Alma Mater Studiorum - Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Ingegneria e Scienze Informatiche

# GESTURE RECOGNITION: UNA PANORAMICA

*Relazione Finale in:*
Programmazione dei Sistemi Embedded

*Relatore:*
Prof. Alessandro Ricci

*Presentata da:*
Mattia Semprini

Sessione di Laurea Unica
Anno Accademico 2016 - 2017

**Sommario**

Per decenni, l'uomo ha interagito con i calcolatori e altri dispositivi quasi esclusivamente premendo i tasti e facendo "click" sul mouse. Al giorno d'oggi, vi è un grande cambiamento in atto a seguito di una ondata di nuove tecnologie che rispondono alle azioni più naturali, come il movimento delle mani o dell'intero corpo. Il mercato tecnologico è stato scosso in un primo momento dalla sostituzione delle tecniche di interazione standard con approcci di tipo *"touch and motion sensing"*; il passo successivo è l'introduzione di tecniche e tecnologie che permettano all'utente di accedere e manipolare informazioni interagendo con un sistema informatico solamente con gesti ed azioni del corpo. A questo proposito nasce la *Gesture Recognition*, una parte sostanziale dell'informatica e della tecnologia del linguaggio, che ha come obbiettivo quello di interpretare ed elaborare gesti umani attraverso algoritmi informatici.

In questa trattazione andrò a spiegare, nei primi due capitoli la storia delle tecnologie Wearable dai primi orologi che non si limitavano alla sola indicazione dell'orario fino alla nascita dei sistemi utilizzati al giorno d'oggi per la Gesture Recognition. Segue, nel terzo capitolo, un'esposizione dei più utilizzati algoritmi di classificazione delle *gesture*.

Nel quarto andrò ad approfondire uno dei primi framework progettati per fare in modo che lo sviluppatore si concentri sull'applicazione tralasciando la parte di codifica e classificazione delle gesture.

Nell'ultima parte verrà esaminato uno dei dispositivi più performanti ed efficaci in questo campo: il Myo Armband. Saranno riportate anche due studi che dimostrano la sua validità.

# Indice

# Capitolo 1

# Introduzione

## 1.1 I sistemi Wearable

Prima di parlare dei dispositivi indossabili occorre dare alcune informazioni basilari riguardanti la grande famiglia dei *dispositivi embedded* di cui fanno parte i *dispositivi wearable*.

I dispositivi embedded (letteralmente *incorporato/immerso*) sono tutti quei sistemi elettronici di elaborazione digitale progettati appositamente per una determinata applicazione. La piattaforma hardware viene ingegnerizzata ad hoc e quindi non è utilizzabile per altri scopi. Essa è parte del sistema ed è in grado di controllare e gestire tutte le funzioni richieste. In questi sistemi si trova un microprocessore *"special purpose"* basato su una tecnologia di tipo *RISC* (acronimo di: *Reduced Instruction Set Computer*). Essi vengono ideati per eseguire un set di istruzioni semplici e lineari, a differenza di quelli *"general purpose"* pensati sul tipo CISC (Acronimo di: *Complex Instruction Set Computer*), che ne prevedono un insieme più esteso, utilizzati nei comuni pc desktop, laptop ecc.

Di questo grande gruppo di sistemi informatici fanno parte i dispositivi *wearable*. Essi, come dice la parola, sono sistemi indossabili finalizzati all'emissione di informazioni finite oppure alla raccolta di dati provenienti dall'ambiente esterno o dal corpo dell'utente con l'ausilio di sensori come accelerometro, giroscopio, magnetometro, sensore EMG ecc. Generalmente queste informazioni vengono trasmesse tramite protocolli di comunicazione come il Bluetooth ad un apparato esterno incaricato di elaborarne l'output come smartphone o ad altri dispositivi wearable (ad esempio smart glasses).

I dispositivi wearable necessitano una progettazione totalmente diversa rispetto a quella dei comuni sistemi informatici. Data la natura del compito

che devono svolgere infatti, avranno caratteristiche specifiche dipendenti dal campo di utilizzo. Tuttavia a prescindere da esso tutti i dispositivi devono rispettare questi requisiti:

- **Efficienza,** a livello di efficienza si intende:
  - **a)** Di dimensioni e peso ridotti al fine di non arrecare fastidio o ingombro all'utente;
  - **b)** Devono essere performanti, non solo dal punto di vista della velocità di elaborazione dei dati ma anche della facilità d'uso;
  - **c)** A ridotto consumo energetico al fine di massimizzare il tempo operativo minimizzandone anche l'emissione di calore;
- **Affidabilità,** come affidabilità si intende:
  - **a)** Gli utenti devono avere accesso ad informazioni il più possibile attendibili.
  - **b)** L'utente che li utilizza dovrebbe poter dipendere dal dispositivo, soprattutto in specifici campi (ad esempio quello dell' *"Health Care"*).

## 1.2 Evoluzione dei dispositivi indossabili

Creata nei primi anni '60, questo tipo di tecnologia inizialmente è stata inventata per aiutare i giocatori dei casinò a vincere in modo facile e con maggiori probabilità. I primi computer indossabili avevano, infatti, una funzione piuttosto illegale ed erano utilizzati da una piccola cerchia ristretta di persone. Con il tempo la tecnologia indossabile si è evoluta acquistando una funzionalità più nobile e negli anni '80, grazie ad aziende come Casio, IBM ecc., sono iniziate le prime sperimentazioni di orologi con funzionalità aggiuntive come il cronometro, la calcolatrice con memorie accessibili anche se estremamente piccole rispetto a quelle odierne.

In particolare nel 1984 l'azienda giapponese Seiko Epson rilasciò il primo computer portatile indossabile, un embrione dei moderni smartwatch con un semplice sistema costruito su un chip.

Nel 1987 nacquero i primi apparecchi acustici digitali creati in aiuto alle persone con problemi di udito, portando per la prima volta i dispositivi wearable nella vita quotidiana anche delle persone non interessate attivamente al settore.

L'azienda statunitense *"Reflection Tecnology"* nel 1989 commercializzò il **primo *smart glass*** denominato *"Private Eye"* [1]. Esso utilizza una tecnologia basata su una matrice di led per creare uno schermo monocromatico con risoluzione 720x280, un vetro vibrante scannerizza in sequenza ogni colonna formata da 280 led, che accendendosi ad intermittenza danno forma a tutti i pixel dello schermo.

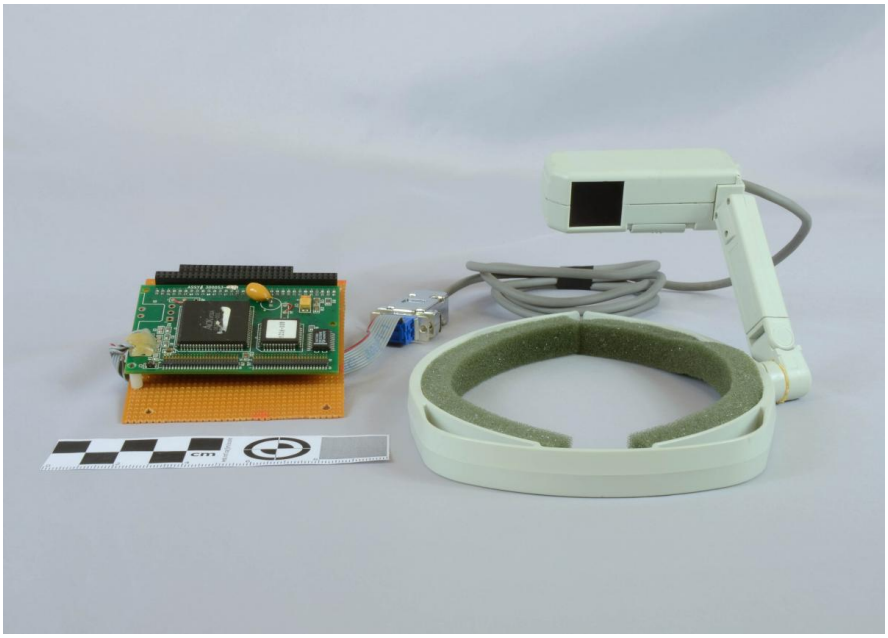


Figura 1.1: Private Eye, il primo prototipo di smartglass

Nel corso degli anni '90 vennero concepite e commercializzate le prime telecamere indossabili, una webcam collocata su una fascetta da mettere in fronte finalizzata a riprese video.

Negli primi anni 2000 sono stati introdotti i dispositivi Bluetooth per la comunicazione senza fili e gli apparecchi medici per il fitness, in grado di leggere i battiti cardiaci e il monitoraggio del proprio corpo durante una seduta di sport.

Nell'ultimo decennio abbiamo assistito ad una notevole impennata dell'interesse a queste tecnologie da parte delle aziende operanti in questo settore che ha portato ad una crescita esponenziale delle vendite, la quale ha raggiunto qualsiasi tipo di utenza in maniera capillare. Qui Figura 1.2 si può vedere un'analisi effettuata nel 2016 da parte di Tractica [2] dove mostra una previsione di crescita che và da 118 milioni di unità nel 2016 a 430 milioni di unità nel 2022, ovvero in soli sei anni quasi la quadruplicazione del mercato.

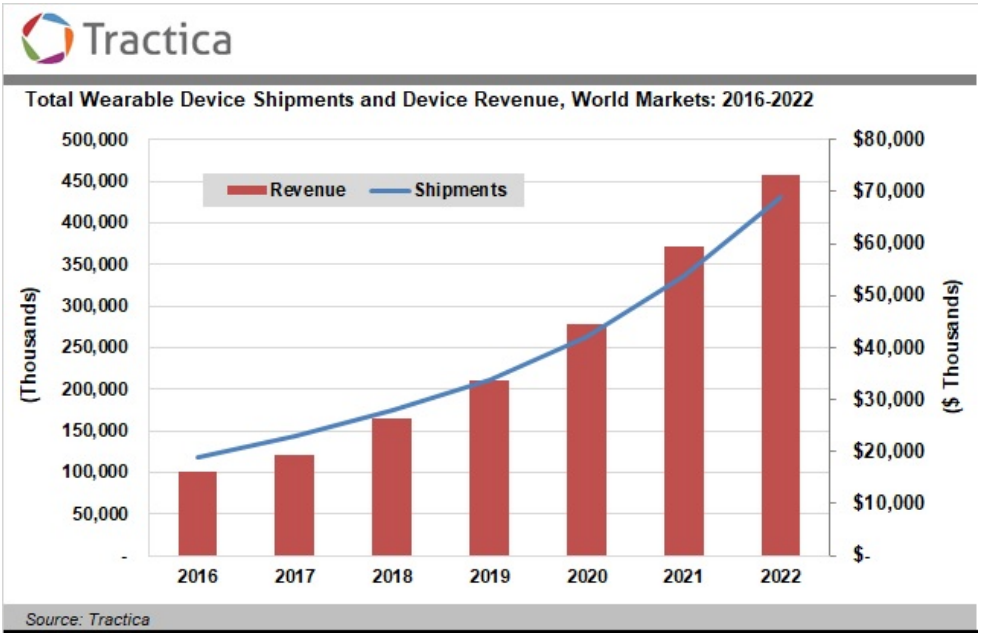


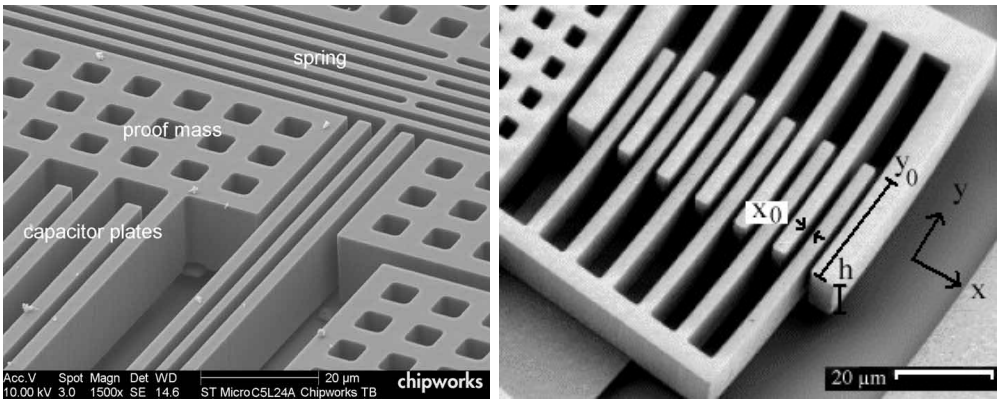

Figura 1.2: Analisi Tractica

Andando ad analizzare nel dettaglio questa rivoluzione, il fulcro iniziale è stato sicuramente l'avvento degli smartphone, le prime aziende che credettero di più in questo settore furono Google, con il sistema operativo Android (2008), ed Apple con il suo iOS (2007) le quali riuscirono a fondere lo stato dell'arte dei dispositivi embedded con quello che erano i sistemi operativi presenti. Queste aziende si sono dovute confrontare con tutte le difficoltà tutt'ora rilevanti derivate dalla progettazione di dispositivi che devono essere prestanti senza eccessivi consumi energetici e soprattutto, nel caso di Android, interfacciarsi con hardware differenti.

Il contributo fondamentale nello sviluppo degli smartphone e di conseguenza anche dei dispositivi wearable è stato portato dall'aggiunta dei vari sensori inerziali quali, accelerometro, giroscopio, magnetometro, gps ecc. e la loro fusione nel cosiddetto *sensor fusion*. Esso è la combinazione di dati pro-

venienti da fonti differenti al fine di aumentare la precisione delle informazioni derivanti dai segnali.

Questa aggiunta è stata resa possibile solo dall'introduzione della cosiddetta tecnologia MEMS (acronimo per *Micro Electro-Mechanical Systems*). Questi micro-sistemi sono un insieme di dispositivi meccanici ed elettronici in forma microscopica, integrati su un substrato di materiale semiconduttore come il silicio, i quali fondono le proprietà elettriche con quelle meccaniche permettendo di creare sensori che misurano grandezze fisiche (come l'accelerazione, il campo elettromagnetico, ecc.) su scala micrometrica.




(a) (b)

Figura 1.3: Accelerometri micrometrici (MEMS)

Gli smartphone non hanno comportato solamente un'evoluzione dal punto di vista dell'hardware ma hanno portato ad un notevole cambiamento nello sviluppo del campo software. La programmazione di questi dispositivi richiede, infatti, uno studio diverso dai normali sistemi informatici dato che è fortemente orientata agli eventi ed al multithreading.

