UNIVERSITÀ DEGLI STUDI DI FIRENZE
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Specialistica in Scienze e Tecnologie
dell'Informazione


Tesi di Laurea

# METODI DI APPRENDIMENTO AUTOMATICO PER L'ANALISI IN LINEA DELLA DEAMBULAZIONE: SVILUPPO E VALUTAZIONE DI UN APPLICATIVO PER SMARTPHONE

AHADU TSEGAYE

Relatore *Prof. Angelo Maria Sabatini*

Co-Relatore *Prof. Maria Cecilia Verri*

Anno Accademico 2010-2011
Revisione 3.40

Ahadu Tsegaye: *Machine Learning Methods for online Gait Analysis: Developement and Validation of a Smartphone Application*, Corso di Laurea Specialistica in Scienze e Tecnologie dell'Informazione © Anno Accademico 2010-2011-v3.40

# 1

## SOMMARIO

[ *TODO: riscrivere alla fine* ]

In questo lavoro si affronta il problema dell'analisi della deambulazione umana mediante l'uso di sensori indossabili e dispositivi di computazione portabili smartphone.

# Parte I

# STATO DELL'ARTE

# 2 INTRODUZIONE

Ipotesi: *è possibile costruire un sistema intelligente e portatile in grado di riconoscere e monitorare i movimenti di un individuo e di fornirne informazioni a riguardo in tempo reale. Per ridurre la complessità del problema di riconoscimento di un movimento generico, prendiamo in considerazione solo gli arti inferiori nella deambulazione entro un intervallo di velocità e pendenza del terreno.*

A questo punto si tratta di riconoscere le 8 fasi della deambulazione normale studiate dalla Chinesiologia (vedi capitolo 3), vale a dire il problema della *segmentazione automatica in tempo reale della deambulazione umana*.

La soluzione del problema qui menzionato, avrebbe risvolti immediati in Medicina per la riabilitazione per la diagnosi e/o assistenza a persone con problemi di deambulazione, nella Robotica e Computer Grafica per la emulazione/simulazione della deambulazione umana, nel mondo dello sport agonistico per l'apprendimento di specifiche tecniche motorie ed in innumerevoli altri settori.

Il problema della segmentazione della deambulazione è stato ampiamente affrontato in letteratura e con svariate combinazioni di materiali e metodi.

Per quanto riguarda i materiali, sono state proposte soluzioni parziali che si basano semplicemente sull'osservazione diretta di un fisiatra oppure su strumenti ottici, basati su sistemi di telecamere e marker (segnali riconoscibili dalle telecamere); strumenti inerziali, basati su sensori fisici: accelerometri, giroscopi, elettromagnetometri ecc. Questi ultimi sono stati posizionati in svariate parti del corpo ed in diverse combinazioni.

Per quanti riguarda i metodi, gli ambiti scientifici nei quali viene affrontato il problema spaziano dalla Cinematica per lo studio delle forze, all'Analisi Matematica per lo studio di curve, alle Macchine a stati finiti per lo studio di sequenze temporali, e più di recente all'Apprendimento Automatico per la capacità di astrarre sulle differenze individuali nel compiere qualsiasi azione.

Nonostante il vasto numero di lavori, non è ancora stata data una soluzione soddisfacente al problema. Tutti i metodi proposti peccano di dipendenza dagli strumenti che usano, vale a dire che variando questi ultimi, variano le prestazioni dei metodi e di dipendenza dai soggetti sui quali vengono fatti gli esperimenti. Questo significa che le soluzioni sin ora proposte sono fatte su misura per specifici casi.

Il nostro obbiettivo è creare un metodo che dipenda il meno possibile dai materiali, che tenga conto della variabilità interpersonale e che abbia una complessità computazionale abbastanza bassa da poter essere usato su uno Smartphone.

Le nostre scelte sono quindi cadute su pochi sensori semplici per la raccolta dei dati e sulle HMM (Hidden Markov Models vedi appendice A.3) per gestirne la componente temporale, creando così un sistema a efficace basso costo, portatile, semplice da utilizzare e .

# 3

# CENNI ALLA CHINESIOLOGIA

κινησισ (kinēsis): mobilità,
λογια (logia): studio di

> Lo studio dei principi su cui si fondano la meccanica e l'anatomia del movimento umano.
>
> *www.merriam-webster.com*[1]

La Chinesiologia studia il movimento umano sotto diversi aspetti: biomeccanico, del controllo motorio e della psicologia del moto.
L'approccio biomeccanico [2] consiste nell'applicazione dei principi della Meccanica allo studio di organismi viventi: principalmente proprietà fisiche di materiali biologici, segnali biologici, modellazioni e simulazioni biomeccaniche.

Per restringere l'ambito, noi ci concentriamo sulle interazioni biomeccaniche dell'apparato locomotore (scheletro e muscoli). La branca della Meccanica Classica (vedi appendice B) che viene utilizzata dalla Biomeccanica è la Cinematica [3] che si occupa di descrivere la posizione ed il moto di oggetti nello spazio, senza riferimento alle forze o masse coinvolte (vale a dire alle cause e agli effetti di tale moto).

## 3.1 ANALISI DEI MOVIMENTI DEL CORPO UMANO

Per un trattamento rigoroso dei movimenti del corpo umano, è necessario stabilire dei piani di riferimento, lungo i quali collocare le diverse parti del corpo (vedi figura 1).

---

**Definizione 1 (Piani che tagliano il corpo umano).** I movimenti del corpo umano vengono descritti in riferimento a tre piani:

1. **Frontale o Coronale**: piano verticale che divide il corpo in parte anteriore e posteriore.
2. **Sagittale**: piano verticale che divide il corpo in parte sinistra e destra.
3. **Trasversale o Orizzontale**: piano orizzontale che divide il corpo in parte superiore ed inferiore.

---

Ad esempio la deambulazione o corsa avviene principalmente lungo il piano sagittale, sollevare le braccia lateralmente comporta un movimento sul piano frontale, mentre la rotazione della testa per guardarsi intorno avviene principalmente lungo il piano trasversale.

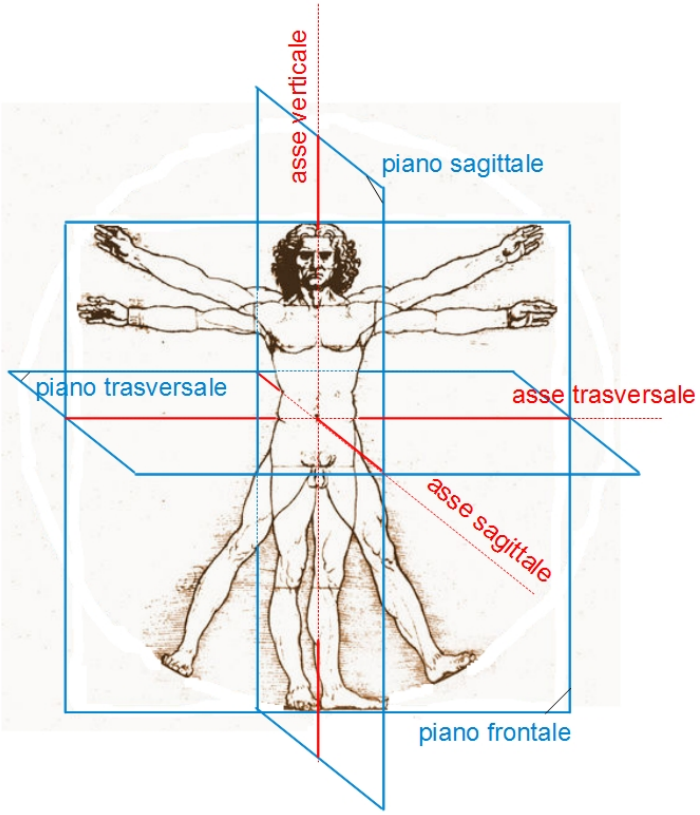


Figura 1.: Immagine riadattata. Originale cortesia di `http://sciencephoto.com`

## 3.2 L'ANDATURA DURANTE LA DEAMBULAZIONE

Viene definita Andatura (Gait) la sequenza di movimenti degli arti inferiori che un animale compie su una superficie solida durante la locomozione [4]. Gli animali possiedono diverse forme di andatura che scelgono in base alla velocità, ed al terreno ed ad altre variabili.
La Deambulazione (o Camminata) è una delle principali forme di andatura degli animali aventi arti inferiori ed avviene tipicamente a velocità inferiore a quelle della Corsa (che a sua volta è una forma di locomozione).

---

**Definizione 2 (Deambulazione Normale).** Una Deambulazione normale (negli esseri umani) è composta di due macro fasi per ciascun piede:

1. `Fase_A_Terra` (stance), in cui il piede supporta tutto il peso del corpo e
2. `Fase_In_Aria` (swing), in cui il piede è in aria e porta avanti il baricentro del corpo, mentre il peso del corpo è sull'altro piede.

I due arti inferiori sono sempre alternativamente nelle due fasi e per circa il 25% del tempo sono in contatto simultaneo con il pavimento.

---

## 3.3 BREVE STORIA DELLO STUDIO DELLA DEAMBULAZIONE UMANA

Un primissimo contributo allo studio dell'andatura umana è stato dato dai fratelli Wilhelm Weber, fisico, e Eduard Weber, anatomista. I Weber, nel loro libro *The Mechanics of Human Motions* [5], pubblicato nel 1836, definiscono e misurano per la prima volta la durata delle fasi della Deambulazione Normale (vedi definizione 2), usando solamente un cronometro ed un telescopio con una scala.
Un grande contributo a questo campo è stato dato dal fisiologo francese Ètienne Jules Marey, che nel 1873 pubblicò il trattato *Animal Mechanism: a Treatise on Terrestrial and Aerial Locomotion* dove con l'uso di scarpe a camera d'aria collegate a un registratore e della Cronofotografia Geometrica[1] e con l'uso di soggetti vestiti con abiti aderenti e neri con bottoni di metallo e strisce riflettenti, riuscìì a misurare la durata del contatto del piede con il suolo, durante la camminata in piano, su un terreno regolare. Inoltre egli introdusse il concetto dell'efficienza energetica del movimento.
Il fotografo inglese Edward Muybridge, nel 1887 con l'uso della fotografia seriale, con 48 fotocamere elettriche sincronizzate riuscìì a catturare la fase in volo di un cavallo al galoppo.
L'anatomista tedesco Wilhelm Braune, ed il matematico tedesco Otto Fisher, negli anni 1890, con l'uso di un sistema a 4 fotocamere, un tubo luminoso attaccato al corpo ripreso ed un sistema di riferimento rettangolare, riuscirono a analizzare per la prima volta l'andatura in 3D ed a stabilire i metodo per il calcolo dei parametri meccanici dello stesso.
Nel 1938 Elftman e 20 anni dopo Frankel diedero grossi contributi agli studi di tipo cinetico sul passo normale e patologico[2].
M.P. Murray, negli anni '60, con l'uso della fotografia a luce interrotta e più tardi

1 più riprese fotografiche vengono impresse sulla stessa fotografia, in modo che possa essere ripresa una sequenza di azioni della persona. Si tratta di un antenato della cinepresa.

2 passo affetto da una qualunque tipo di deformazione rispetto al passo normale

con il sistema 3D, diede dei contributi allo studio cinetico in pazienti normali e patologici.
J Perry con l'uso di goniometri elettronici monoassiali ha sviluppato un nuovo sistema di terminologie per l'andatura sia normale che patologica.

## 3.4 CICLO DI DEAMBULAZIONE (GAIT CYCLE)

L'unità di base per la descrizione dell'andatura nella deambulazione è il ciclo di andatura (Gait Cycle) (vedi figura 2). Si tratta della sequenza di azioni compiute dal corpo dall'istante del contatto di un tallone a terra fino al contatto successivo dello stesso tallone. Il ciclo di andatura ha una durata compresa nell'intervallo di [0.5-2] secondi, in base alla velocità. Il ciclo di andatura è suddiviso in due fasi: una di appoggio, in cui il piede è a contatto con il terreno, ed una aerea.

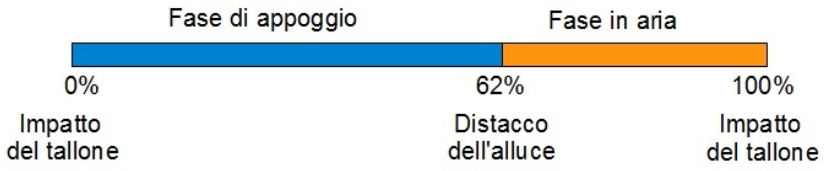


Figura 2.: Ciclo completo di andatura

La fase di appoggio (dal contatto del tallone al distacco dell'alluce dal terreno) viene ulteriormente suddivisa in tre sottofasi:

1. contatto iniziale;
2. appoggio intermedio (detta anche fase di risposta al carico);
3. fase propulsiva (o di contatto finale).

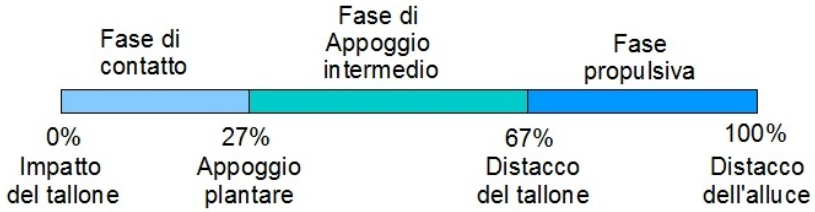


Figura 3.: Ciclo completo di andatura

Anche la fase in cui il piede è in aria, viene suddivisa in tre sottofasi:

1. propulsione iniziale (initial swing),
2. propulsione mediale (mid swing)
3. decelerazione (terminal swing).

| NOME PARAMETRO | UNITÀ DI MISURA | DESCRIZIONE |
|---|---|---|
| Andatura | (sec) | durata di un completo ciclo di andatura |
| Passo | (sec) | durata di un completo passo sinistro o destro |
| Contatto | (sec, %) | durata del periodo in cui i piedi rimangono a contatto con il terreno |
| Contatto a piede singolo | (sec, %) | durata del periodo in cui solo un piede rimane a contatto con il terreno |
| Doppio contatto | (sec, %) | durata del periodo in cui entrambi piede rimangono contemporaneamente a contatto con il terreno |
| Fase aerea | (sec, %) | durata del perodo in cui il piede è in aria |

Tabella 1.: Parametri temporali

### 3.4.1 *Confronto fra Falcata (Stride) e Passo (Step)*

La falcata, che va dal contatto del tallone con il suolo fino al successivo contatto dello stesso piede, è sinonimo di ciclo di andatura (Gait Cycle), mentre il passo comincia dal contatto di un tallone e termina al contatto dell'altro tallone. Una falcata coincide esattamente con due passi (vedi figura 4).

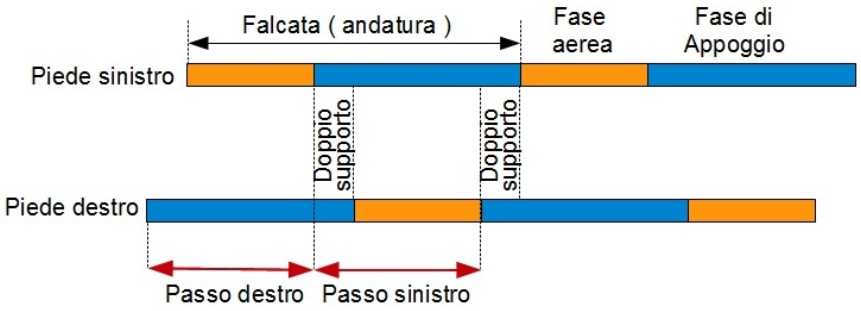


Figura 4.: Andatura e Passo a confronto

## 3.5 PARAMETRI CHE DESCRIVONO IL PATTERN (SCHEMA) DELLA DEAMBULAZIONE

| NOME PARAMETRO | UNITÀ DI MISURA | DESCRIZIONE |
|---|---|---|
| Lunghezza dell'andatura | (cm) | distanza tra due punti di contatto successivi dello stesso tallone |
| Lunghezza del passo | (cm) | distanza tra due punti di contatto successivi di talloni opposti |
| Larghezza del passo | (cm) | distanza laterale tra due punti di contatto successivi di talloni opposti |
| Angolo del piede | (grad) | angolo tra il collo del piede e lo stinco |

Tabella 2.: Parametri spaziali

| NOME PARAMETRO | UNITÀ DI MISURA | DESCRIZIONE |
|---|---|---|
| Cadenza | (passi/ min) | numero di passi al minuto |
| Velocità del passo | (m/s) | numero di metri percorso al secondo |

Tabella 3.: Parametri di velocità

# 4 LO STATO DELL'ARTE

Il mondo scientifico che si è confrontato con il problema della segmentazione della deambulazione, e più in generale sull'analisi della deambulazione ed individuazione delle sue varie fasi, è molto vasto e variegato. Per avere una visuale completa sul panorama di ricerca in questo ambito, si può suddividere lo stato dell'arte per materiali e metodi usati.

## 4.1 MATERIALI

Per materiali si intende tutto l'arsenale fisico, usato per captare, misurare e raccogliere dati riguardanti la deambulazione. I materiali possono, a loro volta, essere suddivisi in quelli che permettono di fare misure dirette dei parametri della deambulazione, ovvero spostamento, velocità e accelerazione sia lineare che angolare di arti e articolazioni; e quelli che permettono di fare misurazioni indirette della deambulazione con sistemi di tracciamento [ *TODO: dare spiegazione* ] del movimento.

### 4.1.1 *Osservazione diretta del paziente*

Il fisiatra è figura medica che si occupa di diagnosi, terapia e riabilitazione di persone con problemi di limitazioni di attività, in questo caso motorie. Un fisiatra è in grado, ad occhio nudo, di fare una stima accurata dei parametri della deambulazione umana.

### 4.1.2 *Stereofotogrammetria*

Metodologia di ricostruzione di oggetti ed del loro moto in tre dimensioni basato sulla sovrapposizione di immagini riprese da più telecamere posizionate intorno all'oggetto (vedi figura 5). Sull'oggetto vengono posizionati dei marker [ *TODO: in italiano?* ] che sono i punti di riferimento per le telecamere. La ricostruzione del moto dell'oggetto avviene mediante il calcolo degli spostamenti relativi dei marker. [ *TODO: METTERE UNA NOSTRA FOTO* ]

### 4.1.3 *Sensori Inerziali*

I sensori inerziali misurano, le forze inerziali (vedi appendice B) che agiscono su di essi (vedi appendice C).

