrisponde sempre all'*algoritmo dei picchi*, ma è necessario specificare quale *Algoritmo di Identificazione dei Segmenti* utilizzare.

2.3 Esposizione della Tassonomia

Viene di seguito fornita una descrizione dettagliata dei principali step illustrati all'interno della tassonomia, ognuno dei quali verrà poi descritto più nello specifico indicando quale parte di algoritmo rappresenta e a quali altre parti della tassonomia è direttamente collegato.
All'interno delle immagini utilizzate per illustrare la tassonomia vengono indicate con delle frecce le strade percorse in almeno uno degli studi analizzati.
Tutte le altre combinazioni sono in ogni caso strade percorribili e possono essere utilizzate per sperimentare nuovi pedometri con configurazioni differenti rispetto a quelle trovate negli studi.
Eventuali percorsi obbligati o strade non percorribili verranno evidenziati durante l'analisi di ognuno degli step principali della tassonomia.
La Tassonomia può essere osservata nella sua interezza nella figura 2.10.

2.3.1 Raccolta dei Dati

La figura 2.2 mostra la prima fase della tassonomia e corrisponde alla raccolta dei dati attraverso l'utilizzo dei sensori.
Come si può vedere nell'immagine, i sensori utilizzabili possono essere l'accelerometro, il giroscopio, il magnetometro e il sensore di orientamento.
Per ognuno di essi viene indicata con una freccia quale frequenza di campionamento utilizzare durante la registrazione dei dati
Come detto in precedenza, le combinazioni inserite corrispondono a quelle trovate all'interno degli studi, ma non si esclude che nuovi valori di frequenza di campionamento o nuove combinazioni possano essere utilizzate generando comunque risultati positivi.
I dati registrati vengono quindi passati alla sezione successiva, nella quale vengono esposti tutti i possibili filtri utilizzabili.

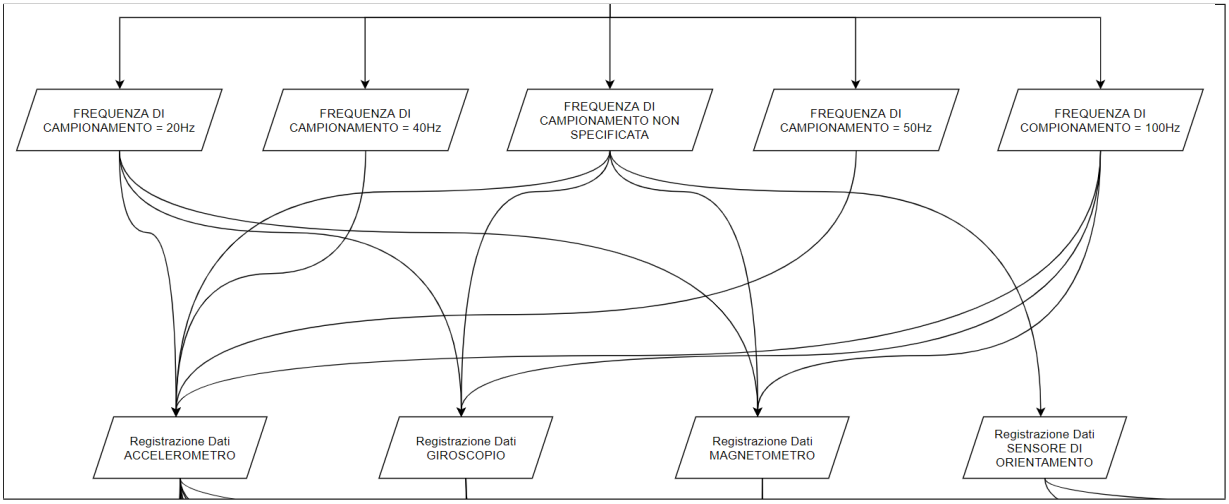

Figura 2.2: Raccolta Dati

2.3.2 Utilizzo dei Filtri

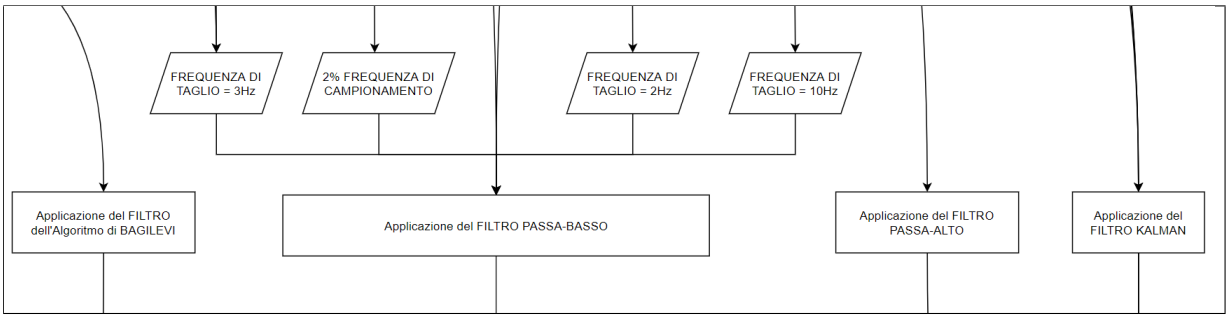

Figura 2.3: Filtri

Questa sezione, illustrata dalla figura 2.3, riassume tutti i filtri esposti all'interno degli studi.

Come descritto precedentemente questi filtri corrispondono al filtro trovato all'interno dell'algoritmo di Bagilevi, al filtro passa-basso, al filtro passa-alto, per il quale è neces-

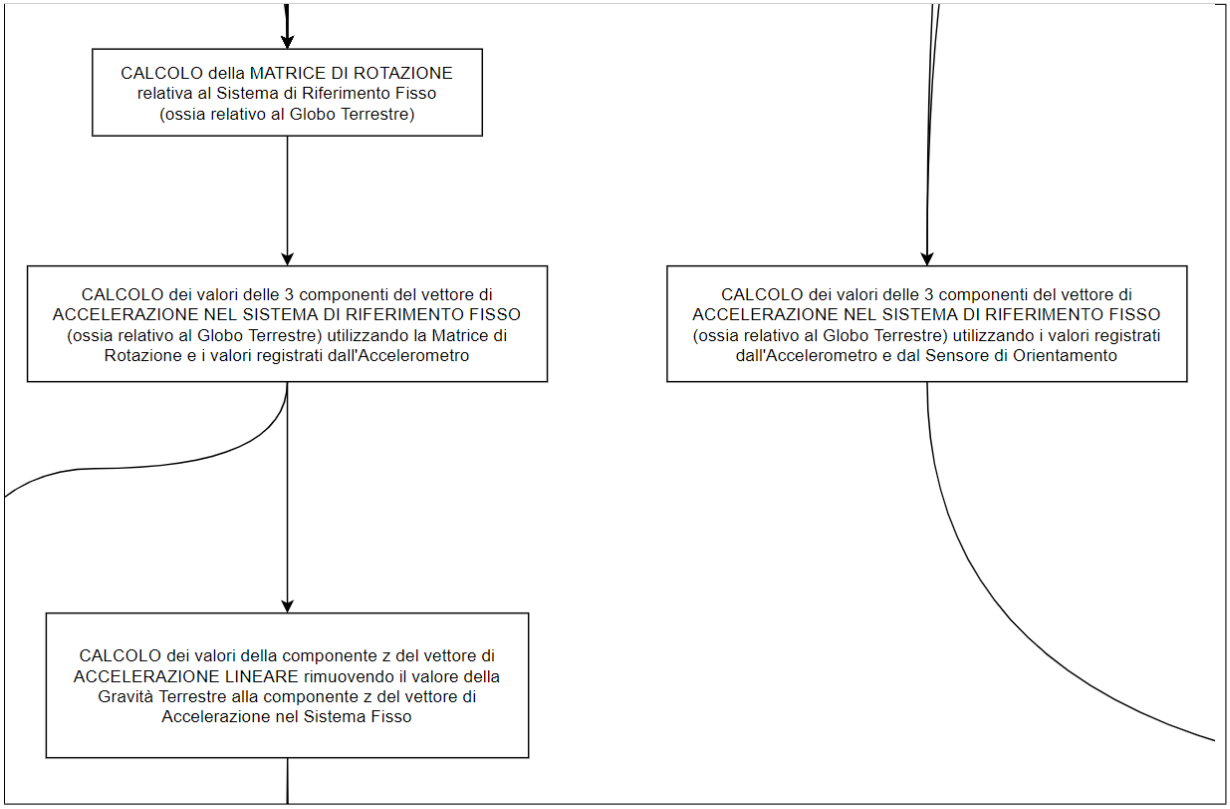

Figura 2.4: Calcolo Sistema di Riferimento Fisso

sario specificare la frequenza di taglio, e al filtro di Kalman.
Le frecce entranti indicano per ogni filtro quali dei sensori elencati precedentemente lo utilizzano. In particolare il filtro di Bagilevi viene utilizzato per filtrare dati provenienti dall'accelerometro, così come anche il filtro di Kalman, il quale però viene utilizzato per filtrare anche dati provenienti da magnetometro e giroscopio.
Il filtro passa-alto viene utilizzato solo con i dati del giroscopio, mentre il filtro passa-basso filtra dati diversi in base alla frequenza di taglio scelta.
Nel caso di una frequenza di taglio pari a 3Hz viene utilizzato per filtrare dati provenienti da accelerometro e giroscopio, nel caso di una frequenza di taglio pari a 10Hz o pari al 2% di quella di campionamento viene utilizzato per filtrare solo i dati dell'accelerometro, mentre nel caso di una frequenza di taglio pari a 2Hz viene utilizzato per filtrare i dati del solo giroscopio.

Anche in questo caso non si esclude che l'applicazione dei filtri ai valori registrati da diversi sensori possa generare risultati comunque positivi. Si può anzi affermare che la differente scelta della frequenze di taglio per il filtro passa-basso porterà nelle sperimentazioni effettuate ad ottenere risultati molto interessanti, come si potrà osservare nell'ultimo capitolo.
I dati filtrati di tutti i sensori vengono quindi passati alla sezione relativa al calcolo della matrice di rotazione, mostrata nell'immagine 2.4, nella quale vengono utilizzati in combinazione per calcolare la suddetta matrice.
Come spiegato in precedenza, questo strumento viene utilizzato per calcolare i valori di accelerazione in un sistema di riferimento fisso.
Questo calcolo può essere però effettuato anche senza l'ausilio della matrice di rotazione, come evidenziato sempre nella stessa immagine 2.4. In questo caso il calcolo avviene prendendo in considerazione i valori, non filtrati, dell'accelerometro e del sensore di orientamento.
Un'ulteriore alternativa permette che i dati dell'accelerometro, in alcuni casi filtrati e in altri casi non filtrati, possano essere passati direttamente a uno dei successivi algoritmi di riconoscimento dei passi.
Questa stessa alternativa è valida anche per i dati filtrati del giroscopio.

2.3.3 Riconoscimento dei Passi in Modalità Real-Time

In questa sezione, illustrata nella figura 2.5, si possono osservare i principali algoritmi utilizzati per riconoscere se i dati registrati dai sensori corrispondono al verificarsi di un passo.
In questo caso i dati vengono elaborati in real-time, ovvero vengono analizzati subito dopo la loro registrazione, identificando direttamente se l'istante del dato registrato corrisponde al compimento di un passo.
In particolare i possibili algoritmi utilizzabili in questa modalità sono due, l'*algoritmo dei picchi* e l'*algoritmo di intersezione con l'asse delle ascisse*.
Entrambi fanno riferimento alla curva che è possibile identificare tracciando in un grafico

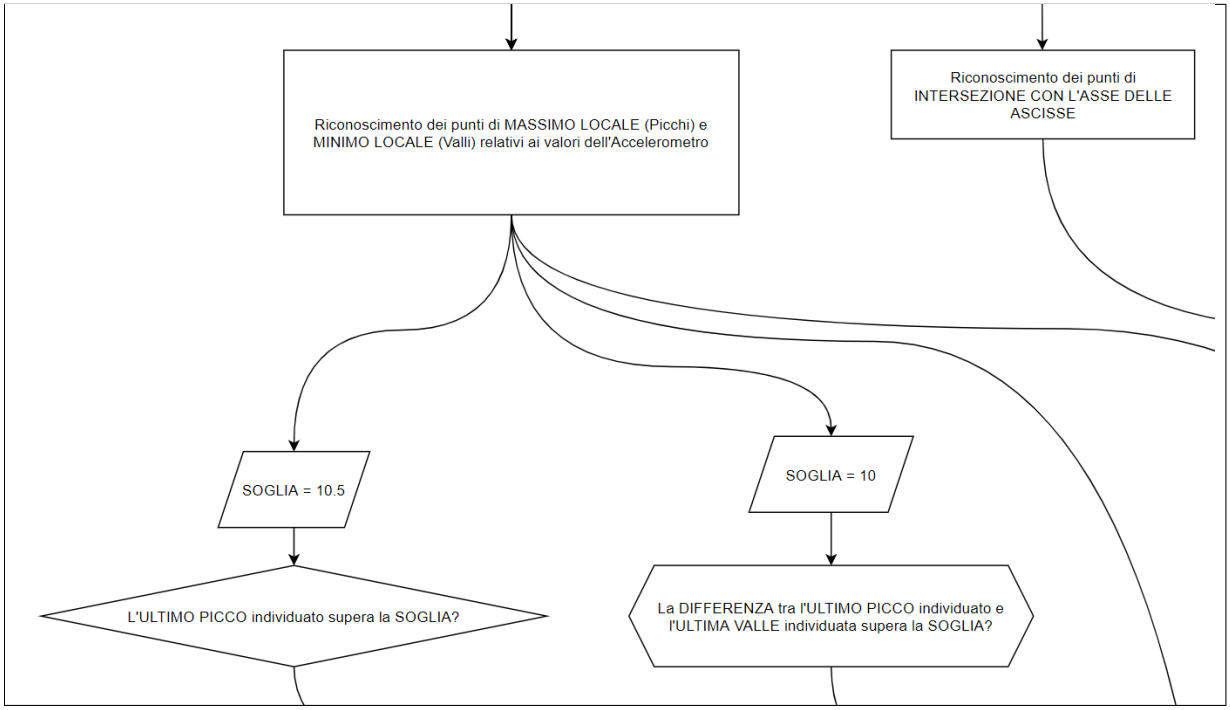

Figura 2.5: Riconoscimento Passo - Real-Time

il susseguirsi dei dati registrati dai sensori, in particolare quelli dell'accelerometro.