Ritornando a parlare dei dispositivi wearable in generale, le loro applicazioni pratiche sono molteplici, che spaziano da compiti generici, come può essere per gli smartwatch (che hanno oramai raggiunto a livello prestazionale gli smartphone), a compiti molto specifici:

- **HEALTHCARE:** esistono già da parecchio tempo dispositivi in grado di tenere sotto controllo i parametri vitali di un paziente. Alcuni esempi di questi dispositivi sono gli strumenti in grado di tener traccia e rappresentare graficamente l'attività elettrica del cuore (elettrocardiogramma), il movimento dei muscoli e dei nervi (elettromiografia) ed

infine dell'attività elettrica del cervello (elettroencefalogramma). Tuttavia l'ambito dei dispositivi appena citati è molto legato ad un ambiente di tipo ospedaliero, occorre quindi citare altri dispositivi, sempre in questo ambito, che possono venir utilizzati quotidianamente anche all'esterno di strutture specifiche. Di particolare importanza, un prodotto in via di sviluppo denominato *"K'Watch Glucose"* [3], un dispositivo in grado di tenere traccia del livello di zucchero nel sangue senza dover prelevare neanche una goccia di sangue.

- **SPORT:** dispositivi in grado di memorizzare e creare statistiche in base agli spostamenti dell'utente, fornendogli informazioni dettagliate sugli allenamenti. Il dispositivo ad oggi più famoso e più venduto è sicuramente il *"Fitbit"* [4].

- **INTRATTENIMENTO:** di questa grande famiglia fanno parte le interfacce di input *"hands-free"*, letteralmente mani libere, con le quali è possibile interagire con una macchina attraverso mezzi naturali, come la voce *(Speech Recognition)* o i gesti *(Gesture Recognition)*. Queste ultime possono anche essere utilizzate in ambiti di realtà aumentata (*"mixed reality"*), in cui l'utente tramite l'utilizzo di smart glasses, non è isolato completamente dalla realtà che tuttavia viene arricchita con informazioni di vario tipo. Allo stesso modo possono venire utilizzati nella realtà virtuale (*"virtual reality"*), dove al contrario l'utilizzatore, tramite l'ausilio di appositi visori VR (Oculus Rift Figura 1.4(b), Playstation VR, HTC Vive Figura 1.4(a), viene completamente immerso in una realtà creata ad hoc.


(a) (b)

Figura 1.4: Visori per la Realtà Aumentata

## 1.3 Gesture Recognition

L'interazione fra uomo e computer (dove per computer si intende un qualsiasi strumento computazionale) tradizionalmente avviene tramite mouse e tastiera per la parte input; monitor e altoparlanti per la parte di output. L'utilizzo di questi strumenti presuppone un certo livello di coordinazione vista-movimento delle mani che in particolari casi, come per portatori di *handicap*, non è possibile avere. Proprio l'impossibilità di un contatto diretto, per svariate motivazioni, ha portato allo studio di una via alternativa e meno impegnativa di interagire con i sistemi elettronico-informatici.

Questa disciplina vede i suoi antenati nei primi anni '80 con i primi esperimenti sullo *Speech Recognition* (riconoscimento vocale), in cui si cercava di trasformare la voce umana in informazioni scritte automaticamente da un computer. Successivamente, grazie ai progressi tecnologici negli anni '90, questa materia si espanse anche allo studio dei gesti umani come nuova tecnologia di input, nacquero quindi i primi sistemi in grado di riconoscere ed interpretare il linguaggio dei segni.

La *Gesture Recognition* (riconoscimento delle gesture) come si può dedurre dal nome, si pone come obbiettivo l'interpretazione dei gesti umani provenienti, non solo dalle mani ma dall'intero corpo.

Un sistema basato su di essa deve essere programmato contestualizzandolo in base all'ambiente in cui deve operare, ottimizzando in questo modo la trasformazione dei dati in informazioni utili. Ad oggi, quasi 20 anni dopo la nascita di questa disciplina, la ricerca sta ancora continuando attivamente, attualmente è focalizzata nello studio dei comportamenti umani attraverso l'identificazione e il riconoscimento della postura, dell'andatura, della prossemica e dell'emozioni.

Il settore che più di tutti ha permesso lo sviluppo e la diffusione capillare della Gesture Recognition è stato quello videoludico: la *Nintendo* si cimentò per prima con la console *"Wii"* tramite il controller *"Wii Remote"* Figura 1.5(a) dimostrando alle persone "comuni" modi alternativi per l'interazione con un dispositivo elettronico. In seguito altre aziende del settore iniziarono a seguire questo trend, nacque quindi il *"Playstation Move"*, un dispositivo simile al Wii Remote, da parte dell'azienda nipponica *Sony* e il *"Kinect"* Figura 1.5(b), che a differenza dei primi due è un dispositivo composto da due telecamere di diverso tipo una normale e una ad infrarossi, da parte di *Microsoft*. La comunità scientifica ha visto in quest'ultimo grandi potenzialità da poter impiegare in campi diversi da quello videoludico come ad esempio "Looking at people" e appunto della "Gesture Recognition". Conseguentemente nel febbraio 2011 Microsoft rilasciò pubblicamente il *Software Development Kit (SDK)*.


(a) (b)

Figura 1.5: I due dispositivi più famosi

Qui sotto si può trovare una lista non esaustiva di possibili applicazioni:

- **Smart surveillance (sorveglianza intelligente):** tramite l'utilizzo di apposite telecamere, la creazione di un sistema di videosorveglianza intelligente, che non si limita al semplice *motion-detect*. Lo scopo è quello, in primo luogo, di rilevare se è presente un soggetto nell'inquadratura, successivamente registrarne i movimenti e da essi capire le sue intenzioni. Un sistema intelligente che riesce a "comprendere" e decidere qualora far scattare o meno l'allarme.

- **Metodi di input innovativi:** possono essere utilizzati per sostituire i classici mouse e tastiera come metodo per passare informazioni ad un calcolatore. In questo modo anche le persone con deficit motori possono facilmente interagire con il proprio pc. Un ulteriore utilizzo può portare al controllo di una casa domotica.

- **Realtà virtuale:** per l'interazione con oggetti in un ambiente virtuale. Esistono già al giorno d'oggi chat room virtuali come *"VR CHAT"* [5] nei quali è possibile interagire dinamicamente con l'ambiente e le altre persone presenti.

- **Gaming:** come già spiegato, sono in sviluppo costante nuovi dispositivi e modi sempre più innovativi per interagire con i videogiochi. Un dispositivo degno di nota è il *"leap motion"*, grazie ad esso è possibile avere una trasposizione virtuale delle mani all'interno dell'ambiente virtuale.

Figura 1.6: Il dispositivo Leap Motion con trasposizione delle mani

Il riconoscimento dei gesti rappresenta l'interfaccia utilizzata dagli utenti per interagire con i sistemi informatici.

I sensori sono il mezzo utilizzato per colmare il divario tra mondo fisico e sistema informativo, mentre gli algoritmi sono la chiave per decodificare e regolare l'interazione dell'utente con il sistema. Tra i gesti utilizzati per interagire con un sistema informatico, i gesti delle mani sono il modo naturale preferito di controllo e interazione degli esseri umani, sia *uomo-uomo* che *uomo-macchina*; questi infatti rappresentano un'interazione fisica veloce ed efficace per ricevere informazioni e inviare comandi ad un oggetto.

# Capitolo 2

# Tecniche e tecnologie utilizzate nella Gesture Recognition

## 2.1 Tecnologie

Prima di parlare dei dispositivi che vengono utilizzati per il riconoscimento gesture, occorre citare i vari sensori che sono in grado di generare segnali sulla base dello spostamento del corpo umano, i quali verranno impiegati per la fase di riconoscimento.

Questi sensori come già detto si basano su una tecnologia *MEMS*, taluni sono presenti in quasi la totalità dei dispositivi, altri invece sono più particolari e quindi vengono montati principalmente in device specifici al riconoscimento.

Ecco una lista dei sensori più comunemente utilizzati:

- **Giroscopio:** è un sensore che viene ampiamente utilizzato in grado di misurare le variazioni della velocità angolare ($^\circ/s$) di un dato oggetto oppure, come nel caso della Gesture Recognition, di un movimento del corpo umano.
  Il giroscopio viene caratterizzato dal numero di assi in grado di misurare, si va da quelli più economici ad un solo asse, a quelli più costosi fino a tre assi.

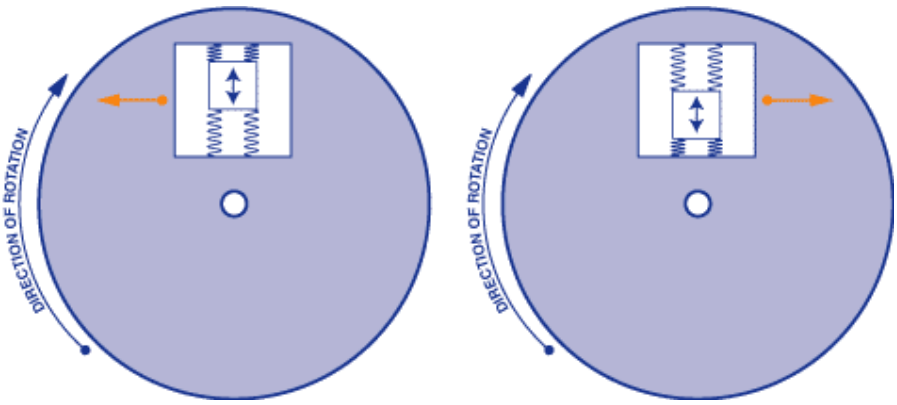



Figura 2.1: Funzionamento del giroscopio

- **Accelerometro:** è uno strumento che viene utilizzato per misurare, come dice la parola, l'accelerazione $(m/s)$. All'interno del sensore risiede una massa nota sospesa attraverso una molla elastica. Quando la massa si muove per via della forza inerziale genera un segnale elettrico che viene misurato rilevandone la velocità.

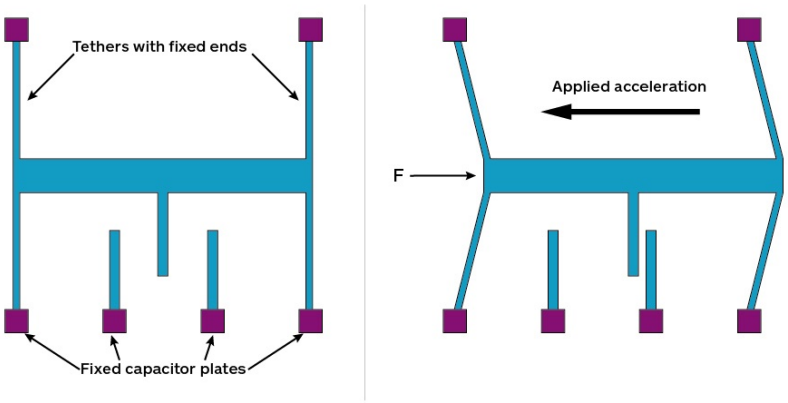



Figura 2.2: Funzionamento dell'accelerometro

- **Magnetometro:** strumento utilizzato per misurare il campo magnetico di un corpo. Solitamente viene sfruttato per capire l'orientamento del dispositivo basandosi sul campo magnetico della terra.
  A differenza del giroscopio che è molto veloce nel calcolare i dati ma fornisce l'informazione solamente nell'istante in cui avviene il movimento, il magnetometro è in grado di leggere l'orientamento anche in assenza di spostamento ma ha un tempo di risposta molto più alto del precedente.

- **GPS:** acronimo per *Global Positioning System*, è uno strumento in grado di calcolare la latitudine e la longitudine della posizione del dispositivo. È in grado di farlo grazie alla comunicazione con satelliti specifici che gli permettono di triangolare la propria posizione in qualsiasi parte del globo.

- **Telecamera:** sensore in grado di convertire la luce in segnali elettrici, che vengono processati e trasformati in un'immagine digitale. Nel campo della Gesture Recognition solitamente ne vengono utilizzate di due tipi: le classiche, utilizzate anche in tutti gli smartphone, e quelle infrarossi, in grado di calcolare la profondità dell'immagine. Esse possono venir utilizzate singolarmente, per avere una trasposizione dell'immagine in due dimensioni, oppure in gruppo rendendo possibile la creazione di una mappatura 3D dell'ambiente.


Figura 2.3: Anche se utilizzato per un diverso scopo, il funzionamento del *"Face Recognition"* del nuovo IPhone X viene impiegato anche nella *Gesture Recognition*

Molto spesso tutti questi sensori vengono utilizzati in combinazione, un esempio è quello del cosiddetto *"Inertial Measurement Unit"* (o *"IMU"*), che generalmente è l'aggregazione di due, *IMU 6 assi* contenente un accelerometro e un giroscopio, oppure tre, *IMU 9 assi* contenente oltre ai due già presenti anche un magnetometro.

## 2.2 Vison Based Gesture Recognition

Fin dal principio della Gesture Recognition [6], il metodo più utilizzato e più sperimentato è quello basato sulla *"vision"* ossia l'analisi di immagini o video. Il funzionamento si basa su telecamere che vengono puntate ad esempio alle mani che catturano immagini o video che verranno successivamente elaborati al fine di tracciare il movimento delle dita, per poi essere classificato in gesture.

Esistono due tipi di modelli che vengono implementati ed utilizzati al giorno d'oggi. Il modello tridimensionale e quello basato sull'aspetto bidimensionale.

Il modello 3D ha diverse tecniche che vengono utilizzate per la rappresentazione di gesture che sono: il *"textured volumentric"*, modello geometrico, modello scheletrico. Generalmente solo due di questi sono utilizzati, il *"textured volumetric"* contenente dati molto dettagliati riguardanti non solo lo scheletro ma anche informazioni riguardanti la pelle, e il modello geometrico meno preciso del precedente per quello che riguarda la pelle, ma contenente dati più significativi sullo scheletro.

Figura 2.4: Punti generati dal proiettore ad infrarossi del Kinect, rilevati dalla telecamera infrarossi che utilizza questa informazione per ricreare una rappresentazione 3D dell'ambiente

I modelli bidimensionali possono essere classificati in due categorie principali: *modelli 2D statici* e *modelli basati sul movimento* (*"motion"*), ognuna

delle quali è suddivisa in sotto-categorie, le più utilizzate sono:

- **Modello basato sui colori:** utilizza dei marker per tenere traccia del movimento del corpo da tracciare.
- **Modello basato sulla forma:** utilizza diverse proprietà geometriche come il perimetro, convessità, allungamento, baricentro ecc.
- **Modello basato sulla sagoma:** viene utilizzato tipicamente sui contorni deformabili.
- **Modello basato sul movimento:** utilizza una sequenza di immagini per tracciare il movimento di un oggetto.