- **Resistori Sensibili alla forza**: misurano la forza esercitata dalla massa della persona, sul piano di deambulazione, accelerata dalla forza di gravità.
- **Giroscopi** misurano l'accelerazione (momento) angolare. Questo può essere fatto meccanicamente con un giroscopio a rotazione o elettronicamente con un giroscopio a vibrazione (vedi appendice C.2).
- **Accelerometri** misurano il peso per unità di massa, questa quantità è nota come forza specifica o g-force. In altre parole, misurando il pe-

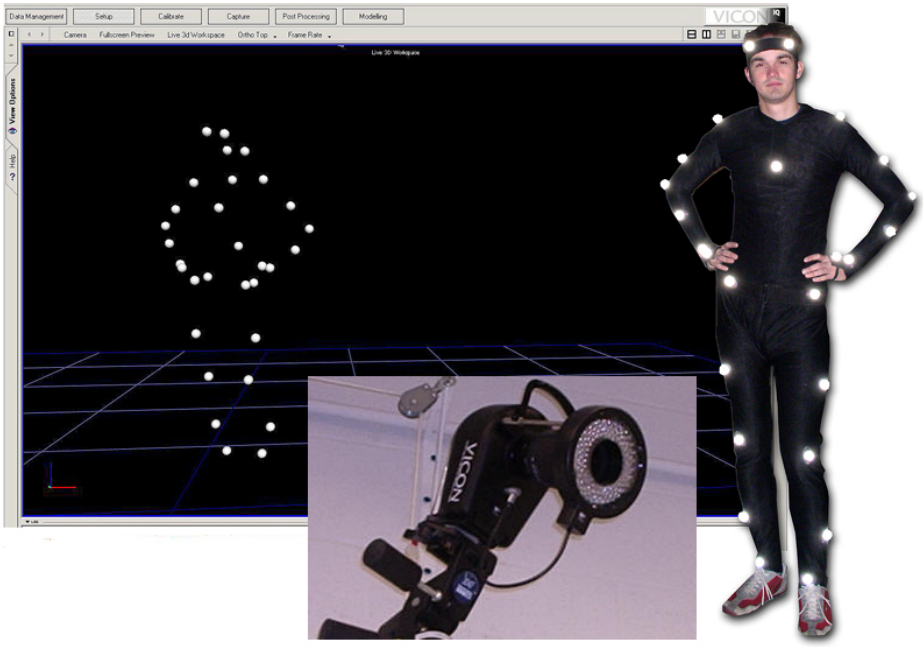


Figura 5.: Strumentazione del sistema stereofotogrammetrico Vicon

| SENSORE | FSR |
|---|---|
| POSIZIONE DEL SENSORE | sotto le scarpe |
| HARDWARE | trasduttori [ *TODO: inserire riferimento appendice* ] progettati come degli interruttori meccanici, sensibili al peso FSR[1] |
| LIMITAZIONI | – non è possibile distinguere le variazioni di peso dovute alla deambulazione da quelle dovute allo spostamento di peso di altra natura.<br>– passi tremolanti riducono l'affidabilità della percezione di un evento che stabilisca l'inizio o la fine di una fase del passo. |

Tabella 4.: Sensori di forza: FSR

| | |
|---|---|
| SENSORE | Giroscopio |
| POSIZIONE DEL SENSORE | coscia, stinco, coscia e stinco |
| HARDWARE | Giroscopi |
| VANTAGGI | – non influenzati dalla forza di gravità;<br>– non influenzati da vibrazioni o scosse dovute alla deambulazione;<br>– meno influenzate (rispetto alle FRS) dal posizionamento sul corpo. |
| LIMITAZIONI | – [ *TODO: verificare* ] non è possibile distinguere le variazioni di peso dovute alla deambulazione da quelle dovute allo spostamento di peso.<br>– passi tremolanti riducono l'affidabilità della percezione di un evento che stabilisca l'inizio o la fine di una fase del passo. |

Tabella 5.: Giroscopi

so il sensore misura l'accelerazione in un riferimento inerziale relativo all'accelerometro stesso.

- **Magnetometri** misurano il campo magnetico. Vi sono più modi per misurare il campo magnetico, quello maggiormente usato detto Fluxgate funziona mediante un meccanismo elettrico (vedi appendice C.3).

I sensori qui presentati sono stati usati per l'analisi della deambulazione in diversi numeri e combinazioni. Una classificazione può essere fatta sulla base della grandezza fisica misurata dal singolo sensore o dall'insieme di sensori scelti. Tale grandezza fisica può essere:

FORZA Il corpo umano esercita una forza (peso) sulla superficie su cui si trova. Un sensore per misurare tale forza, deve essere posizionato, tra il corpo e la superficie. La collocazione naturale di un sensore di forza è sotto la suola delle scarpe. I trasduttori [3] per tale scopo vengono progettati interruttori meccanici dipendenti dal peso o Resistori Sensibili alla forza (FSR).

Metodi di riabilitazione o di aiuto di persone con la sindrome del piede cadente (foot drop) basati sulla Stimolazione Elettrica Funzionale (FES) usavano inizialmente le FSR [6]. Mentre i primi sistemi di questo tipo usavano micro interruttori puramente meccanici, ovvero azionati direttamente dalla pressione del piede, sistemi più recenti sono azionati da valori soglia (threshold) ottenuti mediante uno o più trasduttori FSR posizionati sotto il tallone, il metatarso e l'alluce [7, 6]. Questi metodi a differenza

3 Un trasduttore è un dispositivo che converte un tipo di energia in un altro.

dei primi, sono meno soggetti ad attivazioni dovute a spostamenti di peso non dovuti alla deambulazione. Con un interruttore singolo sotto il tallone si riescono ad ottenere informazioni sugli eventi Tallone_-Alzatoe Contatto_Del_Tallone. Aggiungendo ulteriori sensori sotto il metacarpo o l'alluce si hanno informazioni sugli eventi Piede_A_Terra e Piede_Alzato [6]. Un altra tecnica che è stata usata per misurare la forza è usare suole sensibili alla forza, che ricoprono l'intera suola con una matrice di sensori[8]. L'accuratezza ed affidabilità di tali sistemi dipende soprattutto dai materiali usati.
Un'altra tecnica è basata su un sensore di pressione connesso ad un piccolo tubo, incollato al perimetro esterno della scarpa. In questo modo le variazioni nella forza esercitata sulla suola si manifestano in variazioni pneumatiche, che vengono poi misurate dal sensore di pressione[9].
Sistemi di individuazione di eventi di deambulazione basati su sensori di forza, sono considerati abbastanza standard per essere usati come riferimento per determinare l'accuratezza di metodi novelli per la medesima funzione. Ciò è dovuto al fatto che un sensore posto sotto i piedi sembri essere la scelta più naturale per sistemi di misura della deambulazione.

Danno risultati soddisfacenti per la deambulazione normale 2. Gli svantaggi dei sensori della forza sono l'impossibilità di discernere tra variazioni di carico dovute alla deambulazione da quelle dovute a spostamenti del peso (ad esempio da una gamba all'altra). Una deambulazioni strascicata riduce notevolmente l'affidabilità della individuazione degli eventi della deambulazione [10]. Altri difetti di questi sistemi sono la poca durata, e la limitata applicabilità [11].

ACCELERAZIONE ANGOLARE La maggior parte degli studi che hanno usato un giroscopio hanno optato per lo stesso tipo di sensore unidimensionale nella configurazione a singolo sensore [12, 13, 14] o tre sensori [15]. In tutti i casi i dati sono stati elaborati in differita . Sono state testate innumerevoli combinazioni di posizionamenti dei sensori: coscia [14], stinco [12], coscia e stinco [13] e su entrambe gli stinchi ed una sola coscia [15]. Dal segnale del sensore sono stati calcolati l'angolo dell'articolazione del ginocchio [13], l'angolo dell'articolazione dell'anca mediante l'integrazione della velocità angolare [14]. Uno dei problemi più comuni del giroscopio è il drift (deviazione) causata da una imprecisa calibrazione dello strumento. Un altro approccio è il cosiddetto studio di Wavelet: la stima del Contatto_Del_Tallone, e Tallone_Alzatomediante lo studio analitico del segnale del giroscopio in funzione del tempo [15, 12]. La validazione del sistema è stata fatta su ciascun sistema, confrontando i risultati ottenuti mediante un sistema basato su telecamere noto come Vicon, in condizioni di laboratorio. Un grande vantaggio della analisi del moto in generale mediante il giroscopio, è che questi non subisce l'effetto dell'accelerazione di gravità [16]. Inoltre le vibrazioni dei sensori durante gli esperimenti non influenzano il giroscopio [17]. I giroscopi, rispetto agli FSR, sono meno sensibili al posizionamento sul corpo, un giroscopio posto in qualunque posizione di un segmento del corpo lungo lo stesso piano fornisce con variazioni minime gli stessi valori. Infine nessun tipo di movimento che non riguardi l'asse che percepisce il sensore viene catturato dallo strumento [13].

FORZA E ACCELERAZIONE ANGOLARE Al fine di individuare gli eventi Contatto_-Del_Tallonee Tallone_Alzato, è stato messo a punto un sistema composto da tre FSR per catturare il carico verticale ed un giroscopio per misurare la velocità di rotazione del piede sul piano sagittale (vedi figura

1) [18]. Una volta posizionati gli FSR sotto il tallone, il primo ed il quarto metatarso ed il giroscopio monoassiale sul tallone, il sistema era in grado di individuare le fasi di $Fase\_A\_Terra$ e $Fase\_In\_Aria$ in tempo reale . La validazione è stata fatta in condizioni di laboratorio, anche questa volta con sistemi di cattura del movimento. Il sistema è stato incassato in una suola per poter essere usato con un meccanismo di FES di correzione della ricaduta del piede [19]. I vantaggi principali del sistema qui descritto sono l'affidabilità e la robustezza: riesce a distinguere semplici spostamenti di peso (ad esempio stando fermi, alzandosi o sedendosi) da variazioni di carico dovute alla deambulazione. I difetti del sistema sono collegati ai difetti degli FSR: la loro poca durata.

ACCELEROMETRIA La tecnologia MEMS (Micro-Electro-Mechanical Systems) (vedi appendice C) permette uno sviluppo di accelerometri in miniatura a basso consumo energetico, adatti al monitoraggio della deambulazione [20]. I sensori usati negli esperimenti, misurano l'accelerazione in due [10, 11, 21, 22, 23] o tre dimensioni [16, 24, 25], avendo più sensori monoassiali [16, 21, 22], biassiali [10, 11, 23, 26] o triassiali [25, 24]. Per misurare sia l'accelerazione lineare che rotazionale, i sensori vengono sistemati su coscia [11], stinchi [16, 25, 21], coscia e stinchi [22, 26], coscia, stinchi e bacino [23], piede, coscia, stinchi e bacino [24] o busto [10].
L'elaborazione dei dati provenienti dai suddetti sensori è stata fatta sia in modalità in tempo reale che in differita. L'elaborazione in differita comprende l'analisi temporale dell'accelerazione verticale nel piano sagittale, i cambiamenti da positivo a negativo e viceversa, che sono stati correlati rispettivamente agli eventi $Contatto_Del_Tallone$e $Tallone\_Alzato$[10]. Un altro approccio ha dimostrato che è possibile rilevare gli eventi $Piede\_Alzato$ , $Prima\_Fase\_In\_Aria$ , $Ultima\_Fase\_In\_Aria$ , $Contatto_Del_Tallone$sulla base dei dati della velocità angolare e lineare sul piano sagittale e frontale [21].
Per quanto riguarda i sistemi in tempo reale è stato dimostrato che permettano di rilevare gli eventi $Risposta\_Al\_Carico$ , $Fase\_Intermedia\_A\_Terra$ , $Ultima\_Fase\_A\_Terra$ , $Pre\_Aerea$ [16, 26] oppure $Contatto_Del_Tallone$e $Tallone\_Alzato$[11]. L'uso di accelerometri necessita di elaborazioni ulteriori del segnale per compensare all'influenza della forza di gravità. Il posizionamento dei sensori è un altro aspetto delicato, dovuto al movimento di muscoli durante la deambulazione. Un vantaggio degli accelerometri, è che l'evento $Tallone_Alzato$è individuabile anche in individui che non hanno un contatto con il tallone molto definito [16] o con deambulazione strascicata [10].

ACCELEROMETRIA E ACCELERAZIONE ANGOLARE Il metodo di misurazione può essere basato su un modello bidimensionale su un piano sagittale [17, 27, 28, 29, 30, 31] oppure tridimensionale [32, 33]. Per modelli bidimensionali il gruppo di sensori può essere composto da due sensori inerziali unidimensionali ed un giroscopio bidimensionale [17] oppure da un unico sensore inerziale bidimensionale che misuri le componenti tangenziali e radiali dell'accelerazione [27]. I sistemi tridimensionali usano di solito un giroscopio tridimensionale ed un sensore inerziale tridimensionale [32, 33]. Le unità di sensori sono montati su piedi [28, 33], stinco e coscia [17, 27, 30], o piedi, stinchi e cosce [31, 32]. I dati sono stati elaborati primariamente in differita ed hanno fornito informazioni sull'accelerazione di segmenti di arti, velocità di giunture ed eventi quali $Contatto - Del - tallone$, $Piede\_Alzato$ e $Piede\_A\_Terra$ . La validazione di questi sistemi è stata fatta su soggetti sani [17, 27, 28, 31] e

con problemi di deambulazione. I dati ottenuti sono stati confrontati con quelli prodotti da un sistema ripresa del movimento : WATSMART$^{TM}$ [32], Vicon$^{TM}$ [17], interruttori a piede [28, 29, 30].

ACELEROMETRIA, ACCELERAZIONE ANGOLARE E CAMPO MAGNETICO La misurazione del campo magnetico terrestre mediante un magnetometro fornisce il campo gravitazionale terrestre ed una misura di riferimento per l'orientamento del corpo. Al contrario dell'acelerometria, il campo magnetico non è influenzato dai movimenti del corpo. Sono state prodotte unità contenenti sensori sia uniassiali [34] che biassiali [35] in combinazione con un magnetometro. I sensori sono stati posizionati al piede e stinco di una gamba oppure di entrambe le gambe. I sistemi sono in grado di determinare angoli tra diverse articolazioni in tre dimensioni mediante l'analisi dei dati in differita [35] o mediante l'analisi tempo reale [34]. Tali sistemi hanno permesso di individuare cinque eventi della deambulazione: Risposta_Al_Carico , Fase_Intermedia_A_Terra , Ultima_Fase_A_Terra , Pre_Aerea e Fase_In_Aria con l'uso di misure dell'accelerazione e della sua derivata prima [34]. Anche gli eventi Contatto_Del_Tallonee Piede_Alzato sono stati individuati in corrispondenza di picchi nella rotazione lungo il piano sagittale dello stinco. Un modello di un doppio pendolo può essere usato per calcolare la lunghezza di un passo [36].

INCLINOMETRIA Un sensore di inclinazione può essere usato per determinare l'inclinazione di una parte del corpo. Consiste di un elemento inerziale che percepisce la forza di gravità, come un liquido o un sistema costituito da una massa ed una molla. Quando il sensore è inclinato, la forza di gravità causa uno spostamento dell'elemento inerziale relativo al sistema del sensore. Tale spostamento viene registrato in una resistenza o capacitanza. Questi sistemi non sono abbastanza affidabili perché non distinguono la deambulazione da altri tipi di spostamenti del piede [19]. Inoltre rispondono ad accelerazioni, il che produce degli errori grossolani nei loro dati.

## 4.2 METODI

La grande sfida del rilevamento della deambulazione è di individuare gli eventi della deambulazione mentre la persona sta camminando ovvero in tempo reale . Tradizionalmente il problema è stato affrontato con un insieme di regole basate su valori di soglia sulle emissioni, principalmente di interruttori da piede, e da tutti gli altri sistemi di sensori descritti in 4.1.3. Tutti gli algoritmi pubblicati, consistono di gruppi di euristiche che cercano di identificare alcune caratteristiche dei parametri della deambulazione. Le regole sono state applicate con i seguenti approcci: analisi funzionale di dati non elaborati [15, 37, 38], di dati derivati [19], tecniche di apprendimento induttivo [8, 11, 16, 39] oppure macchine a stati finiti [19, 40].

### 4.2.1 *Analisi*

L'analisi funzionale comprende metodi matematici per disegnare curve che permettono di estrarre caratteristiche corrispondono a determinate fasi della deambulazione. Le derivate prime e seconde dei dati dell'accelerazione verticale e orizzontale dal piede permettono di determinare gli eventi Contatto_Del_-

Tallonee Piede_Alzato soltanto usando delle regole basate su valori di soglia, con un'elaborazione in tempo reale. [37].

Parte II

# LAVORO SVOLTO

# 5 MATERIALI E METODI

## 5.1 INTRODUZIONE

Il lavoro complessivo, può essere suddiviso in tre fasi principali:

1. **creazione ed addestramento di un modello**: scelta di un modello consono al problema e ottimizzazione dei relativi parametri,
2. **implementazione**: stesura applicativo per Android che implementi il modello,
3. **applicazione e test**: abbiamo correlato un parametro spaziale del cammino (la distanza percorsa) a partire da quelli temporali ottenuti con la segmentazione (cadenza e velocità) e successivamente verificandone la correttezza con misurazioni dirette dei parametri spaziali (misurazione della distanza fatta con il GPS).

## 5.2 CREAZIONE ED ADDESTRAMENTO DEL MODELLO

Il sistema adattivo scelto sono le HMM. La scelta di queste ultime è giustificato sia dagli studi che il gruppo di ricerca capitanato dal prof. Sabatini, al centro di Bio-Robotica di Sant'Anna[1] sta portando avanti sulla classificazione automatica del movimento umano, sia dal fatto che lavori come *A Hidden Markov Model-based stride segmentation technique applied to equine inertial sensor trunk movement data* di Pfau [41] abbiano dimostrato le loro ottime potenzialità. Inoltre gli interessi futuri di ricerca scientifica del gruppo saranno orientati sia verso la stima dei parametri di un'attività nota sia verso la classificazione di diverse attività (se si considerano modelli di HMM gerarchici [42]). Entrambi i compiti possono essere portati a termine dalle HMM.

### 5.2.1 *Dati*

Sono stati scelti 6 soggetti sani (con deambulazione normale)[2] e sono stati sottoposti a 5 sessioni di cammino e 5 sessioni di corsa. Le prove sono state compiute su threadmill su un intervallo di velocità [3-7]Km/h per il cammino e [8-12]Km/h per la corsa. Ciascuna sessione è stata della durata di 2 minuti. La prima sessione alla velocità di 3Km/h e con un incremento di 1Km/h in ciascuna sessione successiva.

Le sessioni sono state tutte eseguite posizionando una IMU C sul collo del piede dei soggetti. Il tipo di sensore usato è un giroscopio monoassiale con asse di sensibilità orientato sul piano mediale-laterale (vedi figura 1).
I segnali sono stati raccolti dal sensore ad una frequenza di 100 Hz e filtrati

1 http://www-arts.sssup.it/
2 non patologico

| | |
|---|---|
| **Attività 1** | cammino {3,4,5,6,7}Km/h |
| **Soggetti** | 6 |
| **Attività 2** | corsa {8,9,10,11,12}Km/h |
| **Soggetti** | 5 |
| **Durata** | 2 minuti per attività |
| **Strumenti 1** | IMU, Vicon, Treadmill |
| **Luogo** | Laboratorio |

Tabella 8.: Tabella riassuntiva della raccolti dati

passa-basso[3] a 15Hz mediante un filtro forward-backward[4] di Butterworth[5]. Dai dati raccolti, quelli considerati per lo studio sono quelli compresi nell'intervallo dai 50 ai 110 secondi.
I dati acquisiti con la IMU sono stati sincronizzati con i dati acquisiti mediante stereofotogrammetria. A tale scopo è stato impiegato un sistema ottico di analisi del moto commerciale, Vicon [43], composto di 4 telecamere posizionate intorno al treadmill, in modo da poter tracciare il movimento di 5 marker retroriflettenti[6]. I marker sono dei bulbi sferici in plastica di circa 14mm di diametro, attaccati ad una superficie quadrata della stessa dimensione, alle quali viene applicato del nastro biadesivo, mediante il quale vengono incollate sulla parte del corpo che vuole studiare. I marker sono stati posizionati sull'arto inferiore sinistro: sul tallone, sulla lato esterno della caviglia, sulla giuntura metatarso-falange dell'alluce, sulla coscia e sulla schiena, tra la zona lombare e sacrale. Il sistema Vicon, con il software [ *TODO: Vicon Tracker 1.2* ] ha tracciato la traiettoria in 3D, con un campionamento a 50Hz ed un accuratezza di $\pm 10$ mm. e ricostruito tutta la sequenza di movimenti in 3D degli esperimenti, visualizzandoli a video. Dai dati ottenuti con il sistema Vicon sono stati generati dei segnali di riferimento per le fasi del cammino mediante la procedura descritta in [18].