Il primo dei due algoritmi si occupa di identificare in real-time se il valore registrato corrisponde a un massimo locale o a un minimo locale relativamente alla curva appena descritta. Questo calcolo può avvenire sia tramite l'utilizzo della matrice di rotazione, sia tramite il solo utilizzo dei dati dell'accelerometro.
Il secondo algoritmo si occupa invece di identificare i punti in cui la curva precedentemente descritta incrocia l'asse delle ascisse. In questo caso l'algoritmo funziona con il solo utilizzo dei dati registrati dall'accelerometro.
Nel primo caso gli istanti riconosciuti come picchi vengono sottoposti a un ulteriore controllo in grado di identificare, secondo specifici parametri indicati sempre nella figura 2.5, se corrispondono effettivamente al compimento di un passo da parte dell'utente.
Il secondo algoritmo viene invece utilizzato come correttivo del primo, in quanto viene applicato successivamente ad esso come ulteriore controllo prima dell'identificazione di

un picco come effettivo passo.
In alcuni casi questo ulteriore controllo non viene applicato, ma viene utilizzato solamente il primo algoritmo.
Come verrà illustrato successivamente, nell'implementazione utilizzata per analizzare tutti i diversi pedometri si è data la possibilità di scegliere per ognuno di essi se utilizzare solamente il primo algoritmo o se utilizzarlo in combinazione con il secondo.

2.3.4 Riconoscimento dei Passi in Modalità Non Real-Time

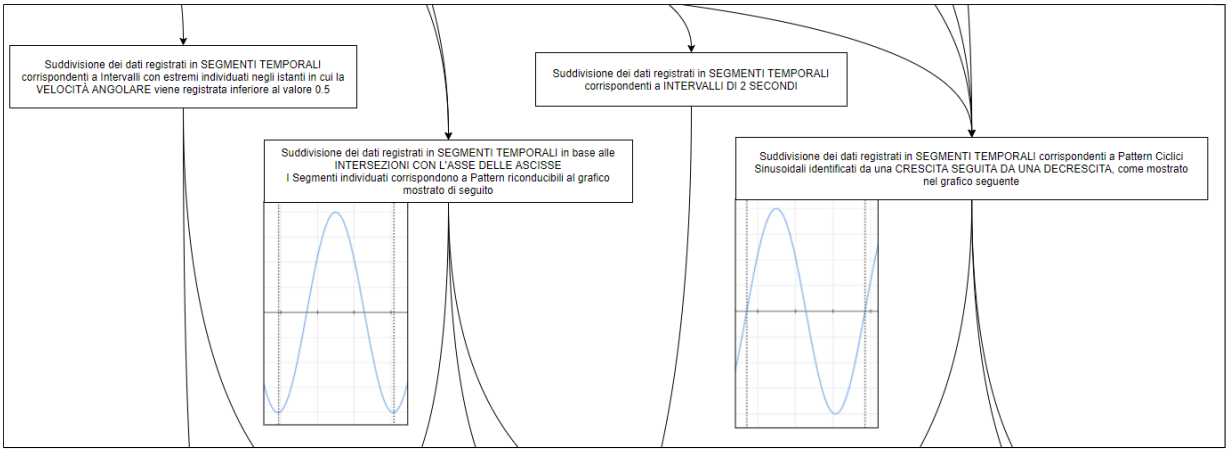

Figura 2.6: Suddivisione in Segmenti (Non Real-Time)

In questa sezione vengono analizzate le tecniche utilizzate per riconoscere i passi utilizzando la madalità non real-time.
Come descritto in precedenza, in questa modalità si ha la necessità di raccogliere un determinato quantitativo di dati, raggruppati in segmenti, prima di poterli analizzare e determinare se hanno le caratteristiche per essere identificati come passi.
Il riconoscimento di questi segmenti può avvenire in diversi modi, illustrati nella figura 2.6, riassumibili principalmente in due categorie: l'identificazione di pattern specifici all'interno della curva relativa ai dati registrati dal giroscopio o dall'accelerometro, e il

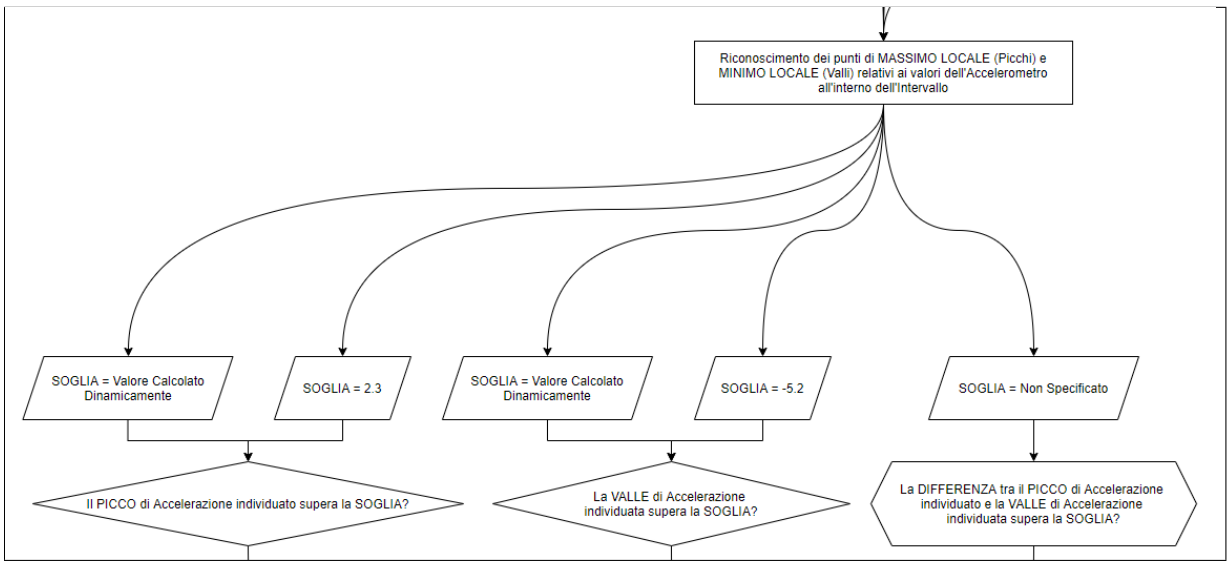

Figura 2.7: Riconoscimento Passo (Non Real-Time) [1/3]

riconoscimento di valori chiave.
In particolare, la prima categoria comprende due specifici pattern. Il primo è riconducibile al grafico più a sinistra della figura 2.6, e viene identificato in base alle intersezioni con l'asse delle ascisse, individuate con l'apposito algoritmo.
Il secondo invece è riconducibile all'altra figura presente sempre nella stessa immagine, e viene identificato in base alla presenza di una crescita seguita da una decrescita.
La seconda categoria comprende invece due algoritmi di suddivisione molto differenti. Il primo infatti identifica i segmenti in base ai valori registrati dal giroscopio, mentre il secondo identifica un nuovo segmento per ogni intervallo di registrazione di durata pari a due secondi.
Una volta identificati i segmenti, essi vengono passati a un algoritmo di riconoscimento dei passi in grado di identificare se il suddetto segmento corrisponde a un passo oppure no.
Questi algoritmi, anche definiti algoritmi di riconoscimento dei passi non real-time, sono molto simili a quelli utilizzati nella variante real-time, con la differenza che vengono applicati a un insieme di dati bene definito invece che ai dati registrati singolarmente.
Nella maggior parte dei casi essi lavorano identificando specifici valori chiave all'interno

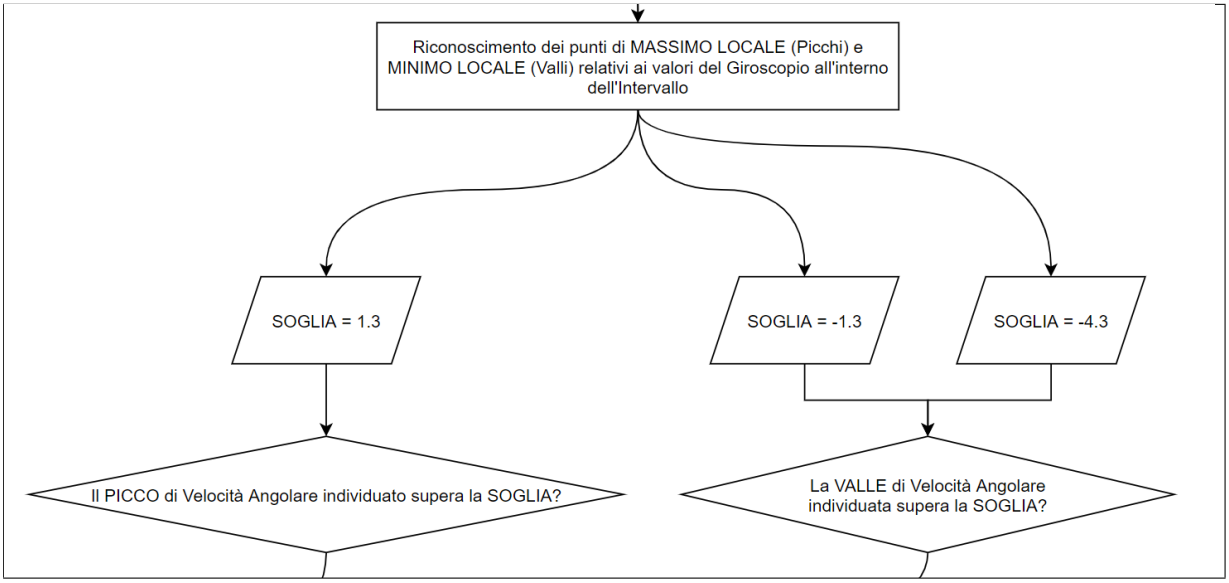

Figura 2.8: Riconoscimento Passo (Non Real-Time) [2/3]

del segmento, da sottoporre successivamente a un algoritmo di decisione in grado di determinare se il segmento corrisponde a un passo oppure no.
In particolare, le figure 2.7 e 2.8 descrivono questo tipo di algoritmo.
Nella prima i valori chiave estratti corrispondono ai massimi e minimi locali relativi all'accelerometro, estratti utilizzando un *algoritmo dei picchi*, e i passi sono identificati in base al superamento di determinati valori soglia da parte di questi valori.
Anche la seconda immagine si basa sull'estrazione dei valori di massimo e minimo locale tramite l'utilizzo di un *algoritmo dei picchi*, ma in questo caso relativo al giroscopio. Anche in questo caso il passo viene identificato in base al superamento da parte di questi valori di determinati valori soglia.
I valori soglia con cui vengono confrontati i valori chiave variano nei vari studi analizzati, e sono riassunti all'interno delle relative immagini.
Altri algoritmi di questo tipo lavorano invece sottoponendo direttamente i dati dei segmenti all'algoritmo di decisione, senza eseguire l'estrazione dei valori chiave.
In particolare questa tipologia di algoritmo identifica un segmento come passo in base agli istanti di inizio e di fine segmento, come illustrato nella figura 2.9.

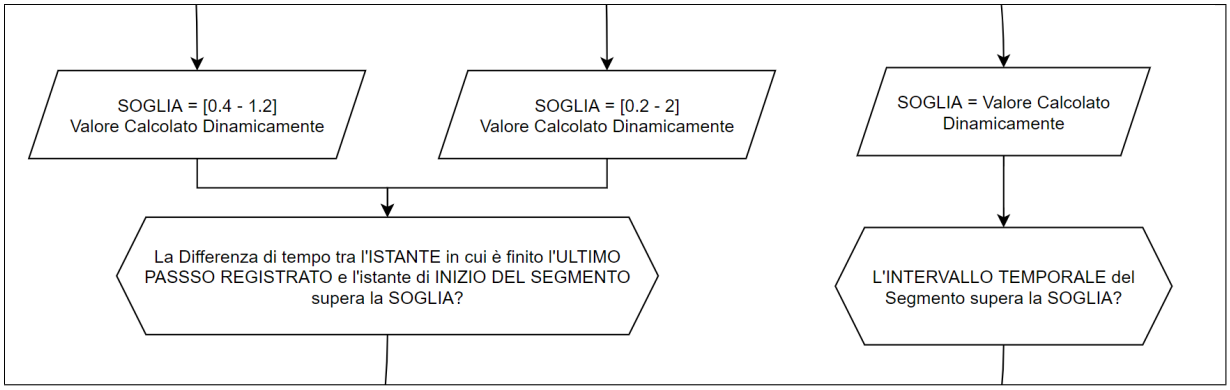

Figura 2.9: Riconoscimento Passo (Non Real-Time) [3/3]