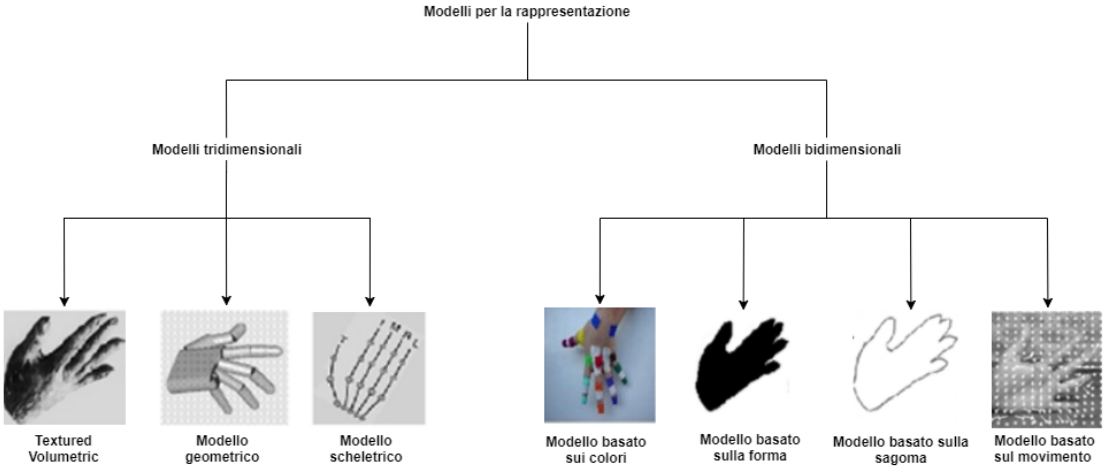



Figura 2.5: Rappresentazione grafica dei vari modelli

La Vision Based Gesture Recognition si basa su tre fasi fondamentali: rilevazione, tracciamento e riconoscimento.

### 2.2.1 Rilevazione

Il primo passo è la rilevazione della mano e la sua segmentazione. La segmentazione è molto importante perché separa i dati rilevanti (la mano) da quelli inutili (lo sfondo). Esistono diversi modi per farlo e alle volte la loro combinazione produce risultati molto più soddisfacenti rispetto a quelli ottenuti dall'utilizzo del singolo metodo:

- **Segmentazione tramite il colore:** è fondamentale la scelta dei colori. Vengono valutati vari spazi di colore (RGB, HSV, YUV, ecc.), solitamente se ne sceglie uno che riesca ad eliminare la componente della luce, al fine di rimuovere le ombre e i riflessi che ogni cambio di illuminazione inevitabilmente crea nell'immagine.

- **Segmentazione tramite la forma:** le caratteristiche del contorno possono essere utilizzate per ricavare diversi dati. Se correttamente rilevato, è in grado di tracciare il movimento ignorando le caratteristiche di illuminazione citate prima. Tuttavia le sue potenzialità possono essere ostacolate perché, basandosi sulla rilevazione del contorno, la segmentazione può portare ad evidenziare anche i bordi degli oggetti nello sfondo. Per questo motivo devono essere utilizzati filtri sofisticati oppure può essere impiegato in combinazione con il metodo precedente per ricavare dati più precisi.

- **Segmentazione tramite pixel:** questo metodo ignora completamente i colori, infatti l'immagine a colori viene trasformata in una a scala di grigi e da essa vengono estratte le informazioni riguardanti il contorno.

- **Segmentazione basata su modelli tridimensionali:** questo metodo sfrutta modelli 3D conosciuti a priori e li adatta all'immagine che gli viene sottoposta. Esso ha il vantaggio di creare uno schema indipendente dal punto in cui la telecamera viene posizionata. I punti e le linee del modello 3D vengono utilizzati per ricavare gli angoli formati dalle articolazioni della mano e di conseguenza calcolarne la posizione.

- **Segmentazione basata sul movimento:** questo metodo rispetto ai precedenti, che sfruttavano un'immagine statica, utilizza un flusso video per rilevare la mano. È quello meno utilizzato fra quelli elencati perché, anche se è molto veloce nella rilevazione dell'oggetto, assume che il solo elemento in movimento sia l'oggetto che deve essere segmentato. Quindi non è adatto a luoghi in cui lo sfondo è dinamico.

### 2.2.2 Tracciamento

La seconda fase è quella del tracciamento dei dati, si occupa appunto di tenere traccia dei movimenti che la mano effettua per poi essere processati, nella terza fase, con i vari algoritmi di classificazione. Se la prima fase (rilevamento) è svolta in maniera veloce, ossia se si riesce ad eseguire in modo tale che passino in output lo stesso quantitativo immagini preso in input, allora si può utilizzare il metodo di rilevamento anche per questa fase. Purtroppo

la difficoltà intrinseca nel tracciamento delle mani rende molto complicata questa condizione perché esse possono cambiare drasticamente posizione nel giro di pochi frame.

Il tracciamento può essere definito come la differenza di un punto o di un segmento della mano attraverso due frame. Questa differenza può essere considerata come una traiettoria, esse sono molto importanti perché possono essere utilizzate sia non elaborate *(RAW)* oppure venir classificate nella fase successiva. Un esempio delle prime è il disegno virtuale nel quale basta seguire la traiettoria grezza fornita; nel secondo caso, dopo la fase di classificazione, possono essere utilizzate da applicazioni più complesse.

Diversamente possono essere sfruttate nella fase precedente quando si utilizza un rilevamento basato sul modello tridimensionale. Esse infatti sono in grado di fornire al rilevamento una regione in cui la mano da tracciare si "dovrebbe" trovare, riducendo drasticamente lo spazio in cui cercarla aumentando di conseguenza le prestazioni.

### 2.2.3 Riconoscimento

Questa fase è comune in tutto il settore della Gesture Recognition, il suo scopo è classificare i dati provenienti dalla fase precedente in un'informazione finita. Una gesture può essere considerata come una transizione da uno stato di riposo, ad uno di movimento per poi ritornare in uno stato di riposo. Ognuna segue pattern temporali precisi e possiamo utilizzare algoritmi ad hoc per classificarli. La materia viene approfondita nel Capitolo 3.

## 2.3 Altre tecnologie per il riconoscimento

Ecco una lista non esaustiva di tecnologie che vengono utilizzate nel campo della Gesture Recognition:

### Tecnologia Radar

Anche se l'utilizzo di onde radio per la rilevazione degli oggetti viene utilizzato da ormai un secolo (ad esempio nel *Sonar*) nel campo della Gesture Recognition è ancora agli albori. Il funzionamento è tanto semplice quanto efficace, un sensore emette onde elettromagnetiche a frequenza molto elevata, gli oggetti nel suo raggio vengono colpiti dalle onde e le rispediscono al mittente. Da queste onde si riescono a derivare tantissime informazioni come ad esempio: la distanza e la forma in base al tempo le quali vengono ricevute, il materiale di cui è composto ricavato dall'intensità in cui esse arrivano al sensore, ecc.

Questa tecnologia è in grado di captare i movimenti millimetrici rendendola molto efficace nel riconoscimento delle gesture. Tutta via uno svantaggio risiede nel fatto che l'oggetto da tracciare deve essere relativamente vicino al sensore.

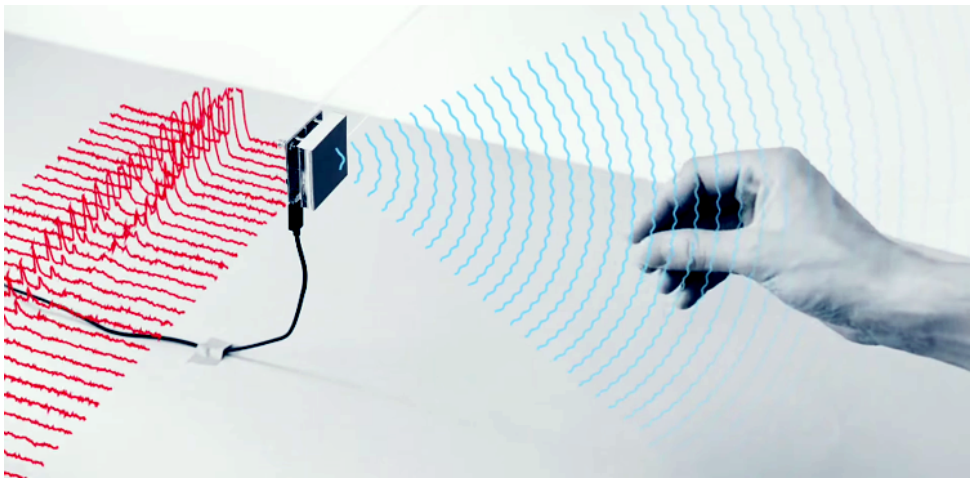


Figura 2.6: Dispositivo ancora in via di sviluppo ma che sta dando risultati molto soddisfacenti è il Project Soli da parte di Google. [7]

### Glove Based

Questa tecnologia insieme a quella basata sulla *"vison"*, sono state le prime ad essere pensate e implementate nel campo della Gesture Recognition. Al giorno d'oggi esistono molte tipologie di *guanti sensoriali* che si basano tutte sui sensori di flessione (*Flex sensors*), essi individuano la misura del piegamento o flessione della mano e/o delle dita. Questa tipologia di sensore soffre di alcune problematiche: i sensori di qualità sono molto costosi e quindi utilizzati solo in campi particolari, quelli più economici sono soggetti a rotture frequenti. Un'altra problematica è legata al fatto che non sono grado di distinguere fra stiramento e rotazione, e devono essere accompagnati da un sensore in grado di capire quando avviene una rotazione (giroscopio/magnetometro).

Il guanto di per sé si rileva molto efficace e genera dati molto precisi tuttavia risulta scomodo dato che, spesso, deve essere indossato in combinazione ad una sorta di bracciale all'interno del quale si trovano i componenti elettronici (microcontrollore e sensori inerziali) Figura 2.7.


Figura 2.7: Si può notare che oltre al guanto è presente anche il bracciale

#### Tecnologia basata su impulsi elettrici

Questa tecnologia è relativamente recente, negli ultimi anni infatti i sensori EMG in grado di rilevare gli impulsi elettrici provenienti dai nervi, contrazione dei muscoli ecc., sono diventati sempre più precisi e di dimensioni più ridotte. Tuttavia l'utilizzo di questi sensori necessita una conoscenza approfondita riguardante i segnali elettrici provenienti dal corpo umano e per divenire efficaci devono essere calibrati e personalizzati su ogni individuo. Inoltre l'interfaccia tra i muscoli ed i sensori è influenzata da molte fonti di rumore, come interferenze della linea di alimentazione, *crosstalk*, perspirazioni e impedenza di contatto cutaneo. Nonostante questi limiti alcune

aziende si sono lanciate nell'impresa e sono riuscite a creare dispositivi come il Myo Armband Capitolo 5 che semplificano incredibilmente l'utilizzo di questi sensori.

# Capitolo 3

# Algoritmi di classificazione usati nella Gesture Recognition

L'obbiettivo della Gesture Recognition è, come si può intuire dal nome, quello di riuscire ad interpretare e classificare segnali provenienti da sensori, processati o grezzi, a seguito di un segno fatto dal corpo umano. In questo capitolo andrò ad enunciare, senza entrare nei dettagli, alcuni metodi di *machine learning* che vengono utilizzati per classificare questi segnali.

Una prima categorizzazione che possiamo fare è quella di suddividere in due generi i tipi di gesture: Statiche e Dinamiche. La classificazione nel primo caso è un compito non particolarmente complesso, per interpretare una gesture statica basterebbe, infatti, semplicemente un classificatore generico basato su corrispondenze in un *template*. Nel caso di gesture dinamiche, invece, bisogna prendere in considerazione anche il fattore temporale che richiede tecniche più avanzate in grado di gestirlo.

Fortunatamente il problema della classificazione delle serie temporali è già stato affrontato ed esistono algoritmi di machine learning che possono essere impiegati anche nel campo della Gesture Recognition.

Un modo per classificare questi algoritmi è quello di considerarne il loro risultato. Ne esistono tante categorie ma le più utilizzate sono:

- **Algoritmi ad apprendimento non supervisionato:** tecnica che consiste nel fornire al sistema una serie di dati non annotati manualmente dall'utente. Esso dovrà classificarli in base alle loro caratteristiche comuni autonomamente. Uno degli algoritmi più utilizzato sono il *K-Means* e le *reti neurali artificiali (artificial neural network)*.

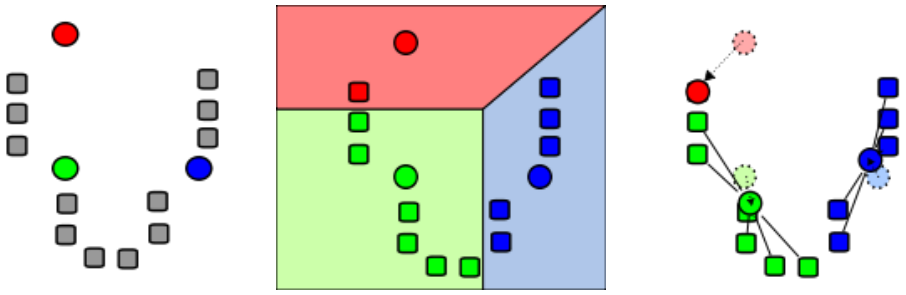
- **Algoritmi ad apprendimento supervisionato:** al contrario di quella precedente, è una tecnica che consiste nel fornire al sistema dati annotati manualmente. Nella fase di training/insegnamento vengono forniti esempi realizzati ad-hoc costituiti da una serie di input e di output e l'obbiettivo del sistema è quello di costruire dei modelli in grado di classificare i nuovi dati basandosi sugli esempi forniti. Gli algoritmi più utilizzati sono: *Hidden Markov Model*, *K-Nearest Neighbors* e *Neural Network*.

- **Algoritmi ad apprendimento per rinforzo:** questi sono algoritmi avanzati e non saranno approfonditi. Brevemente il loro funzionamento si basa sull'adattamento alle mutazioni dell'ambiente. Il sistema riceve una risposta dall'esterno a seconda dell'output elaborato che consiste in una valutazione delle prestazioni. Un output corretto comporta la ricezione da parte del sistema di un *"premio"*, viceversa riceverà una *"penalità"* nel caso in cui darà una risposta errata. L'obbiettivo del sistema è quello di, cambiando le scelte, massimizzare il numero di premi.