Il grafico del segnale del giroscopio posizionato sul piede ha una struttura abbastanza definita e ripetitiva (vedi grafico 6). Assumendo di cominciare dal piede in aria, qualche istante prima che stia per atterrare con il tallone, si ha il grafico nel punto di coordinata $(0, 0)$. Da qui questo in circa 30 secondi raggiunge un picco negativo di circa -200°/secondi($min_1$). Da questo vertice risale a V (con la stessa pendenza ma di segno positivo) a 0°/sec. Successivamente per circa 50 secondi il piede rimane a velocità angolare 0°/secondi, dopo di che la curva precipita verso il suo minimo, tracciando una curva concava. Dopo 50 secondi circa di precipizio questa giunge a -450°/sec ($min$). Appena raggiunto il minimo, risale quasi verticalmente (in meno di 50 secondi) al valore massimo, circa 450 ($max$). Dal massimo riscende allo 0°/sec in due tratti, di cui il secondo è più lungo e di pendenza maggiore, simile a quello con cui la curva è risalita al picco massimo, ma decrescente. Da questo punto lo stesso scenario si ripete ad

3 Un filtro passa-banda, è uno strumento elettronico che lascia passare frequenze in un intervallo e blocca il passaggio o di frequenze (o le attenua) fuori dall'intervallo. Un passa-basso lascia passare le frequenze basse, e smorza quelle a frequenze alte.
4 Il filtro viene passato in entrambe le direzioni per avere un segnale simmetrico
5 Detto anche filtro a magnitudine massimamente piatta nella banda che passa, in modo da avere meno disturbo possibile.
6 Dispositivo o superficie che riflette la luce alla sua sorgente minimizzando la dispersione di luce

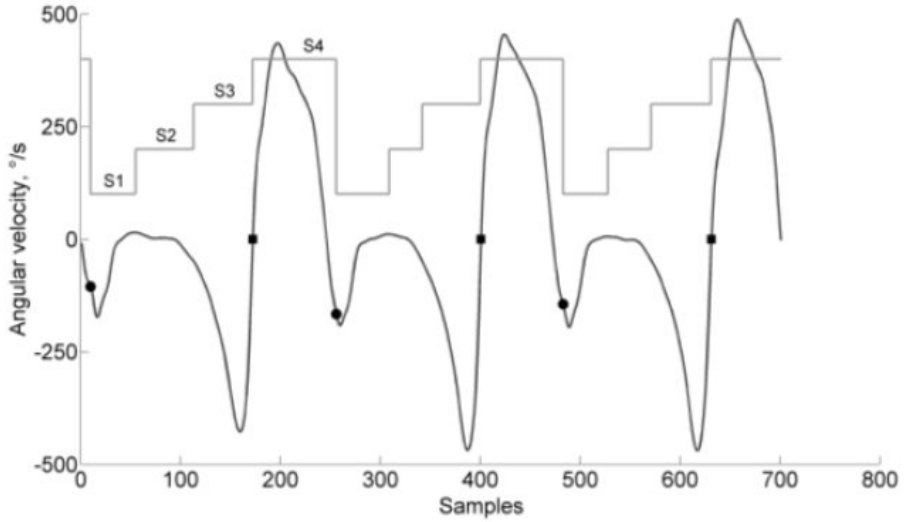


Figura 6.: Grafico del segnale di un giroscopio posizionato sul piede

| Stato | inizio | fine |
|---|---|---|
| $S_1$ | $t_{HS}$ | $t_{FF}$ |
| $S_2$ | $t_{FF}$ | $t_{HO}$ |
| $S_3$ | $t_{HO}$ | $t_{FO}$ |
| $S_4$ | $t_{FO}$ | $t_{HS}$ |

Tabella 9.: Le quattro fasi della deambulazione e gli eventi (tempo iniziale e finale) che li definiscono.

ogni passo, se non si modificano le condizioni di esecuzione del movimento. In un ciclo di cammino possono essere identificate fino a 8 fasi (vedi capitolo 3). Per tenere basso il numero di parametri è stato deciso di accorpare le fasi riducendole a 4 (vedi tabella 9). La prima fase $S_1$ è quella nell'intervallo fra gli eventi $t_{HS}$ a $T_{FF}$, la seconda $S_2$ è quella compresa fra gli eventi $t_{FF}$ a $t_{HO}$, la terza $S_3$ fra $t_{HO}$ a $t_{FO}$ e l'ultima $S_1$ è quella che chiude il ciclo $t_{FO}$ a $t_{HS}$.

Per semplicità si può indicare uno stato $S_i$ con l'evento da cui ha inizio, ad esempio con lo stato HS indico lo stato $S_1$.

Gli eventi ($t_{HS}$, $t_{FF}$, $t_{HO}$, $t_{FO}$ ) sono determinati mediante soglie, sulla velocità angolare $\omega_k$, ricavate da studi pregressi [28] come mostra la Tabella 10, dove $\widetilde{\omega_k}$ è il valore non filtrato del giroscopio.

Si assume che i segnali del giroscopio siano modellati mediante una HMM:

$$\mathrm{Deamb\_HMM} =< N, M, \mathbf{A}, \mathbf{B}, \pi > \tag{5.1}$$

dove:

1. $N = 4$ è la cardinalità dell'insieme degli stati $S = \{HS, FF, HO, FO\}$,
2. $M = |V|$ è la cardinalità dell'insieme finito dei valori di osservazione $\omega$ (°/sec) del giroscopio,
3. **A** è la matrice di transizione (vedi appendice A),
4. **B** è la matrice di probabilità di emissione delle osservazioni $\omega$ per ciascuno stato. Ad ogni stato è stata associata una funzione di densità di probabilità gaussiana univariata.

| Evento | regola | |
|---|---|---|
| $t_{HS}$ | $\omega_k \leqslant 0$ e $min_1 \leqslant \omega_k$ | $\max_{\forall k} \lvert\widetilde{\omega_k} - \omega_k\rvert$ |
| $t_{FF}$ | $\lvert\omega_k\rvert \geqslant ??°/sec$ | |
| $t_{HO}$ | $\lvert\omega_k\rvert \geqslant 50°/sec$ | |
| $t_{FO}$ | $\omega_k = 0$ | se $\omega$ da negativo è diventato positivo e cresce fino al suo massimo. |

Tabella 10.: Regole mediante le quali vengono ricavati gli eventi che delimitano le 4 fasi del cammino.

5. $\pi$ è il vettore di probabilità a priori.

A questo punto sono state addestrate le HMM. Come prima operazione, è stato etichettato un sottoinsieme dei dati: ad una sequenza $\Omega$ di dati è stato associato un vettore $Y$ di etichette. Questo è stato fatto con l'uso di un sistema stereofotogrammetrico (sistema di telecamere Vicon).

Dato che nella deambulazione normale le uniche transizioni tra stati sono quelle tra stati contigui o quelle che riportano nello stesso stato, il modello migliore il tipo di HMM scelto è il modello left-right (vedi appendice A.3.2). Ciò significa che la matrice di transizione gode della proprietà di avere transizioni solo verso stati successivi (da qui il nome da sinistra - a destra):

$$a_{ij} > 0 \quad \text{sse} \quad j = i \quad \text{oppure} \quad j = i+1 \quad \text{oppure} \quad (i = N \quad \text{e} \quad j = 1) \tag{5.2}$$

Una volta deciso il tipo di HMM da adottare, si devono stimare i parametri del modello. La stima (dei parametri del modello) della massima verosimiglianza (Maximum Likelihood Estimation - MLE) $\ell(\Omega, \mathtt{Deamb\_HMM})$ non è un problema convesso dato. Il problema dei massimi locali viene aggirato con un'attenta inizializzazione dei parametri.

Assumendo che possa avvenire solo una transizione tra stati contigui, in ciascun ciclo di deambulazione possono essere fatte delle stime approssimative dei paramtri della HMM:

$$\begin{aligned}
\pi_i &= N_i / N_{tot} \\
a_{ij} &= C/N_i \quad \text{dove} \quad j = (i+1)\%Q \\
\mu_i &= \frac{1}{N_i} \sum_{t=1}^{T} \Omega(t) \\
\sigma_i &= \sqrt{\frac{1}{N_i - 1} \sum_{t=1}^{T} (\Omega(t) - \mu_i)^2}
\end{aligned} \tag{5.3}$$

dove $N_i$ è il numero di valori etichettati con l'etichetta $i$-esima nel vettore di etichette $Y$ e $C$ è il numero di cicli di deambulazione nel training set.
[ *ERR: VERRI: POCJO CHIARO - I parametri di emissione ( $\mu$ e $\sigma$) per ciascuno stato $i$ sono stimati calcolando delle medie e deviazioni standard empiriche dagli output osservabili dei quali si conosce lo stato che li ha emessi.*
] La validazione è stata fatta con l'approccio leave-one-subject-out cross-validation (LOOCV). Il metodo consiste nell'addestrare un modello su P-1 ($P = 6$) soggetti e testarne la validità su 1 soggetto. Ripetendo P volte la procedura per ciascuno dei soggetti alla fine i risultati vengono combinati.[ *TODO: spiegare come più in dettaglio* ]

| **A** | HS | FF | HO | TO |
|---|---|---|---|---|
| HS | 0.985 | 0.015 | 0.0 | 0.0 |
| FF | 0.0 | 0.998 | 0.002 | 0.0 |
| HO | 0.0 | 0.0 | 0.991 | 0.009 |
| FO | 0.007 | 0.0 | 0.0 | 0.993 |

Tabella 11.: Matrice di Transizione

| **B** | $\mu[°/\text{SEC}]$ | $\sigma^2[°/\text{SEC}]$ |
|---|---|---|
| HS | −52.2 | 12711.8 |
| FF | −11.8 | 898.7 |
| HO | −180.1 | 35806.8 |
| FO | 233 | 14980.3 |

Tabella 12.: Matrice di Emissione

Tutta la prima fase della creazione addestramento e validation incrociata del modello è stata fatta off-line, mediante un Mathworks MATLAB (R2008a) ed l'HMM toolbox [44]. Nella fase di test, viene usato l'algoritmo di decodifica di Viterbi 4 sui dati del giroscopio, per trovare la sequenza di stati che con maggior verosimiglianza ha prodotto le osservazioni (sequenze di valori del giroscopio). Il modello left-right può portare a considerare cicli di deambulazione aggiuntivi (insertions), o meno di quelli effettivi (deletions). Il problema delle inserzioni viene risolto mediante l'applicazione di un'euristica: per i presupposti del tipo di deambulazione considerata (velocità e pendenza del terreno).

I parametri del modello stimati da una delle esecuzioni della validazione incrociata è il seguente: Dati: HMM

Le seguenti sono un esempio di osservazioni: 12400 dati Giroscopio

| | $\pi$ |
|---|---|
| HS | 0.180 |
| FF | 0.278 |
| HO | 0.279 |
| FO | 0.264 |

Tabella 13.: Distribuzione di probabilità iniziale

| time | O |
|---|---|
| 1 | 0.37694 |
| 2 | 0.37694 |
| 3 | 0.37694 |
| 4 | 0.37694 |
| 5 | −1.9058 |
| ⋮ | ⋮ |
| 24800 | −16.831 |

Tabella 14.: Osservazioni Giroscopio: velocità angolari

# 6 APPLICATIVO ANDROID

## 6.1 INTRODUZIONE

La seconda parte del lavoro, è relativa all'implementazione di algoritmi di segmentazione online su uno Smartphone Android. L'applicativo in oggetto, sulla base di un modello HMM addestrato offline (vedi sezione 5.2), impiega una versione dell'algoritmo di Viterbi [45] per elaborare a run-time i dati giroscopici acquisiti dalla IMU (e trasferiti via Bluetooth), effettuando la segmentazione e consentendo output di tipo grafico (colorando in modo differente la traccia del segnale giroscopico in funzione della fase del cammino), e di data logging.

## 6.2 METODOLOGIA DI PROGRAMMAZIONE USATA: AGILE

Per l'implementazione dell'algoritmo di Viterbi e le funzioni utilizzate nella fase di analisi del segnale, è stata utilizzata una tecnica nota come Agile Programming. Una tecnica che permette lo sviluppo di programmi in cicli di lavoro iterativi ed incrementali in termini di funzionalità.

1. Scrittura di uno Unit Test[1]. Lo Unit Test istanza l'oggetto dalla classe da testare, ed invoca il metodo con valori per i quali è noto il comportamento (teorico) del metodo da testare.
   Il test viene eseguito, per la prima volta, prima di aver implementato il metodo da testare, ed ovviamente fallisce[2]. Il passo successivo è di implementare il minimo indispensabile per far funzionare il test. Se il test viene superato dal codice scritto, viene scritto un altro test, ed un altro pezzo di codice, iterando la procedura finché non si ha tutto il programma desiderato.
   A titolo di esempio di seguito verrà illustrato lo sviluppo della classe che gestisce le operazioni di tipo statistico:

```
1 // la classe da costruire
2 public class StatisticsOperations{...}
3 // l'insieme di test
4 public class StatisticsOperationsTest extends TestCase {...}
```

Codice 6.1: Sviluppo di una classe mediante la tecnica di programmazione Agile

   Le operazioni necessarie per lavorare sulle HMM erano:
   - verificare la validità di valori di probabilità
   - verificare la completezza di un insieme di alternative probabilistiche
   - calcolare la Probabilità di un dato assumendo quanto estratto da una distribuzione Normale ($\mathcal{N}(x, \mu, \sigma)$) con media $\mu$ e varianza $\sigma^2$ note.

1 Per Unit si intende la più piccola parte di un programma che può essere testata, nei linguaggi orientati ad oggetti un metodo costituisce un'unità. Per Unit testing si intende la verifica del funzionamento di singole Unit.

2 Eclipe IDE ™utilizza il framework JUnit che semplifica la gestione degli Unit Test.

Creazione delle le firme dei metodi

```
1  public class StatisticsOperations{
2      public static boolean areCompleteProbabilisticAlternatives(
           ArrayList<Object> alternatives) {...}
3      public static boolean isValidProbabilityValue(double value)
           {...}
4      public static double gaussian(double x, double mu, double
           sigma_square) {...}
5  }
```

Codice 6.2: Firme dei metodi da implementare

2. Creazione di un test vuoto (destinato a fallire).
3. Implementazione dei test per cinque casi rappresentativi dei degli intervalli esterni ed interni a quello di riferimento (0,1).

```
1  public class StatisticsOperationsTest extends TestCase {
2      ... // metodi setUp e tearDown che per ora non uso
3
4      public void testIsMinusOneACorrectProbabilityValue() {
5          boolean expected = false;
6          boolean actual = StatisticsOperations.
               isValidProbabilityValue(-1);
7          assertEquals(expected, actual);
8      }
9      public void testIsZeroACorrectProbabilityValue() {
10         boolean expected = true;
11         boolean actual = StatisticsOperations.
               isValidProbabilityValue(0);
12         assertEquals(expected, actual);
13     }
14     public void testIsPointFiveACorrectProbabilityValue() {
15         boolean expected = true;
16         boolean actual = StatisticsOperations.
               isValidProbabilityValue(.5);
17         assertEquals(expected, actual);
18     }
19
20     public void testIsOneACorrectProbabilityValue() {
21         boolean expected = true;
22         boolean actual = StatisticsOperations.
               isValidProbabilityValue(1);
23         assertEquals(expected, actual);
24     }
25
26     public void testIsTwoACorrectProbabilityValue() {
27         boolean expected = false;
28         boolean actual = StatisticsOperations.
               isValidProbabilityValue(2);
29         assertEquals(expected, actual);
30     }
31
32     public void testIsRandNumACorrectProbabilityValue() {
```

```
33          boolean expected = false;
34          double randomProbabilityValue = Math.random();
35          if (randomProbabilityValue >= 0 &&
                randomProbabilityValue <= 1){
36              expected = true;
37          }
38          boolean actual = StatisticsOperations.
                isValidProbabilityValue(randomProbabilityValue);
39          assertEquals(expected, actual);
40      }
41
42  }
```

Codice 6.3: Casi di test di contorno

4. Sviluppo del corpo del metodo che si sta testando. Questa fase (la più importante) è notevolmente agevolata dalle precedenti. Il programmatore ha chiare la funzione da implementare e le relative problematiche.

```
1  public class StatisticsOperations{
2      public static boolean areCompleteProbabilisticAlternatives(
           ArrayList<Object> alternatives) {...}
3      public static boolean isValidProbabilityValue(double value)
           {
4          if (value >= 0 && value <= 1)
5              return true;
6          else
7              return false;
8  }
9      public static double gaussian(double x, double mu, double
           sigma_square) {...}
10 }
```

Codice 6.4: Implementazione dei metodi da testare

La esecuzione del test sul metodo riscritto porta ad uno dei seguenti esiti:

1. superamento di tutti i test, quindi lo sviluppo del metodo è completato,
2. fallimento di almeno un test, il metodo deve essere corretto ed i test devono essere nuovamente eseguiti.

## 6.3 ORGANIZZAZIONE DEL PROGRAMMA

### 6.3.1 *Android Manifest*

AndroidManifest.xml (vedi Appendice D.1.6) è il file che viene usato come punto di partenza e di riferimento dal compilatore Dalvik (vedi Appendice D.1.3).

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="imu.Interface" android:versCode="1" android:versName="1.0">
```

Codice 6.5: AndroidManifest testata

Il nome del Java package, serve come identificativo unico per l'applicativo.

```
3     <uses-permission android:name="android.permission.BLUETOOTH" />
4     <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
5     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
          />
6     <uses-sdk android:minSdkVersion="8" />
7     ...
```

Codice 6.6: AndroidManifest permessi

Dato che il nucleo di Android-OS è Linux (vedi Appendice D) usa una politica di Access Control[3] molto simile. Tutti i servizi che non sono forniti dal contesto di questo particolare applicativo, quindi i servizi che fornisce l'applicativo stesso, e quelli forniti dal sistema operativo per tutti gli applicativi, possono essere utilizzati solo se l'utente da il permesso di utilizzarli. Per questo motivo l'applicativo deve elencare tutte le richieste di permessi di cui necessita. L'elenco deve essere fatto nella porzione iniziale dell'`AndroidManifest` Dichiarazione dei permessi che l'applicativo deve avere dall'utente per accedere ad alcune parti dell'API[4] di Android e per interagire con altri applicativi. La riga 4 chiede il permesso di fare la discovery (ricerca) di dispositivi e di associarsi a dispositivi trovati, la riga precedente invece il permesso di connettersi a dispositivi associati [5]. L'ultimo permesso serve per poter scrivere su un dispositivo d'immagazzinamento esterno, questo ci serve per poter salvare ad esempio l'output dell

```
8     <application android:icon="@drawable/spinningtop02"
9        android:label="@string/app_nameicon"
10       android:name="imu.objects.MailBoxes">
```

Codice 6.7: AndroidManifest dichiarazione dell'attività principale

Vengono dichiarati il nome dell'applicativo, l'icona, ed un'etichetta. In questo punto si dichiara anche il nome dell'oggetto che può essere visibile a livello globale di applicativo. Infatti l'oggetto `MailBoxes` che è un vettore di `MailBox` (vedi 6.4.4) che sono a loro volta utilizzati per lo scambio di dati tra thread.

```
11       <activity android:name=".IMUinterface" android:configChanges="
             orientation">
12          <intent-filter>
13             <action android:name="android.intent.action.MAIN" />
14             <category android:name="android.intent.category.LAUNCHER" />
15          </intent-filter>
16       </activity>
17       <activity android:name=".Saveactivity" android:label="@string/
             save_name" />
18       <activity android:name=".SensSetupactivity" android:label="@string/
             sens_name" />
19       <activity android:name=".Provaactivity" android:label="@string/
             prova_name" />
```

3 Accesso controllato dell'utente di un filesystem o di un servizio

4 Application Programming Interface: interfaccia ad un codice che viene messo a disposizione come servizio in modo che terzi possano usarlo per sviluppare altro software, avvolte anche in altri linguaggi. In dettaglio una API nei linguaggi orientati ad oggetti, sono concepiti come l'insieme di tutti i metodi pubblici che le classi pubbliche offrono.

5 L'associazione è la prima connessione, nella quale vi è una forma di identificazione e associazione dei due dispositivi. Per connessione invece si intendono le connessioni successive alla prima, in cui un dispositivo conosce già l'altro e, più rapidamente, possono iniziare una conversazione.