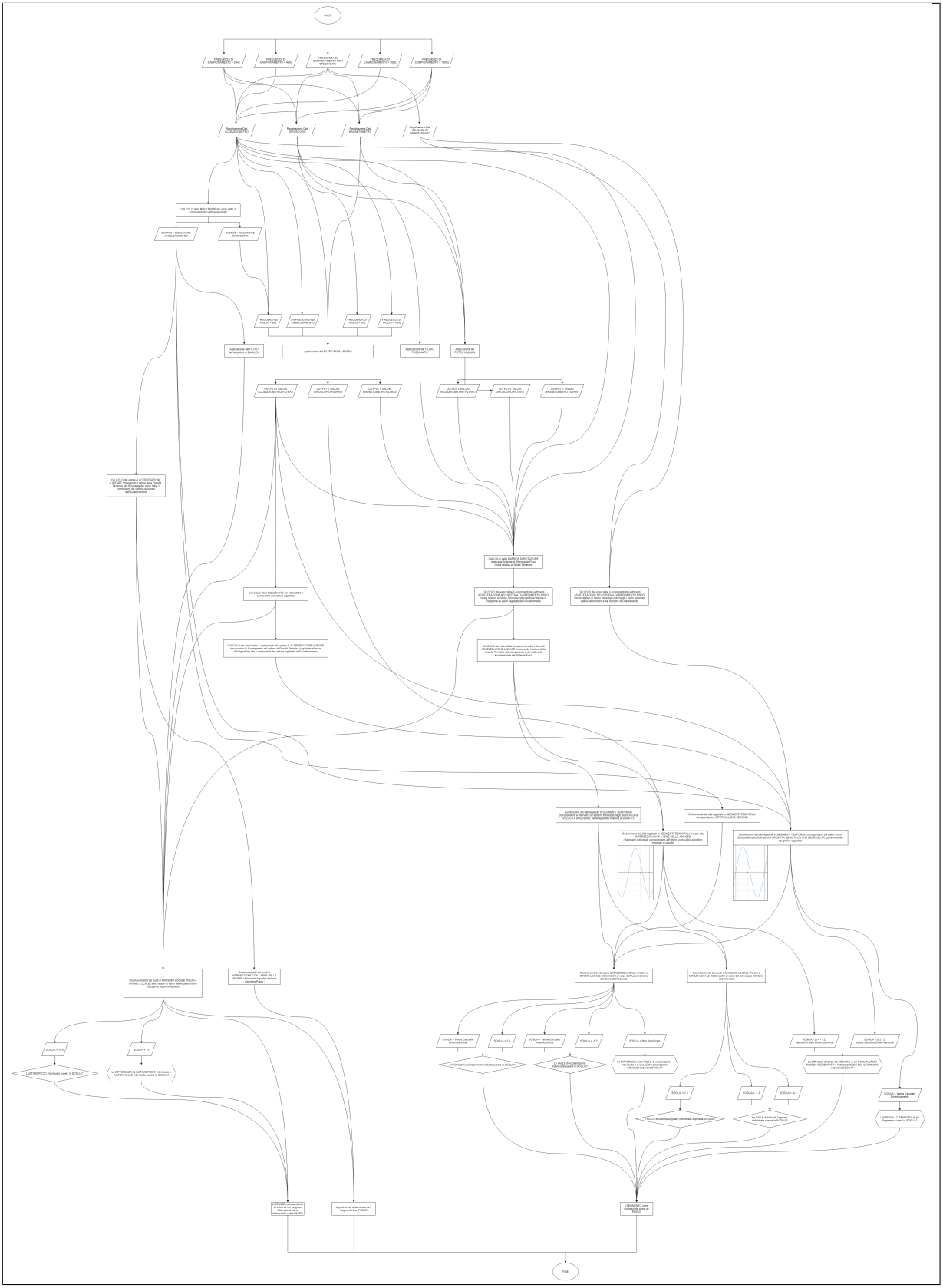

Figura 2.10: Tassonomia Completa

Capitolo 3

Implementazione dell'Applicazione

3.1 Scopo dell'Applicazione

La creazione della tassonomia descritta nel capitolo precedente ha reso evidente la ripetitività degli algoritmi proposti negli studi analizzati, facendone emergere gli step principali e le variabili discriminanti in grado di differenziare i diversi flussi di esecuzione.
Si è quindi deciso di utilizzare queste informazioni per implementare un'applicazione Android, sviluppata utilizzando il linguaggio *Java* all'interno dell'ambiente di sviluppo *"Android Studio"*, in grado di replicare tutte le possibili strade percorribili all'interno della tassonomia, in modo da poterle direttamente confrontare ed ottenere di conseguenza risultati relativi alla loro reale stabilità ed efficacia.
L'applicazione è stata infatti fatta in modo da poter utilizzare tutte le possibili combinazioni riscontrabili all'interno della tassonomia. Partendo da quest'ultima si è quindi sviluppato un diagramma di flusso molto compatto, comprendente tutte le possibili strade, dal quale si è partiti per lo sviluppo dell'applicazione
La stesura del diagramma di flusso ha reso evidente come la prima sostanziale differenza tra le diverse implementazioni sia la scelta della modalità di riconoscimento del passo.
In questo studio si è deciso di implementare esclusivamente le diramazioni dipendenti dalla scelta della modalità *real-time*. Saranno quindi necessari ulteriori studi per ottenere risultati relativi alle diramazioni dipendenti dalla scelta della modalità *non real-time*.

3.2 Principali Funzionalità

L'applicazione è stata implementata in modo da permetterne l'utilizzo in due modalità distinte: modalità *"Live Testing"* e modalità *"Offline Tesing"*.
Il primo caso permette di scegliere una determinata configurazione da utilizzare come pedometro e di vedere in tempo reale il numero di passi registrati dall'applicazione durante una camminata.
La seconda modalità permette invece di registrare i dati rilevati dai sensori durante una camminata e, successivamente, utilizzarli per determinare il numero di passi contati da diverse configurazioni di pedometri, confrontandole direttamente.
Viene di seguito fornita una descrizione più dettagliata delle principali funzionalità sviluppate all'interno dell'applicazione.
Un'analisi più approfondita della struttura architetturale dell'applicazione e delle componenti Android utilizzate per la sua implementazione varrà fornita nella sezione successiva.

3.2.1 Visualizzazione Real-Time

Questa funzionalità corrisponde alla prima delle due modalità citate precedentemente. Come prima cosa questa modalità permette di scegliere una precisa configurazione di pedometro da utilizzare. La configurazione deriva dalla scelta di:

1. Frequenza di Campionamento
   (a) 20Hz
   (b) 40Hz
   (c) 50Hz
   (d) 100Hz
   (e) Massima Disponibile

2. Modalità di registrazione dei Dati
   (a) Real-Time
   (b) Non Real-Time
3. Algoritmo di Riconoscimento dei Passi
   (a) Algoritmo dei Picchi
   (b) Algoritmo dei Picchi + Algoritmo di Intersezione con l'Asse delle Ascisse
   (c) Massima Disponibile
4. Filtro
   (a) Nessun Filtro
   (b) Filtro Passa-basso
   (c) Matrice di Rotazione
   (d) Algoritmo di Bagilevi

   Nel caso in cui venga scelto il filtro passa-basso viene resa disponibile la scelta della frequenza di taglio:
5. Frequenza di Taglio
   (a) 2Hz
   (b) 3Hz
   (c) 100Hz
   (d) 2% della Frequenza di Campionamento

La possibilità di scelta della modalità di registrazione dei dati è stata inserita per predisporre l'applicazione all'implementazione della modalità *non real-time*.
Come detto in precedenza, infatti, l'unica modalità di registrazione dei dati disponibile nell'applicazione è quella *real-time*.
In base alle scelte fatte viene quindi generata una specifica configurazione, utilizzabile come pedometro.

Una volta impostata la configurazione, l'applicazione si sposta su una diversa schermata dalla quale è possibile vedere in tempo reale i passi registrati dall'applicazione con quella specifica configurazione.
In questa schermata l'applicazione rende inoltre visibile un grafico, aggiornato anch'esso in tempo reale, rappresentante la curva generata dai dati registrati dall'accelerometro.

3.2.2 Registrazione di un Test

Questa funzionalità, come tutte le successive, fa parte della seconda modalità di utilizzo dell'applicazione.
In particolare questa funzionalità permette di raccogliere tutti i dati registrati dai principali sensori durante una camminata eseguita dall'utente.
Una volta eseguita la camminata viene chiesto all'utente di inserire il numero di passi effettivamente effettuati e, facoltativamente, qualche nota aggiuntiva relativa al tipo di camminata effettuata.
Ognuna di questa registrazioni, anche definita *"Test"*, viene quindi salvata in locale nel dispositivo.

3.2.3 Import & Export dei Test

Come detto in precedenza, lo sviluppo dell'applicazione ha avuto come principale obiettivo il confronto delle diverse configurazioni di pedometri.
Per raggiungere questo obiettivo si è quindi pensato di raccogliere i dati di diversi tipi di camminate eseguite da diverse persone, come verrà ampiamente descritto nel successivo capitolo.
Al fine di limitare al massimo gli spostamenti e gli incontri fisici, specialmente durante il delicato periodo di pandemia di Covid-19 affrontato durante la stesura della presente Tesi, si è pensato di fare in modo che ogni utente fosse in grado di registrare i propri test

senza la necessità di incontrare altre persone.
É stato quindi necessario implementare una funzionalità che permettesse operazioni di import e di export dei diversi test, in modo da poter permettere lo scambio dei test tra diversi dispositivi Android.
In particolare l'applicazione permette queste due distinte funzionalità:

1. *Export* di un Test
   Accedendo a una determinata schermata è possibile ottenere una lista di tutti i test salvati all'interno dell'applicazione.
   Selezionando uno o più di essi è possibile premere un bottone che permette di scegliere quale applicazione di messaggistica, installata sul dispositivo, utilizzare per inviare i test selezionati a un altro dispositivo. Da questa schermata è inoltre possibile eliminare ogni singolo test.

2. *Import* di un Test
   Premendo uno specifico bottone è possibile accedere al *File Manager* del dispositivo attraverso il quale è possibile navigare fino a raggiungere una qualsiasi cartella contenente un test registrato.
   Cliccando su di esso l'applicazione importa il file automaticamente al suo interno, aggiungendolo agli altri test registrati precedentemente.
   Questa funzionalità viene prevalentemente utilizzata per importare i test ricevuti da altri dispositivi.

3.2.4 Confronto Offline di Molteplici Configurazioni

Questa funzionalità è probabilmente la più degna di nota all'interno dell'applicazione. Essa permette infatti di effettuare un confronto diretto e visivo del comportamento delle diverse configurazioni applicate allo stesso insieme di dati.
Accedendo a una determinata schermata dell'applicazione è possibile selezionare un numero, compreso tra 1 e 6, di diverse configurazioni che si vuole testare.
In particolare, le diverse configurazioni utilizzabili sono in totale 14, che derivano da