## 3.1 K-Means

L'algoritmo K-Means è uno delle tecniche di apprendimento non supervisionato più semplice. L'obbiettivo dell'algoritmo è quello di dividere $n$ osservazioni in $k$ "cluster". Le osservazioni vengono raggruppate nei cluster con l'obbiettivo di minimizzare l'errore di clustering, ossia la somma delle distanze di ogni dato rispetto al proprio centro di cluster. Questo algoritmo generalmente è veloce (tempo polinomiale $N^x$), solitamente il numero di iterazioni è minore del numero di punti. In casi particolari però può essere estremamente lento, una ricerca [8] ha dimostrato che esso può raggiungere anche un tempo computazionale esponenziale ($N^n$).

Questo algoritmo può essere utilizzato nella Gesture Recognition per classificare sia le gesture statiche che quelle dinamiche. Durante la fase di training all'algoritmo vengono forniti in input un set di informazioni (ad esempio immagini), esso assegna $k$ punti e inizia a spostarli basandosi sulla similarità dei dati. All'arrivo di nuovi elementi, che devono essere classificati, calcola la distanza dai vari centri di cluster e restituisce quello più vicino.



(a) (b) (c)

Figura 3.1: Funzionamento dell'algoritmo k-means

Figura 3.1(a) Si scelgono 3 punti preferibilmente non troppo vicini tra di loro che saranno i punti centrali dei 3 cluster. Figura 3.1(b) Si associa ogni osservazione al cluster più vicino a loro. Figura 3.1(c) Viene spostato il centro dei cluster basandosi sul baricentro rispetto alle osservazioni. Si ripete iterativamente la seconda e la terza fase fino a che il punto centrale è in equilibrio e quindi non viene più spostato, oppure sulla base di criteri scelti dall'utente.

## 3.2 K-Nearest Neighbors

Il K-Nearest Neighbors è l'algoritmo più semplice di apprendimento supervisionato. Esso è un metodo di classificazione che semplicemente confronta i dati da catalogare con gli esempi forniti nella fase di training e cerca quelli più simili. Il *k-NN* è un tipo di algoritmo detto *lazy* (pigro) dove gli esempi vengono salvati localmente e tutta la computazione viene eseguita durante la fase di classificazione.

La fase di training semplicemente consiste nell'ordinare i vettori funzione *"feature vector"* (ad esempio le immagini) e le varie annotazioni (*classi* che in nel nostro caso sono le gesture). Durante la fase di classificazione i dati vengono catalogati basandosi sulla maggioranza del numero dei suoi vicini e quindi sulla loro somiglianza.

Durante la fase di classificazione un parametro $K$, intero positivo, viene definito dall'utente. Se $K = 1$ i dati vengono assegnati semplicemente alla classe del vettore più simile. La scelta di $K$ dipende dalle caratteristiche dei dati che dovranno essere classificati, infatti decidere di utilizzare un valore di $K$ grande abbassa il rumore (dati non voluti che possono compromettere la classificazione) ma rende i confini delle classi più sfumati/ambigui.

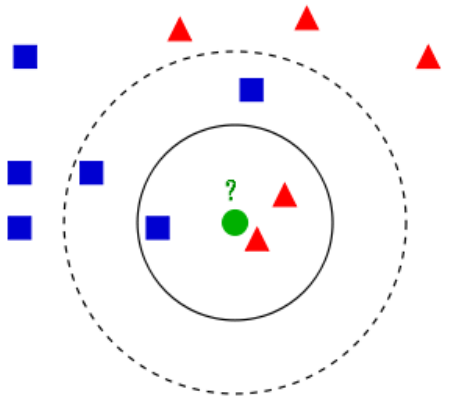



Figura 3.2

La Figura 3.2 mostra un esempio di classificazione basata su questo algoritmo. L'obbiettivo è quello di classificare il pallino verde. Come mostra la figura possiamo avere due tipi di classificazione: *quadrato blu* e *triangolo rosso*. Scegliendo un $K = 3$ (cerchio riga continua) possiamo notare che verrà scelta la classe "triangolo rosso" perché i primi 3 elementi si suddividono in: un quadrato blu e due triangoli rossi. Tuttavia scegliendo un $K = 5$ (cerchio riga tratteggiata) la scelta ricadrà sulla classe "quadrato blu". È cruciale quindi uno studio delle caratteristiche dei dati, avvalendosi eventualmente anche di tecniche statistiche come la *convalida incrociata* (*Cross Validation*).

## 3.3 Hidden Markov Model

Il *modello di Markov nascosto* fu introdotto negli anni '90 ed è diventato velocemente il metodo di riconoscimento di pattern temporali più utilizzato.

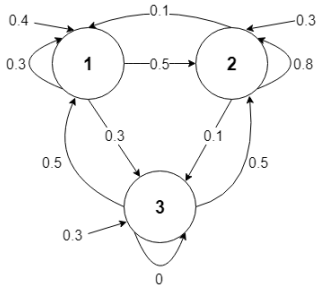
Esso si basa sulla *catena di Markov*. Una catena di Markov è un automa a stati finiti nel quale la probabilità di transizione di stato è data solo ed esclusivamente dallo stato subito precedente e non dall'intera storia degli stati passati (*proprietà di Markov*). Questo automa si dice *"memoryless"* letteralmente senza memoria perché non viene salvato nulla durante il processo.

La catena di Markov impone inoltre che ogni risultato finale (*output symbol* o *simbolo in uscita*) possa essere dato da un solo stato, per questo si dice che queste catene sono deterministiche. Può essere rappresentata come una matrice di transizione $A = \{a_{a,b}\}$ di dimensione $NxN$ dove ogni elemento della matrice $a_{a,b}$ equivale alla possibilità di passare dallo stato $a$ allo stato $b$. Essendo una matrice di transizione ogni riga deve avere come

risultato 1: $\sum_{b=1}^{N} a_{a,b} = 1$ e non possono essere presenti valori negativi. Inoltre deve essere presente una distribuzione della probabilità di stato iniziale $\pi = (\pi_1, \ldots, \pi_N)$ dove $\pi_x$ è la possibilità che lo stato iniziale sia $x$.

$$A = \begin{pmatrix} 0.3 & 0.5 & 0.2 \\ 0.1 & 0.8 & 0.1 \\ 0.5 & 0.5 & 0 \end{pmatrix}, \quad \tilde{\pi}_0 = \begin{pmatrix} 0.4 \\ 0.3 \\ 0.3 \end{pmatrix}$$

(a) Matrice e distribuzione iniziale




(b) Rappresentazione grafica della matrice e delle distribuzioni iniziali

Un Hidden Markov Model può essere considerato come una generalizzazione di una catena di Markov nella quale gli stati non possono essere osservati direttamente. L'HMM è non deterministico ed è impossibile risalire al set di input semplicemente guardando l'output (per questo si dice *hidden/nascosto*) perché per ogni output symbol non c'é un solo stato che lo produce.

Ogni transizione di stato è formata da: lo stato da cui parte la transizione, la probabilità che la transazione avvenga, lo stato in cui si arriva dopo la transizione ed eventualmente il simbolo in uscita che viene generato.

In un contesto di Gesture Recognition possiamo considerare ogni stato come un movimento che si esegue, la transizione come la probabilità che quel movimento evolva in un altro, l'output symbol possiamo vederlo come un'osservazione, e tutte le sequenze di queste osservazioni compongono una gesture.

## 3.4 Rete neurale artificiale

Le Reti Neurali Artificiali (*Artificial Neural Network* o *ANN*) possono essere utilizzate sia come metodo di *apprendimento supervisionato*, sia *non supervisionato*. Esse sono modelli matematici che si inspirano alle reti neurali biologiche, ossia quelle che compongono il cervello. Questi sistemi proprio come il cervello sono formati da neuroni artificiali in grado di svolgere calcoli che si scambiano dati l'un l'altro tramite collegamenti proprio come avviene

con le sinapsi (il neurone artificiale riceve i dati, li elabora e li invia ad un altro neurone).

I modelli prodotti dalle reti neurali, sono molto efficienti ma proprio come nel caso del funzionamento del neurone umano non è possibile spiegare cosa effettivamente avviene al loro interno. La definizione inglese per queste reti infatti è *Black Box (scatola nera)* perché a differenza degli altri modelli, nei quali si può a grandi linee esaminare il percorso che dall'input genera l'output, nelle reti neurali non è possibile spiegare come esso sia stato prodotto.

Utilizzato nella Gesture Recognition, durante la fase di training vengono forniti in input i feature vector e viene annotato anche l'output nell'eventualità che l'apprendimento sia supervisionato, successivamente la rete neurale inizia ad addestrare i neuroni. Alla fine di questa fase l'algoritmo è in grado di classificare nuovi dati basandosi sulle operazioni eseguite nei neuroni.

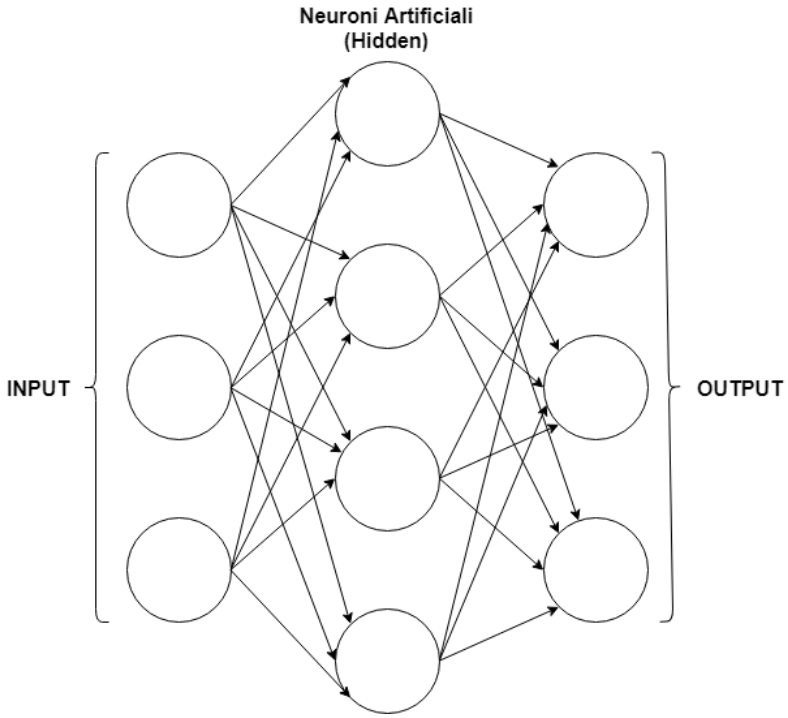



Figura 3.3: Rappresentazione grafica di una rete neurale

# Capitolo 4

# Un esempio di piattaforma: il Georgia Tech Gesture Toolkit

Come in tutti i settori informatici soprattutto in quello della Gesture Rrecognition, per facilitare e velocizzare lo sviluppo di applicazioni, è necessario avere un'infrastruttura generale, un *framework*, capace di fornire allo sviluppatore un certo grado di astrazione. Il framework fornisce una serie di API (Application Programming Interface) che permettono al developer di concentrarsi nello sviluppo dell'applicazione specifica senza dover sprecare risorse nella parte di classificazione della gesture, comune in tutte le applicazioni.

Un carattere intrinseco della Gesture Recognition, come già detto, è quello di fondarsi su pattern temporali, alla base degli HMM. Essi venivano già ampiamente utilizzati nel *voice recognition* e negli anni 2000, si iniziò a pensare di trasformare toolkit che sfruttavano questi modelli adattandoli al riconoscimento delle gesture. Nacque così il Georgia Tech Gesture Toolkit ($GT^2K$) il quale vedeva le sue fondamenta nel framework del riconoscimento vocale dell'Università di Cambridge.

Il $GT^2K$ creato appunto dal *"Georgia Institute of Tecnology"* [9] è il framework che per primo permise ai ricercatori di concentrarsi nello sviluppo di sistemi basati sulla Gesture Recognition, invece di devolvere tempo e risorse sullo studio e la creazione di un sistema di riconoscimento ad hoc per ogni applicazione. Esso infatti, permette di "ingorare" il livello più basso, trascurando la parte di classificazione della gesture e consente di pensare ad essa come un oggetto di alto livello.

Il $GT^2K$ è un'infrastruttura che non è designata alla prelevazione dei dati ma devono venirgli forniti, essi possono essere sia filtrati che grezzi ($RAW$). Questi dati vengono utilizzati per creare informazioni finite che in seguito

sono date all'applicazione ad-hoc per essere trasformate in azioni specifiche.

Il processo del GT$^2$K si suddivide in 4 fasi: preparazione, training, validazione, e riconoscimento della/delle gesture.

Analizzerò in dettaglio queste fasi prendendo come esempio un set ristretto di gesture che utilizzate in combinazione potranno essere sfruttate per controllare un veicolo (*attenzione, stop, avanza, arretra e rallenta*).

## 4.1 Preparazione

In questa fase è necessario che l'utente fornisca al sistema un insieme di modelli, interpretazioni grammaticali ed esempi per ogni gesture che dovrà essere classificata.

### 4.1.1 Sviluppo degli Hidden Markov Models

Per ogni gesture viene utilizzato un HMM diverso, generalmente, nella fase di *"insegnamento"* del modello non vengono inseriti o rimossi gli stati, ma solo aggiornate le probabilità di transizione fra di essi.

La progettazione di questi modelli richiede una conoscenza approfondita della struttura dei dati difficilmente intuibile da un utente non esperto. Inoltre per determinare le corrette strutture dei modelli potrebbero essere necessarie alcune sperimentazioni pratiche che generalmente richiedono tempo considerevole.

Per questo motivo il toolkit fornisce all'utente meno esperto degli strumenti per generare automaticamente queste strutture, dando comunque la possibilità ai più esperti di creare il proprio modello e fornirlo al framework.

### 4.1.2 Specifica della Grammatica

Il caso più semplice è denominato "Isolated Gesture Recognition" (*Riconoscimento gesture isolato*) nella quale il riconoscimento viene eseguito su un gesto alla volta. Tuttavia non è questo il caso tipico, molto più spesso si deve eseguire un riconoscimento del tipo "Continuous Recognition" (*Riconoscimento continuo*) ossia un continuo input di dati che devono essere classificati. Questo può venir rappresentato al GT$^2$K con interpretazioni grammaticali ben precise. Esse sono in grado di far sfruttare al toolkit la conoscenza della struttura dei dati e, sapendo quale gesto è stato appena eseguito, ristringere il pool di gesti futuri possibili. Inoltre queste interpretazioni grammaticali permettono di definire gesture complesse come sequenza

di gesture più semplici.