```
20        <activity android:name=".EmbeddedSensDataPlottingvActivity"
21           android:label="@string/prova_name" android:configChanges="
                 orientation"
22           android:theme="@android:style/Theme.NoTitleBar.Fullscreen" />
23        <activity android:name=".TemporalDataPlottingvActivity"
24           android:configChanges="orientation" android:theme="@android:style/
                 Theme.NoTitleBar.Fullscreen" />
```

Codice 6.8: AndroidManifes elenco di tutte le attività

Sezione di `AndroidManifest` in cui si elencano i componenti fondamentali (vedi Appendice D.2) utilizzati nell'applicativo. Il nostro applicativo, che ha una struttura molto semplice, ed utilizza solo `Activity` ed `Intent`. Il l'Activity principale (Main Activity) è `IMUInterface`.
La dichiarazione di una Activity è composta da un nome e da una classe Java. Inoltre sono dichiarate altre impostazioni iniziali: se sono gestiti o meno i 2 orientamenti dello schermo, se lo schermo deve essere visualizzato senza la barra dei titoli in testa allo schermo.

### 6.3.2 *Codice sorgente*

Tutto il codice sorgente (Java) deve essere contenuto nella cartella *src* del progetto. Abbiamo strutturato il codice in 5 package (cartelle di Java):

- **activities**: le Activity, corrispondono indicativamente ciascuna ad una schermata sullo Smartphone,
- **objects** : dati, modelli o contenitori su cui fare operazioni,
- **services**: operazioni fondamentali dell'applicativo, che qui coincidono agli algoritmi che si applicano alla HMM.
- **tests**: test prodotti dall'utilizzo della metodologia Agile (vedi 6.2),
- **util**: operazioni di supporto al resto del codice.

### 6.3.3 *Risorse*

Le risorse sono costituite da tutti i file che fanno parte dell'applicativo, ma non sono codice sorgente (vedi Appendice D.1.6) vale a dire file xml, immagini, testo ecc. La sezione più importante delle risorse è la cartella **layout**. Questa contiene una serie di file xml, che in teoria, per ciascuna Activity dovrebbe essere creato un file xml da mettere nella cartella layout, con lo stesso nome della Activity.

#### *Layout*

Un file di layout (impaginazione) è un documento xml in cui viene descritta la struttura dell'interfaccia utente di una Activity. Layout deve essere composto di elementi grafici predefiniti in Android oppure deve essere di tipo `Veiw` o `ViewGoup`[6]. Ogni file di layout deve contenere esattamente un elemento radice. Ogni elemento grafico ha un identificativo unico, con cui può essere richiamato dal codice sorgente, tramite l'indicizzazione automatica. Eclipse ™permette di definire un layout sia in modalità xml che in modalità grafica (WYSIWYG[7]). Ad esempio il layout della activity principale (ImuInterface) è il seguente

6 Sia la View che la ViewGroup appartengono al package android.view e sono componenti importante dell'interfaccia utente

7 What You See Is What You Get

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="fill_parent"
   android:background="#111111" android:layout_height="fill_parent">
```

Codice 6.9: LinearLayout:impaginazione della schermata principale

Un `LinearLayout` è un tipo di layout che può contenere altri layout, in sostanza serve ad organizzare oggetti. Sono impostate alcune delle proprietà (orientamento, larghezza ecc) dell'oggetto.

```
   <TextView android:layout_height="wrap_content"
      android:text="@string/ImuActivity_title" android:textStyle="bold"
      android:id="@+id/AppName" android:layout_width="320dip">
   </TextView>
```

Codice 6.10: TextView: oggetto contenente testo

Questo è un oggetto che contiene testo non modificabile dall'utente. Qui viene visualizzato il titolo della schermata, contenuto nella variabile di stringa `"@string/ImuActivity_title"`.

```
   <EditText android:id="@+id/editText1"
      android:layout_height="wrap_content"
      android:layout_width="match_parent">
   </EditText>
   <EditText android:id="@+id/editText2"
      android:layout_height="wrap_content"
      android:layout_width="match_parent">
   </EditText>
   ....
```

Codice 6.11: EditText:oggetto contenente campi di testo modificabili

Vi sono subito sotto il titolo della schermata, quattro campi di testo (di cui sono riportate solo 2 a titolo illustrativo).

```
   <LinearLayout android:background="#111111"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:layout_width="fill_parent"
      android:orientation="horizontal">
      ...
      <Button android:id="@+id/button2" android:layout_height="wrap_content"
         android:text="Start" android:gravity="center_vertical|
             center_horizontal"
         android:layout_width="55dip">
      </Button>
      ...
   </LinearLayout>
   ...
```

Codice 6.12: Esempio di LinearLayout contenente un Button

La parte seguente della schermata, percorrendola dall'alto verso il basso, è un altro `LineraLayout` contenente altri campi di testo e bottoni.
La schermata risultante dal layout è quella dell'immagine 7.

### *File grafici*

I file grafici sono distribuiti su tre cartelle **drawable-hdpi**, **drawable-ldpi**,**drawable-mdpi** in cui salvare formati a definizioni diverse delle stesse immagini per diversi dispositivi.


Figura 7.: Schermata della Activity iniziale dell'applicativo

#### *Valori*

Il file più importante in questa cartella è `strings.xml` in cui vengono salvate tutte le stringhe da visualizzare sulle varie schermate. L'utilità maggiore di mantenere le stringhe in un file unico, è l'internazionalizzazione dell'applicativo. Con il tipo di mercato mondiale che hanno gli applicativi Android, avere il testo tradotto in più lingue è necessario.

#### *Altri File*

Qualunque tipi di file, che non sia compatibile con quelli menzionati precedentemente, vanno nella cartella `raw`. Ad esempio, in fase di sviluppo, per testare l'algoritmo di Viterbi, abbiamo usato un file contenente dati di un giroscopio, acquisito nella fase precedente del lavoro.

### 6.3.4 *Librerie*

Le librerie usate sono quelle di Android 2.2 (vedi Appendice D.1.2).

## 6.4 FUNZIONAMENTO

### 6.4.1 *Interfaccia*

La linea guida per lo sviluppo della UI (Interfaccia Utente) è stata quella della massima ergonomicità, in linea con la filosofia di sviluppo di interfacce utente di Android.
La navigazione tra le schermate dell'applicativo è guidata da widget che sono oggetti grafici della UI di un programma, che hanno lo scopo di facilitare all'utente l'interazione con il programma.
All'avvio del programma, viene visualizzato sullo schermo del dispositivo, un menu dal quale è possibile

### 6.4.2 *Comunicazione Bluetooth*

`BluetoothAdapter` Rappresenta l'adattatore Bluetooth del dispositivo locale e permette di fare le operazioni più importanti sullo stesso:

- Ricerca (discovery) di dispositivi,
- Query su una lista di dispositivi connessi,
- istanziazione di un dispositivo Bluetooth (`BluetoothDevice`) usando un indirizzo MAC noto,
- creazione di una server-socket (`BluetoothServerSocket`) per accettare richieste di connessione da altri dispositivi Bluetooth.

### 6.4.3 *Gestione della connessione*

`ConnectThread` è un `\Runnable` che gestisce le connessioni mediante thread. Dati un dispositivo `BluetoothDevice`, un adattatore `BluetoothAdapter` ed il numero identificativo del dispositivo, se questo non è connesso, viene creata una socket RFComm con l'UUID pronta per una comunicazione. All'avvio del thread (con il metodo `run()`) avviene la connessione. Il thread lancia a sua volta un thread al quale delega la gestione delle socket create (`manageConnectedSocket`).

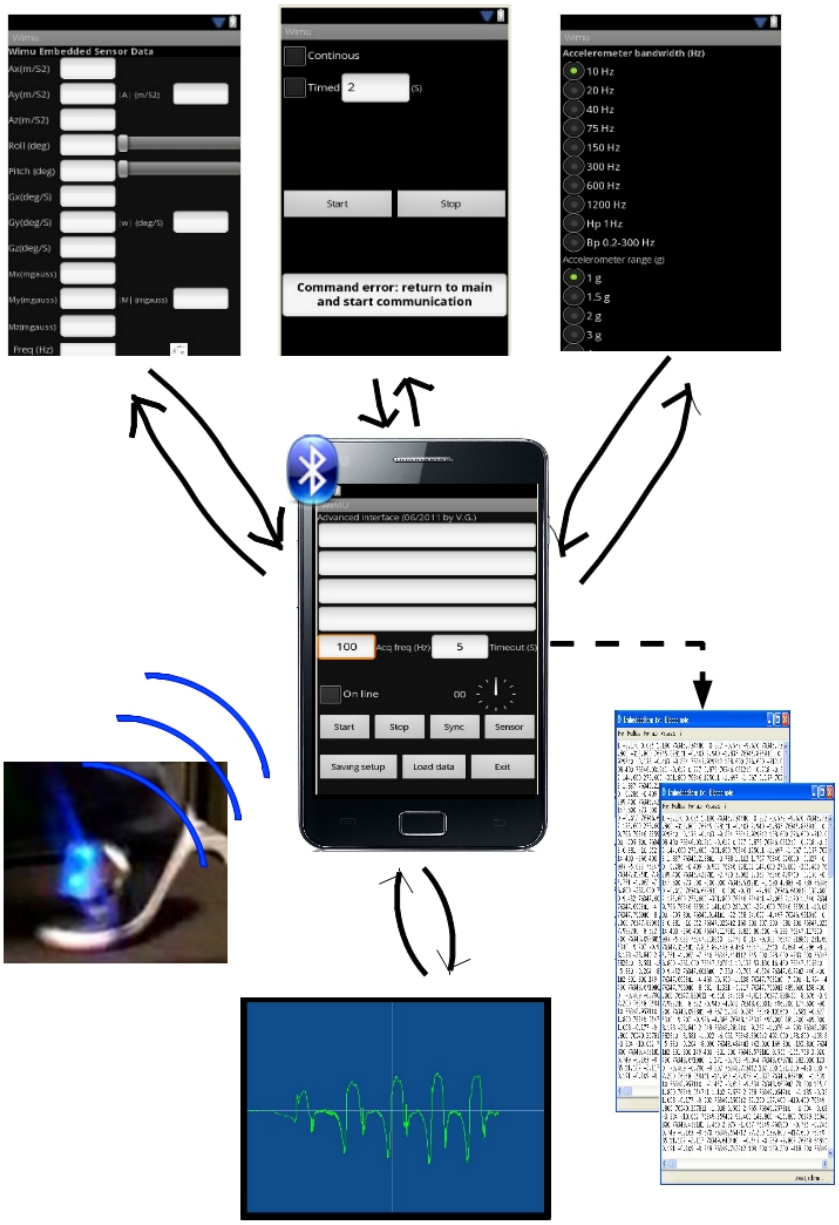


Figura 8.: Collegamento fra le schermate dell'applicativo

Quest'ultimo invia un segnale di inizio acquisizione dati sui canali creati e li acquisisce se disponibili.

### 6.4.4 *Gestione Thread*

Il sistema operativo Android è multithread. Un applicativo che risponda rapidamente alle richieste dell'utente deve necessariamente smistare i compiti a più thread, dando la precedenza ai thread di risposta all'utente. Il più importante di questi è lo UI Thread (User Interface Thread). Una regola di base della programmazione di applicativi Android è, nella creazione di un'activity, che si deve assolutamente evitare di sovraccaricare, o ancora peggio, eseguire istruzioni potenzialmente bloccanti all'interno dello UI Thread. La prassi per la gestione di operazioni con un comportamento imprevedibile dal punto di vista temporale, come ad esempio le operazioni di rete o di lettura e scrittura di file, è di incaricare altri thread che possano, se necessario, essere eseguiti in background, o fermati se necessario.
Abbiamo ritenuto utile avere un protocollo di comunicazione fra thread mediante una struttura che abbiamo chiamato mailbox, in cui $n$ thread possano inserire e consumare dati.

#### *MailBox*

L'oggetto `MailBox` è un buffer di tipo generico, a cui si può accedere per ora solo mediante un inserimento ed un prelievo sincronizzati, ma ciò potrebbe essere facilmente esteso a 3 modalità:

1. **Sincronizzata**: l'inserimento e prelievo di dati dalla mailbox è bloccante, un thread inserisce un dato se c'è spazio nel buffer, un altro thread utilizza il dato, se esiste. Ciascuno attende il proprio turno per compiere l'azione. La sincronizzazione viene gestita con un meccanismo di semafori che Java fornisce. I metodi di accesso sono:

```
1     synchronized public void put(T object)
2     synchronized public T get()
```

   Un thread alla volta può prendere, in mutua esclusione, l'accesso in scrittura al buffer, chiamando per primo il metodo `put` (vedi algoritmo **??**). Il valore contenuto nel buffer è sempre visibile a tutti i thread. Questo risultato è stato ottenuto dichiarando la variabile con la parola chiave `volatile`.

```
1     volatile private T buffer = null;
```

   Normalmente, ciò che avviene al momento in cui un thread prende l'accesso in mutua esclusione ad una variabile, è che si copia nel proprio stack di lavoro la variabile e modifica quella e solo prima di rilasciare la variabile la aggiorna all'esterno. Altri thread che fossero abilitati a vedere la variabile nel momento in cui è in modifica, vedrebbero un valore incongruente con il valore reale della stessa. La parola chiave volatile, impedisce ai thread che la modificano di crearsene una copia, e li vincola a lavorare direttamente sulla variabile[8] garantendone la visibilità in tutti i

8 la parola chiave volatile impedisce un possibile ripristino del valore di una variabile, il che rende potenzialmente pericolosa per la perdita di dati

momenti.

```
synchronized public void put(T object){
    //se il buffer è vuoto
    if (buffer == null){
        //inserisci l'oggetto
        buffer = object;
        //notifica tutti i thread in attesa sul buffer
        this.notifyAll();
    }else{
        try{
            //altrimenti attendi una notifica sul buffer
            this.wait();
            buffer = object;
        }catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Un thread può prelevare il dato immesso da un altro nella mailbox, usando il metodo `get()`(vedi algoritmo **??**). Mentre il dato viene prelevato, questo non può essere modificato. Appena il thread ha preso il dato, lo elimina dal buffer.

```
synchronized public T get(long d){
    T temp = null;
    //il buffer è vuoto
    if(buffer == null){
        try {
            //attendi una notifica sul buffer
            this.wait();
            //salva il contenuto del buffer
            temp = buffer;
            //svuoto il buffer
            buffer = null;
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    } else {
        temp = buffer;
        buffer = null;
        //notifica tutti i thread in attesa sul buffer
        this.notifyAll();
    }
    return  temp; }
```

Queste due operazioni implicano che devono essere eseguite in ordine, ed alternati ($put(dato_i)$, $get()$, $put(dato_j)$, $get()$ ecc ).

2. **Temporizzata**: ciò che avviene è molto simile al caso precedete: i thread vengono sincronizzati sul buffer. La differenza è che attendono solo una predeterminata quantità di tempo prima di inserire dati nuovi nel buffer.

Ciò significa che sovrascrivono qualunque dato non sia stato consumato in tempo.

3. **Condizionata**: la variante, in questo caso, sta nel superare l'attesa in scrittura nel momento in cui si verifichi una determinata condizione.

*Test per il protocollo di comunicazione MailBox*

Il primo esperimento riuscito sul protocollo è stato quello di due thread, un produttore ed un consumatore che comunicano tramite una `MailBox`. Il produttore genera una sequenza ordinata di numeri dispari, crea un oggetto `Point` nel seguente modo:

```
new Point(x++, Math.sin(x))
\end{verbatim}

L'oggetto così creato viene inserito dallo stesso produttore nella
    MailBox. Il consumatore può tentare di prelevare l'oggetto
    chiamando \verb|get()|. Mandando in stampa su console una
    sessione di comunicazioni si ha:

\begin{table}%
\centering
\begin{tabular}{l l}
put:& (70913.0, 0.921)\\
get:& (70913.0, 0.921)\\
put:& (70915.0, -0.737)\\
get:& (70915.0, -0.737)\\
put:& (70917.0, -0.307)\\
get:& (70917.0, -0.307)\\
put:& (70919.0, 0.993)\\
get:& (70919.0, 0.993)\\
put:& (70921.0, -0.519)\\
get:& (70921.0, -0.519)\\
put:& (70923.0, -0.561)\\
get:& (70923.0, -0.561)\\
put:& (70924.0, 0.393)
\end{tabular}
\caption{Sequenza di scritture su console da parte di un thread che
    crea la coppia (x, sin(x)) e da un altro che consuma il valore
    usando il protocollo MailBox}
\end{table}

Ho reimplementato lo stesso esperimento su Android, con una semplice
     activity per visualizzare il punto generato dal thead
    produttore, come il centro di un cerchio.
\begin{figure}
    \centering
        \includegraphics[width=.6\textwidth]{imgs/
            MailBoxCommTestAndroid.jpg}
    \caption{Prova di funzionamento del protocollo di comunicazione
        fra thread implementato su Android}
    \label{fig:mailbox_comm_test_android}
\end{figure}

```

```
\subsection{HMM}
L'interfaccia \verb|HiddenMarkovModel| è pensata per fornire uno scheletro per tutti i tipi di HMM.
Chi implementa l'interfaccia, dovrebbe partire dalla creazione di un insieme di stati $S$ ed un insieme di simboli di osservazione $V$. A questo punto le dimensioni delle strutture dati che conterranno i parametri sono note, e le implementazioni dei seguenti metodi accessori agli stessi dovrebbero assicurarsi che vengano rispettate tali dimensioni.
\begin{itemize}
    \item \verb|setA(ArrayList<Object> mtx)|, \verb|getA()|: rendere accessibile la matrice di transizioni
    \item \verb|setB(ArrayList<Object> mtx)|,  \verb|getB()|: rendere accessibile la matrice di emissioni
    \item \verb|setPi(ArrayList<Double> vec)|, \verb|getPi()|rendere accessibile il vettore di prior
    \item \verb|areValidObservations(ArrayList<Object> observations)|, chi implementa l'interfaccia potrebbe anche decidere di verificare la validità di una nuova osservazione, verificandone l'appartenenza all'insieme delle osservazioni.
\end{itemize}
\subsubsection{Scheletro di un HMM}
Una prima implementazione dell'interfaccia è la classe \verb|HMM| che obbliga chi la vuole usare ad implementarla solo passando al costruttore un insieme di stati. Questo viene ottenuto rendendo privato il costruttore di default
\begin{lstlisting}[language=java, style=eclipse, caption=Costruttori HMM , label=code:hmm1]
private HMM() {...}
public HMM(HashMap<String, Object> states) {...}
```

Codice 6.13: Creazione di un oggetto Point

L'assegnazione dei parametri avviene solo dopo una serie di controlli sui dati in ingresso che mirano a garantire una serie di proprietà degli stessi.
Assegnazione della matrice di transizioni(vedi codice **??**):
per prima cosa la matrice deve essere quadrata con dimensione pari al numero di stati. Successivamente a tale verifica si deve verificare la validità dei valori di probabilità di transizione per ciascuna coppia di stati, nonché la loro completezza come alternative probabilistiche (vale a dire che la loro somma è uguale a 1). Questa è un operazione costosa, ma necessaria, per garantire un minimo di correttezza.

```
public void setA(ArrayList<Object> mtx) {
        boolean isFitTransitionMtx = true;

        if (mtx != null &&
                // la matrice di transizione deve essere quadrata
                // della cardinalità e della stessa dimensione
                // dell'insieme degli stati
                mtx.size() == S.size() &&
             ((ArrayList<Object>) mtx.get(0)).size() == S.size()) {

            // ogni elemento della matrice deve essere un valore
            // di probabilità valido:
```

```
            for (int i = 0; i < S.size(); i++){
                isFitTransitionMtx &=
                    StatisticsOperations.
                        areCompleteProbabilisticAlternatives((
                        ArrayList<Object>) transitionsMtx.get(i));
            }
            // solo a questo punto assegno i valori alla matrice
            if (isFitTransitionMtx){
                A = transitionsMtx;
            }
        }else {
            //lancio un errore oppure
            //forzo l'utente a inizializzare
            //A correttamente
        }
    }
```

Codice 6.14: Impostazione della matrice di transizione

La stessa procedura viene eseguita per l'assegnazione delle prior (vedi codice 6.15):

```
public void setPI(ArrayList<Object> prior) {
    if (prior != null &&
            prior.size() == S.size() &&
            StatisticsOperations.
                areCompleteProbabilisticAlternatives(prior)){
        PI = prior;
    } else {
        //lancio un errore oppure
        //forzo l'utente a inizializzare
        //A correttamente
    }
}
```

Codice 6.15: Impostazione del vettore delle prior

L'implementazione delle matrici di emissione è delegato a HMM specializzate, ad emissioni discrete o continue.

### *HMM ad emissioni discrete*

Un'implementazione concreta di un HMM (utilizzabile come oggetto) è quella della HMM ad emissioni discrete

```
public class DiscreteHMM extends HMM {...}
```

Codice 6.16: Firma della classe HMM ad emissioni discrete

Questa eredita dalla classe `HMM` tutto ciò che contiene, e fornisce una sua implementazione della matrice di emissioni (vedi codice 6.17). Anche in questo caso vengono eseguiti dei controlli sui valori che vengono assegnati: la dimensione della matrice di emissioni $B : N \times M$ dove $N = |S|$ (Cardinalità dell'insieme degli stati dell'HMM) e $M = |V|$ cardinalità dell'insieme dell'alfabeto di osservazioni; i valori assegnati forniscono un'insieme completo di alternative probabilistiche.