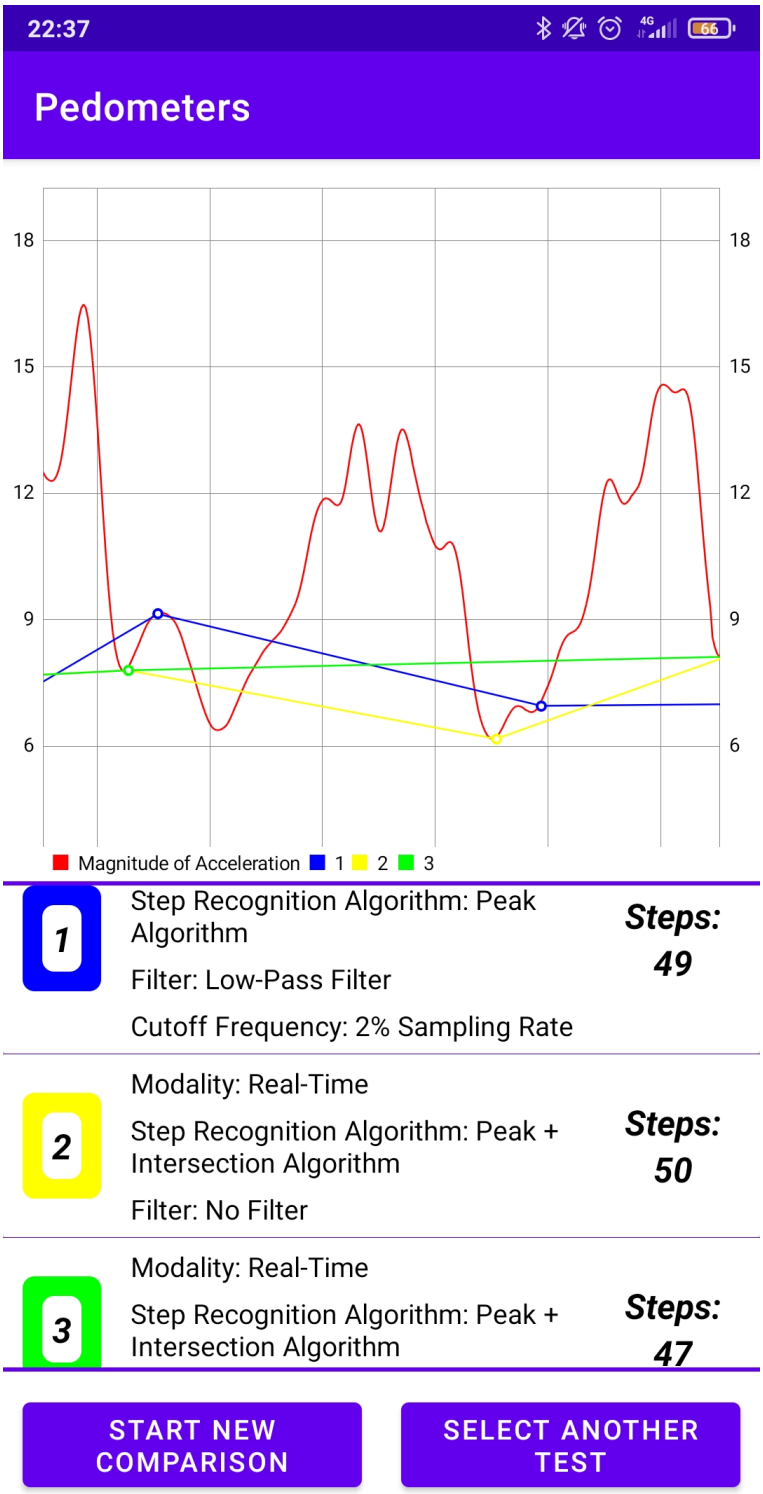

Figura 3.1: Screenshot del Confronto Offline

tutte le possibili combinazioni dei parametri esposti precedentemente nella funzionalità di *Visualizzazione Real-Time*, escluso quello relativo alla scelta della *Frequenza di Campionamento*.
L'esclusione di questo parametro deriva dalla sua necessità di essere definito prima della registrazione dei dati, in quanto definisce la frequenza con cui essi vengono registrati.
Per questo motivo i test utilizzati all'interno dei questa funzionalità vengono registrati utilizzando la massima Frequenza di Campionamento disponibile sul dispositivo.
Una volta selezionate le diverse configurazioni l'applicazione permette di scegliere il test da utilizzare per confrontarle. Ognuno dei test riporta il numero di passi effettivamente compiuti durante la sua registrazione e le eventuali note aggiuntive inserite.
La scelta del test permette quindi di accedere a una nuova schermata, particolarmente complessa, nella quale è possibile avere un riscontro visivo dell'applicazione delle diverse configurazioni a quel specifico test.
In particolare l'applicazione esegue l'algoritmo di ognuna delle configurazioni scelte ai dati salvati nel test, elaborando di conseguenza il numero passi calcolati da ognuna di esse con quello specifico test.
Come si può vedere nell'immagine 3.1, la schermata viene quindi divisa in due sezioni.
Nella sezione in basso viene visualizzato un elenco delle configurazioni scelte, ognuna delle quali ben riconoscibile grazie a una descrizione riassuntiva delle principali caratteristiche che le caratterizzano, e affiancata dal numero di passi contati dalla stessa.
Nell'altra sezione viene invece mostrato un grafico contenente la curva generata dai dati del test registrati dall'accelerometro, e, per ognuna delle configurazioni, un insieme di punti collegati da una linea, ognuna delle quali rappresentata con un colore differente.
Ognuno dei punti evidenziati delle varie linee rappresenta un istante in cui quella determinata configurazione ha rilevato un passo durante l'analisi del test.
Cliccando su una delle configurazioni elencate è possibile evidenziare la rispettiva linea all'interno del grafico, in modo da renderla ben visibile rispetto alle altre.
In questo modo è possibile avere un riscontro visivo ben chiaro degli istanti in cui ciascuna configurazione rileva i passi, e si ha la possibilità di confrontare direttamente le diverse configurazioni, visualizzando in che modo ciascuna di esse reagisce agli stessi valori in input.

Partendo da questo grafico si può quindi determinare il comportamento delle varie configurazioni, rilevando i punti in cui vengono rilevati *falsi passi positivi* o non vengono rilevati passi effettivamente effettuati, ed avere quindi un primo riscontro sui limiti di ciascuna di esse.
Un esempio esaustivo di questa funzionalità è illustrato nella figura 3.1, nella quale è possibile vedere il confronto di tre diverse configurazioni applicate a un test durante il quale erano stati effettuati 50 passi.
Osservando la sezione in basso dell'immagine si può notare come le configurazioni abbiano generato rispettivamente tre diversi conteggi dei passi: 49, 50 e 47.
Osservando la parte superiore, contenente il grafico, si può quindi notare come le tre configurazioni abbiano individuato correttamente il primo passo, seppur in istanti leggermente differenti, mentre il secondo passo è stato individuato solamente dalle prime due configurazioni, ed è stato erroneamente ignorato dalla terza.

3.3 Struttura Architetturale e Implementazione

In questa sezione viene fornito un dettagliato approfondimento relativo alle componenti di Android Studio utilizzate per sviluppare l'applicazione e alle principali scelte implementative poste alla base della struttura architetturale della stessa.

3.3.1 Classi di Utility

La parte fondamentale dello sviluppo dell'applicazione è sicuramente incentrata sull'implementazione dei diversi algoritmi di conteggio dei passi.
Utilizzando la tassonomia e il diagramma di flusso descritti nei capitoli precedenti sono state quindi implementate diverse classi *utility* relative ad ognuno dei principali step descritti nella sezione 2.3.
In questo modo si è potuto replicare il flusso di esecuzione delle diverse configurazioni rappresentate all'interno del diagramma combinando l'utilizzo delle varie classi.
In particolare le classi utility implementate sono le seguenti:

1. Classe *Sensore*
   Questa classe è stata implementata in modo tale da avere la possibilità di contenere all'interno di un singolo oggetto tutte le informazioni relative ai dati raccolti da un singolo sensore.
   Durante l'esecuzione dell'algoritmo ogni sensore fisico viene quindi associato ad un singolo oggetto.
   All'interno di ognuno di questi oggetti vengono quindi salvati tutti i dati registrati dal sensore e, successivamente, tutti i valori derivati da essi, quali ad esempio il valore di risultante calcolato su di essi, o il risultato dell'applicazione di un particolare filtro sempre sugli stessi.
   Le classi contenenti le funzioni di calcolo e di filtro sono indipendenti da questa

classe, e verranno analizzate di seguito.
In questo modo è possibile avere per ogni sensore un oggetto con il quale si ha la possibilità in ogni istante di eseguire operazioni di salvataggio o di recupero delle sue informazioni più rilevanti, tramite l'utilizzo di specifiche funzioni di *get* e di *set*.

2. Classe *Configurazione*
   Questa classe è stata invece implementata per avere un oggetto nel quale salvare le informazioni relative alla scelta di una specifica configurazione di pedometro.
   Questa classe viene quindi utilizzata quando si utilizzano le funzionalità di *Visualizzazione Real-Time* [3.2.1] e di *Confronto Offline di Molteplici Configurazioni* [3.2.4], durante le quali è necessario specificare la scelta dei parametri di una o più configurazioni.
   In questo modo si ha per ogni configurazione un oggetto all'interno del quale vengono salvati tutti i parametri scelti, che possono essere letti in qualsiasi momento durante l'esecuzione degli algoritmi.
   Si è scelto inoltre di inserire in questi oggetti il salvataggio di valori calcolati utilizzando più sensori, quali ad esempio la matrice di rotazione, e di valori relativi agli istanti in cui vengono identificati valori chiave, quali ad esempio i valori di massimo o di minimo locale.
   Analogamente alla classe precedente, si ottiene in questo modo un oggetto indipendente per ogni configurazione scelta con il quale è possibile in ogni momento eseguire operazioni di lettura e scrittura dei valori appena descritti, specifici di ogni diversa configurazione.

3. Classe *Filtri*
   Questa classe è completamente indipendente dalle altre, ed ha l'unico compito di fornire specifici metodi, ognuno relativo ad uno specifico filtro, da utilizzare per filtrare i dati registrati dai sensori.
   Una volta istanziato un oggetto di questa classe è quindi possibile utilizzarlo per richiamare uno dei suoi metodi in modo da poterlo utilizzare per filtrare determinati valori con uno specifico filtro.

4. Classe *Calcoli*

   Anche questa classe svolge un lavoro simile a quello svolto dalla classe precedente. In questo caso però la classe offre metodi da utilizzare per eseguire specifici calcoli, quali ad esempio il calcolo della matrice di rotazione o il calcolo della risultante.
   Come nel caso della classe *Filtri* anche in questo caso, una volta effettuato l'utilizzo di uno dei suoi metodi, i valori calcolati vengono salvati all'interno di un oggetto istanza di una delle prime due classi descritte, ovvero *Sensore* o *Configurazione*.

5. Classe *Riconoscimento Valori Chiave*

   Questa classe fornisce invece un insieme di funzioni corrispondenti ai diversi metodi di riconoscimento dei valori chiave illustrati nella tassonomia descritta nei capitoli precedenti, ognuna delle quali viene utilizzata in base alla configurazione scelta.
   In questo caso gli oggetti istanza di questa classe sono legati a un singolo oggetto istanza della classe *Configurazione*. L'utilizzo dei metodi presenti all'interno degli oggetti istanza di questa classe determina infatti il valore degli attributi presenti in un oggetto istanza della classe *Configurazione*, quali ad esempio gli istanti in cui vengono identificati i valori di massimo o di minimo locale.

6. Classe *Individuazione Passo*

   Questa classe ha un funzionamento ed un utilizzo molto simile a quella precedente, con la differenza che implementa le funzioni corrispondenti ai diversi metodi di individuazione dei passi illustrati nella tassonomia.
   Anche in questo caso l'utilizzo di un oggetto istanza di questa classe è legato a un singolo oggetto istanza della classe *Configurazione*.
   L'utilizzo di questa classe è l'ultimo step all'interno della ricerca degli istanti corrispondenti a un passo. Nel caso in cui l'utilizzo di uno dei metodi presenti in questa classe ritorni un valore pari a *true* significa che in quel determinato istante una specifica configurazione ha individuato un passo.

3.3.2 Activity

Le *"Activity"* sono una componente fondamentale dell'utilizzo di Android Studio, in quanto corrispondono alle finestre contenenti l'interfaccia utente delle applicazioni, e sono quindi il principale mezzo con cui gli utenti possono interagire con esse.
Nell'applicazione qui presentata sono state implementate diverse activity, ognuna delle quali relativa a una delle funzionalità dell'applicazione descritte in precedenza.
In particolare le activity implementate sono le seguenti:

1. Activity *Main Activity*
   Questa activity corrisponde all'activity principale dell'applicazione, ovvero quella che viene lanciata al suo avvio.
   La funzione di questa activity è di mettere a disposizione dell'utente la scelta della funzionalità da utilizzare, offrendo un elenco di bottoni attraverso i quali è possibile raggiungere ciascuna di esse.
   Quest'activity gestisce inoltre la funzionalità di *Import dei test* [3.2.3], in quanto qessa non richiede nessun tipo di interfaccia grafica, e può quindi essere gestita direttamente dalla *Main Activity*.
   Questa activity è sempre raggiungibile da tutte le activity descritte qui di seguito utilizzando il tasto *Indietro* del dispositivo.

2. Activity *Send Test*
   Questa activity gestisce invece la funzionalità di *Export dei test* [3.2.3].
   In questo caso nella schermata viene mostrato un elenco contenente tutti i test salvati localmente sul dispositivo.
   Tramite l'utilizzo delle *checkbox* è possibile selezionarne uno o più di uno e, tramite la pressione di uno specifico bottone, viene chiesto all'utente di scegliere quale applicazione di messaggistica utilizzare per inviare i test selezionati a un altro dispositivo.
   Da questa stessa schermata è inoltre possibile, tenendo premuto un determinato test, eliminarlo definitivamente.

3. Activity *Live Testing*
   Questa activity è utilizzata per implementare la funzionalità di *Visualizzazione*

```
if(configurazione.getReal_time()==0) { //real-time
    switch(configurazione.getFiltro()){
        case 0: //bagilevi
            if(accelerometer_event) {
                accelerometro.setRisultante_filtrato(filtri.filtroBagilevi(
                accelerometro.getVettore_3_componenti()));
                passo_inidividuato = individuazione_passo.
                individuazionePassoRealTime_Bagilevi(riconoscimento_valori_chiave.
                riconoscimentoPuntiMassimoMinimoLocaleRealTime_Bagilevi(accelerometro
                .getRisultante_filtrato(), istante)); }
            break;
        case 1: //passa-basso
            if(accelerometer_event) {
                accelerometro.setVettore_3_componenti_filtrato(filtri.
                filtroPassaBasso(accelerometro.getVettore_3_componenti(), alpha));
                accelerometro.setRisultante_filtrato(calcoli.risultante(accelerometro
                .getVettore_3_componenti_filtrato()));
                if(riconoscimento_valori_chiave.
                riconoscimentoPuntiMassimoMinimoLocaleRealTime(accelerometro.
                getRisultante_filtrato(), istante))
                    passo_inidividuato = individuazione_passo.
                    individuazionePassoRealTime_SogliaPicco_DifferenzaPiccoValle();
                else passo_inidividuato = false; }
            break;
        case 2: //nessun filtro
            if(accelerometer_event) {
                accelerometro.setRisultante(calcoli.risultante(accelerometro.
                getVettore_3_componenti()));
                if(riconoscimento_valori_chiave.
                riconoscimentoPuntiMassimoMinimoLocaleRealTime(accelerometro.
                getRisultante(), istante))
                    passo_inidividuato = individuazione_passo.
                    individuazionePassoRealTime_SogliaPicco_DifferenzaPiccoValle();
                else passo_inidividuato=false; }
            break;
        case 3: //matrice_rotazione
            if(accelerometro.getIstanziato() && magnetometro.getIstanziato()) {
                calcoli.matriceRotazione(accelerometro.getVettore_3_componenti(),
                magnetometro.getVettore_3_componenti());
                accelerometro.setVettore_3_componenti_sistema_fisso(calcoli.
                accelerazioneSistemaFisso(accelerometro.getVettore_3_componenti()));
                if(riconoscimento_valori_chiave.
                riconoscimentoPuntiMassimoMinimoLocaleRealTime(accelerometro.
                getVettore_3_componenti_sistema_fisso()[2], istante))
                    passo_inidividuato = individuazione_passo.
                    individuazionePassoRealTime_SogliaPicco_DifferenzaPiccoValle();
                else passo_inidividuato=false; }
            break; }
    //algoritmo picchi + correzione algoritmo intersezioni
    if(configurazione.getAlgoritmo_di_riconoscimento()==1) {
        if(accelerometer_event) {
            accelerometro_ascisse.setRisultante(calcoli.risultante(
            accelerometro_ascisse.getVettore_3_componenti()));
            accelerometro_ascisse.setRisultante_lineare(calcoli.accelerazioneLineare(
            accelerometro_ascisse.getRisultante()));
            riconoscimento_valori_chiave.
            riconoscimentoPuntiIntersezioneAsseAscisseRealTime(accelerometro_ascisse.
            getRisultante_lineare(), istante);
            passo_inidividuato = passo_inidividuato && individuazione_passo.
            individuazionePassoRealTime_IstanteIntersezione(); }
        else passo_inidividuato=false; }
    if(passo_inidividuato) stepDetected(); }
```

Figura 3.2: Snippet relativo al Testing di una Configurazione

Real-Time [3.2.1] e si occupa unicamente di alternare la visualizzazione di due differenti Fragment.
I *"Fragment"* sono un'altra componente di Android Studio molto simile alle activity, e per essere utilizzati devono essere inseriti all'interno di una di esse. Per questo motivo possono essere identificati come sub-activity.
In particolare i fragment vengono utilizzati per inserire all'interno di un'activity una porzione riutilizzabile di interfaccia grafica, con la possibilità di essere sostituiti in ogni momento da altri fragment, permettendo quindi all'activity di passare in modo semplice da un'interfaccia grafica ad un'altra.
In questa activity i fragment sono stati utilizzati per alternare la visualizzazione di due specifiche interfacce grafiche.
La prima di esse, definita *Enter Settings Fragment*, viene utilizzata per permettere all'utente di scegliere i valori da assegnare ai vari parametri utilizzati per identificare una specifica configurazione, parametri specificati nella descrizione della funzionalità 3.2.1.
La seconda interfaccia grafica, definita *Pedometer Running Fragment*, è invece utilizzata per permettere all'utente di visualizzare in *real-time* l'individuazione dei passi da parte del pedometro.
In particolare viene visualizzato un grafico rappresentante i valori raccolti in tempo reale dall'accelerometro, accompagnato da un valore numerico, ben visibile, indicante il numero di passi identificati dall'algoritmo fino a quel momento.
L'immagine 3.2 mostra il codice *Java* utilizzato dall'applicazione per identificare i passi.
In particolare questo frammento di codice viene eseguito ogni volta che uno dei sensori registra un nuovo valore e, in base ai parametri della configurazione scelta, l'algoritmo segue un flusso di esecuzione differente.
Ognuna delle funzioni presenti nei diversi casi viene richiamata da una delle classi utility elencate precedentemente.
L'utilizzo dei fragment permette di poter passare da una delle due interfacce grafiche all'altra in ogni momento, permettendo all'utente di stoppare un pedometro ed avviarne un altro in modo semplice e veloce.