Esempio:

```
Muoviti in avanti = avanza rallenta stop
Muoviti indietro = arretra rallenta stop
Comando = attenzione <Muoviti in avanti | Muoviti indietro>
```

Nel caso di "`Muoviti in avanti`" oppure "`Muoviti indietro`" il riconoscimento della gesture sarà del tipo "continuous recognition".

### 4.1.3 Esempi pratici per ogni gesture

Come detto prima il framework non si occupa di recuperare i dati, ma li riceve da sensori esterni, questi dati vengono immagazzinati in vettori chiamati *"feature vector"* (vettori funzione), una gesture può essere rappresentata come una sequenza di questi array. Per fare in modo che il $GT^2K$ capisca i dati che riceve, questi vettori devono essere propriamente annotati dall'utente.

Ad esempio utilizzando una telecamera oppure un sensore che restituisca vettori funzione composti da tre valori $(x, y, t)$ dove $t$ è l'unità temporale, $x$ e $y$ valori reali che indicano la posizione della mano dell'utilizzatore e registrando per 300 volte i dati dei sensori, i vari vettori possono essere annotati in questo modo:

```
(x , y , 0) -> (x , y , 60) attenzione
(x , y , 61) -> (x , y , 180) avanza
(x , y , 181) -> (x , y , 230) rallenta
(x , y , 231) -> (x , y , 250) arretra
(x , y , 251) -> (x , y , 300) stop
```

## 4.2 Training/Validation

A seguito della prima fase di preparazione, nella quale tutte le fasi vengono annotate manualmente dall'utente, avviene la fase di addestramento in modo totalmente autonomo, nella quale viene chiesto al developer semplicemente di scegliere un metodo di formazione e la configurazione di alcuni parametri specifici dipendenti da esso. Questo processo crea gli HMM che verranno poi utilizzati nella fase effettiva dell'applicazione di riconoscimento.

Il toolkit consente di evitare i dettagli di basso livello, fornendo di default alcuni metodi di training standard ma permette anche, agli utenti più esperti, di integrare i propri. Questi metodi vengono suddivisi in due tipi, uno di formazione (*"training set"*) e uno di validazione (*"validation set"*). Il primo è quello utilizzato per addestrare il modello, mentre il secondo per valutarne le prestazioni esaminando dati non ancora presi in considerazione.

La valutazione del modello tramite il validation set aiuta a stimare quanto esso riesca a generalizzarsi rispetto a nuovi dati, inoltre serve per determinare se e quanto sia presente il problema dell'*overfitting*. Esso letteralmente significa "eccessivo adattamento" viene causato quando il training set è effettuato troppo a lungo o svolto su uno scarso numero di risultati. Il modello prodotto di conseguenza potrebbe adattarsi a caratteristiche specifiche al solo insieme di formazione, che non hanno riscontro negli altri casi. Facendo così in modo che le prestazioni nel riconoscimento del set di allenamento/formazione aumentino, mentre quelle sui dati nuovi peggiorino.

I metodi standard che vengono forniti all'utente sono la convalida incrociata (*cross validation*) e la *leave one out*. La prima prende un campione di dati in una percentuale predeterminata come training set (es 60%) i restanti (in questo caso il 40%) vengono utilizzati come validation set (misurare le performance). Leave one out invece, prende un singolo dato come validazione e utilizza i restanti come training, questo processo viene poi ripetuto per ogni cambiamento del set di dati lasciando sempre un dato come validazione e i restanti come training. A livello prestazionale la cross validation è più veloce mentre la leave one out crea dati più attendibili ma è estremamente più lento dato che itera per $n - 1$ volte tutto il set di dati.

Ritornando all'esempio di prima, una persona viene osservata e vengono catturate 100 sequenze. La convalida incrociata prenderà 60 sequenze come training e le restanti 40 verranno utilizzate per valutare le prestazioni del sistema. Leave one out, invece, prenderà 99 delle sequenze come training ed effettuerà il riconoscimento sull'unica lasciata fuori. Questo processo si ripeterà per altre 99 volte; nella seconda iterazione la parte di addestramento verrà effettuata sulla sequenza numero 1,3 fino a 100, lasciando la seconda come validazione.

## 4.3 Performance del sistema

Le performance del sistema derivano dalla fase precedente (training/-validation), vengono salvate nella forma di una matrice confusione (anche detta tabella di errata classificazione) e sono basate sul livello di ***accuratezza*** del riconoscimento. Questa tabella è formata da tre tipi di problema: *inserimento, cancellamento e infine sostituzione.*

- **L'errore d'inserimento** avviene quando il sistema, durante uno scenario di riconoscimento continuo, riconosce l'avvenimento di una gesture anche se essa non è stata eseguita.
- **L'errore di cancellazione** al contrario avviene quando il sistema non riesce a riconoscere una gesture anche se essa è stata eseguita.
- **L'errore di sostituzione** avviene quando viene riconosciuta la gesture sbagliata.

L'accuratezza del sistema si può definire con questa operazione:

$$ACCURATEZZA = \frac{N - I - C - S}{N}$$

Dove $N$ indica il totale degli esempi, $I$ il numero di errori di inserimento, $C$ gli errori di cancellazione, $S$ quelli di sostituzione.

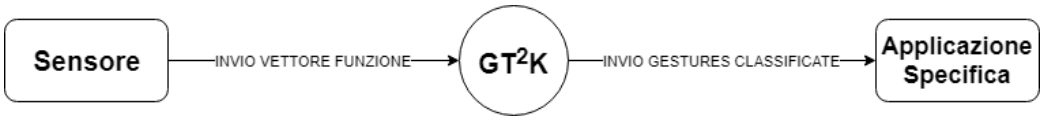
Data la natura di questi problemi, l'errore di inserimento e quello di cancellazione possono avvenire solo durante un riconoscimento del tipo continuo, quindi nel caso di *"Isolated Gesture Recognition"* $I$ e $C$ equivarrebbero a 0 trasformando l'operazione in: $ACCURATEZZA = \frac{N-S}{N}$

```
------------------ Overall Results ------------------
%Corr=99.00, Acc=99.00 [H=297, D=1, S=2, I=0, N=300]
------------------ Confusion Matrix -----------------
          a   a   r   a   s
          t   v   a   r   t
          t   a   l   r   o
              n   l   e   p
              z   e   t
              a   n   r
                  t   a
                  a
          -   -   -   -   -
att       58  0   0   2   0
avanza    0   119 0   0   0
rallenta  0   0   50  0   0
arretra   0   0   0   20  0
stop      0   0   0   0   50
=====================================================
```

Figura 4.1: Matrice confusione generata dal nostro esempio.

## 4.4 Riconoscimento

In questa fase i modelli generati nelle fasi precedenti possono venire utilizzati nel riconoscimento di nuovi dati. Quando il sistema riceve nuovi dati relativi a gesture non riconosciute calcola in automatico la loro percentuale di somiglianza con ogni modello salvato e successivamente fornisce questa informazione che può essere utilizzata dall'applicazione specifica per eseguire azioni.




Ritornando all'esempio si potrebbe implementare il sistema appena generato per controllare un veicolo. Il toolkit riceve i dati dai sensori, calcola le varie possibilità, invia le gesture classificate al robot ed esso decide quale azione eseguire.

## 4.5 Applicazioni

Dato che come già detto si tratta di un framework che funge come interprete dei dati, per dimostrare la sua l'efficienza e la versatilità nella sua pubblicazione nel 2003 [9] sono state presentati quattro diversi scenari di utilizzo. Il primo si tratta di un sistema di Gesture Recognition all'interno di un'automobile, il secondo riguarda Riconoscimento del linguaggio dei segni, il terzo è un *"Blink Pattern Recognition"* ed infine un *"Workshop Activity Recognition"*. Approfondirò di seguito i primi due citati.

### 4.5.1 Controllo di un sistema informativo di un'automobile

Uno studio eseguito dall'*ANSA* ha dimostrato che in Italia 3 incidenti su 4 sono dovuti a distrazioni da parte del conducente [10], e il maggior responsabile di essi è derivante da un utilizzo improprio dello smartphone oppure del sistema informativo dell'auto. Per esempio una semplice risposta al telefono richiede che almeno una mano rilasci il volante e gli occhi, per una frazione di secondo, distolgono l'attenzione dalla strada per vedere chi chiama e rispondere o meno alla chiamata.

È stato pensato un sistema in grado di minimizzare le distrazioni e che permetta al guidatore di controllare i vari dispositivi con un semplice movimento delle mani senza, cosa fondamentale, distogliere lo sguardo dalla strada.

Una delle sfide più importanti nell'implementazione di un'applicazione di questo genere tramite l'utilizzo di telecamere come fonti di dati, sono le inevitabili variazioni di luminosità durante la guida, per esempio il passaggio in un tunnel, oppure la semplice sterzata talvolta cambia la condizione di luce. Un sistema di questo genere è tassativo che non venga influenzato da questi fattori, quindi sono state impiegate telecamere con un'apertura dell'obbiettivo che capta solo la luce infrarossa, puntate ad una matrice composta da 72 led ad infrarossi (*"IR LED"*).

### Preparazione

**Sviluppo degli HMM:** Per fare in modo che il guidatore si distragga il meno possibile le gesture devono essere semplici, sono quindi state scelte otto gesture ognuna in una direzione diversa, e dopo alcune sperimentazioni sono stati adottati otto HMM ad otto stati.

**Specifica della grammatica:** data la natura di questo esperimento, solamente gesture del tipo "Isolated Gesture Recognition" potranno essere eseguite in qualunque ordine. La grammatica che è stata fornita al programma è quindi:

```
Gesture = su | giu | sinistra | destra | alto-sinistra |
alto-destra | basso-sinistra | basso-destra
```

**Raccolta dati e annotazioni:** il vettore funzione corrisponde esattamente alla matrice dei LED, e quindi è rappresentato da un vettore composto da 72 elementi booleani dove 0 corrisponde a LED non visibile e 1 a LED visibile. Il sistema è stato osservato e annotato opportunamente per 251 frame.

### Addestramento e valutazione prestazionale del sistema

Come addestramento/validazione è stato scelto il metodo "leave one out", che ha creato una matrice confusione dove si può notare che vengono classificate correttamente 249 frame su 251, con una precisione del 99.20

```
---------------- Overall Results ------------------
%Corr=99.20, Acc=99.20 [H=249, D=0, S=2, I=0, N=251]
---------------- Confusion Matrix -----------------
           d   u   l   r   u   u   d   d
           o   p   e   i   p   p   o   o
           w       f   g   l   r   w   w
           n       t   h   e   i   n   n
                       t   f   g   l   r
                           t   h   e   i
                               t   f   g
                                   t   h
                                       t
           -   -   -   -   -   -   -   -
down      25   0   0   0   0   0   0   0
 up        0  25   0   0   0   0   0   0
left       0   0  36   0   0   0   0   0
right      0   0   0  31   0   0   0   1
up_left    0   0   0   0  34   0   0   0
up_right   0   0   0   0   0  27   0   1
dwn_left   0   0   0   0   0   0  33   0
dwn_right  0   0   0   0   0   0   0  38
=================================================
```

Figura 4.2: Matrice confusione generata sulla base dei 251 frame

### 4.5.2 Riconoscimento del linguaggio dei segni mobile Americano

L'obbiettivo di questo progetto è quello di riconoscere un semplice vocabolario che compone il *"Contact Sign"* (una sottoparte della Lingua dei Segni Americana ASL). Questo progetto non si limita alla raccolta di dati solo da una fonte, infatti sono stati utilizzati 2 diversi sensori:

- **Accelerometro:** sono stati impiegati 2 accelerometri a 3 assi, uno montato sul torso dell'utente e un altro nel polso, in grado di cogliere informazioni che ad una sola telecamera risulterebbero nascoste. Come ad esempio la rotazione o il movimento in verticale in direzione della telecamera stessa.

- **Telecamera:** una telecamera montata su un cappello in direzione delle mani, in grado di fornire informazioni non ottenibili dagli accelerometri. Come ad esempio la posizione relativa orizzontale oppure il gesto della mano.

L'obbiettivo è che l'aggiunta di più sensori aumenti l'accuratezza del sistema diminuendo il rumore *"noise"* (senza l'aggiunta di filtri).

### Preparazione

**Sviluppo degli HMM:** Le gesture del vocabolario sono state rappresentate da due diversi HMM. Un modello a cinque stati rappresenta le gesture più corte *Il mio, me, parlare, esci, calibrazione*, il secondo modello, a dieci stati, rappresenta quelle più lunghe *computer, aiuto.*

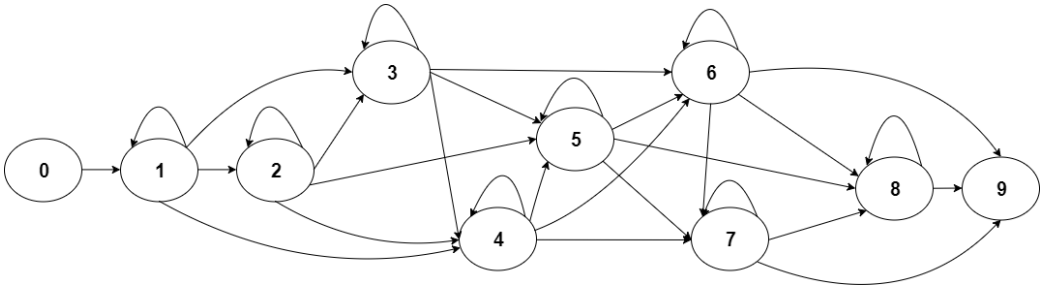



Figura 4.3: Questa immagine rappresenta il modello a 10 stati utilizzato per modellare *computer* e *aiuto* ossia le gesture più complesse.

**Specifica della grammatica:** per questo progetto sono state considerate 72 frasi composte da 7 parole, ognuna delle quali inizia e termina con una gesture precisa (*calibrazione* ed *esci*), e con una permutazione delle 5 parole centrali (*il mio, computer, mi, aiuta, a parlare*) formano frasi diverse. La grammatica sottoposta al toolkit è questa:

```
parola = il mio | computer | mi | aiuta | a parlare
frasi = (calibrazione parola parola parola parola parola esci)
```

**Raccolta dati e annotazioni:** la raccolta dei dati è avvenuta combinando i dati provenienti dai entrambe le fonti (accelerometro e telecamera). L'accelerometro invia vettori funzione nella forma di $(x,y,z)$ ognuna per asse cardinale, la telecamera invece invia più vettori ognuno per indicare: coordinate centrali $(x,y)$, massa, eccentricità, angolo, assi maggiori $(x,y)$ e assi minori $(x,y)$.