```java
public void setB(ArrayList<Object> mtx) {
        boolean isFitEmissionMtx = true;
        // il numero di righe della matrice di
        // emissione deve essere pari al numero di stati
        if (mtx != null &&
                mtx.size() == S.size()&&
            // il numero di colonne di mtx essere
            // pari al numero di simboli osservabili
            ((ArrayList<Object>) mtx.get(0)).size() == V.size()) {
            for (int i = 0; i < S.size(); i++) {
                // la somma di tutte le probabilità di osservazione
                // deve essere pari a 1
                isFitEmissionMtx &= StatisticsOperations
                        .areCompleteProbabilisticAlternatives((
                            ArrayList<Object>) mtx
                                .get(i));
            }
            if (isFitEmissionMtx) {
                B = mtx;
            }
        } else {
            //lancio un errore oppure
            //forzo l'utente a inizializzare
            //B correttamente
        }
}
```

Codice 6.17: Impostazione della matrice di emissioni

### *HMM ad emissioni continue*

L'implementazione della HMM ad emissioni continue, nella gerarchia di classi, è un'altra diramazione a partire dalla classe `HMM`. Una HMM ad emissioni continue ha una distribuzione di probabilità sulle emissioni. Dato che la distribuzione più comunemente utilizzata è quella Gaussiana (o Normale), è stata implementata una classe con tale distribuzione.

```java
public class NormalHMM extends HMM
```

Codice 6.18: Firma della classe HMM ad emissioni continue

### *Operazioni sulle HMM*

La classe di base che gestisce le operazioni sulle HMM discrete è

```java
public class HMMOperations
```

Codice 6.19: Firme dell'operatore delle HMM

Questa permette di eseguire gli algoritmi più importanti sulle HMM, come l'algoritmo Forward-Backward (vedi Appendice 2, 3), l'algoritmo di Viterbi (vedi Appendice 4) e la sua variante offline.

Per le operazioni sulle HMM continue la classe che implementa gli algoritmi sopra menzionati è

`ContinuousHMMOperations`

Dopo aver creato ed inizializzato una `NormalHMM`, si può richiamare ad esempio la segmentazione nel seguente modo:

```
// creo l'HMM ed il suo operatore
ContinuousHMMOperations contHMMOp
NormalHMM nHMM
...
// qui creo i parametri
// states, A, Pi, emissionMtx
...
// inizializzo la HMM
// con i parametri creati
nHMM = new NormalHMM(states);
nHMM.setA(A);
nHMM.setPI(Pi);
nHMM.setContB(emissionMtx);
//collego l'oeratore con l'HMM
contHMMOp = new ContinuousHMMOperations(nHMM);
...
// un modo per acquisire osservazioni di
// prova è di leggerli da un file
observation = reader.readLine()
...
// finalmente eseguo la segmentazione
contHMMOp.onlineViterbi(observation)
```

Codice 6.20: Applicazione dell'operatore su HMM

# 7

# VALIDAZIONE

La terza fase del lavoro è la validazione. Abbiamo utilizzato un semplice meccanismo di triangolazione per fare una rapida verifica di funzionamento del lavoro totale. In dettaglio, data la cadenza (passi/minuto) e la velocità di deambulazione (m/sec), si può stimare la distanza percorsa. Allo stesso momento, si può usare un dispositivo GPS[1] per stimare la distanza percorsa e confrontare i due risultati. Ci si aspetta di avere degli errori considerevoli sull'approssimazione della distanza del GPS, in quanto ha un'accuratezza di ($\pm$ 15m).

La IMU è stata posizionata sul collo del piede di un soggetto, il quale è stato sottoposto a 6 sessioni di cammino su threadmill. Ciascuna sessione era della durata di 1:30 minuti. La velocità del threadmill è stata regolata a 2Km/h per la prima sessione ed aumentata di 1 km/h ad ogni sessione fino a 8Km/h.
La segmentazione di ciascuna sessione di cammino è stata salvata su un file, dal quale è stata calcolata la cadenza. Per calcolare la cadenza è sufficiente conoscere contare il numero di cicli di deambulazione al minuto. Quello che abbiamo fatto è calcolare la cadenza usando ognuno dei 4 eventi, determinati dalla segmentazione, per calcolare la cadenza. Facendo la media dei 4 risultati abbiamo ottenuto una buona stima della cadenza. Questa operazione è stata ripetuta per ciascuna velocità.
Abbiamo ottenuto una distribuzione di valori di cadenza per ciascuna velocità. A questo punto applicando una regressione lineare abbiamo trovato una relazione tra cadenza e velocità.

Dopo aver caricato bene le batterie dello Smartphone, siamo usciti dal laboratorio ed abbiamo fattola la prova per le strade di Pontedera. Il soggetto degli esperimenti, ha indossato anche in questo caso la IMU sul collo del piede. La velocità di cammino, qui è stata scelta del soggetto. Sullo Smartphone è stato attivato il segmentatore e su un altro il GPS. La distanza ed il percorso fatto tra i due punti è stata stimata dal GPS.

I dati del GPS, sono riferiti a coordinate geografiche[2] che sono state convertite per poter calcolare distanze in metri.

1 Global Positioning System
2 longitudine, latitudine

# 8

# RISULTATI E CONCLUSIONI

Parte III

# APPENDICE

# A

# SULLE HMM

I HMM sono strumenti di modellazione di sequenze temporali. Un comune esempio di applicazione è il riconoscimento della comunicazione verbale (Speech Recognition)[46]. Una sequenza temporale non è altro che un segnale.

**Definizione 3 (Segnale).** Un segnale è l' output osservabile di un processo, che in base al numero di sorgenti di provenienza viene detto

- **Puro**, se a sorgente unica
- **Corrotto** altrimenti

Un segnale può essere di natura discreta, o continua rispetto al tempo. Ad esempio un segnale che consista in una sequenza di caratteri viene raggruppato fra i segnali discreti, mentre la comunicazione verbale fra i segnali continui.

La sorgente di un segnale (il processo), a sua volta, può essere stazionaria, se le sue proprietà statistiche non variano nel tempo, o non stazionaria altrimenti.

## A.1 PROBLEMA: MODELLARE SEGNALI

Il problema della modellazione dei segnali è di fondamentale importanza, in molti settori. Le motivazioni principali sono la riproduzione dei segnali e la scoperta delle loro sorgenti. Esistono varie tipologie di modelli dai quali scegliere per modellare al meglio un dato segnale. La suddivisione maggiore è quella fra modelli deterministici e stocastici.

- Modelli deterministici: descrivono il segnale mediante le sue proprietà note. Ad esempio la luce ha una velocità pari a $300.000 km/s$.
- Modelli stocastici: descrivono le proprietà statistiche del segnale. Un esempio di modello statistico sono i processi gaussiani, i processi marcoviani e le HMM. In questi modelli si assume che il segnale possa essere caratterizzato come un Processo Parametrico Stocastico, con parametri stimabili in modo algoritmico.

I modelli deterministici descrivono i comportamenti di tutti i parametri di un segnale, in tutte le condizioni possibili. In molti casi però, i segnali di interesse sono abbastanza complessi, o oscurati, da non poterne conoscerne tutti i parametri, quindi i modelli deterministici risultano inefficaci. L'alternativa sono i modelli stocastici che descrivono solo un sottoinsieme dei parametri che caratterizzano un segnale, oppure una risultante di questi, e dato che il segnale si comporta solo in parte come i parametri descritti, il modello può fornire solo una descrizione probabilistica del segnale.

## A.2 PROCESSI DI MARKOV DISCRETI

La tipologia più semplice di modello di Markov sono quelli discreti.

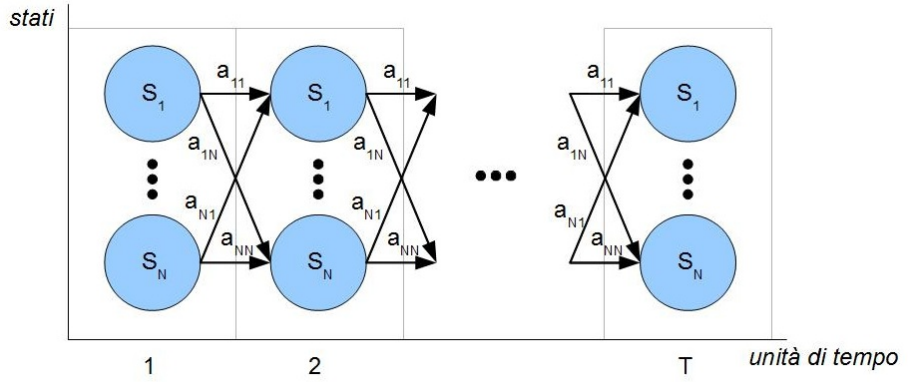


Figura 9.: Rappresentazione a lattice di Processo Discreto di Markov

**Definizione 4 (Processi di Markov Discreti).** $< N, M, A, \pi >$
Dato un insieme di stati $S = \{s_1, \ldots, s_N\}$ sono detti Processi di Markov Discreti dei sistemi che ad ogni istante di tempo $t$ assumono uno stato, ovvero lo stato corrente $q_t = s_k \in S$.

- $q_0 = \max_{0 \leqslant i \leqslant N} \pi_i$
- $q_t = \max_{0 \leqslant j \leqslant N} A_{q_{t-1},j} \quad 1 \leqslant t \leqslant T$

---

[ *TODO: fare rappresentazione ad automi a stati finiti* ]
Si può dare una descrizione probabilistica completa dei processi di Markov Discreti con la seguente equazione:

$$\wp[q_t = s_j | q_{t-1} = s_i, \ldots, q_0 = S_k] \tag{A.1}$$

Un caso speciale di (A.1) in cui la probabilità che il sistema si trovi in un determinato stato nel presente dipende solo dall'istante di tempo precedente e non da tutta la storia a partire dal primo istante di tempo.

---

**Definizione 5 (Modelli di Markov del Primo Ordine).**

$$\wp[q_t = S_j | q_{t-1} = S_i] \tag{A.2}$$

---

Le Catene di Markov ad ogni istante temporale compiono una transizione di stato con una probabilità nota. Tale probabilità è descritta nella Matrice delle Probabilità di Transizione.

---

**Definizione 6 (Matrice delle Probabilità di transizione).**

$$A = a_{i,j} = \wp[q_t = S_j | q_{t-1} = S_i] \quad \text{con} 1 \leqslant i, j \leqslant N \tag{A.3}$$

---

Proprietà delle $a_{ij}$ derivanti dal fatto che sono dei valori di probabilità:

1. $a_{ij} \geqslant 0$

2. $\sum_{j=1}^{N} a_{ij} = 1$

All'istante di tempo iniziale $t = 0$, lo stato della Catena marcoviana è determinato da una distribuzione di probabilità. Un vettore $\pi : 1 \times N$ che a ciascuno stato fa corrispondere la probabilità che sia lo stato iniziale, questa è nota come probabilità a Priori o probabilità dello stato iniziale.

---

**Definizione 7 (Distribuzione di probabilità dello stato iniziale).**

$$\pi_i = \wp(q_1 = S_i) \quad con 1 \leqslant i \leqslant N \tag{A.4}$$

---

## A.3 HMM

Avvolte le osservazioni sono funzioni probabilistiche di qualche stato nascosto (cioè esiste un processo stocastico nascosto che produce una sequenza di osservazioni).

---

**Definizione 8 (Matrice di probabilità delle Osservazioni).**

$$\begin{aligned} &B = \{b_j(k)\} \\ &\text{dove } b_j(k) = \wp[v_k \quad \text{all'istante } t | q_t = s_j] \quad \text{con } 1 \leqslant j \leqslant N \quad , 1 \leqslant k \leqslant M \end{aligned} \tag{A.5}$$

---

---

**Definizione 9 (HMM a Osservazioni discrete).**

$$HMM =< N, M, A, B, \pi > \tag{A.6}$$

dove:

1. $N = |S|$ è l'insieme degli stati $S = \{s_1, \ldots, s_N\}$,
2. $M = |V|$ è un insieme finito di Simboli di Osservazione $V = \{v_1, \ldots, v_M\}$ ,
3. $A$ è la matrice di transizione definita in (A.3),
4. $B$ è la matrice di probabilità dei Simboli di Osservazione definita in (A.5),
5. $\pi$ è il vettore di probabilità a priori definto in (A.4).

---

Un HMM può essere usato per generare una sequenza di osservazioni $O = \{O_1, O_2, \ldots, O_T\}$ nel seguente modo:

### A.3.1 *HMM ad emissioni continue*

Nel caso in cui le osservazioni siano continue è necessario avere un modello che associ ad ogni stato una distribuzione di emissione, invece che un singolo valore. Un esempio di distribuzione è la gaussiana mono variata. In questo caso la definizione della matrice di emissione descritta in (A.5) diventa

$$B = b_j(x) = \mathcal{N}(x, \mu_j, \sigma_j) = \frac{1}{\sigma_j \sqrt{2\pi}} \mathbf{e}^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}} \quad per 1 \leqslant j \leqslant N \tag{A.7}$$

**Algorithm 1** genera sequenze con HMM

1: $q_1 = \arg \max_{1 \leqslant i \leqslant N} \pi_i$
2: **for** $t = 1, \ldots, T; t++$ **do**
3: $\quad o_t = \max_{1 \leqslant s \leqslant N} [b_{q_t, s}]$
4: $\quad q_t = \arg \max_{1 \leqslant j \leqslant N} a_{i,j}$
5: **end for**
6: **return** {modello}

La formula può essere generalizzata a più gaussiane multivariate dette misture gaussiane

$$B = b_j(x) = \sum_{m=1}^{M} c_{j,m} \mathcal{N}(x, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \quad \text{per } 1 \leqslant j \leqslant N \tag{A.8}$$

Ovviamente è possibile applicare qualunque tipo di distribuzione al posto della gaussiana.
Dalla definizione (A.8) segue

$$\int_{-\infty}^{\infty} b_j(x) dx = 1 \quad \text{per } 1 \leqslant j \leqslant N \tag{A.9}$$

### A.3.2 *Tipi di HMM*

Vi sono casi particolari di HMM che sono considerati importanti per la loro ricorrenza nella descrizione di sistemi naturali. Il più generico tipo di HMM è noto come Ergodico, in cui ogni stato è connesso ad ogni altro stato, quindi la matrice di transizione è una matrice senza zeri.
Un modello più significativo è il modello Bakis o Sinistra-Destra. In questo modello l'aumentare del tempo, causa una variazione monotona crescente modulo N sull'indice degli stati. Questo modello è conforme ai segnali le cui proprietà variano con il tempo. Le matrici di transizione dei modelli Sinistra-Destra, godono della proprietà

$$\begin{aligned} a_{i,j} &= 0 \quad \forall j < i \\ a_{i,j} &= 0 \quad j > i + \Delta \end{aligned} \tag{A.10}$$

La seconda condizione serve ad evitare che vi siano grossi balzi in avanti sugli stati, di solito $\Delta = 2$. Inoltre le probabilità iniziali hanno il vincolo

$$\pi_i = 1 \Leftrightarrow i = 1 \tag{A.11}$$

supponendo di avere gli stati ordinati da 1 ad N

## A.4 TIPI DI PROBLEMI CHE SI AFFRONTANO CON LE HMM

Generalmente con le HMM si affrontano 3 tipologie di problemi.

1. **Valutazione**: dati O e $\lambda$, calcolare $\wp(O|\lambda)$ in modo ottimale.
2. **Decodifica**: dati O e $\lambda$, trovare la sequenza di stati $Q = q_1, q_2, \ldots, q_T$ "migliore", secondo un criterio di ottimalità, che possa aver generato O.
3. **Apprendimento**: dato $\lambda$, regolarne i parametri per massimizzare $\wp(O|\lambda)$?

*Valutazione*

Il problema della valutazione consiste nel calcolare la probabilità che una certa sequenza di osservazioni sia stata prodotta da un dato modello. La soluzione del problema permette, dati diversi modelli candidati $\lambda_1, \ldots, \lambda_n$ ed una sequenza di osservazioni O, di scegliere $\lambda_k$ con $\max_{1 \leqslant i \leqslant n} \{\wp(O|\lambda_i)\}$.

Una soluzione naïve della Valutazione è quello di enumerare tutte le possibili sequenze di stati (detti anche percorsi) $Q = q_1, \ldots, q_T$ dove $q_i \in S = s_1, s_2, \ldots, s_N$ e calcolare per ciascuna la probabilità che l'osservazione sia stata prodotta da essa e moltiplicare il risultato per la probabilità che quel particolare percorso venga scelto.

$$\begin{aligned} \wp(O|\lambda) = & \sum_{1 \leqslant t \leqslant T, i \in \{\text{tutti } Q\}} \wp(O_t|Q_i, \lambda)\wp(Q_i|\lambda) \quad \text{dove} \\ & \wp(O|Q_i, \lambda) = \prod_{1 \leqslant t \leqslant T} \wp(O_t|q_{i,t}, \lambda) \\ & = b_{q_i,1}(O_1) * \ldots * b_{q_i,T}(O_T) \quad \text{e} \\ & \wp(Q_i|\lambda) = \pi_{q_1} * a_{q_1,q_2} * \ldots * a_{q_T-1,q_T} \end{aligned} \tag{A.12}$$

l'algoritmo ha complessità esponenziale, $O(N^T)$.

La soluzione ottima a questo problema è data dall'algoritmo di programmazione lineare noto come Forward. L'idea su cui si basa è di considerare dei percorsi parziali per rappresentare osservazioni parziali. Vengono definite le variabili $\alpha_{i,t}$ come la probabilità di aver osservato la sequenza parziale $O_1, \ldots, O_t$ e di essere nello stato $S_i$ all' istante temporale t

$$\alpha_{i,t} = \wp(o_1, \ldots, o_t, q_T = S_i|\lambda) \tag{A.13}$$

Dalla definizione (A.13) ricaviamo la matrice $\alpha(N \times T)$ nella quale vengono inserite le probabilità parziali per ogni combinazione stato-tempo, di modo che al passo temporale successivo il calcolo possa essere basato su tali valori e non ricalcolando tutto dal primo istante di tempo.

La complessità dell'algoritmo Forward è $O(NT)$. Il netto miglioramento è dovuto al fatto che i risultati parziali vengono riutilizzati, limitando il numero di calcoli da svolgere ad ogni istante temporale a N. La struttura grafica su cui si basa l'algoritmo Forward è detta struttura a traliccio.

*Decodifica*

Nella decodifica, si cerca la sequenza di stati $Q = q_1, \ldots, q_T$ che ha generato una data sequenza di osservazioni $O = o_1, \ldots, o_T$. Dato che, al contrario della Valutazione, non esiste un'unica soluzione al problema delle Decodifica, quello che si fa è di stabilire un criterio di ottimalità in funzione del quale fare la ricerca.

Un possibile criterio di ottimalità della sequenza è quello di scegliere gli stati $q_t$ che all'istante di tempo t sono i più probabili. Tale criterio massimizza il numero atteso di stati corretti individualmente.

Per risolvere il problema necessitiamo di due variabili $\beta$ e $\gamma$. La prima è definita come risultato dell' algoritmo Backward, che computa il processo inverso dell'algoritmo Forward.

$$\beta_{i,t} = \wp(o_{t+1} \ldots o_T|q_t = S_i, \lambda) \tag{A.14}$$

**Algorithm 2** Forward

1: {1. Inizializzazione}
2: **for** $j = 1$ to $N = |S|$ **do**
3: $\alpha_{j,1} = \pi_j b_j(o_1)$ { *La probabilità congiunta di partire dal* j*-esimo stato* }
4: **end for** { *ed emettere il primo segnale dallo stesso*}

5: {2. Induzione}
6: **for** $t = 1$ to $T - 1$ **do**
7: **for** $j = 1$ to $N$ **do**
8: $\alpha_{j,t+1} = [\sum_{i=1}^{N} \alpha_{i,t} a_{i,j}] b_j(o_{t+1})$ { *La probabilità congiunta di arrivare al* }
9: **end for** { j*-esimo stato ed emettere il* $t+1$*-esimo segnale*}
10: **end for**
11: {3. Terminazione}
12: **for** $j = 1$ to $N$ **do**
13: $\wp(O|\lambda) = \sum_{j=1}^{N} \alpha_{j,T}$
14: **end for**

**Algorithm 3** Backward

1: {1. Inizializzazione}
2: **for** $j = 1$ to $N = |S|$ **do**
3: $\beta_{j,T} = 1$
4: **end for**
5: {2. Induzione}
6: **for** $t = T - 1$ to $1$ **do**
7: **for** $i = 1$ to $N$ **do**
8: $\beta_{i,t} = \sum_{j=1}^{N} a_{i,j} b_j(o_{t+1}) \beta_{j,t+1}$
9: **end for**
10: **end for**

La seconda variabile, $\gamma$ è definita come la probabilità di essere in uno stato $S_i$ al tempo $t$, data un'osservazione $O$ ed un modello $\lambda$.

$$\gamma_{i,t} = \wp(q_t = S_i|O, \lambda) \tag{A.15}$$

$\gamma$ può essere calcolata in funzione delle variabili di Forward e Backward:

$$\gamma_{i,t} = \frac{\alpha_{i,t}\beta_{i,t}}{\wp(O|\lambda)} = \frac{\alpha_{i,t}\beta_{i,t}}{\sum_{j=1}^{N} \alpha_{j,t}\beta_{j,t}} \tag{A.16}$$

$\alpha_{i,t}$, fornisce le probabilità delle osservazioni parziali fino a $t$, mentre $\beta_{i,t}$, da $t+1$ in poi, essendo correntemente nello stato $S_i$. Il denominatore dell'equazione

(A.16) è un fattore di normalizzazione che rende $\gamma$ un valore di probabilità, quindi sarà valida la proprietà

$$\sum_{i=1}^{N} \gamma_{i,t} = 1 \tag{A.17}$$

Usando $\gamma$ è possibile trovare lo stato q che individualmente è il più probabile al tempo t:

$$q_t = \arg \max_{0 \leqslant i \leqslant N} \gamma_{i,t} \tag{A.18}$$

Anche se (A.18) massimizza il numero di stati più probabili scegliendo quelli che individualemente sono i più probabili, potrebbe creare problemi nel momento in cui si considera una sequenza di stati. Se la HMM ha anche transizioni nulle, la sequenza generata da (A.16) potrebbe essere non valida, perché non vi è nessun controllo sulla probabilità di co-occorrenza di stati.
Il criterio di ottimlalità può essere cambiato ad individuare la miglior sequenza di lunghezza T che massimizzi la probabilità di tutti gli stati nella sequenza. Per trovare il cammino migliore, viene usato un metodo di programmazione dinamica detto algoritmo di Viterbi. L'algoritmo di Viterbi individua la migliore sequenza di stati che rispondano a una data sequenza di osservazioni:

$$\delta_{i,t} = \max_{q_1,\dots q_{t-1}} \wp(q_1 \dots q_t = S_i, o_1 \dots o_t | \lambda) \tag{A.19}$$

Nella matrice $\delta$ vengono inserite le probabilità, mentre per tenere traccia del percorso migliore si usa una variabile $\psi$:

$$\psi_{i,t} = \arg \max_{q_1,\dots q_{t-1}} \wp(q_1 \dots q_t = i, o_1 \dots o_t | \lambda) \tag{A.20}$$

La procedura completa per trovare il percorso di stati più probabile è la seguente:

### A.4.1 *Algoritmo di Viterbi in Tempo Reale*

Il problema dell'algoritmo di Viterbi in molte applicazioni reali, è la latenza. I sistemi che utilizzano l'algoritmo hanno spesso necessità di avere risultati immediati, in tempo reale. Trovare la sequenza di stati più verosimile che ha generato una sequenza di osservazioni è (come mostrato dall'algoritmo 4) fattibile con l'algoritmo di Viterbi, tracciando percorsi nella sequenza temporale di stati ed una volta arrivato a termine (tempo finale T), ricostruendo a ritroso il cammino migliore. Un problema significativo dell'algoritmo è che assume che la sequenza temporale sia finita. Vi sono molti casi pratici in cui l'applicazione dell'algoritmo sarebbe utile, per i quali i dati sono un flusso continuo ed incessante di dati. Una possibile soluzione è l'applicazione del'algoritmo di Viterbi a finestre successive di dati, dopo la quale restituire solo una parte iniziale degli decodificati [47], [48](perché hanno una probabilità maggiore di essere corretti). Questa soluzione, conduce a una decodifica sub-ottimale.
Un'altra soluzione consiste nel confrontare più percorsi su una finestra temporale in espansione, finché le soluzioni non convergono. Nel lavoro [49] per la localizzazione automatica di un soggetto via video, la finestra temporale viene dinamicamente ridimensionata in base a un'euristica che bilanci latenza e accuratezza. Questo tipo di approccio non garantisce la convergenza dei percorsi considerati. L'approccio che noi abbiamo usato nel lavoro è quello proposto da Bloit et al [45].

**Algorithm 4** Viterbi

{Inizializzazione}
**for** $i = 0$ to $N$ **do**
  $\delta_{i,1} = \pi_i b_i(o_1)$
  $\psi_{i,1} = 0$
**end for**
{Iterazione}
**for** $t = 2$ to $T$ **do**
  **for** $i = 1$ to $N$ **do**
    $\delta_{i,t} = \max_{1 \leqslant j \leqslant N} [\delta_{j,t-1} a_{j,i}] * b_i(o_i)$
    $\psi_{i,t} = \arg \max_{1 \leqslant j \leqslant N} [\delta_{j,t-1} a_{j,i}]$
  **end for**
**end for**
{Terminazione}
$P* = \max_{1 \leqslant i \leqslant N} [\delta_{i,T}]$
$q_T* = \arg \max_{1 \leqslant i \leqslant N} [\delta_{i,T}]$
{Backtracking}
**for** $t = T - 1$ to $1$ **do**
  $q*_t = \psi_{q*_{t+1},t+1}$
**end for**

[1]

**Definizione 10 (Cammino locale).** Viene detta cammino locale, la sequenza di stati $s(a, b, i)$ ottenuta applicando l'algoritmo di Viterbi alla finestra temporale dall'istante temporale $a$ all'istante $b$ (con $a < b$) e compiendo il backtracking da uno stato arbitrario $i$ al tempo $b$.

**Definizione 11 (Punto di Fusione).** Si definisce punto di fusione,l'istante temporale $\tau < T$ t.c per $a \leqslant t \leqslant \tau$, tutti i cammini locali appartenente all'insieme dei cammini locali $CL = \{s(a, b, i), \forall i \in S\}$ (dove $S$ è l'insieme degli stati dell'HMM), sono uguali.

Un punto di fusione gode della seguente proprietà: i cammini locai fino al punto di fusione (che per definizione sono tutti uguali) sono sempre uguali al cammino globale (quello ottenibile con l'algortimo di Viterbi originale 4)[1].

$a = 0$, $b = 0$

*Apprendimento*

Il problema consiste nel configurare i parametri di $\lambda$ per massimizzare la probabilità di una sequenza di dati osservati. Non è noto un metodo analitico per risolvere il problema, infatti, data una sequenza finita di osservazioni, non esiste un modo ottimo di stimare i parametri di $\lambda$. Possiamo però scegliere $\lambda$ in modo da massimizzare localmente $\wp(O|\lambda)$, con un procedimenti iterativi come

1 per la dimostrazione consultare [45]

- Baum-Welch
- Expectation-Modification
- Tecniche basate sul gradiente

La procedura iterativa è detta riestimazione, e consiste in un miglioramento graduale (in base ad un criterio) ed un aggiornamento dei parametri. Per descivere le riestimazione, è necessario definire la variabile $\xi$, come la probabilità di essere in un certo stato in un istante di tempo ed essere in un altro nel successivo istante di tempo:

$$\begin{aligned}\xi_t(i,j) &= \wp(q_t = S_i, q_{t+1} = S_j | O, \lambda) \\ &= \frac{\alpha_{i,t} a_{i,j} b_i(o_t) \beta_{j,t+1}}{\wp(O|\lambda)} = \frac{\alpha_{i,t} a_{i,j} b_i(o_t) \beta_{j,t+1}}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{i,t} a_{i,j} b_i(o_t) \beta_{j,t+1}}\end{aligned} \tag{A.21}$$