4. Activity *New Test*

   Questa activity viene utilizzata per implementare la funzionalità di *Registrazione di un Test* [3.2.2].

   Tramite l'utilizzo di un apposito bottone è possibile avviare la registrazione dei dati rilevati dai vari sensori e, successivamente, tramite la pressione di un secondo bottone, è possibile stoppare la registrazione e salvare il test. Da questa schermata viene anche data la possibiltà di indicare il numero di passi effettivamente effettuati ed eventuali note aggiuntive relative al tipo di camminata effettuata.

   I test registrati vengono poi successivamente salvati in locale nel dispositivo, come verrà dettagliatamente descritto nella sezione successiva.

5. Activity *Select Configurations To Compare*

   Questa activity, insieme alle due successive, viene invece utilizzata per implementare la funzionalità di *Confronto Offline di Molteplici Configurazioni* [3.2.4], occupandosi di fornire all'utente un schermata in cui selezionare le diverse configurazioni da confrontare.

   In particolare, viene utilizzato più volte, per un massimo di sei volte, il fragment *Enter Settings Fragment*, descritto in precedenza.

   Tramite l'utilizzo di uno specifico bottone l'utente salva la configurazione visualizzata e passa alla visualizzazione di un nuovo fragment in cui inserire un'ulteriore configurazione.

   Una volta scelte tutte le configurazioni da confrontare viene lanciata una nuova activity, descritta qui di seguito, dalla quale è possibile selezionare il test da analizzare.

6. Activity *Select Test*

   Rimanendo nell'ambito della funzionalità di *Confronto Offline di Molteplici Configurazioni* [3.2.4], questa activity si occupa quindi di permettere all'utente di selezionare il test da analizzare.

   Il funzionamento di questa activity è molto semplice, in quanto viene mostrato un elenco di tutti i test presenti in locale sul dispositivo, ognuno dei quali è selezionabile tramite un apposito bottone posizionato a fianco di ciascuno di essi.

Ovviamente la selezione può essere fatta solo su un singolo test.
Una volta scelto il test da analizzare viene infine lanciata un'ultima activity, descritta qui di seguito.

7. Activity *Configurations Comparison*
   Questa activity corrisponde quindi all'ultima parte di implementazione della funzionalità di *Confronto Offline di Molteplici Configurazioni* [3.2.4], e corrisponde anche alla sua parte principale.
   Come già visto nella descrizione delle funzionalità, da questa activity è possibile effettuare un vero e proprio confronto del comportamento delle diverse configurazioni su uno stesso test.
   Poiché la descrizione della schermata di confronto di diverse configurazioni è stata già ampiamente approfondita, si rimanda il lettore alla sezione 3.2.4.

3.3.3 DataBase e Archiviazione dei Test

Come già ampiamente descritto fino a questo momento, i test registrati dall'applicazione vengono salvati localmente nel dispositivo.
In particolare, l'applicazione utilizza due sistemi paralleli di salvataggio dei test.
Il primo di questi è un semplice DataBase, implementato utilizzando una specifica libreria, denominata *"Room"*, la quale, sfruttando la potenza di *SQLite*, consente un accesso semplice e veloce allo stesso DB.
All'interno di questo DataBase vengono salvati, per ogni test, l'insieme dei valori registrati dai vari sensori, formattati in una stringa in formato *json*, il numero di passi effettivamente effettuati, e le eventuali note aggiunte.
L'aggiunta di un test all'interno del DataBase avviene sia in seguito alla registrazione di un nuovo test, sia in seguito all'importazione di un test registrato da un differente dispositivo.
Il secondo sistema di salvataggio dei test consiste invece nella scrittura di un file *.txt* in una directory dedicata all'archiviazione dei file dell'applicazione, non accessibile esterna-

mente da altre applicazioni.
Le informazioni salvate in questi file sono del tutto identiche alle informazioni salvate all'interno del DataBase. In questo caso però tutto il file è formattato seguendo il formato *json*.
Anche in questo caso la creazione del file avviene sia in seguito alla registrazione di un nuovo test, sia in seguito all'importazione di un test registrato da un differente dispositivo.
La scelta dell'utilizzo di entrambi questi metodi deriva da due distinte necessità.
L'utilizzo del DataBase garantisce un accesso semplice e veloce ai dati salvati all'interno dei test, mentre l'accesso ai file *.txt* salvati nella directory è molto più lento.
La connessione con il DataBase viene quindi stabilita durante l'utilizzo delle principali funzionalità di esecuzione dei pedometri precedentemente descritte, in modo da garantire un'esecuzione il più fluida possibile.
I file *.txt* sono invece stati utilizzati per permettere lo scambio di test tra diversi dispositivi.
Particolare attenzione è stata prestata nel garantire che i dati salvati all'interno del DataBase fossero costantemente coerenti con quelli salvati nella directory del dispositivo, specialmente dopo ogni singola operazione di registrazione di un nuovo test o di importazione di un test registrato da un diverso dispositivo.

3.3.4 Altre Componenti

Altre componenti di Android Studio rilevanti che sono state utilizzate nello sviluppo dell'applicazione sono la classe *"AsyncTask"* e la classe *"Adapter"*.
La prima di queste è una classe astratta utilizzata per permettere l'esecuzione di *task asincroni*, ovvero operazioni eseguite in un *thread* che lavora in background rispetto al thread principale, contenente l'interfaccia grafica, e che si interfaccia con esso restituendogli i risultati calcolati.
In quest'applicazione questa classe è stata sfruttata per implementare nuove classi da utilizzare per eseguire tutte le operazioni di scrittura e di lettura dei campi salvati al-

l'interno del DataBase, in modo tale da non bloccare l'interfaccia grafica durante la loro esecuzione.
La classe *Adapter* viene invece utilizzata per creare un collegamento tra i dati salvati all'interno di un'applicazione e i dati mostrati in una specifica schermata.
Nel caso di questa applicazione questa classe è stata utilizzata tutte le volte che si ha avuto la necessità di mostrare l'elenco dei test salvati o l'elenco delle configurazioni da confrontare.

Capitolo 4

Test e Risultati

4.1 Raccolta dei Dati ed Esecuzione dei Test

Lo sviluppo dell'applicazione descritta nel capitolo precedente ha permesso di avere uno strumento completo ed affidabile con il quale poter confrontare il comportamento di diverse implementazioni di pedometri su uno stesso campione di dati.
Al fine di ottenere risultati oggettivi è quindi stata stilata una lista di specifici test da sottoporre a un determinato campione di persone.
In particolare, 11 diversi tester hanno dovuto compiere 6 differenti camminate, di 50 passi ciascuna, registrando, tramite l'utilizzo dell'applicazione Android, un differente test per ognuna di esse.
Per ottenere dati il più omogenei possibile sono state date specifiche direttive ai tester. É stato quindi infatti loro di tenere il dispositivo in mano in modalità *texting*, e di disattivare le notifiche del dispositivo al fine di evitare il verificarsi di vibrazioni in grado di compromettere i dati registrati dai sensori.
I dati registrati dai tester sono successivamente stati analizzati, sempre tramite l'applicazione, utilizzando 14 diverse configurazioni di pedometri, generando risultati significativi.
All'interno di questa sezione verranno analizzati i test registrati per confrontare i diversi algoritmi, con un approfondimento relativo alle caratteristiche dei tester e alle diverse configurazioni testate.
Nelle sezioni successive verranno invece analizzati e discussi i risultati ottenuti.

4.1.1 Test Effettuati

Per ottenere un'indicazione corretta dell'effettiva efficacia e precisione delle diverse configurazioni di pedometri sono quindi state effettuate 6 diverse camminate, ognuna delle quali caratterizzata da una diversa caratteristica specifica. In particolare, le camminate con cui sono stati effettuati i test sono le seguenti:

1. Camminata Normale.
2. Camminata in Salita.
3. Camminata in Discesa.
4. Camminata con Passi Irregolari.
5. Camminata con Passi Stretti.
6. Corsa.

Ognuno dei tester ha quindi dovuto effettuare un numero pari a 50 passi per ognuna di queste camminate, raccogliendo per ognuna di esse i valori registrati dai diversi sensori. La scelta delle caratteristiche delle sei differenti camminate è stata fatta in modo tale da avere una chiara indicazione dell'affidabilità di ciascuna configurazione anche in situazioni *limite*, che si discostano da una camminata standard.

4.1.2 Campione di Tester

Per registrare i diversi test è stato utilizzato un campione di 11 diverse persone, di età compresa tra i 22 e i 26 anni, e con età media pari a 24 anni. Le informazioni dettagliate relative ai diversi tester sono riassunte all'interno della tabella 4.1.
Come si può vedere all'interno di essa, per effettuare i test sono stati utilizzati 9 diversi

Tester	Dispositivo	Età
I	Xiaomi Redmi Note 5	25
II	Xiaomi Redmi Note 5	24
III	Xiaomi Redmi S2	23
IV	Samsung Galaxy S8	25
V	Samsung Galaxy A40	24
VI	Samsung Galaxy A40	24
VII	Honor View 20	26
VIII	Honor 10	22
IX	Huawei P20 Lite	23
X	Google Pixel 4a	23
XI	Asus Zenfone 5Z	25

Tabella 4.1: Elenco Tester

dispositivi Android, commercializzati da 6 diverse marche produttrici di smartphone. Osservando la tabella si può anche notare che due coppie di tester, in particolare i tester I-II e i tester V-VI, hanno fatto uso dello stesso dispositivo per registrare i test.
L'utilizzo di più differenti dispositivi ha permesso di ottenere indicazioni ancora più affidabili sulla reale efficacia dei diversi algoritmi, in quanto in ognuno di essi vengono utilizzati sensori differenti.

4.1.3 Configurazioni Testate

I test registrati sono quindi stati analizzati tramite l'utilizzo degli algoritmi implementati all'interno dell'applicazione Android.
Nello specifico, sono state utilizzate 14 diverse configurazioni, corrispondenti a tutte le possibili combinazioni definibili all'interno dell'applicazione in base ai parametri espressi nella sezione 3.2.1 escluso quello relativo alla scelta della *Frequenza di Campionamento*. I dettagli delle varie combinazioni sono riassunti all'interno della tabella 4.2.