È importante sottolineare che questi sensori inviano dati ad una frequenza differente è quindi necessario sincronizzare i due flussi.

### Addestramento e valutazione prestazionale del sistema

Come metodo di addestramento/validazione è stato utilizzato ancora una volta il "Leave One Out", ed è stato dimostrato come la combinazione di questi due sensori ha aumentato drasticamente la percentuale di accuratezza

del sistema. La percentuale di accuratezza della sola telecamera durante il test è stata del 52.38%, del solo accelerometro è stata 65.87%, invece la percentuale dopo la combinazione delle due fonti è 90.48%

# Capitolo 5

# Un caso di studio: il Myo Armband

## 5.1 Il Dispositivo

Figura 5.1: Myo Armband

Il *Myo Armband* prodotto dall'azienda statunitense *"Thalmic Lab"*, è un bracciale che va indossato nell'avambraccio, ed al momento si tratta di uno dei dispositivi più evoluti nel campo della "Gesture Recongintion".

È un prodotto pensato per il settore customer, è possibile utilizzarlo in diversi campi, da quelli professionali come ad esempio presentazioni virtuali, controllo remoto di sistemi informatici, a quello dell'entertainment come ad esempio controllo della musica/film, oppure droni e videogiochi [11].

Il Myo pesa circa 93 grammi, è formato da otto blocchi rettangolari connessi tra loro in ognuno dei quali sono presenti tre sensori elettromiografici. Essi sono in grado di rilevare l'attività muscolare e nervosa attraverso segnali elettrici che produce durante l'esecuzione di una gesture. Al suo interno possiamo trovare anche un *IMU* (Inertial Measurament Unit) a 9 assi composto da un magnetometro, un accelerometro e un giroscopio a 3 assi, in grado di rilevare i movimenti del braccio come ad esempio la sua rotazione. Contiene anche un processore *ARM Cortex M4* (*RISC*) che si occupa della gestione del dispositivo. È fornito infine, a livello di connettività, di un modulo Bluetooth 4.0 LE (Low Energy), in grado di mantenere una connessione stabile e molto performante con qualsiasi dispositivo compatibile, garantendo al contempo un'autonomia di svariate ore d'utilizzo. A livello di output è dotato di due led in grado di fornire all'utente informazioni riguardanti lo stato della connessione e della batteria [12] e un modulo in grado di fare vibrare il dispositivo a 3 intensità *(breve, media e lunga)*.

Il Myo è in grado di riconoscere 5 gesture basilari: *"double tap"*, *"fist"*, *"finger spread"*, *"wave-in"* e *"wave-out"*. Non è un dispositivo *"user-indipendant"* infatti ad ogni cambiamento di utente, spostamento o ogni volta che viene indossato deve essere ricalibrato attraverso una gesture predefinita (*"Wave Out"*) al termine della quale viene restituito un feedback all'utente attraverso una vibrazione. Inoltre, perché il dispositivo sia performante deve essere indossato per un breve periodo di tempo prima dell'utilizzo (fase di *"Warm UP"*).


Figura 5.2: Gesture predefinite del dispositivo

Il rilascio del dispositivo nel 2014 ha avuto un grandissimo impatto nel campo del'*HCI* (*Human Computer Interaction*) e furono pubblicate tantissime applicazioni sviluppate appositamente per il Myo, qui sotto si può trovare una lista non esaustiva dei successi del prodotto:

- Nel novembre del 2014 fu rilasciato un progetto che si poneva come obbiettivo la traduzione della lingua dei segni automatica. [13]
- Nel 2015 fu pubblicato un progetto che puntava a telecontrollare un veicolo con il solo utilizzo del Myo. [14]
- Sempre nel 2015 fu creato un prototipo che utilizza il Myo come vero e proprio strumento musicale. [15]
- Ancora nel 2015 l'azienda produttrice pubblicò il primo videogioco "Icarus Rising" che richiede l'utilizzo del Myo armband come controller e un visore della realtà virtuale (come ad esempio l'Oculus Rift) nel quale il giocatore controlla una navicella in modo del tutto naturale con il semplice movimento delle mani. [16]

## 5.2 MYO Connect

Prima di passare ad analizzare l'SDK che l'azienda fornisce, occorre citare un altro importante componente. Il *Myo Connect* è un elemento fondamentale che ci permette di interfacciare applicazioni predefinite con il dispositivo. È un software presente per le piattaforme Microsoft *Windows* e *MacOS* che di base può essere utilizzato per interagire con varie applicazioni che spaziano dal campo *entertainment* (ad esempio Netflix, Youtube, Spotify, ecc) a quello *business* (come per esempio PowerPoint, Google Slide e Adobe Reader), semplicemente utilizzando le 5 gesture precaricate sul dispositivo. Esso permette anche di giungere ad un livello più basso, offrendo la possibilità di creare script personalizzati scritti in LUA [17] permettendone l'utilizzo anche in altri software al di fuori di quelli predefiniti. Questi script, però, non permettono di creare nuove gesture.

## 5.3 SDK

L'azienda offre diverse versioni del *Software Development Kit* per le piattaforme più utilizzate: Windows/MacOS [18], Android [19], IOs [20].

Il dispositivo fornisce di base due tipi di dati all'applicazione:

- **Dati di tipo spaziale:** un insieme di dati che servono a determinare il movimento del braccio, essi possono essere di 2 forme: **orientamento** che rappresenta in che direzione viene puntato il dispositivo, fornito sotto forma di *quaternione*; e un vettore **accelerazione** che rappresenta la velocità appunto di accelerazione in un determinato periodo di tempo, sotto forma di vettore a 3 dimensioni.
- **Dati di tipo gestuale:** rappresentano il tipo di gesture che l'utente sta eseguendo.

In questo capitolo andrò ad analizzare le API fornite dal SDK, tralasciando ogni singola implementazione ma utilizzando il linguaggio di programmazione *Java (Android)* per riportare semplici esempi di come i vari metodi forniti possono essere utilizzati.

#### Classe *HUB*

Questa è la classe più importante del SDK, si occupa di tutto quello che riguarda la gestione della connessione ad uno o più dispositivi e le varie politiche di blocco. Il suo design è basato su di un *singleton* perché è rigorosamente necessario che ce ne sia una sola istanza. Fra le funzionalità che offre possiamo trovare:

- Inizializzazione dell'hub (tramite singleton).
- Connessione al/ai dispositivo/i più vicino/i.
- Connessione ad un dispositivo specifico identificato dall'indirizzo fisico *(MAC address)*.
- Disconnessione di un dispositivo (identificato sempre tramite *MAC address*).
- Impostazione delle politiche di blocco, si può scegliere fra *None* nella quale gli eventi *"Pose"* (identificano le gesture) vengono inviati anche a dispositivo bloccato e *Standard* che al contrario non vengono trasmessi.
- Aggiunta o rimozione di uno o più *DeviceListener*.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Hub myHub = Hub.getInstance();

    if (!myHub.init( context: this, getPackageName())) {
        /*se l'hub non può essere inizializzato non si può utilizzare il dispositivo
        * quindi chiudi l'applicazione*/
        Toast.makeText( context: this,  text: "L'hub non può essere inizializzato",
                Toast.LENGTH_SHORT).show();
        finish();
    }
    myHub.attachToAdjacentMyo(); //connessione al dispositivo piu vicino

    myHub.setLockingPolicy(Hub.LockingPolicy.NONE); /*imposto la politica di invio dati facendo
                                                    * in modo che l'applicazione riceva dati
                                                    * anche quando il MYO è bloccato*/

    myHub.addListener(this.myDeviceListener);       //attacco all`hub un listener da me creato

}

@Override
protected void onDestroy() {
    super.onDestroy();
    /*al termine dell'applicazione bisogna deregistrare il listener, altrimenti avverranno le
    * callback anche dopo la chiusura dell'app*/
    Hub.getInstance().removeListener(this.myDeviceListener);
    /*inoltre bisogna deallocare l'hub, con questo metodo inoltre ci si disconnette dal
    * dispositivo rendendolo disponibile ad altre connessioni*/
    Hub.getInstance().shutdown();
}
```

### Classe *Myo*

In questa classe possiamo trovare la rappresentazione virtuale di un dispositivo Myo. Al suo interno è presente una lista di metodi che vengono utilizzati per capire conoscere lo stato del dispositivo. Questa classe non può essere istanziata direttamente, il costruttore, infatti, viene chiamato direttamente dalla classe HUB quando viene invocato un metodo per la connessione al MYO (ad esempio *attachToAdjacentMyo*).

```
//se un MYO è stato connesso all'hub sono in grado di interagire con esso
if (myMYO!=null){
    /*posso sapere in che braccio è indossato (se non è avvenuta la fase di
    * sincronizzazione mi restituisce UNKNOWN)*/
    System.out.println(myMYO.getArm().toString());
    //posso azionare la vibrazione
    myMYO.vibrate(Myo.VibrationType.MEDIUM);
    //conoscere se è stata eseguita una gesture (fra quelle di default)
    System.out.println(myMYO.getPose().toString());
    //posso portare il dispositivo nello stato di "LOCK" forzatamente
    myMYO.lock();
}
```

**Classe enumerativa *Pose*:**

Questa classe enumerativa rappresenta l'insieme di gesture basilari che sono preregistrate nel dispositivo: *REST*, *FIST*, *WAVE_IN*, *WAVE_OUT*, *FINGERS_SPREAD*, *DOUBLE_TAP*, *UNKNOWN*.

**Interfaccia *DeviceListener***

Qui si trovano tutte le *callback* che vengono invocate ogni qualvolta il Myo genera dati. Una lista delle chiamate che possono essere intercettate sono:

- A seguito di una connessione *(onConnect)*.
- A seguito di uno sblocco/blocco (*onUnlock*/ c textitonLock).
- A seguito di una gesture (*onPose*).
- A seguito della generazione di dati provenienti dai sensori inerziali (*onOrientationData*, il quale fornisce un *quaternione* rappresentante la rotazione/onAccelerometerData che fornisce dati sotto forma di un vettore a 3 dimensioni/*onGyroscopeData* anch'esso fornisce i dati sotto forma di un vettore a 3 dimensioni).
- A seguito della generazione di dati provenienti dai sensori *EMG* (SOLO PER WINDOWS/MAC/iOS).

**Classe *Quaternion***

In questa classe possiamo trovare la modellazione di un quaternione [21]. Un quaternione è un importante tipo di dato che viene utilizzato per la

modellazione di rotazioni, dato che non soffre del problema tipico nell'utilizzo degli angoli di Eulero, ossia il blocco cardanico (*gimbal lock*).

### Classe *3Vector*

In questa viene rappresentato un vettore a 3 dimensioni $(x, y, z)$, che viene utilizzato per la rappresentazione di un punto sul piano cartesiano. Esso come detto prima viene impiegato nella descrizione dei dati provenienti dai singoli sensori.

```
private DeviceListener myDeviceListener = new AbstractDeviceListener() {

    @Override
    public void onConnect(Myo myo, long timestamp) {
        statoDispositivoTW.setText("Il dispositivo è bloccato");
    }

    /*Metodo invocato quando il dispositivo viene sbloccato, la gesture predefinita
            * per lo sblocco è il DOUBLE_TAP*/
    @Override
    public void onUnlock(Myo myo, long timestamp) {
        statoDispositivoTW.setText("Il dispositivo è sbloccato");
    }

    /*Metodo invocato quando il dispositivo viene bloccato.*/
    @Override
    public void onLock(Myo myo, long timestamp) {
        statoDispositivoTW.setText("Il dispositivo è bloccato");
    }

    /*Metodo invocato quando il dispositivo rileva una gesture.
    * ATTENZIONE: la chiamata di questo metodo è condizionata alle politiche di sblocco del HUB,
    * ossia se l'HUB è nello stato Hub.LockingPolicy.NONE il metodo verrà chiamato ongi volta
    * che il dispositivo rileva una gesture. Viceversa se l'HUB è nello stato
    * Hub.LockingPolicy.STANDARD esso verrà invocato solo se il dispositivo è sbloccato.*/
    @Override
    public void onPose(Myo myo, long timestamp, Pose pose) {
        switch (pose){
            case UNKNOWN:
                Toast.makeText(getApplicationContext(), text: "Il dispositivo non ha " +
                        "riconosciuto la gesture", Toast.LENGTH_LONG).show();
                break;
            case FIST:
                Toast.makeText(getApplicationContext(), text: "Il dispositivo ha riconosciuto" +
                        "il pugno", Toast.LENGTH_LONG).show();
                break;
            case FINGERS_SPREAD:
                Toast.makeText(getApplicationContext(), text: "Il dispositivo ha riconosciuto" +
                        "la mano aperta", Toast.LENGTH_LONG).show();
                break;
            case DOUBLE_TAP:
                Toast.makeText(getApplicationContext(), text: "Il dispositivo ha riconosciuto" +
                        "il double tap. Blocco del dispositivo!", Toast.LENGTH_LONG).show();
                myo.lock();
                statoDispositivoTW.setText("Il dispositivo è bloccato");
                break;
        }
    }
```

```
    /*Metodo invocato quando il MYO invia dati provenienti dall'IMU, viene rappresentato da un
    * quaternione*/
    @Override
    public void onOrientationData(Myo myo, long timestamp, Quaternion rotation) {
        //Calcolo gli angoli di Eulero provenienti dal quaternione
        float roll = (float) Math.toDegrees(Quaternion.roll(rotation));
        float pitch = (float) Math.toDegrees(Quaternion.pitch(rotation));
        float yaw = (float) Math.toDegrees(Quaternion.yaw(rotation));

        //bisogna aggiustare i dati nel caso in cui il dispositivo sia indossato al contrario
        if(myo.getXDirection()==XDirection.TOWARD_ELBOW){
            roll *= -1;
            pitch *= -1;
        }

        //adesso posso mostrare i dati nella textView
        quaternioneTW.setText("Quaternione orientamento-> roll: "+ roll
        + " pitch: " + pitch + " yaw: " + yaw);
    }