Possiamo correlare $\gamma$ e $\xi$, nel seguente modo

$$\gamma_{i,t} = \sum_{j=1}^{N} \xi_t(i,j) \tag{A.22}$$

---

**Definizione 12.** Numero di transizioni attese

- Numero di transizioni attese da $S_i$

$$\sum_{t=1}^{T-1} \gamma_{i,t} \tag{A.23}$$

- Numero di transizioni attese da $S_i$ a $S_j$

$$\sum_{t=1}^{T-1} \xi_t(i,j) \tag{A.24}$$

- $\overline{\pi}_i$ = numero di volte nello stato $S_i$ a tempo $t = 1$ atteso

$$= \gamma_{i,1} \tag{A.25}$$

- $\overline{a}_{i,j} = \dfrac{\text{numero di transizioni attese dallo stato} S_i \text{allo stato} S_j}{\text{numero di transizioni attese dallo stato} S_i}$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_{i,t}} \tag{A.26}$$

- $\overline{b}_j(k) = \dfrac{\text{numero di volte nello stato} S_j \text{osservando} v_k}{\text{numero di volte atteso nello stato} S_i}$

$$= \frac{\sum_{t=1}^{T} \gamma_{j,t} \quad \text{t.c. } o_t = v_k}{\sum_{t=1}^{T} \gamma_{j,t}} \tag{A.27}$$

Dato il modello $\lambda = (A, B, \pi)$ indico il modello riestimato con $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$. L.Baum et al[50] hanno dimostrato che nel processo di riestimazione è vera una delle seguenti alternative:

1. $\lambda = \overline{\lambda}$
2. $\wp(O|\overline{\lambda}) > \wp(O|\lambda)$

Nel secondo caso è possibile sostituire $\overline{\lambda}$ a $\lambda$, ripetendo l'iterazione, finché non si verifica una condizione d'arresto, ed il risultato della procedura è detto stima di massiva verosimiglianza (maximum likelihood estimate) dell'HMM

**Algorithm 5** Baum-Welsh o Maximum Likelihood Expectation.

**Require:** maxlikelihood($\lambda = A, B, \pi$)

1: **repeat**

2: $\boxed{\pi_i = \gamma_1(i)}$ $\boxed{a_{ij} = \dfrac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}}$ $\boxed{b_j(k) = \dfrac{\sum_{t=1, O_t = V_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}}$

3: **if** $\lambda = \tilde{\lambda}$ **then**

4: **return** $\lambda$

5: **else if** $\wp(O|\tilde{\lambda}) > \wp(O|\lambda)$ **then**

6: maxlikelihood($\tilde{\lambda}$)

7: **end if**

8: **until** {una condizione limite}

## A.5 PROBLEMI DI IMPLEMENTAZIONE DI HMM

Vi sono diversi problemi che si devono affrontare per implementare le HMM ed i vari algoritmi sinora descritti. Alcuni di questi sono

- Ridimensionamento (scaling): la procedura di riestimazione, comporta una lunga sequenza di prodotti di valori di probabilità. Ciò fa in modo che i valori tendano esponenzialmente a zero, quindi vi è un inevitabile problema di underflow. L'unico modo di ovviare al problema è quello di ridimensionare i valori, moltiplicandoli per un coefficiente che non dipenda dallo stato, ma solo dal tempo. Alla fine del processo, i coefficienti di ridimensionamento vengono eliminati.
  Ad esempio nella procedura di riestimazione si calcola la matrice di transizione con l'equazione (A.26)

$$\overline{a_{i,j}} = \frac{\sum_{t=1}^{T-1} \alpha_{i,t} a_{i,j} b_j(o_{t+1}) \beta_{j,t+1}}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \alpha_{i,t} a_{i,j} b_j(o_{t+1}) \beta_{j,t+1}} \tag{A.28}$$

Usando un coefficiente di ridimensionamento è $c_t =$

$$\frac{1}{\sum_{j=1}^{N} \alpha_{i,t}} \tag{A.29}$$

si ottiene un $\alpha_{j,t}$ scalato

$$\hat{\alpha}_{j,t} = \frac{\sum_{j=1}^{N} \hat{\alpha}_{j,t-1} a_{i,j} b_j(o_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_{i,t-1} a_{i,j} b_j(o_t)} \tag{A.30}$$

Nel momento in cui vengono calcolati i $\beta_{j,t}$, vengono eliminati i fattori di ridimensionamento:

$$\hat{\beta}_{j,t} = c_t \beta_{j,t} \tag{A.31}$$

La modifica più importante deve essere applicata all'algoritmo Forward, perché si è interessati al valore di probabilità. In questo caso non è possibile semplicemente sommare $\hat{\alpha_{j,t}}$, perché sono valori scalati e privi di significato se presi singolarmente. In questo caso viene usata la seguente proprietà:

$$\begin{gathered} \prod_{t=1}^{T} c_t \sum_{i=1}^{N} \alpha_{i,T} = 1 \\ \prod_{t=1}^{T} c_t \wp(O|\lambda) = 1 \Leftrightarrow \wp(O|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t} \end{gathered} \tag{A.32}$$

Qui introduciamo il logaritmo della probabilità, in modo che il valore sia calcolabile su un computer

$$\log(\wp(O|\lambda)) = -\sum_{t=1}^{T} \log c_t \tag{A.33}$$

- Molteplici sequenze di osservazione. Nelle HMM Sinistra-Destra, non si possono usare singole sequenze di osservazioni per addestrare il modello, perché questo tipo di modello tende a uscire molto facilmente da uno stato, quindi ad ogni stato corrispondono pochissime osservazioni. Ciò implica che per una quantità di dati sufficiente a fare una stima affidabile dei parametri del modello si devono usare più sequenze di osservazioni.
- Stime dei parametri iniziali. Un problema tutt'ora irrisolto è come scegliere i valori dei parametri iniziali in modo tale che i massimi locali corrispondano al massimo globale della funzione di verosimiglianza (likelihood). Normalmente quello che si fa è scegliere valori casuali oppure uniformi per poi iniziare la procedura di riestimazione. Invece per quanto riguarda i parametri B è necessaria una buona stima iniziale, che solitamente viene fatta con un processo di segmentazione e media delle osservazioni in stati.
- Insufficienza di dati. Un problema frequente è che il numero di osservazioni è troppo basso per consentire di avere una stima abbastanza buona dei parametri del modello. Una possibile soluzione è quella di aumentare

il numero di dati, ma ciò è spesso impraticabile. L'approccio inverso è quello di ridurre il numero di parametri, come ad esempio il numero di stati. Ciò è in teoria sempre praticabile, ma spesso poco sensato, in quanto vi sono delle motivazioni valide per avere quei parametri. Una terza alternativa è quella di interpolare tra un insieme di stime di parametri con un'altro da un modello per il quale si ha un numero sufficiente di dati di addestramento. L'idea è quella di progettare insieme al modello, anche una versione ridotta dello stesso, per il quale il numero di dati in possesso sia sufficiente. Date le stime per i parametri del modello $\lambda = (A, B, \pi)$ come per la versione ridotta $\lambda' = (A', B', \pi')$ il modello interpolata è ottenuto come

$$\tilde{\lambda} = \epsilon\lambda + (1 - \epsilon)\lambda' \tag{A.34}$$

dove $\epsilon$ rappresenta un peso che viene associato ai parametri del modello, ed $(1 - \epsilon)$ il peso associato a quelli del modello ridotto. Il valore di $\epsilon$ viene determinato in funzione dei dati di addestramento. Mercer et al [**?** ] hanno dimostrato che è possibile stimare l'$\epsilon$ ottimo mediante l'algoritmo Forward-Backward, espandendo la HMM a partire da (A.34).

- Scelta della dimensione e tipo del modello. Si tratta della scelta dei parametri che si deve fare all'inizio per rappresentare al meglio il problema con le HMM. Il tipo di HMM, Ergodico o Sinistra-Destra, la dimensione del modello cioè numero di stati, l'alfabeto di osservazione, discreti o continui, a distribuzione singola o a misture di distribuzioni. Non vi è un metodo standard, o migliore di prendere queste decisioni, ma devono essere fatte in base al tipo di segnale che si sta modellando.

# B

# CENNI DI MECCANICA CLASSICA

## B.1 CINEMATICA

La Cinematica[51] è lo studio del moto di corpi materiali, senza considerare le cause e conseguenze di tale moto. La parte della Meccanica che si occupa delle cause e conseguenze del moto dei corpi è la Dinamica o Cinetica. La Cinematica fornisce una descrizione geometrica dei possibili moti.
Il soggetto mediante il quale vengono condotti gli studi in Cinematica è la particella, vale a dire un corpo immaginario, che occupa un singolo punto dello spazio. Un insieme di particelle le cui distanze relative rimangono invariate in ogni riferimento spaziotemporale.

### B.1.1 *Moto rettilineo di una particella*

Le quantità principali di cui si occupa sono

*Posizione*

Il vettore posizione della particella P nel punto A: $\mathbf{r} = (x_A, y_A, z_A)$ con magnitudine $|\mathbf{r}| = \sqrt{x_A^2 + y_A^2 + z_A^2}$ (m).

*Spostamento*

Lo spostamento da A a B di P: $\mathbf{r}_{AB} = \mathbf{r}_B - \mathbf{r}_A$ (m)

*Distanza*

La distanza $s = \int_{t_1}^{t_2} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2} dt$ (m) dove t è il tempo.

*Velocità*

La **velocità**

- **media** $\bar{\mathbf{v}} = \frac{\Delta \mathbf{r}}{\Delta t}$ (m/s) con $\Delta t > 0$
- **istantanea** $\mathbf{v} = \lim_{\Delta t \to 0} \frac{\Delta \mathbf{r}}{\Delta t}$ con magnitudine $|\mathbf{v}| = \frac{ds}{dt}$ (m/s)

*Accelerazione*

L'accelerazione

- **media** $\bar{\mathbf{a}} = \frac{\Delta \mathbf{v}}{\Delta t}$ con $\Delta t > 0$ ($m/s^2$)
- **istantanea** $\mathbf{a} = \lim_{\Delta t \to 0} \frac{\Delta \mathbf{v}}{\Delta t}$ ($m/s^2$)

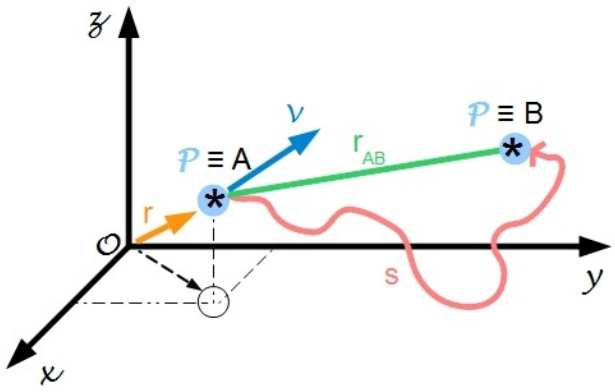


Figura 10.: Moto Rettilineo di una particella

## B.2 MOTO ANGOLARE DI UNA PARTICELLA

*Posizione*

Il vettore posizione della particella P nel punto A rispetto ad un asse di rotazione $O - z$ è $\mathbf{r}(t)$. La posizione angolare del punto P è $\mathbf{r}_\perp(t) = r_{\perp x}\cos\theta i + r_{\perp x}\sin\theta j$ $(rad)$

*Velocità*

La velocità angolare è data da: $\omega = \frac{d\theta}{dt}$ $(rad/s)$

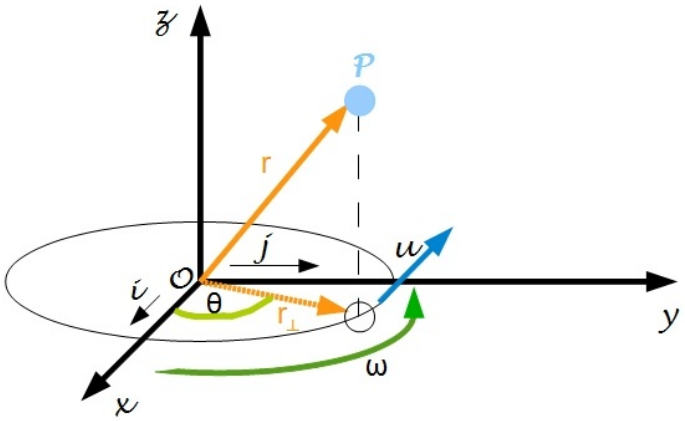


Figura 11.: Moto Angolare di una particella

*Accelerazione*

L'accelerazione angolare è data da: $\alpha = \frac{d\omega}{dt}$ $(rad/s^2)$

## B.3 DINAMICA (CINETICA)

Branca della meccanica che si occupa di forze che producono, arrestano o modificano il moto di corpi. Le due leggi fondamentali della Dinamica sono quelle di Newton, in particolare la seconda:

$$F = ma \tag{B.1}$$

# C

# SENSORI

Principalmente in questo lavoro usiamo un giroscopio monoassiale. In letteratura, l'analisi della deambulazione viene affrontata mediante accelerometri, giroscopi e magnetometri.

## C.1 ACCELEROMETRO

Un accelerometro (vedi figura 12) è un dispositivo elettromeccanico che misura le forze di accelerazione. Tali forze possono essere sia statiche, come la forza di gravità, che dinamiche, causate muovendo l'accelerometro.
Gli usi immediati di un accelerometro sono

- misurando l'accelerazione statica delle forza di gravità, si può calcolare l'angolo a cui è inclinato lo strumento.
- misurando l'accelerazione dinamica si può analizzare il modo in cui si sta muovendo il dispositivo

Un applicazione industriale importante è l'airbag nelle macchine, la cui apertura scatta se l'accelerometro percepisce una brusca frenata. Un'altra applicazione è quella implementata dalla Apple, nei suoi portatili per la protezione del disco rigido: se il portatile dovesse cadere mentre acceso, l'accelerometro capta la caduta libera ed il sistema operativo si auto termina immediatamente in modo che la testina non sia sul disco.

## C.2 GIROSCOPIO

Il giroscopio è uno strumento per misurare l'accelerazione di rotazione (momento angolare) di un corpo. Vi sono diversi tipi di giroscopi, meccanici, a vibrazione, a fibre ottiche ecc.. Un disco rotante in assenza di torsione esterna, mantiene la direzione della sua rotazione Quando viene applicata una torsione viene applicata al disco, ad angolo con il suo asse di rotazione, il disco ruota sul piano determinato dalle due assi (rotazione iniziale e torsione) nella direzione che va dall'asse di rotazione iniziale a quello della torsione.
Il tipo di giroscopio che usiamo in questo lavoro è il cosiddetto giroscopio piezoelettrico, o MEMS (Micro Electro Mechanical System) o a vibrazione. Si basa sul principio di Coriolis: un oggetto che vibra, continua a vibrare sullo stesso piano se la struttura che lo sostiene è in rotazione. La misurazione della velocità angolare avviene nel seguente modo: un elemento piezoelettrico (oggetto di forma tubolare) oscilla a causa di una rotazione, quindi viene misurata la forza di Coriolis sulla sezione longitudinale dell'elemento, dopo essere stata convertita in un voltaggio elettrico dallo stesso elemento piezoelettrico.

## C.3 MAGNETOMETRO

Il Magnetometro è uno strumento che misura il campo magnetico. Questo può essere fatto in diversi modi. Il metodo pre elettronico, è quello inventato da Coulomb ed usa un ago magnetico sospeso.
Il metodo elettronico chiamato elettromagnetometro o Magnetometro Fluxgate

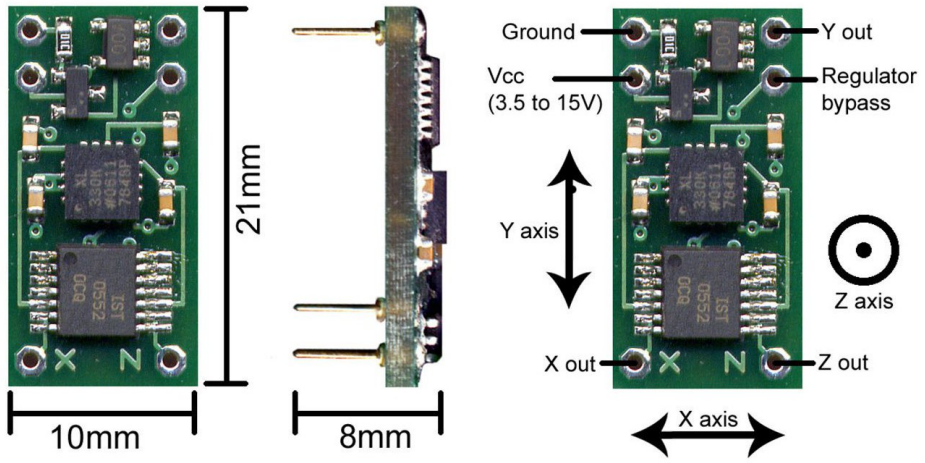


Figura 12.: Accelerometro triassiale, cortesia di `http://www.dimensionengineering.com`

è basato sulla saturazione di materiali magnetici. Questi ha un centro in ferro, ed intorno ad esso due fili conduttori. Attraverso il primo filo fluisce corrente elettrica. Il ferro è un elemento magnetico, ma in condizioni normali gli assi magnetici sono orientati in direzioni casuali e la forza magnetica totale è prossima a zero. Nel momento in cui comincia a fluire corrente nel filo, gli assi si allineano e creano un campo magnetico percepibile come aumento del campo magnetico creato dalla corrente nel filo. La quantità di forza magnetica che può produrre il ferro è limitata, il ferro giunge ad un livello di saturazione, dopo di che cambia bruscamente polarità, al che giunge alla saturazione e cambia polarità e così via. Questo processo induce corrente nel secondo filo che avvolge il ferro. Se la procedura avvenisse in un ambiente magneticamente neutrale il voltaggio nei due file dovrebbe combaciare, altrimenti vi sarà un dislivello proporzionale al campo magnetico di disturbo. L'intensità del campo magnetico terrestre superficiale è circa 50,000 nanoTesla.

## C.4 IMU

Inertial Measurement Unit, (Unità di Misura Inerziale), è l'integrazione di più sensori. Questo fornisce le misure fatte dai sensori interni con eventuali correzioni fatte in base alla temperatura interna dello strumento stesso Le IMU sono usate come sistemi di navigazione inerziali di aerei, missili.
La IMU che abbiamo usato per ha la seguente scheda di definizione:

| IMU | | |
|---|---|---|
| **Sensore** | **Intervallo di misurazione** | **Risoluzione** |
| Accelerometro triassiale | x[±1g-3g],y[1.5g-2g/8g], z[2g-16g], | [12-14 bit] |
| Giroscopio triassiale | [±2000-1600°/sec], | [12-16 bit] |
| Magnetometro triassiale | [±4 gauss], | 12 bit |
| Termometro | [-55-155°/C] | 12bit |
| **Connettività** | | Bluetooth per le brevi distanze |
| **Frequenza di campionamento** | | $\geqslant$ 300 Hz |
| **Dimensioni strumento** | | 60 × 30 × 40 mm |

Tabella 15.: Scheda tecnica IMU

# D

# SU ANDROID

Android è una piattaforma completa[1] totalmente open source[2] progettata per dispositivi mobili. Android è di proprietà della società Open Handset Alliance, con Google come maggiore azionario. L'obbiettivo di Google è accelerare lo sviluppo della tecnologia mobile ed offrire all'utente un'esperienza sempre più ricca ed allo stesso tempo meno costosa. Android è pensato per essere pronto all'uso dal punto di vista di tutti i possibili attori:

- **Utenti**: I dispositivi hanno una configurazione di default che permette un funzionamento immediato e performante ma che può in un secondo momento essere profondamente riconfigurato su misura.
- **Sviluppatori**: Uno sviluppatore ha bisogno soltanto dell' Kit di sviluppo di Android (Android SDK[3]), che comprende anche un emulatore, ma permette anche di sviluppare su un vero dispositivo. Uno sviluppatore ha accesso al codice dell'intera piattaforma Android.
- **Manufattori**: Android è portabile[4] e (eccetto alcuni driver per specifici hardware) permette di far funzionare dispositivi immediatamente. I venditori non sono tenuti a rendere disponibile alla comunità le proprie aggiunte. In molti casi dispositivi Bluetooth e Wi-Fi, sono gestiti da codice proprietario. Ma dato che lo sviluppo di codice viene regolato da una API (Application Programming Interface ovvero un'Interfaccia di Programmazione di un'Applicazione), il problema è facilmente gestibile.

Android è ottimizzato per dispositivi mobili, che ovviamente hanno il requisito fondamentale della dimensione ridotta. Gli obbiettivi dei progettisti del sistema erano la massimizzazione della durata della batteria, ottimizzazione della memoria, ottimizzazione delle risorse computazionali.

## D.1 ANDROID OS

### D.1.1 *Linux Kernel*

Il sistema operativo Android [52] si basa sulla versione 2.6 di Linux [53] per i servizi centrali di sistema come la sicurezza, la gestione della memoria e dei processi, lo stack di rete ed i modelli dei driver (vedi figura 13). Il Kernel funziona anche da livello di astrazione tra l'hardware e lo stack di software. Tutte le applicazioni Android vengono eseguite in processi Linux separati, dopo aver avuto i premessi richiesti dal sistema Linux

1 Comprensiva di tutto il software necessario per un dispositivo mobile

2 L'intero stack di Android, vale a dire i moduli Linux del sistema operativo, le librerie native, il framework e gli applicativi, è completamente gratuito e modificabile. Viene distribuito sotto licenze business-friendly (Apache/MIT), in modo che chiunque possa estenderlo, modificarlo ed usarlo liberamente.

3 Software Developement Kit

4 Android non fa nessun tipo di assunzione sul tipo di dispositivo su coi verrà montato.


Figura 13.: Architettura di sistema di Android, cortesia di `http://developer.android.com`

### D.1.2 *Librerie Native Android*

Le librerie di Android sono principalmente composte da librerie C/C++ della comunità open source. Queste librerie vengono esposte sotto forma di servizi di sistema per i programmatori che vogliano usarli come funzioni senza conoscerne i dettagli implementativi a livello di application framework (vedi figura 13). Le librerie principali sono:

- **Librerie Standard di (ANSI) C**: un implementazione BSD[5] della libreria Standard di C (*libc*), ottimizzata per dispositivi basati sul sistema Linux. Alcuni esempi di servizi della libreria sono l'allocazione di memoria, la gestione dell'input/output ecc.
- **Librerie Media**: basate sulle OpenCORE [54] di PacketVideo[6], versione open source della libreria CORE$^{TM}$della stessa compagnia. Queste librerie supportano la visualizzazione (playback) e la registrazione dei formati audio e video ed immagini statiche più popolari (MPEG4, H.264, MP3, AAC, AMR, JPG, e PNG).
- **Surface Manager**: gestisce l'accesso al sottosistema di visualizzazione e compone, in modo trasparente all'utente, la grafica 2D con quella 3D di applicazioni multiple.
- **LibWebCore**: un motore per un web browser, che può essere usato sia dal browser di Android che da una vista del web incorporata in un applicativo. LibWeb [55] è una libreria/toolkit per sviluppare applicazioni Web scritte in Perl.
- **SGL**: motore grafico 2D.
- **Librerie 3D**: un'implementazione basata sulle API di OpenGL[7] [56]. Le librerie usano l'acceleratore grafico 3D, dove disponibile, e il rasterizzatore[8] altamente ottimizzato per programmi 3D incluso nella distribuzione. OpenGL è un ambiente per sviluppare grafica sia 2D che 3D, interattiva e portabile.
- **FreeType** [57]: rendering di font con tecnologia bitmap e vettoriale
- **SQLite** un motore per un database relazionale potente e leggero a disposizione di tutte le applicazioni.
- **OpenSSL** [58]: è un insieme di strumenti Open Source che implementano il Secure Sockets Layer (SSL v2/v3) ed i protocolli Transport Layer Security (TLS v1) ed infine una libreria di crittografia generica di ottimo livello.

### D.1.3 *Tempo di esecuzione di Android: Dalvik*

Il linguaggio Java [59], JDK[9] Tools [60] e le librerie Java sono gratuite, mentre la Java Virtual Machine non lo è. Dato che questo andava contro la politica

5 Berkley Software Distribution e licenza Apache/MIT che a differenza della licenza GNU non obbliga sviluppatori a ridistribuire i loro codici alla comunità

6 PacketVideo è il membro fondatore Open Handset Alliance

7 Open Graphics Library

8 Trasformatore di un oggetto grafico dalla sua descrizione vettoriale in una descrizione visuale, vale a dire pixel o punti che possano essere visualizzati su uno schermo o stampati.

9 Java Developement Kit

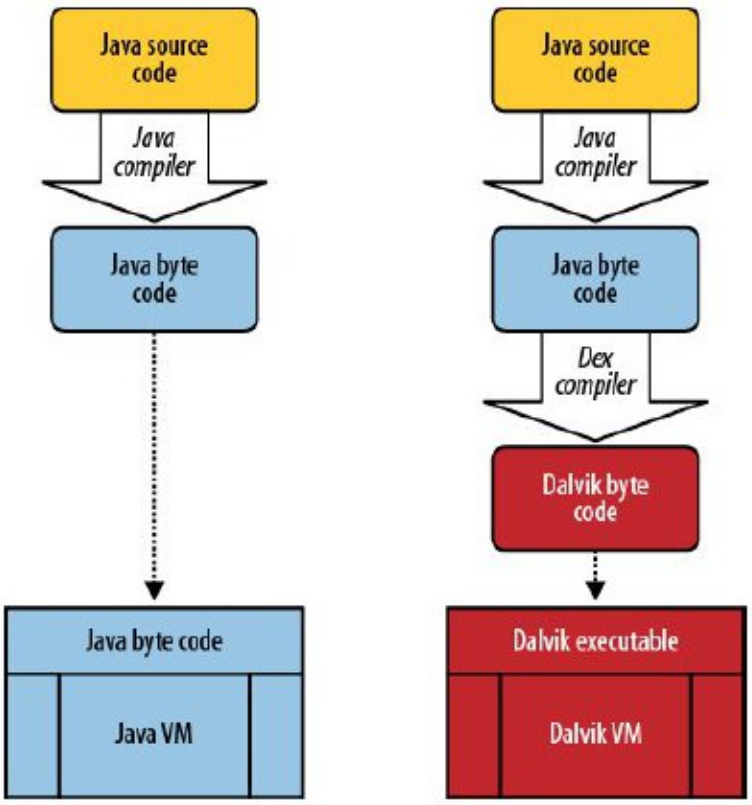


Figura 14.: Comparazione del processo di compilazione di un file Java in ambiente Android con quello classico. Immagine cortesia di [63]

del progetto, Google[10] ha sviluppato una versione ex-novo della Java Virtual Machine, ad-hoc per Android[11]. I problemi principali che il gruppo di sviluppo hanno affrontato sono quelli della durata della batteria e le risorse (memoria e ram) limitate.

### *Java e Android*

Normalmente il codice Java viene compilato e poi il bytecode viene eseguito sulla JVM, sotto Android il bytecode viene ricompilato con il compilatore Dalvik (vedi sezione D.1.3) che produce un Dalvik-bytecode detto Dex, che viene eseguito dal Dalvik VM (vedi figura 14).

Il processo è automatizzato dall'IDE[12] (Eclipse o ANT [64]) che si usa. La distribuzione Java di Android non è standard: è una variante di Java Standard Edition, in cui le Java AWT e Swing sono state sostituite da Android UI[13], appositamente ottimizzate per gli schermi e le schede grafiche di dimensioni ridotte dei dispositivi.

10 Dan Bernstein ed il team di sviluppo
11 Fino al 2005, non vi erano alternative alla JVM della Sun, poi sono nate OpenJDK [61] e Apache Harmony [62]
12 I Developement Environment
13 User Interface

### D.1.4 *Application Framework*

Questa è la parte della piattaforma che permette di sviluppare applicativi Android, fornendo servizi (sensori, posizionamento, telefonia, Wifi ecc) ed abbondante documentazione in merito.

### D.1.5 *Applications*

Le applicazioni sono i programmi sviluppati dal mondo di sviluppatori Android. Questi possono essere sia già istallati all'acquisto del dispositivo, sia scaricati dai mercati Android.