Test	Modalità	Algoritmo di Riconoscimento Passi	Filtro	Frequenza di Taglio
I	Real-Time	Picchi	Nessun Filtro	—
II	Real-Time	Picchi + Intersezione Asse Ascisse	Nessun Filtro	—
III	Real-Time	Picchi	Matrice di Rotazione	—
IV	Real-Time	Picchi + Intersezione Asse Ascisse	Matrice di Rotazione	—
V	Real-Time	Picchi	Algoritmo di Bagilevi	—
VI	Real-Time	Picchi + Intersezione Asse Ascisse	Algoritmo di Bagilevi	—
VII	Real-Time	Picchi	Passa-basso	2Hz
VIII	Real-Time	Picchi + Intersezione Asse Ascisse	Passa-basso	2Hz
IX	Real-Time	Picchi	Passa-basso	3Hz
X	Real-Time	Picchi + Intersezione Asse Ascisse	Passa-basso	3Hz
XI	Real-Time	Picchi	Passa-basso	10Hz
XII	Real-Time	Picchi + Intersezione Asse Ascisse	Passa-basso	10Hz
XIII	Real-Time	Picchi	Passa-basso	2% Frequenza di Campionamento
XIV	Real-Time	Picchi + Intersezione Asse Ascisse	Passa-basso	2% Frequenza di Campionamento

Tabella 4.2: Elenco Configurazioni Testate

4.2 Generazione dei Grafici

La raccolta dei dati da parte dei tester ha portato ad ottenere un totale di 66 test, 11 per ognuna delle 6 diverse camminate, un numero ampiamente sufficiente con cui testare le diverse configurazioni ed ottenere informazioni attendibili relative alla loro effettiva precisione nei diversi contesti.
Partendo da questi test si è quindi utilizzata l'applicazione Android per calcolare, tramite la funzionalità di *Confronto Offline di Molteplici Configurazioni* [3.2.4], il numero di passi identificati da ciascuna delle 14 configurazioni per ognuno dei 66 test.
I numeri calcolati sono stati quindi utilizzati per generare un insieme di grafici di tipo *box and scatter plot*, dai quali è possibile ottenere informazioni rilevanti relativamente all'efficacia di ciascuna configurazione.
Questo tipo di grafico permette infatti di avere una chiara visione della distribuzione dei valori di un determinato insieme di dati.
In particolare, l'utilizzo di un rettangolo permette di evidenziare gli estremi dell'intervallo nel quale è distribuita la maggior parte dei dati, mentre un segmento all'interno del rettangolo ne indica il valore medio.
Nella particolare versione di grafico scelta è inoltre possibile vedere l'effettivo posizionamento di ciascuno dei dati registrati, mettendo in evidenzia i valori *outlier*, ovvero tutti quei valori chiaramente distanti dagli altri valori registrati.
In totale sono stati generati 7 differenti grafici di questo tipo: uno per ognuna delle sei diverse camminate, e uno comprensivo dell'insieme di tutti i dati registrati dai 66 test.
In ognuno dei grafici è stata indicata con una linea rossa orizzontale tratteggiata il numero di passi effettivamente effettuato durante i test, pari sempre a 50.
L'asse delle ordinate indica infatti il numero di passi identificati, mentre nell'asse delle ascisse sono indicate le 14 diverse configurazioni testate, identificabili attraverso la tabella 4.2.
L'efficacia di ciascuna configurazione è quindi valutabile, grazie alla struttura dei grafici, confrontando il valore medio calcolato sui dati di ciascuna di esse con l'effettivo numero

di passi effettuati, rapportando il tutto all'ampiezza del rettangolo.
Un rettangolo ampio indica infatti un distribuzione dei dati molto disomogenea, e corrisponde quindi a un algoritmo poco stabile e molto soggetto ad errore.
Un rettangolo più stretto indica invece un distribuzione molto limitata, e identifica quindi un algoritmo molto stabile e in grado di garantire risultati sempre coerenti.
Un configurazione può quindi essere ritenuta affidabile solo nel caso in cui venga rappresentata da un rettangolo stretto e diviso da un segmento posizionato in prossimità del valore 50, rappresentato nei grafici dalla linea rossa tratteggiata.
É stato inoltre generato un ottavo grafico, differente dai precedenti, raffigurante, tramite un *bar chart*, l'*errore medio* di ognuna delle configurazioni.
Vengono di seguito mostrati gli otto grafici generati, descrivendo i dati riportati all'interno di ciascuno di essi.

4.3 Analisi dei Grafici

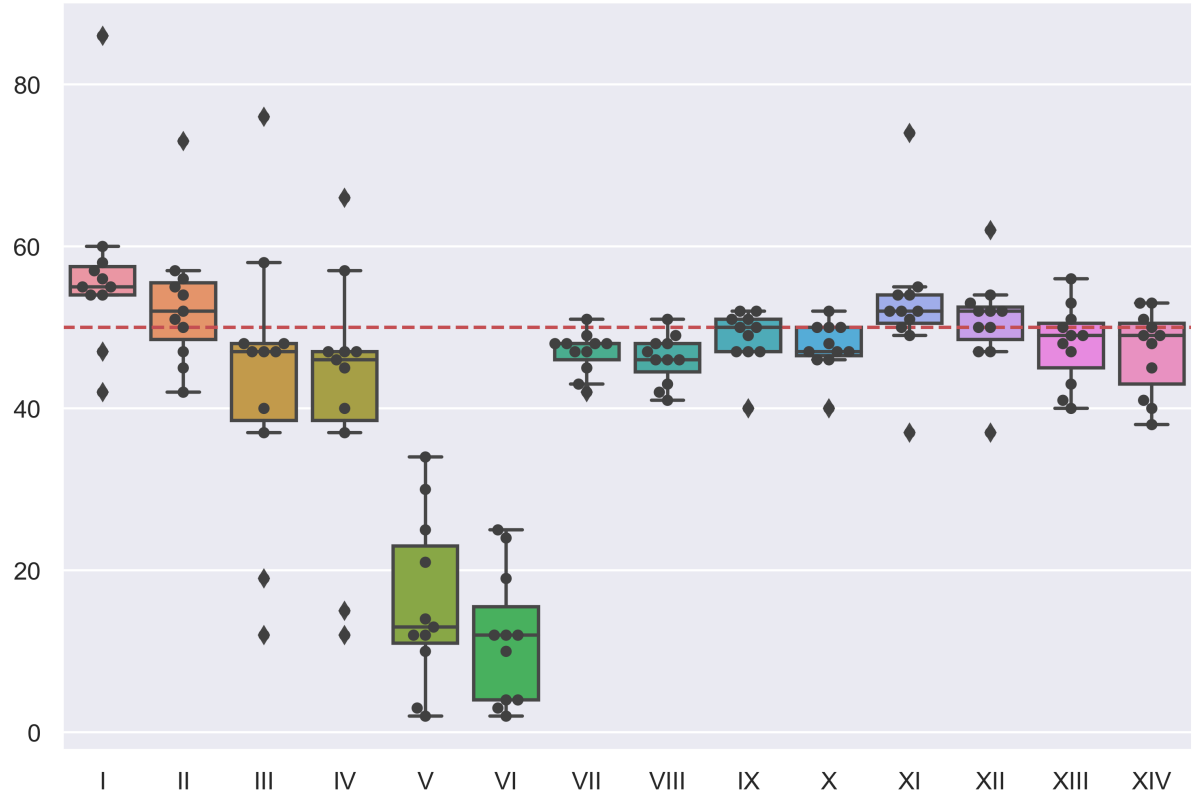

Figura 4.1: Camminata Normale

Il grafico 4.1 rappresenta i dati raccolti durante i test effettuati eseguendo una *Camminata Normale*.
Dall'immagine si può vedere come l'Algoritmo di Bagilevi fornisca risultati del tutto deludenti, mentre l'utilizzo di Nessun Filtro e l'utilizzo della Matrice di Rotazione fornisce risultati molto più precisi, in quanto la loro media si avvicina al valore 50.
La loro ampiezza tuttavia denota dei risultati poco stabili, specialmente nel caso dell'utilizzo della Matrice di Rotazione, nella quale si sono spesso registrati anche valori outlier.

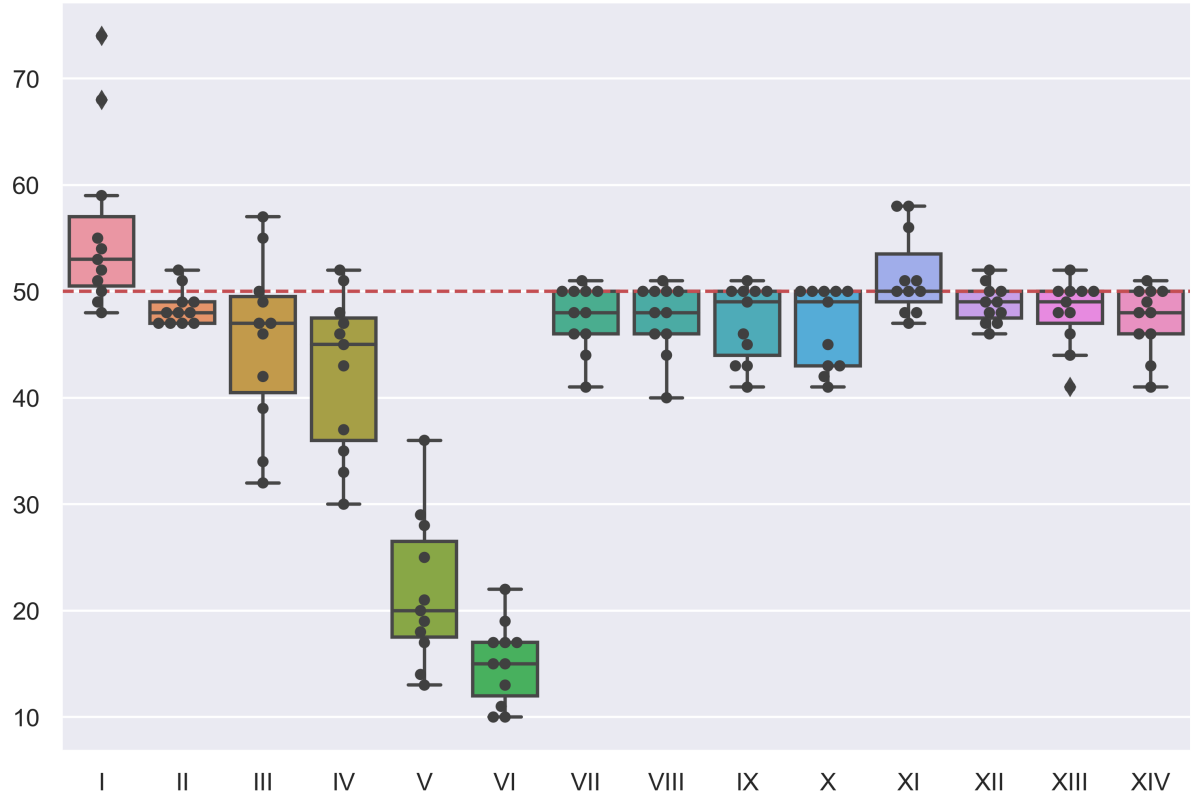

Figura 4.2: Camminata in Salita

Per quanto riguarda invece l'utilizzo del Filtro Passa-basso si può notare come in ogni casso esso fornisca risultati ottimali, specialmente quando utilizzato con una Frequenza di Taglio bassa.
In tutti i casi si può notare come l'utilizzo dell'Algoritmo di Intersezione dell'Asse delle Ascisse come correttivo non abbia portato a particolari miglioramenti, se non nel caso dell'utilizzo di Nessun Filtro.

Il grafico 4.2 rappresenta invece i dati raccolti durante i test effettuati eseguendo una *Camminata in Salita*.
I dati raccolti sono molto simili a quelli rappresentati nel grafico precedente, con la principale differenza che in questo caso non sono quasi mai stati registrati valori outlier.

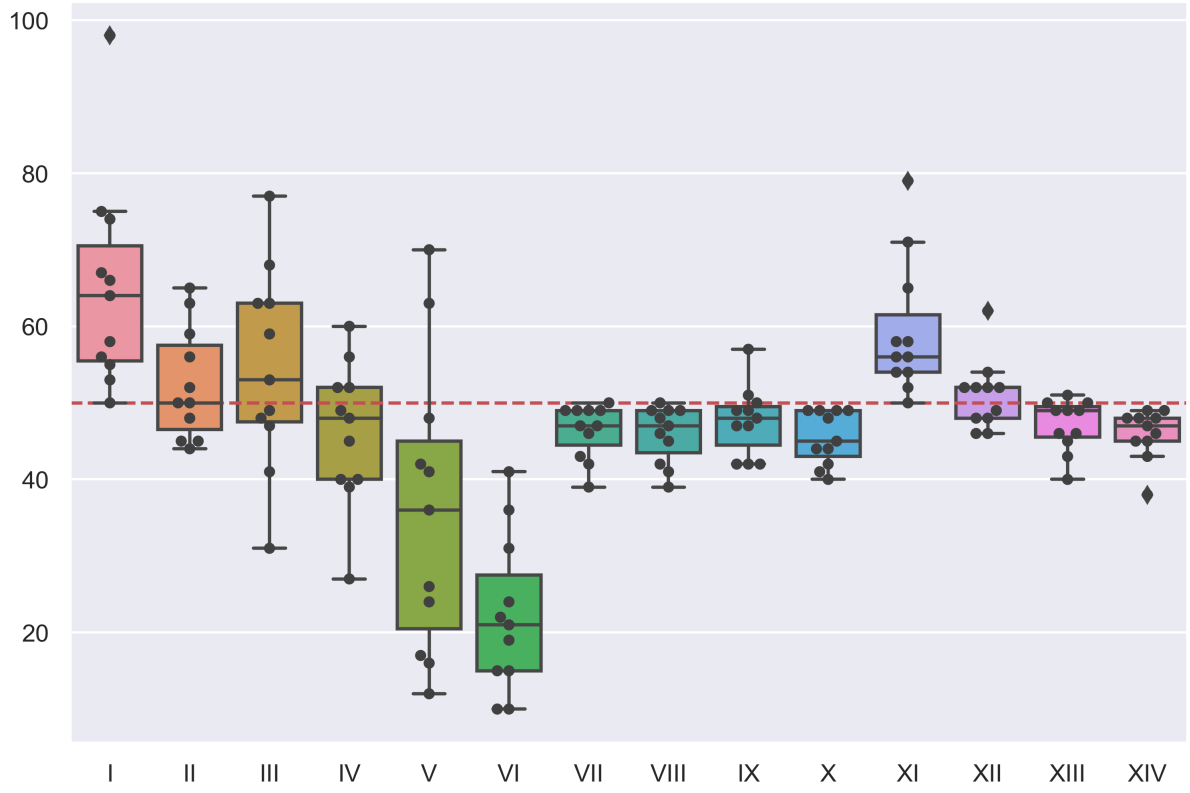

Figura 4.3: Camminata in Discesa

Anche in questo caso si può quindi notare la totale inefficienza dell'Algoritmo di Bagilevi. L'utilizzo della Matrice di Rotazione fornisce invece risultati leggermente peggiori, mentre l'utilizzo di Nessun Filtro fornisce dei risultati leggermente migliori, specialmente nel suo utilizzo combinato con l'Algoritmo di Intersezione dell'Asse delle Ascisse.
Anche l'utilizzo del Filtro Passa-basso fornisce risultati molto simili, anche se in questo caso presenta rettangoli leggermente più ampi quando utilizzato con una Frequenza di Taglio più bassa.