    /*Metodo invocato quando il MYO invia dati provenienti dall'accelerometro, questi dati
    * vengono rappresentati da un vettore a 3 dimensioni*/
    @Override
    public void onAccelerometerData(Myo myo, long timestamp, Vector3 accel) {
        //mostro nella text view i 3 dati
        acclerometroTW.setText("Vettore accelerometro-> x: "+accel.x()+ " y: "+ accel.y()
        + " z: " + accel.z());
    }

    /*Metodo invocato quando il MYO invia dati provenienti dal giroscopio, questi dati vengono
    * rappresentati da un vettore a 3 dimensioni*/
    @Override
    public void onGyroscopeData(Myo myo, long timestamp, Vector3 gyro) {
        //mostro nella text view i 3 dati
        giroscopioTW.setText("Vettore giroscopio-> x: "+gyro.x()+ " y: "+ gyro.y()
                + " z: " + gyro.z());
    }
};
```

## Conclusioni

Purtroppo a mio parere questo SDK al momento non è totalmente completo. Ci sono discrepanze fra le varie versioni, peculiarità non ottimale quando si deve progettare un'applicazione cross platform. Un altro grave difetto si trova nella versione di Android che, a differenza delle altre due, non ha nessun accesso ai dati provenienti dai sensori elettromiografici, questo rende impossibile la creazione di nuove gesture. Fortunatamente esistono metodi per accedervi, analizzando i dati provenienti dal modulo bluetooth [22].

## 5.4 Sperimentazioni

### 5.4.1 Valutazione Myo come metodo di input per ambienti virtuali

Come già detto in precedenza, il dispositivo può essere utilizzato come periferica di input anche nell'ambiente videoludico, specialmente quando lo si abbina ad un visore della realtà virtuale (es Oculus Rift). Questo test è stato svolto proprio per capire se, tramite l'utilizzo di un visore, fosse più comodo l'utilizzo dei classici mouse e tastiera oppure l'ausilio del Myo durante una sessione di gioco [23].

Il test è stato strutturato in due fasi. Nella prima fase è stato utilizzato, come metodo di input, la tastiera; nella seconda invece è stato utilizzato il dispositivo Myo. Entrambe sono state svolte con l'ausilio del visore e con lo stesso videogioco *"Need For Speed: Most Wanted"* nello stesso livello, per minimizzare le differenze. Alla fine di ogni fase al soggetto sono state poste 6 semplici domande:

1. Valuta con un voto da 0 a 5 le seguenti affermazioni:
   a) È stato semplice accelerare.
   b) È stato semplice decelerare/muoversi in retromarcia.
   c) È stato semplice sterzare a destra.
   d) È stato semplice sterzare a sinistra.
   e) È stato semplice usare la nitro.
   f) È stato semplice l'utilizzo del freno a mano.
2. Se hai votato 0 in una delle affermazioni precedenti, spiega la motivazione.
3. Se hai votato 5 in una delle affermazioni spiega la motivazione.
4. È stato semplice giocare?
5. Hai mai giocato prima a questo videogioco? *(SI/NO)*
6. Se alla domanda 5 hai risposto si, lo hai mai giocato con un visore per la realtà virtuale? *(SI/NO)*

Occorre notare che tutti e dieci i partecipanti erano studenti di informatica, quindi familiari nell'utilizzo di mouse e tastiera.

### Fase 1: Test con tastiera

La prima fase si è svolta con l'utilizzo della tastiera, i comandi sono così impostati: accelerare, decelerare, sterzare a destra, sterzare a sinistra nei tasti direzionali; freno a mano nella barra spaziatrice; nitro nel tasto shift.

**Risultati prima fase** Alla fine della prima fase si è notato che la maggioranza dei partecipanti non si è trovato a proprio agio nell'utilizzo della tastiera per via del visore. Come si può notare nella figura Figura 5.3, che mostra graficamente la media della *prima domanda*, la maggioranza dei candidati non ha trovato grosse difficoltà nell'operare con i tasti direzionali, ma ne ha trovate nell'utilizzo degli altri comandi.

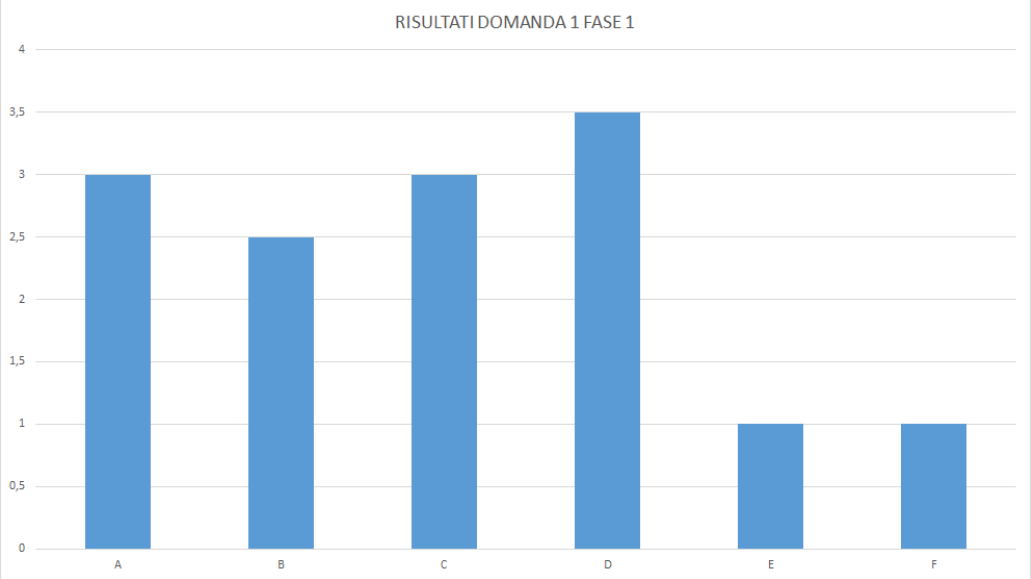



Figura 5.3: Grafico risultati prima domanda

Nella *terza domanda*, solo 3 persone hanno dato, ad una delle affermazioni del quesito 1 voto 5 perché, come spiegato da loro, molto familiari nel giocare a questo videogioco con la tastiera. Tuttavia va notato che questi soggetti non hanno mai distolto la mano dai tasti direzionali, per raggiungere gli altri comandi.

Nella *domanda 4*, il senso comune è stato quello di aver riscontrato difficoltà nell'abituarsi ai comandi anche se, come riportato nel *quesito 5*, 9 persone su 10 avevano già giocato in precedenza a questo titolo e 3 di loro lo avevano sperimentato anche con il visore.

### Test con il Myo

La seconda fase si è svolta con l'utilizzo del Myo come metodo di input, i comandi sono così impostati: ***"Wave-left"*** per reimpostare il centro virtuale, ***"Wave-right"*** per la nitro, ***braccio in alto*** per accelerare e ***in basso*** per decelerare ed infine ***braccio a destra/sinistra*** per sterzare.

**Risultati seconda fase** Al contrario della prima fase, la maggioranza dei soggetti ha dato un voto pari a 5 a tutte le affermazioni della *domanda 1*.

Come dimostra il grafico in Figura 5.4 l'utilizzo del Myo ha portato un miglioramento su tutti i fronti rispetto l'utilizzo della tastiera.

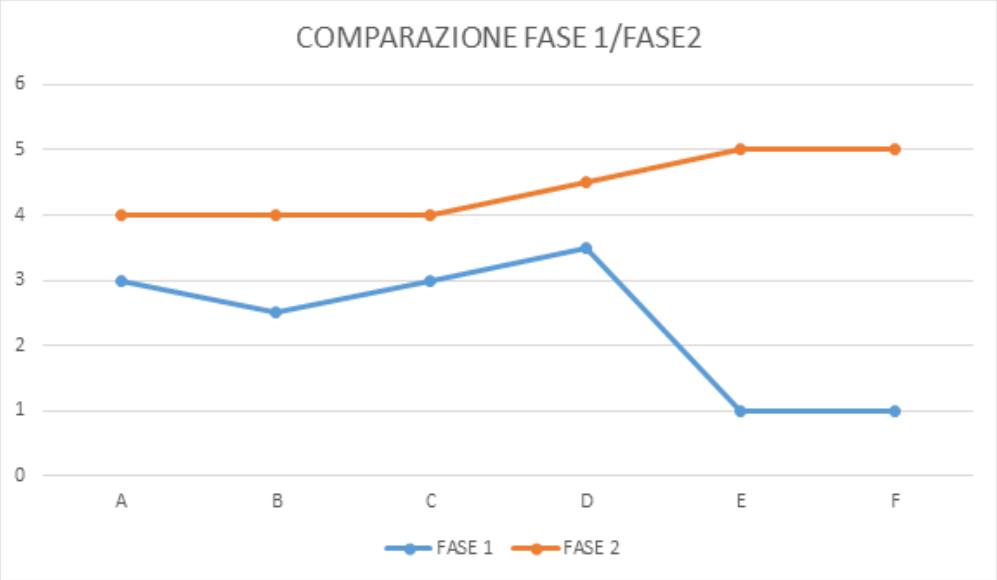



Figura 5.4: Grafico comparativo delle due fasi

Quasi tutti i soggetti hanno risposto alla *domanda 3* dicendo che all'inizio del test hanno avuto difficoltà nel ricordare i comandi ma dopo breve tempo si sono abituati e sono divenuti molto più abili a giocare rispetto all'utilizzo della tastiera, dato che il movimento della mano risulta più naturale quando viene indossato un visore della realtà virtuale.

**Conclusioni** Da questo test possiamo convenire che in un ambiente virtuale il Myo è risultato molto più efficace rispetto all'utilizzo della classica tastiera. Concludendo si può affermare che l'accuratezza dei suoi sensori e la semplicità dei comandi rendono il comparto videoludico (in VR) molto più immersivo, naturale e gratificante.

### 5.4.2 Creazione di nuove gesture

Questo progetto si pone come obbiettivo di utilizzare il dispositivo MYO per riconoscere sei nuove gesture diverse da quelle predefinite [24].

Il processo di riconoscimento in questo progetto viene suddiviso in più fasi: *acquisizione del segnale*, *filtraggio*, *determinazione finestra temporale della gesture*, *estrazione feature*, *classificazione* ed infine *invio della gesture classificata*.

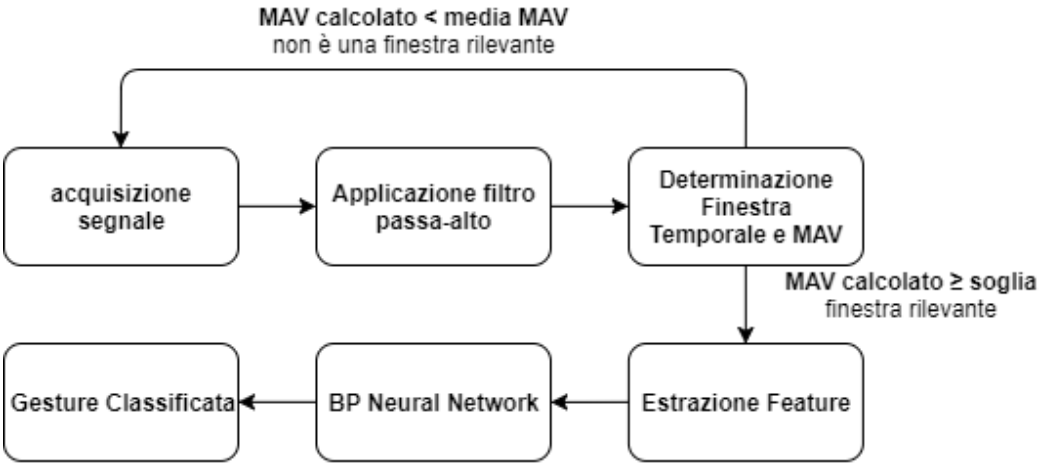



Figura 5.5: Flusso dei dati

Per fare in modo che il processo di riconoscimento sia efficace e performante, una delle fasi più critiche è quella dell'elaborazione dei segnali EMG e la conseguente estrazione delle feature che dovranno essere successivamente classificate. Al momento esistono una miriade di algoritmi che sono stati sviluppati e sperimentati per lo scopo. In questo progetto viene utilizzato un algoritmo *"ibrido"* [25] basato su due domini differenti: il primo è pensato sulla ***temporalità del segnale***, viene implementato grazie alla sua semplicità e velocità (esempi di questi algoritmi sono: *Mean Absolute Value "MAV", Root Mean Square "RMS", ecc.*); il secondo è basato sulla ***frequenza del segnale*** dato che i valori estratti da quest'ultimo sono molto robusti e quindi di rumore *"noise"* ridotto (esempi di questi algoritmi sono: *Mean Power Frequency "MPF", fast Fourier transform "FFT", ecc.*).

**Acquisizione e filtraggio del segnale** I dati provenienti dal Myo subiscono una fase di amplificazione e di filtraggio già prima di essere visibili al sistema. Tuttavia, data la natura debole del segnale proveniente dai sensori elettromiografici, durante la fase di trasmissione via Bluetooth possono insorgere interferenze che portano ad una contaminazione, aggiungendo rumore

(*noise*) al segnale. Quindi prima di poter utilizzarlo deve essere sottoposto ad un'altra fase di filtraggio per eliminare queste interferenze.

Conoscendo la frequenza effettiva del segnale proveniente dai sensori, che parte da *45hz* e arriva a massimo *195hz*, e sapendo che il Bluetooth trasferisce dati ad una frequenza di *200hz*, può venire utilizzato tranquillamente un filtro passa-alto con un taglio di *45hz*. Esso fa in modo che le interferenze sotto i *45hz* vengano scartate facendo passare solo i dati utili nell'intervallo *45-200hz*.

**Determinazione finestra temporale** Per fare in modo che il riconoscimento delle gesture avvenga garantendo prestazioni real-time è necessario che la determinazione della finestra temporale sia il più veloce e precisa possibile. In questo progetto per determinare le varie finestre viene fissata una *soglia* e successivamente utilizzato il Mean Absolute Value (MAV) come metodo di giudizio per determinare l'inizio e la fine di un'azione.

Dato che il Myo invia i dati separati per ogni sensore (quindi 8 segnali diversi, uno per ogni sensore) si deve prima calcolare il MAV di ogni finestra di ciascun sensore:

$$MAV_i = \frac{1}{N}\sum_{K=1}^{N}|x(k)|, \qquad \text{con } i = 1, 2, \ldots, 8$$

Dove $i$ corrisponde al numero dei sensori, $N$ corrisponde alle finestre temporali e $x$ corrisponde ai dati contenuti nella finestra temporale $k$.

Successivamente vengono sommati per creare un unico valore:

$$MAV = \sum_{i=1}^{8} MAV_i$$

Se il valore calcolato risulta minore della *soglia* la finestra viene scartata. È da tenere a mente i che segnali provenienti dai sensori possono variare da individuo a individuo, quindi è necessario una fase preliminare di raccolta dati per determinare la giusta soglia che possa adattarsi alla maggioranza delle persone.

**Estrazione feature** Come detto prima esistono tantissimi algoritmi per riuscire ad estrarre le feature dalle varie finestre temporali. Questo progetto si basa su un algoritmo che tratta in modo combinato il dominio temporale e quello delle frequenze già sperimento nella ricerca [25] con piccoli adattamenti per aumentarne le prestazioni.

L'algoritmo impiegato si suddivide in 3 fasi:

1. Si estraggono gli spettri di potenza su base temporale;