#### *APK*

Un applicazione Android è un singolo file, detto APK file. Questi ha tre componenti principali:

1. **Eseguibile Dalvik** Il codice Java compilato come descritto in figura (vedi figura 14).
2. **Risorse** Tutto cio che è in un applicativo Android, ma non è codice Java: file XML, immagini, audio, video ecc.
3. **Librerie** In un applicativo possono essere incluse librerie di codice nativo, ad esempio in C/C++

### D.1.6 *Struttura di un Android App*

Ogni applicativo per Android deve una determinata struttura di cartelle e file per funzionare. Il file più importante è l'AndroidManifest. Questo file funziona da collante e da indice per comprendere le componenti dell'applicativo. Contiene i permessi che ha come applicativo, di interagire con il resto del sistema operativo. Lavorando in ambiente di sviluppo Eclipse SDK ™, con il plugin per Android SDK Manager, la creazione di un nuovo progetto (Android Project), genera la struttura del programma:

- **src** : codice java
- **gen** : file auto generati per la gestione delle risorse
- **Android 2.2** (Libreria) : tutta la libreria di Android
- **assets**: risorse che non vengono auto indicizzate in R
- **bin**: file binario
- **AndroidManifest.xml**

## D.2 LE COMPONENTI PRINCIPALI DI UN APPLICATIVO ANDORID

Lo sviluppo di programmi (Java) per applicativi Android è necessariamente vincolato dal fatto che l'interazione dell'utente avviene mediante lo schermo del dispositivo, la durata della batteria è limitata, la capacità computazionale è ridotta ecc. Gli sviluppatori di Android hanno creato un framework per sviluppare applicativi, che risolve la maggior parte dei problemi del programmatore. L'impostazione di base del framework è quella della programmazione ad eventi, con un meccanismo di callback (riferimento a un codice) asincrono. Le componenti principali sono:

1. **Acitivity**: un'attività è la logica che gestisce una schermata singola che l'utente vede. Gli applicativi hanno di solito molteplici activity che permettono all'utente di navigare l'applicativo secondo la sua logica,
2. **Intent**: messaggi asincroni inviati tra le componenti principali,
3. **Service**: logica dell'applicativo,
4. **Content Provider**: interfaccia per lo scambio di dati tra applicativi,
5. **Broadcast Receiver**: metodo per gestire chiamate a livello di sistema in modo asincrono,
6. **Application Context**: contesto in cui tutta l'applicazione esiste.

### D.2.1 *Activity*

Ogni applicativo Android ha una *main activity*, che definisce la logica della schermata iniziale. Nell'ottica di ottimizzare le risorse del dispositivo, le activity sono state progettate in modo da consumare il minimo. Quando viene lanciata una activity, viene creato un processo Linux, viene allocato dello spazio per gli oggetti UI, costruire oggetti Java a partire dalle definizioni XML (inflation), posizionare oggetti sullo schermo. Per evitare di incorrere in questo costo ogni volta che si ricapita su una schermata, le activity sono state progettate per avere un ciclo di vita, gestito da un activity Manager. Quest'ultimo si occupa di creare, gestire e distruggere le activity, all'occorrenza. Ogni activity attraversa i seguenti stati (vedi figura 15):

1. **Starting**: l'activity non esiste in memoria. I metodi della classe `Activity` che permettono di gestire l'evento di creazione di una activity sono `onCreate()`, `onStart()` ed `onResume()` tutti per andare nello stato Running.
2. **Running**: l'activity è sullo schermo e l'utente ci sta interagendo. In ogni dato istante di tempo, può esistere solo un'activity in questo stato. Tra tutte le activity, quella nello stato Running ha la massima priorità per l'utilizzo della risorse, per minimizzare i tempi di risposta all'utente. Il metodo per gestire l'evento è `onPause` per andare nello stato Paused.
3. **Paused**: l'activity è ancora visibile, ma l'utente non vi può interagire. Non è uno stato molto comune, perché date le dimensioni ridotte dello schermo, generalmente le activity occupano tutto lo schermo o niente. Ad esempio quando appare una dialog box su una schermata, la schermata è nello stato Paused. Tutte le activity attraversano questo stato nel momento in cui vengono fermate. Queste activity sono tra quelle a priorità più alta, perché sono ancora visibili, e la transizione ad un'altra activity deve essere compiuto in modo fluido. Le callback dello stato sono `onResume()` per tornare nello stato Running e `onStop()` per andare nello stato Stopped.
4. **Stopped**: un'activity si trova in questo stato se non è più visibile ma è ancora in memoria. Queste possono essere distrutte oppure tenute in memoria per essere ripristinate nello stato Running. La seconda operazione è molto meno costosa della creazione ex-novo di un'activity. Le callback di questo stato sono le stesse dello stato Starting ed il metodo `onDestroy()` per andare nello stato Destroyed.
5. **Destroyed**: l'activity viene rimossa dalla memoria, se l'Activity Manager decide che questa non verrà usata per abbastanza tempo da rendere più conveniente la ricreazione della stessa al suo trattenimento in memoria.

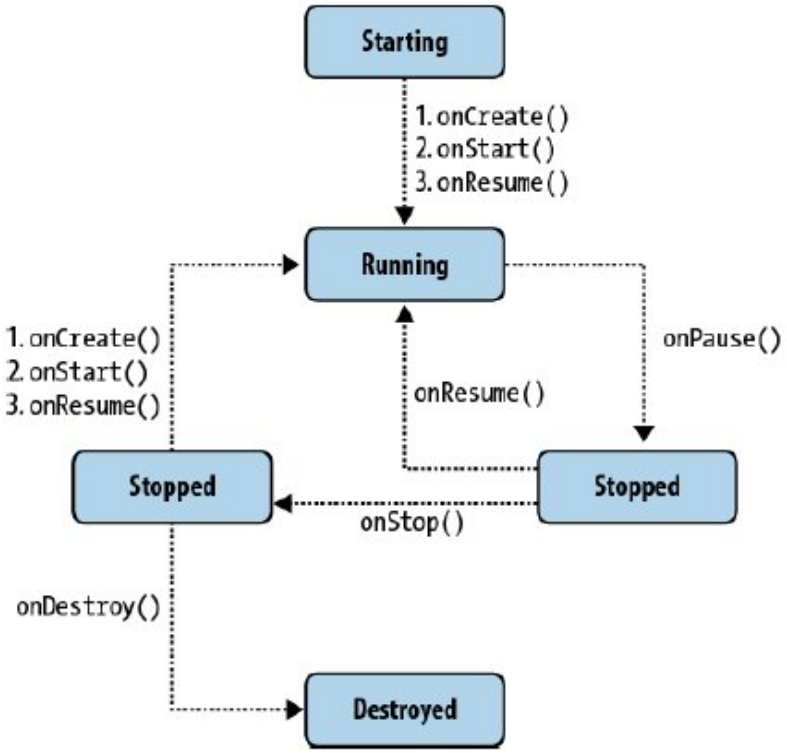


Figura 15.: Ciclo di vita di una Activity, cortesia di [63]

### D.2.2 *Intent*

Le Intent possono essere viste, come dice il nome, delle intenzioni di creare Activity che un mittente comunica. Queste potrebbero essere usate da un'Activity per creare un'altra activity, oppure per far partire un servizio o per inviare un messaggio in broadcast. Questi possono essere espliciti se il mittente dichiara il ricevente, o impliciti se il mittente dichiara solo il tipo di ricevente. Nel secondo caso ci potrebbero essere dei riceventi in competizione per l'esecuzione del messaggio, ed il sistema lascia all'utente la scelta del esecutore.

### D.2.3 *Servizi*

Questi non hanno un interfaccia utente ed il loro ciclo di vita o esecuzione è trasparente a chi utilizza il sistema. Il ciclo di vita di un servizio è molto semplice: inizialmente il servizio viene creato, ed il suo primo stato è detto Starting. Da qui le callback da usare per intercettare la transizione in Running sono `onCreate()` ed `onStart()`. Dallo stato Running con la callback `onDestroy()` il servizio va nell'ultimo stato in cui si può trovare: Destroyed.
I service che sono particolarmente impegnativi dal punto di vista computazionale dovrebbero essere eseguiti su un proprio thread, eseguibile in background, e non quello della UI, in modo da non rallentare l'interfaccia grafica.

### D.2.4 *Content Provider*

Android, per ragioni di sicurezza, esegue ogni applicativo nella propria 'sandbox' compartimento stagno, in modo da confinare i dati usati da un programma a

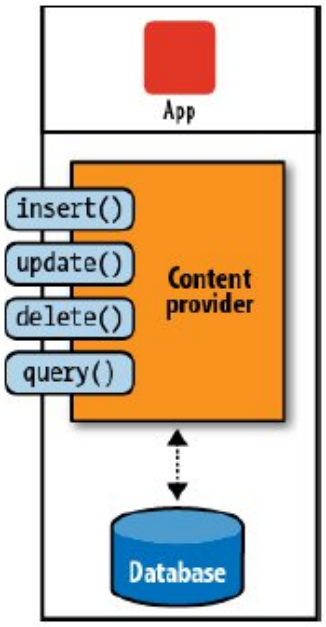


Figura 16.: CRUD di un Content Provider, cortesia di `http://developer.android.com`

quest'ultimo. Mediante gli Intent è possibile scambiare piccole quantità di dati tra applicativi diversi, la condivisione di quantità ingenti di dati persistenti viene fatta tramite i Content Provider. Per facilitare il compito questo componente aderisce all'interfaccia CRUD: il Content Provider è interfacciato ad una base di dati ed implementa i metodi `insert()`, `delete()`, `update()`, `query()`.

### D.3 BROADCAST RECEIVER

Implementazione del pattern Observer (tipo particolare del protocollo Publish/Subscribe) in cui c'è un servizio di prenotazione su un certo evento. Un programma si registra al servizio e nel momento in cui viene lanciato l'evento per il quale si è registrato, il codice viene lanciato. Il sistema operativo lancia eventi in broadcast in continuazione: il sistema è stato avviato, la batteria è scarica, un sms è in arrivo ecc. Ciascuno di questi eventi scatena il lancio dei programmi registrati, o per usare il nome del pattern, i programmi che osservavano l'evento.

### D.4 APPLICATION CONTEXT

Il contesto di un'applicativo Android è l'ambiente in cui i processi con tutti le componenti vengono eseguiti. Il ciclo di vita di un contesto parte con la sua creazione al lancio dell'applicativo, e termina nel momento in cui questi viene terminato. Per avere un riferimento al contesto è sufficiente chiamare `Context.getApplicationContext()` oppure `Activity.getApplication()`

### D.5 INTEFACCIA UTENTI (UI)

[ *TODO: Completare* ] `http://developer.android.com/guide/topics/ui/index.html`

### D.5.1 *Layout XML*

### D.5.2 *Eventi di Input*

### D.5.3 *Menu*

Per default ogni Activity ha un menu di opzioni o azioni, a cui l'utente può accedere premendo un taso fisicamente disegnato sullo schermo.

### D.5.4 *Barra delle Azioni*

### D.5.5 *Dialoghi*

# E

## @TODO LIST

Instructions on how to use this list: instead of writing a TODO right in the middle of the thesis, I should write it in this list and refer to it with a label such as `label{TODO:keyword}`where keyword has to be unique.

- Collect all TODOs.
- Stop using the `eng_to_ita` file. Define a word the first time and use it with an acronym if possible.
- Insert relevant words in TeXnicCenter's dictionary so that I can notice any typing mistakes.
- Tableofcontents: make parts centered, capters bold.

[1] *kinesiology*, 2011. [Online]. Available: http://www.merriam-webster.com/dictionary/kinesiology (Cited on page 7.)

[2] S. J. Hall, *Basic Biomechanics*. McGraw-Hill, 2006. (Cited on page 7.)

[3] (2011) kinematics. [Online]. Available: http://www.britannica.com/EBchecked/topic/318099/kinematics (Cited on page 7.)

[4] M. Hildebrand, "The quadrupedal gaits of vertebrates," *BioScience*, vol. 39, pp. 766–775, 1989. (Cited on page 9.)

[5] W. E. Weber and E. H. Weber, *Mechanik der Menschlichen Gehwerkzauge, The Mechanics of Human Motion*. Gottinger, Gottingen, 1836. (Cited on page 9.)

[6] G. M. Lyons, R. T. Sinkjaer, J. H. Burridge, and D. J. Wilcox, "A review of portable fes-based neural orthosis for the correction of drop foot." *IEEE Transactions on Neural Systems and Rehabilitation Engeneering*, vol. 10, pp. 260–279, 2002. (Cited on pages 15 and 16.)

[7] P. Strojnik, A. Kralj, and I. Ursic, "Programmed six-channel electrical stimulator for complex stimulation of leg muscles during walking," *IEEE Transactions on Biomedical Engineering*, vol. BME-26, pp. 112–116, 1979. (Cited on page 15.)

[8] M. M. Skelly and H. J. Chizeck, "Real-time gait event detection for paraplegic fes walking," *IEEE Engineering in Medicine and Biology Society*, vol. 9, pp. 59–68, 2001. (Cited on pages 16 and 18.)

[9] J. Nilsson, V. P. Stokes, and A. Thorstensson, "A new method to measure foot contact," *Journal of Biomechanics*, vol. 18, pp. 625–627, 1985. (Cited on page 16.)

[10] G. M. L. A. Mansfield, "The use of accelerometry to detect heel contact events for use as a sensor in fes assisted walking," *Medical Engineering Physics*, vol. 25, pp. 879–885, 2003. (Cited on pages 16 and 17.)

[11] Y. Shimada, S. Ando, T. Matsunaga, A. Misawa, T. Aizawa, T. Shirahata, and E. Itoi, "Clinical application of acceleration sensor to detect the swing phase of stroke gait in functional electrical stimulation," *The Tohoku Journal of Experimental Medicine*, vol. 207(3), pp. 197–202, 2005. (Cited on pages 16, 17, and 18.)

[12] B. Coley, B. Najafi, A. Paraschiv-Ionescu, and K. Aminian, "Stair climbing detection during daily physical activity using a miniature gyroscope," *Gait & Posture*, vol. 22(4), pp. 287–294, 2005. (Cited on page 16.)

[13] K. Tong and M. H. Granat, "A practical gait analysis system using gyroscopes," *Medical Engineering & Physics*, vol. 21(2), pp. 87–94, 1999. (Cited on page 16.)

[14] S. Miyazaki, "Long-term unrestrained measurement of stride length and walking velocity utilizing a piezoelectric gyroscope," *IEEE Transactions on Bio-Medical Engineering*, vol. 44(8), pp. 753–759, 1997. (Cited on page 16.)

[15] K. Aminian, B. Najafi, C. Bula, P. F. Leyvraz, and P. Robert, "Spatio-temporal parameters of gait measured by an ambulatory system using miniature gyroscopes," *Journal of Biomechanics*, vol. 35(5), pp. 689–99, 2002. (Cited on pages 16 and 18.)

[16] R. Williamson and B. J. Andrews, "Gait event detection for fes using accelerometers and supervised machine learning," *IEEE Transactions on Rehabilitation Engineering*, vol. 8(3), pp. 312 – 319, 2000. (Cited on pages 16, 17, and 18.)

[17] R. E. Mayagoitia, A. V. Nene, and P. H. Veltink, "Accelerometer and rate gyroscope measurement of the kinematics: an inexpensive alternative to optical motion analysis systems," *Journal of Biomechanics*, vol. 35(4), pp. 537–542, 2002. (Cited on pages 16, 17, and 18.)

[18] I. P. Pappas, M. R. Popovic, T. Keller, V. Dietz, and M. Morari, "A reliable gait phase detection system," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 9(2), pp. 113–125, 2001. (Cited on pages 17 and 24.)

[19] ——, "A reliable gyroscope-based gait-phase detection sensor embedded in a shoe insole," *Ieee Sensors Journal*, vol. 4(2), pp. 268–274, 2004. (Cited on pages 17 and 18.)

[20] J. J. Kavanagh and H. B. Menz, "Accelerometry: a technique for quantifying movement patterns during walking," *Gait & Posture*, vol. 28(1), pp. 1–15, 2008. (Cited on page 17.)

[21] J. R. W. Morris, "Accelerometry-a technique for the measurement of human body movements," *Journal of Biomechanics*, vol. 6(6), pp. 729–732, 1973. (Cited on page 17.)

[22] A. T. Willemsen, J. A. van Alsté, and H. B. Boom, "Real-time gait assessment utilizing a new way of accelerometry," *Journal of Biomechanics*, vol. 23(8), pp. 859–863, 1990. (Cited on page 17.)

[23] A. T. Willemsen, C. Frigo, and H. B. K. Boom, "Lower extremity angle measurement with accelerometers-error and sensitivity analysis," *IEEE Transactions on Biomedical Engineering*, vol. 38(12), pp. 1186–1193, 1991. (Cited on page 17.)

[24] M. D. Duric, S. Dosen, M. Popovic, and D. B. Popovic, "Sensors for assistive system for restoring of the walking," *Proceedings 52nd ETRAN Conference, Society for Electronics, Telecommunications, Computers, Automatic Control and Nuclear Engineering*, vol. 63(1), pp. 978–986, 2008. (Cited on page 17.)

[25] K. Liu, T. Liu, K. Shibata, Y. Inoue, and R. Zheng, "Novel approach to ambulatory assessment of human segmental orientation on a wearable sensor system," *Journal of Biomechanics*, vol. 42(16), pp. 2747–1752, 2009. (Cited on page 17.)

[26] M. D. Duric, "Automatic recognition of gait phases from accelerations of leg segments," *9th Symposium on Neural Network Applications in Electrical Engineering*, vol. 63(1), pp. 121–124, 2008. (Cited on page 17.)

[27] H. Dejnabadi, B. M. Jolles, and K. Aminian, "A new approach to accurate measurement of uniaxial joint angles based on a combination of accelerometers and gyroscopes," *IEEE Transactions on Biomedical Engineering*, vol. 52(8), pp. 1478–1484, 2005. (Cited on page 17.)

[28] A. M. Sabatini, C. Martelloni, S. Scapellato, and F. Cavallo, "Assessment of walking features from foot inertial sensing," *IEEE Transactions on Biomedical Engineering*, vol. 52, pp. 486–494, 2005. (Cited on pages 17, 18, and 25.)

[29] J. M. Jasiewicz, J. H. Allum, J. W. Middleton, A. Barriskill, P. Condie, B. Purcell, and R. C. Li, "Gait event detection using linear accelerometers or angular velocity transducers in able-bodied and spinal-cord injured individuals," *Gait & Posture*, vol. 24(4), pp. 502–509, 2006. (Cited on pages 17 and 18.)

[30] T. K. Lau H, "The reliability of using accelerometer and gyroscope for gait event identification on persons with dropped foot," *Gait & Posture*, vol. 27(2), pp. 248–257, 2008. (Cited on pages 17 and 18.)

[31] T. Liu, Y. Inouea, and K. Shibata, "Development of a wearable sensor system for quantitative gait analysis," *Measurement*, vol. 42(7), pp. 978–988, 2009. (Cited on page 17.)

[32] G. Wu and Z. Ladin, "The study of kinematic transients in locomotion using the integrated kinematic sensor," *IEEE Transactions on Rehabilitation Engineering*, vol. 4(3), pp. 193–200, 1996. (Cited on pages 17 and 18.)

[33] P. H. Veltink, P. Slycke, J. Hemssems, R. Buschman, G. Bultstra, and H. Hermens, "Three dimensional inertial sensing of foot movements for automatic tuning of a two-channel implantable drop-foot stimulator," *Medical Engineering and Physics*, vol. 25(1), pp. 21–28, 2003. (Cited on page 17.)

[34] R. Williamson and B. J. Andrews, "Sensor systems for lower limb functional electrical stimulation (fes) control," *Medical Engineering & Physics*, vol. 22(5), pp. 313–325, 2000. (Cited on page 18.)

[35] K. J. O'Donovan, R. Kamnik, D. T. O'Keeffe, and G. M. Lyons, "An inertial and magnetic sensor based technique for joint angle measurement," *Journal of Biomechanics*, vol. 40(12), pp. 2604–2611, 2007. (Cited on page 18.)

[36] S. W. Lee, K. Mase, and K. Kogure, "Detection of spatio-temporal gait parameters by using wearable motion sensors," *IEEE Engineering Medical Biol Soc.*, vol. 7, pp. 6836–6839, 2005. (Cited on page 18.)

[37] M. Hanlon and R. Anderson, "Real-time gait event detection using wearable sensors," *Gait & Posture*, vol. 30(4), pp. 523–527, 2009. (Cited on pages 18 and 19.)

[38] C. M. O'Connor, S. K. Thorpe, M. J. O'Malley, and C. L. Vaughan, "Automatic detection of gait events using kinematic data," *Gait & Posture*, vol. 25(3), pp. 469–474, 2007. (Cited on page 18.)

[39] C. A. Kirkwood, B. J. Andrews, and P. Mowforth, "Automatic detection of gait events: a case study using inductive learning techniques," *Journal of Biomedical Engineering*, vol. 11(6), pp. 511–516, 1989. (Cited on page 18.)

[40] P. C. Sweeney, G. M. Lyons, and P. H. Veltink, "Finite state control of functional electrical stimulation for the rehabilitation of gait," *MEDICAL AND BIOLOGICAL ENGINEERING AND COMPUTING*, vol. 38(2), pp. 121–126. (Cited on page 18.)

[41] T. Pfau, M. Ferrari, K. Parsons, and A. Wilson, "A hidden markov model-based stride segmentation technique applied to equine inertial sensor trunk movement data," *Journal of Biomechanics*, vol. 41, pp. 216–220, 2008. (Cited on page 23.)

[42] J. Lester, T. Choudhury, and G. Borriello, "A practical approach to recognizing physical activities," *Pervasive Computing*, 2006. (Cited on page 23.)

[43] V. M. Systems. (1989) Vicon, oxford uk. [Online]. Available: http://www.vicon.com/products/system.html (Cited on page 24.)

[44] K. Murphy. (1998) Hidden markov model (hmm) toolbox for matlab. [Online]. Available: http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html (Cited on page 27.)

[45] X. R. Julien Blot, "Short-time viterbi for online hmm decoding : Evaluation on a real-time phone recognition task," *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference*, pp. 1–8, 4/4/2008. (Cited on pages 29, 57, and 58.)

[46] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *PROCEEDINGS OF THE IEEE*, vol. 77, no. 2, pp. 257–287, 1989. (Cited on page 51.)

[47] A. Seward, "Low latency incremental speech transcription in the synface project," *EUROSPEECH*, vol. 2003, pp. 1141–1145, 2003. (Cited on page 57.)

[48] M. Ryynanen and A. Klapuri, "Automatic bass line transcription from streaming polyphonic audio," *ICASSP 2007*, vol. IV, pp. 1437–1440, 2007. (Cited on page 57.)

[49] H. Ardö, K. Åström, and R. Berthilsson, "Real time viterbi optimization of hidden markov models for multi target tracking," *IEEE Workshop on Motion and Video Computing*, pp. 1–8, Feb-2007. (Cited on page 57.)

[50] L. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of the probabilisti functions of markov chains," *The Annals of Mathematical Statistics*, vol. 41, pp. 164–171, 1970. (Cited on page 60.)

[51] R. G. Douglas, *Clasical Mechanics*. Cambridge University Press, 2006. (Cited on page 63.)

[52] Google. (2011) Android developers. [Online]. Available: http://developer.android.com (Cited on page 71.)

[53] I. Linux Kernel Organization. (2003) Linux kernel 2.6, 17/12/2003. [Online]. Available: http://www.kernel.org/pub/linux/kernel/v2.6/ (Cited on page 71.)

[54] P. V. Inc. (2007) Opencore. [Online]. Available: http://www.packetvideo.com/press_releases/11_05_2007.html (Cited on page 73.)

[55] C. Kong. (2000) Libweb 0.02 a perl library/toolkit for building web applications. [Online]. Available: http://cpan.uwinnipeg.ca/htdocs/LibWeb/ (Cited on page 73.)

[56] OpenGL.org. (Relese 4.2 August 2011) The industry's foundation for high performance graphics from games to virtual reality, mobile phones to supercomputers. [Online]. Available: http://www.opengl.org/documentation (Cited on page 73.)

[57] D. Turner, R. Wilhelm, W. Lemberg, and the FreeType contributors. (Relese 2.4.7 October 2011) The free type project. [Online]. Available: http://www.freetype.org/ (Cited on page 73.)

[58] E. A. Young, T. J. Hudson., and the OpenSSL community. (Relese 2.4.7 October 2011) Cryptogpaphy and ssl/tls toolkit. [Online]. Available: http://www.openssl.org/ (Cited on page 73.)

[59] ——. (Relese 7 October 2011) Oracle. [Online]. Available: http://www.java.com/ (Cited on page 73.)

[60] O. Corporation. (Relese October 2011) Oracle. [Online]. Available: http://download.oracle.com/javase/1.5.0/docs/tooldocs/ (Cited on page 73.)

[61] ——. (Bylaws Relese October 2011) Open source imlementation of the java platform, standard edition and related projects. [Online]. Available: http://openjdk.java.net/ (Cited on page 74.)

[62] A. Harmony™. (Resese 5.0, 2010) Open source java se. [Online]. Available: http://openjdk.java.net/ (Cited on page 74.)

[63] M. Gargenta, *Learning Android*. O'Reilly Media, 2011. (Cited on pages 74 and 77.)

[64] A. S. Foundation. (Release 1.8.2 December 2010) The apache ant project. [Online]. Available: http://ant.apache.org/ (Cited on page 74.)