Passando quindi al grafico 4.3 si possono osservare i dati raccolti durante i test effettuati eseguendo una *Camminata in Discesa*.
Le osservazioni relative all'Algoritmo di Bagilevi sono sempre in linea con quelle fatte per i grafici precedenti, mentre l'utilizzo di Nessun Filtro e della Matrice di Rotazione ha

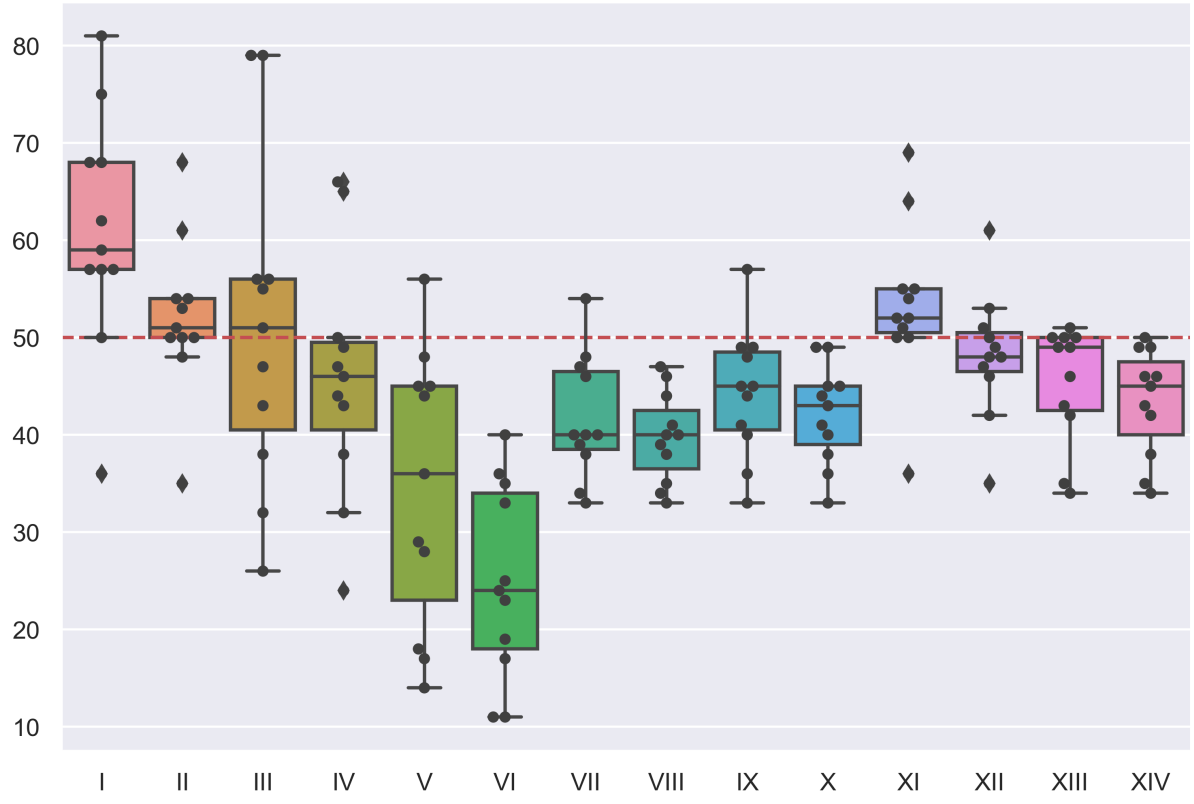

Figura 4.4: Camminata con Passi Irregolari

fornito in questo caso risultati molto meno coerenti, in quanto la loro rappresentazione risulta molto più ampia.
L'utilizzo del Filtro Passa-basso si conferma essere il metodo in grado di fornire i dati più precisi, tranne nel caso in cui venga utilizzato con una Frequenza di Taglio pari a 10Hz, in quanto in questo caso fornisce risultati leggermente sovrastimati.

Il grafico 4.4 rappresenta invece i dati raccolti durante i test effettuati eseguendo una *Camminata con Passi Irregolari*.
Come si può vedere dall'immagine, questo tipo di camminata ha messo in difficoltà la maggior parte degli algoritmi, mettendo in risalto i loro limiti.
L'algoritmo di Bagilevi si è confermato essere il meno accurato, mentre l'utilizzo della

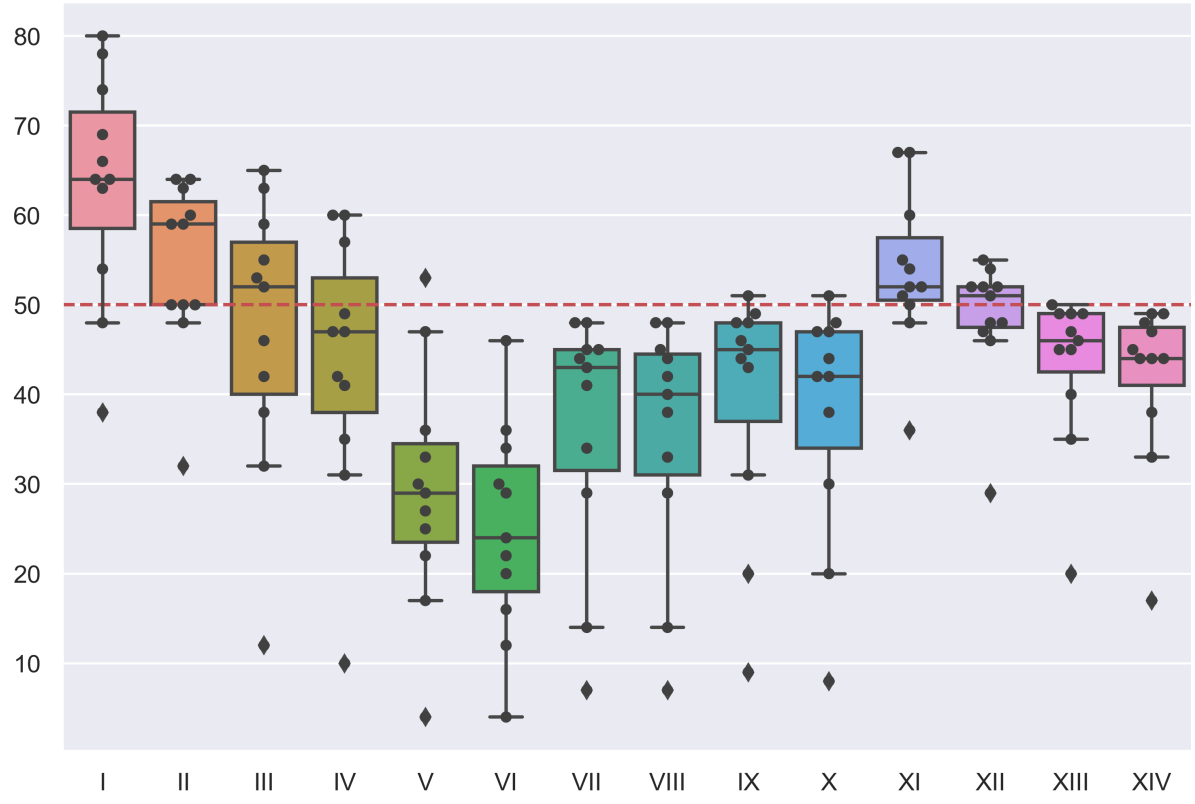

Figura 4.5: Camminata con Passi Stretti

Matrice di Rotazione ha generato risultati molto disomogenei, leggermente corretti dall'utilizzo dell'Algoritmo di Intersezione dell'Asse delle Ascisse.
L'utilizzo di Nessun Filtro ha generato invece risultati parecchio sovrastimati, che sono però risultati essere molto più accurati se corretti dall'Algoritmo di Intersezione, nonostante si sia verificato qualche outlier.
Anche in questo caso l'utilizzo del Filtro Passa-basso sembra aver generato i risultati più precisi, seppur spesso leggermente sottostimati.

Anche il grafico 4.5, relativo ai dati raccolti durante i test effettuati eseguendo una *Camminata con Passi Stretti*, ha messo in difficoltà la maggior parte degli algoritmi.
Essendo i risultati del tutto simili a quelli mostrati nell'immagine 4.4, si rimanda il let-

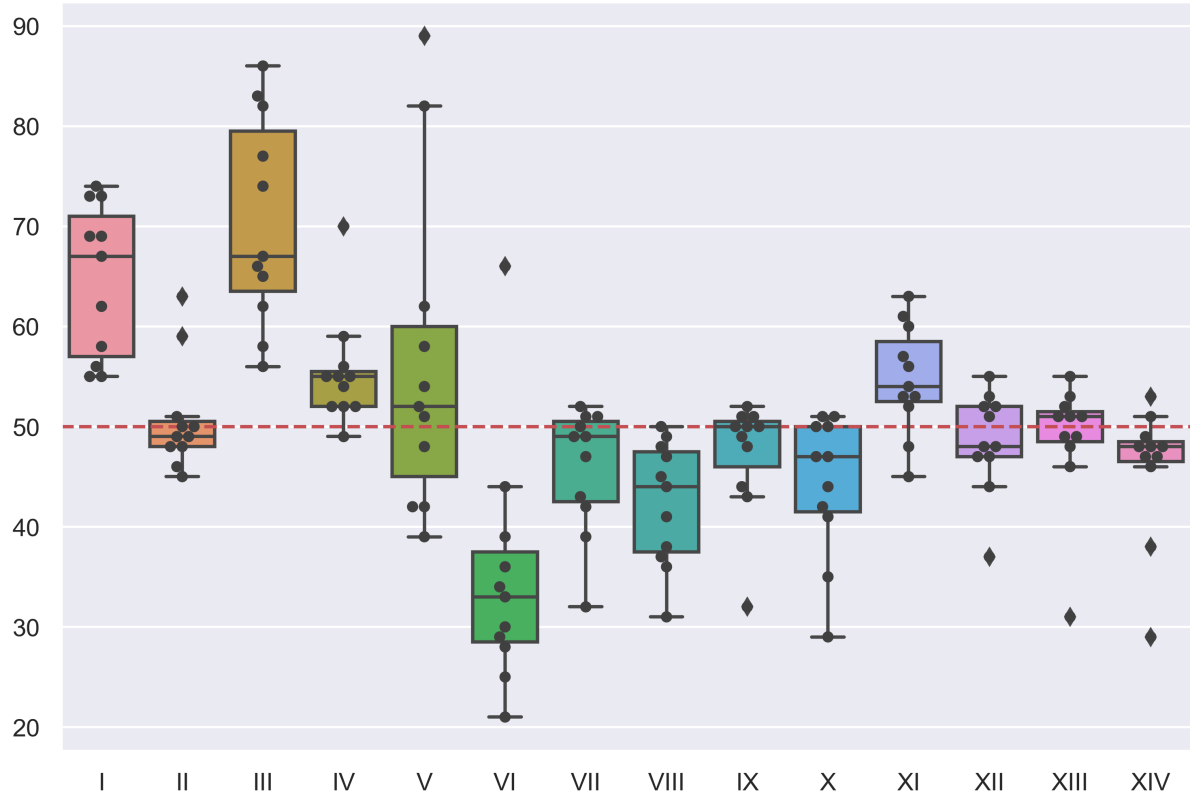

Figura 4.6: Corsa

tore all'analisi dei dati fatta per quel grafico, presente nel precedente paragrafo appena concluso.

Il grafico 4.6 rappresenta invece i dati raccolti durante i test effettuati eseguendo una *Corsa*.
Anche in questo caso molti degli algoritmi hanno calcolato risultati poco accurati, specialmente nel caso dell'utilizzo di Nessun Filtro e dell'utilizzo della Matrice di Rotazione. In entrambi i casi però l'utilizzo dell'Algoritmo di Intersezione dell'Asse delle Ascisse ha fornito una correzione particolarmente accurata, fornendo risultati molto più precisi e compatti.
Anche in questo caso l'algoritmo di Bagilevi si conferma essere il meno preciso, mentre

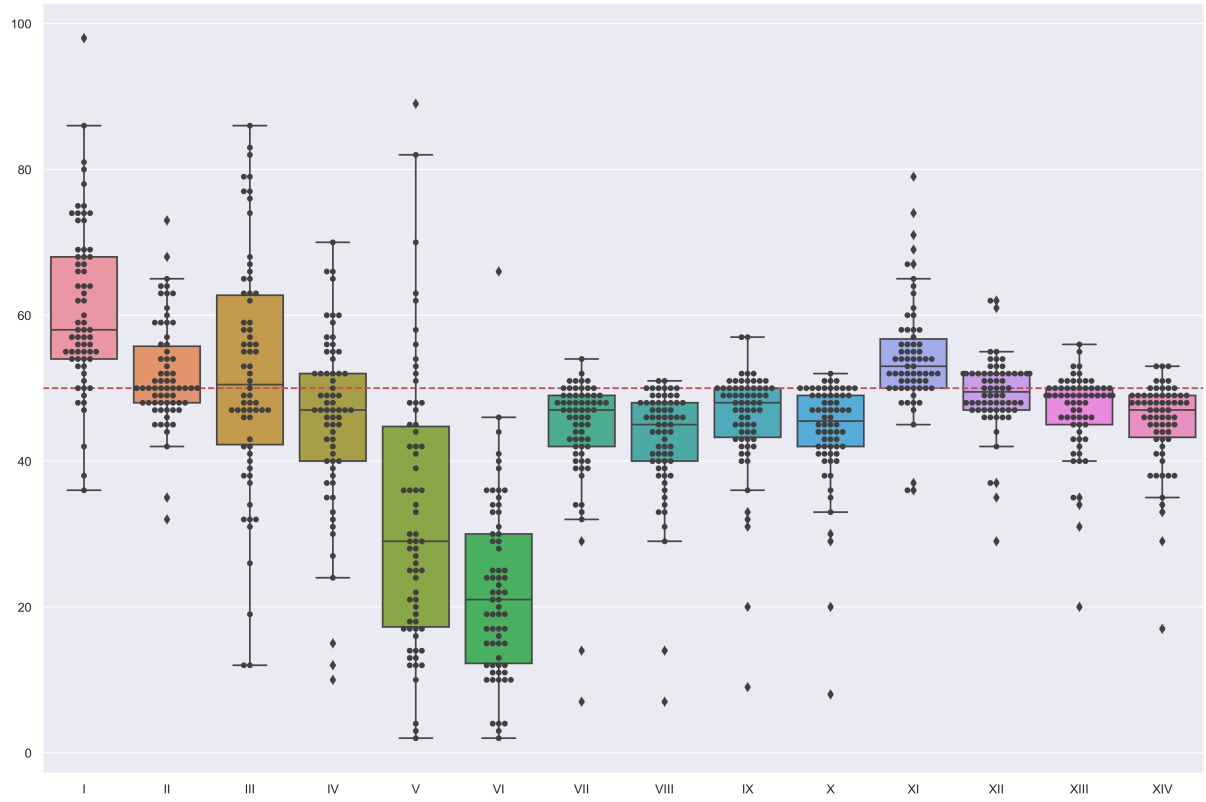

Figura 4.7: Raggruppamento Test

l'utilizzo del Filtro Passa-basso presenta i risultati miglior,i specialmente se utilizzato con Frequenze di Taglio pari al 2% della Frequenza di Campionamento.