2. Utilizzando la tecnica della *similarità del coseno* (calcolo della similitudine tra due vettori effettuata calcolando il coseno tra di loro) si crea un vettore orientamento, stimato fra lo spettro di potenza originale estratto dai sensori EMG e il loro *cepstrum* (il risultato della *trasformata di Fourier* applicata allo spettro in decibel);
3. Vengono moltiplicate le feature estratte dalla finestra temporale corrente con $N$ (prestabilito) finestre temporali precedenti, per fare in modo di avere un segnale più preciso.

**Classificazione delle gesture** Per la fase di classificazione si è scelto di utilizzare un algoritmo di apprendimento supervisionato, il *BP Neural Network* (BP sta per *Back Propagation*). Esso è uno dei tipi di rete neurale più utilizzato, si tratta di un algoritmo di ottimizzazione basato sulla discesa del gradiente per minimizzare la funzione obbiettivo.

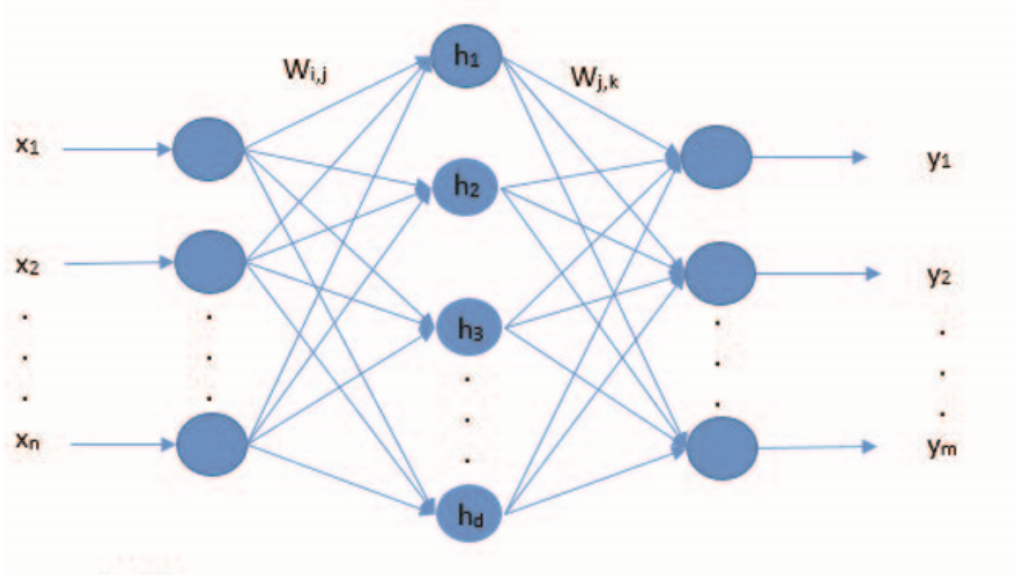



Figura 5.6: Rappresentazione grafica di una BP neural network

La fase di training prende degli esempi che vengono forniti alla rete come input ($X$) e, dato che è un apprendimento supervisionato, viene fornito anche il valore atteso in output ($Y$). La differenza fra il valore di output proveniente dalla rete e il valore atteso fa in modo che i due gradienti ($W_{i,k}$ e $W_{j,k}$)

diminuiscano, riducendo la possibilità che un prossimo esempio simile prenda lo stesso percorso.

Arrivati alla fine della fase di training la rete neurale è pronta per ricevere e processare autonomamente nuovi dati.

**Sperimentazione e Test** Per la sperimentazione ed i test sono stati impiegati cinque soggetti, ognuno dei quali doveva svolgere le sei gesture per 30 volte come *pool di training* e altre 50 volte come *pool di testing*. Il set di training è stato filtrato e processato come spiegato nelle prime due fasi, e in seguito fornito alla rete neurale. Successivamente si è preso il set di testing, anch'esso processato e filtrato ed infine si è verificato il corretto funzionamento del sistema.

| Gesture | 1 | 2 | 3 | 4 | 5 | 6 | unknow |
|---|---|---|---|---|---|---|---|
| 1 | 95.76% | 0.84% | 0 | 0 | 0 | 0.03% | 3.37% |
| 2 | 2.03% | 94.26% | 0 | 0.43% | 0 | 0 | 3.28% |
| 3 | 0 | 0 | 93.65% | 0.25% | 0.58% | 1.04% | 4.48% |
| 4 | 1.07% | 0 | 0 | 93.77% | 0 | 0 | 5.16% |
| 5 | 0 | 0 | 0 | 0 | 98.90% | 0 | 1.10% |
| 6 | 0.71% | 0 | 0 | 0 | 1.01% | 91.50% | 6.78% |

(a) Test soggetto 1

| Gesture | 1 | 2 | 3 | 4 | 5 | 6 | unknow |
|---|---|---|---|---|---|---|---|
| 1 | 91.49% | 0 | 0 | 0 | 3.43% | 0 | 5.08% |
| 2 | 0 | 95.21% | 0 | 0 | 0 | 0.10% | 4.69% |
| 3 | 0 | 0 | 88.48% | 0 | 0 | 0.84% | 10.68% |
| 4 | 1.68% | 0 | 0 | 95.94% | 0 | 0 | 2.38% |
| 5 | 0 | 0 | 0.77% | 0 | 98.43% | 0.12% | 0.68% |
| 6 | 0 | 0 | 7.86% | 0 | 0 | 85.08% | 7.06% |

(b) Test soggetto 2

| Gesture | 1 | 2 | 3 | 4 | 5 | 6 | unknow |
|---|---|---|---|---|---|---|---|
| 1 | 93.63% | 0.42% | 0 | 0 | 1.22% | 0.02% | 4.71% |
| 2 | 1.02% | 94.74% | 0 | 0.22% | 0 | 0 | 4.02% |
| 3 | 0 | 0 | 90.05% | 0.13% | 0.29% | 0.37% | 9.16% |
| 4 | 1.38% | 0 | 0 | 94.86% | 0 | 0 | 3.76% |
| 5 | 0 | 0 | 0.39% | 0 | 98.67% | 0.06% | 0.88% |
| 6 | 0.36% | 0 | 3.43% | 0 | 0.55% | 88.65% | 7.01% |

(c) Test complessivo

Tramite il sistema si è potuto notare che esistono caratteristiche che cambiano da persona a persona. Dalle prime due tabelle Figura 5.7(a), Figu-

ra 5.7(b) si può notare come la qualità dei segnali cambia ma combinando tutti i risultati provenienti dai test dei vari soggetti il sistema è riuscito a riconoscere la gesture corretta in circa il 93% dei casi Figura 5.7(c).

# Capitolo 6

# Conclusioni

Negli ultimi anni l'utilizzo del corpo umano come metodo di interazione fra utente e macchina è, e continua ad essere, materia di grande interesse. Come è stato già sottolineato, l'attività di ricerca è incentivata dalla crescente domanda di applicazioni "Hands Free" e dal continuo ingresso nel mercato di nuove tecnologie. Tuttavia esistono limitazioni intrinseche ad esse ed ognuna ha delle problematiche che devono essere risolte prima che queste possano risultare davvero efficaci.

Allo stato dell'arte sono stati studiati un'infinità di approcci validi che vengono utilizzati per affrontare la sfida del riconoscimento delle gesture ma nessuno di essi è utilizzabile universalmente.

Una ricerca di Juniper Research [26] osserva che *«La realtà virtuale e gli wearable hanno indicato il modo in cui le gesture e il tatto sono in grado di fornire nuovi modi per interagire con la tecnologia»*. Ma spetta alle aziende investire e agli utenti accettare questo inevitabile futuro. Lo studio sottolinea, però, che questa "rivoluzione" non avverrà in maniera né automatica né omogenea. I dispositivi più tradizionali (es. smartphone e computer) faranno più fatica a essere accettati e si stima che meno del 5% includeranno una qualche forma di gesture o controllo del movimento.

Lo scopo che mi sono posto nella stesura del mio scritto è stata quella di fare una panoramica generale sulla Gesture Recognition, una materia che ingloba tantissimi campi: dall'elettronica al Machine Learning e all'intelligenza artificiale.

In particolare, nella prima parte ho analizzato le tecnologie hardware e software utilizzate nella Gesture Recognition, che hanno dimostrato non solo le potenzialità ma anche le varie limitazioni a cui è soggetta.

Nella seconda parte ho analizzato il funzionamento del primo vero framework sviluppato appositamente per questo ambito di ricerca, avvalendomi

di due sperimentazioni pratiche che hanno dimostrato la sua efficienza.

Nell'ultima parte ho presentato uno studio del Myo Armband, un dispositivo che ho testato di persona, esaminandolo dal punto di vista di un programmatore che vuole integrare un sistema di Gesture Recognition tramite l'utilizzo di questo device. Ho analizzato il suo SDK e ho mostrato che, anche a distanza di quattro anni, non è del tutto completo e necessita, a mio parere, di una revisione. Nella parte conclusiva ho analizzato due lavori trovati nella letteratura, il primo mostra l'efficienza del dispositivo in un ambiente di realtà virtuale; il secondo mostra un progetto volto a fare riconoscere al dispositivo gesture non predefinite con ottimi risultati.

La mia opinione è quella che ora come ora, la tastiera e il mouse continueranno ancora a essere il modo principale con cui interagire con i computer, mentre il touchscreen quello per usare gli smartphone. La differenza tra quello che esiste oggi e un futuro innovativo dipenderà da quante aziende "coraggiose" investiranno seriamente nella ricerca, per integrare le gesture e il controllo del movimento di propri dispositivi, non solo sotto forma di "opzione".

# Ringraziamenti

Un ringraziamento speciale va alla mia famiglia che mi ha sostenuto nella scelta di questa università pur avendo alle spalle un passato con un istituto alberghiero.

Ringrazio la mia cugina Sandra che, anche se totalmente ignorante in materia, ha cercato in tutti i modi di aiutarmi nella correzione della sintassi di questo scritto.

Ringrazio Manni che mi ha aiutato nella parte iniziale, quando avevo la sindrome del foglio bianco.

Ringrazio infine il mio relatore il Prof. Alessandro Ricci per la sua disponibilità nel chiarire qualsiasi dubbio sorgesse durante lo sviluppo di questa tesi.

# Bibliografia


[1] Reflection Technology Private Eye Display.
*https://www.google.com/culturalinstitute/beta/asset/reflection-technology-private-eye-display/QgFnZtDAdVz0CQ*

[2] Wearable Device Shipments to Reach 430 Million Units Annually by 2022
*https://www.tractica.com/newsroom/press-releases/wearable-device-shipments-to-reach-430-million-units-annually-by-2022/*

[3] Painless Blood-Free Glucose Monitor - K'Watch Glucose
*http://www.pkvitality.com/ktrack-glucose/*

[4] Fitbit Website
*https://www.fitbit.com*

[5] VR Chat Create and play in Virtual Worlds
*https://www.vrchat.net/*

[6] Siddharth S. Rautaray, Anupam Agrawal
*Vision based hando gesture recognition for human computer interaction: a survey*

[7] Google, Project Soli
*https://atap.google.com/soli/*

[8] A.Vattani
*k-means Requires Exponentially Many Iterations Even in the Plane*

[9] Tracy Westeyn, Helene Brashear, Amin Atrash, Thad Starner
*Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition*

[10] Incidenti, in Italia 3 su 4 sono per distrazioni al volante
*http://www.ansa.it/canale_motori/notizie/sicurezza/2017/07/18/incidenti-in-italia-3-su-4-sono-per-distrazioni-al-volante-_45226d79-2a88-4b7e-890c-d6752287a910.html*

[11] Market delle App del MYO
*https://market.myo.com/*

[12] Significato dei led del MYO
*https://support.getmyo.com/hc/en-us/articles/202724369-What-are-the-different-LED-status-descriptions-for-the-Myo-armband*

[13] Varadach Amatanon, Suwatchai Chanhang, Phornphop Naiyanetr, Sanitta Thongpang
*Sign Language-Thai Alphabet Conversion Based on Electromyogram (EMG)*

[14] Mithileysh Sathiyanarayanan, Tobias Mulling, Bushra Nazir
*Controlling a Robot Using a Wearable Device (MYO)*

[15] Kristian Nymoen, Mari Romarheim Haugen, Alexander Refsum Jensenius
*MuMYO - Evaluating and Exploring the MYO Armband for Musical Interaction*

[16] Icarus Rising
*https://market.myo.com/app/55392b70e4b02f8140d20a84/icarus-rising*

[17] Sito internet del linguaggio LUA
*https://www.lua.org/*

[18] SDK Windows
*https://developer.thalmic.com/docs/api_reference/platform/index.html*

[19] SDK Android
*https://developer.thalmic.com/docs/api_reference/android/index.html*

[20] SDK IOs
*https://developer.thalmic.com/docs/api_reference/ios/index.html*

[21] Breve spiegazione del quaternione
*http://developerblog.myo.com/quaternions/*

[22] Analizzatore bluetooth per estrazione manuale dati EMG
*https://github.com/meleap/myo_AndoridEMGProgettoMIT*

[23] Seema Rawat, Somya Vats, Praveen Kumar
*Evaluating and Exploring the MYO ARMBAND*

[24] Shuzhan He, Chenguang Yang, Min Wang, Long Cheng, Zedong Hu 2017
*Hand Gesture Recognition using MYO Armband*

[25] Rami N. Khushaba, Ahmed Al-Ani, Ali Al-Timemy, Adel Al-Jumaily
*A Fusion of Time-Domain Descriptors for Improved Myoelectric Hand Control*

[26] Sito internet Juniper Research *https://www.juniperresearch.com/home*