Infine l'ultimo grafico di questo tipo, illustrato nell'immagine 4.7, rappresenta il raggruppamento di tutti i dati calcolati in tutte le diverse condizioni, permettendo di visualizzare il comportamento di ogni algoritmo rispetto alla totalità dei dati registrati da ciascuno di essi.
Da questo grafico si può quindi notare come l'algoritmo di Bagilevi sia definitivamente il meno preciso, in quanto registra dati completamente incoerenti tra loro.
Si può inoltre notare come l'utilizzo di Nessun Filtro e della Matrice di Rotazione generi dati particolarmente incoerenti se utilizzati con il solo Algoritmo dei Picchi, mentre generi dati molto più accurati se utilizzati con l'ulteriore ausilio dell'Algoritmo di Inter-

sezione dell'Asse delle Ascisse, specialmente nel caso dell'utilizzo di Nessun Filtro.
L'utilizzo del Filtro Passa-basso si conferma invece essere il migliore, registrando dati molto coerenti tra loro e spesso molto vicini al valore 50, e registrando inoltre un numero molto limitato di valori outlier.
Questo grafico è strettamente legato al grafico 4.8, nel quale viene riportato l'*errore medio* di ognuna delle configurazioni, corrispondente alla media delle distanze tra i dati registrati e il valore 50, calcolate in valore assoluto.
Questi valori sono un ulteriore strumento per valutare l'effettiva efficacia di ciascun algoritmo.
Come è facile intuire, più questo valore si avvicina allo 0 e più la corrispondente configurazione fornisce risultati precisi.

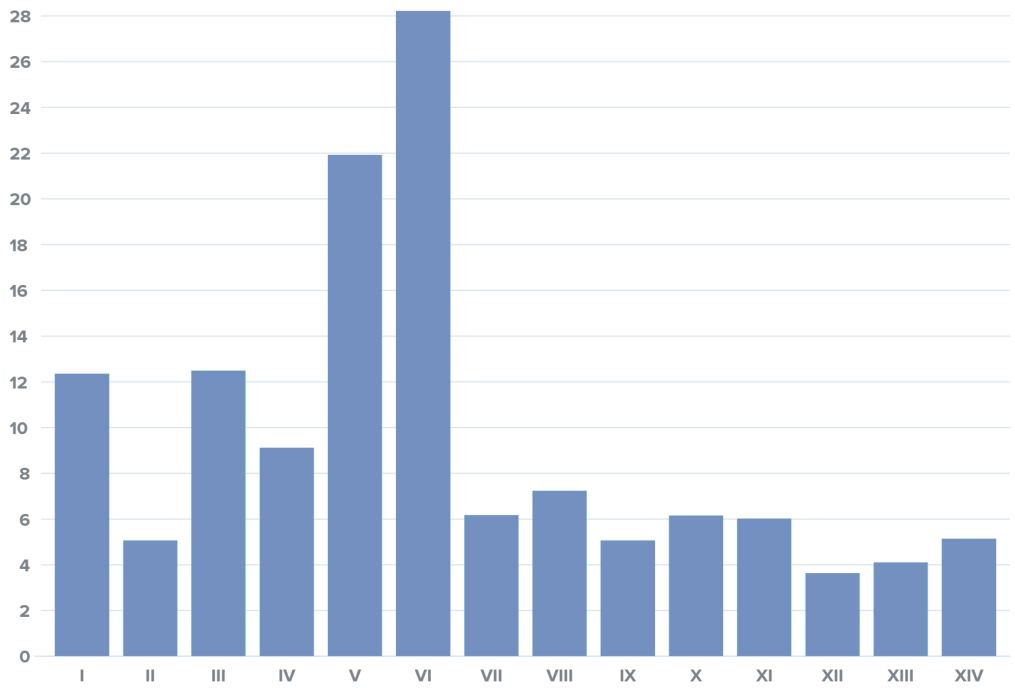

Figura 4.8: Errore Medio dei Test

4.3.1 Considerazioni sui Risultati

In conclusione, la generazione dei grafici ha permesso di ottenere dati oggettivi relativi alla precisione e all'efficacia di ciascuna configurazione.
In particolare si è potuto notare come l'Algoritmo di Bagilevi sia in ogni situazione il peggiore, in quanto fornisce risultati altamente incoerenti e quasi sempre completamente distanti dal valore corretto, oltre ad essere soggetto ad un errore medio particolarmente elevato.
Si è invece potuto notare come il Filtro Passa-basso sia il migliore nella quasi totalità dei casi, indipendentemente dalla Frequenza di Taglio scelta.
L'utilizzo dell'Algoritmo di Intersezione dell'Asse delle Ascisse come correttivo genera miglioramenti rilevanti quando utilizzato nelle configurazioni relative a Nessun Filtro e alla Matrice di Rotazione, mentre non risulta essere particolarmente rilevante quando utilizzato nelle configurazioni relative all'utilizzo del Filtro Passa-basso.
Anche l'analisi dei valori di errore medio presenti nel grafico 4.8 fornisce indicazioni completamente concordi con quanto considerato fino ad ora.
In generale si è potuto notare che la maggior parte delle configurazioni ha avuto difficoltà nel conteggio dei passi durante le camminate più particolari, ovvero quella irregolare e quella composta da passi stretti.
Questa caratteristica è sicuramente dovuta all'estrema particolarità dei dati registrati dai sensori durante queste camminate caratterizzate da condizioni *limite*.
Anche in questo caso le configurazioni più affidabili sono state comunque quelle caratterizzate dall'utilizzo del Filtro Passa-basso.
In definitiva la configurazione migliore è senza dubbio quella composta dall'utilizzo del Filtro Passa-basso con Frequenza di Taglio pari a 10Hz, calcolato con l'Algoritmo dei Picchi e corretto con l'Algoritmo di Intersezione, la quale ha generato risultati particolarmente coerenti tra loro e quasi sempre vicini al valore corretto, senza generare quasi mai valori outlier e generando il valore di errore medio in assoluto più basso.

Conclusioni

Questo studio è stato realizzato con lo scopo di sintetizzare in un unico schema le principali caratteristiche delle implementazioni di pedometri trovate in letteratura, e di fornire un confronto oggettivo sulla reale efficacia di ciascuna di esse.
Durante la fase di analisi degli studi si è potuta notare una particolare ridondanza nelle scelte implementative proposte al loro interno.
Si è scelto quindi di approfondire questa caratteristica, che è da subito sembrata essere particolarmente rilevante.
La realizzazione della Tassonomia ha quindi reso disponibile uno strumento molto dettagliato con cui poter analizzare le diverse implementazioni proposte.
Questa analisi ha portato all'identificazione di uno schema algoritmico ben definito condiviso dalla quasi totalità delle implementazioni proposte, distinte soltanto dalla scelta di specifiche variabili discriminanti.
L'identificazione di questo particolare schema è probabilmente dovuta a condivise caratteristiche e problematiche riscontrate dai diversi sviluppatori durante l'implementazione dei pedometri.
Una futura ulteriore analisi di questa struttura potrebbe portare all'identificazione di uno *standard* da utilizzare durante la fase di implementazione di un nuovo pedometro.
L'implementazione dell'applicazione ha invece fornito un potente strumento da poter utilizzare per testare e confrontare tutte le possibili combinazioni di implementazioni di pedometri.
L'applicazione è stata infatti sviluppata in modo tale da poter essere in futuro aggiornata inserendo qualsiasi tipo di nuovo strumento da utilizzare all'interno di un pedometro.
Una prima implementazione da aggiungere può sicuramente essere l'introduzione della

modalità *non real-time*, per la quale l'applicazione è già stata predisposta.
Il codice sorgente dell'applicazione è stato caricato all'interno di un Repository GitHub, e può essere raggiunto al seguente link:
https://github.com/GiacomoNeriUnibo/Pedometers.git.
Infine, la generazione dei vari grafici ha permesso di avere un chiaro riscontro visivo dell'effettiva precisione delle diverse implementazioni di pedometri.
Il tipo di grafico scelto permette infatti di avere una visione completa della distribuzione dei valori calcolati da ognuna di esse.
Come descritto in precedenza, l'analisi dei grafici ha fatto emergere, per quanto riguarda la precisione di calcolo dei passi, un'evidente superiorità delle implementazioni che fanno uso del Filtro Passa-basso rispetto a tutte le altre implementazioni.

Bibliografia

[1] https://www.ieee.org/.

[2] https://github.com/.

[3] https://github.com/bagilevi/android-pedometer.

[4] F. Gu, K. Khoshelham, J. Shang, F. Yu, and Z. Wei. Robust and accurate smartphone-based step counting for indoor localization. *IEEE Sensors Journal*, 17(11):3453–3460, 2017.

[5] W. Hongman, Z. Xiaocheng, and C. Jiangbo. Acceleration and orientation multisensor pedometer application design and implementation on the android platform. In *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pages 249–253, 2011.

[6] S. Jayalath and N. Abhayasinghe. A gyroscopic data based pedometer algorithm. In *2013 8th International Conference on Computer Science Education*, pages 551–555, 2013.

[7] W. Kang, S. Nam, Y. Han, and S. Lee. Improved heading estimation for smartphone-based indoor positioning systems. In *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, pages 2449–2453, 2012.

[8] G. Liu. Design and implementation of pedometer based on g-sensor. In *2018 3rd International Conference on Smart City and Systems Engineering (ICSCSE)*, pages 436–438, 2018.

[9] Pablo Sebastián Navarro Morales. Smart movement detection for android phones. 2016.

[10] M. Oner, J. A. Pulcifer-Stump, P. Seeling, and T. Kaya. Towards the run and walk activity classification through step detection - an android application. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1980–1983, 2012.

[11] T. O. Oshin and S. Poslad. Ersp: An energy-efficient real-time smartphone pedometer. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2067–2072, 2013.

[12] M. Pan and H. Lin. A step counting algorithm for smartphone users: Design and implementation. *IEEE Sensors Journal*, 15(4):2296–2305, 2015.

[13] Dario Salvi, Carmelo Velardo, Jamieson Brynes, and Lionel Tarassenko. An optimised algorithm for accurate steps counting from smart-phone accelerometry. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4423–4427. IEEE, 2018.

[14] J. S. Sheu, G. S. Huang, W. C. Jheng, and C. H. Hsiao. Design and implementation of a three-dimensional pedometer accumulating walking or jogging motions. In *2014 International Symposium on Computer, Consumer and Control*, pages 828–831, 2014.

[15] A. K. Siddanahalli Ninge Gowda, S. R. Babu, and D. C. Sekaran. Umoisp: Usage mode and orientation invariant smartphone pedometer. *IEEE Sensors Journal*, 17(3):869–881, 2017.

[16] N. Strozzi, F. Parisi, and G. Ferrari. A novel step detection and step length estimation algorithm for hand-held smartphones. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7, 2018.

[17] A. Suprem, V. Deep, and T. Elarabi. Orientation and displacement detection for smartphone device based imus. *IEEE Access*, 5:987–997, 2017.

[18] B. Wang, X. Liu, B. Yu, and R. Jia. Posture recognition and adaptive step detection based on hand-held terminal. In *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*, pages 1–5, 2018.

[19] S. Wu and H. Wu. The design of an intelligent pedometer using android. In *2011 Second International Conference on Innovations in Bio-inspired Computing and Applications*, pages 313–315, 2011.

[20] X. Yang and B. Huang. An accurate step detection algorithm using unconstrained smartphones. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pages 5682–5687, 2015.

Ringraziamenti

Mi è doveroso dedicare questo spazio del mio elaborato a tutte le persone che hanno contribuito alla realizzazione dello stesso.

Vorrei innanzitutto ringraziare il mio relatore, il Dott. Montori Federico, che mi ha fatto appassionare al mondo Android durante il corso di *Laboratorio di Applicazioni Mobili* e che mi ha guidato durante tutto il percorso di realizzazione dell'elaborato, dalle fasi di ricerca fino alla sua stesura.

Ringrazio infinitamente tutti i miei amici che non hanno esitato a dare la loro disponibilità per effettuare tutti i test di camminata, necessari al completamento dell'elaborato.

Un ringraziamento particolare va al mio amico Claudio che ha contribuito con le sue idee a superare i primi ostacoli incontrati durante le fasi di ricerca.

Ringrazio inoltre i miei colleghi Gianluca, Emanuele, Simone e Matteo, con i quali ho condiviso gioie e fatiche di questi tre anni universitari.

Ringrazio infine la mia fidanzata Giulia per avermi costantemente incoraggiato e supportato durante questa ultima fase del mio percorso di studi.