POLITECNICO DI MILANO

Corso di Laurea Magistrale in Ingegneria Informatica

Dipartimento di Elettronica e Informazione

NUOVO APPROCCIO AL COMANDO EMG PER PROTESI COMPLETE DI ARTO SUPERIORE

AI & R Lab

Laboratorio di Intelligenza Artificiale e Robotica del Politecnico di Milano

Relatore: Chiar.ma Prof.ssa Giuseppina GINI

Tesi di Laurea di:

Adrian Alejandro ORTIZ, matricola 784218

Anno Accademico 2012 – 2013

Alla mia famiglia e a Simona

Sommario

La progettazione di protesi di arto superiore è un procedimento molto complesso che ha avuto negli ultimi anni numerosi sviluppi e miglioramenti. Nonostante ciò, restano ancora molti aspetti che possono essere migliorati e perfezionati in modo che l'utente possa avere il pieno controllo della protesi.

Questo lavoro di tesi è stato svolto presso il Dipartimento di Elettronica, Informazione e Bioingegneria del Politecnico di Milano e si colloca all'interno di un più ampio progetto svolto in collaborazione con il Dipartimento di Meccanica del Politecnico di Milano.

Lo scopo finale del progetto è stato quello di realizzare un sistema completo, innovativo e a basso costo, in grado di controllare una protesi attiva di arto superiore durante la fase di avvicinamento e presa a vari oggetti posti di fronte alla protesi. Per l'acquisizione di informazioni che riguardano l'ambiente, è stato utilizzato un sensore Kinect, mentre per l'acquisizione di informazioni sul tipo di movimento che il paziente vuole effettuare, sono stati utilizzati alcuni elettrodi collegati ai muscoli residui della spalla atti a captare il tipo di movimento in accordo con la volontà del paziente.

Grazie a queste informazioni si è riuscito a simulare, tramite un controllo innovativo, il movimento della protesi secondo le intenzioni del paziente, arrivando ad afferrare oggetti posti di fronte ad esso.

Il progetto nasce come punto di partenza per la realizzazione di una protesi attiva soprattutto grazie all'utilizzo di componenti comuni, facilmente reperibili e dal prezzo accessibile che pongono questo lavoro di tesi come una soluzione innovativa in quelle applicazioni che hanno come obiettivo la restituzione di funzionalità e indipendenza ai soggetti interessati da amputazioni totali agli arti superiori.

Indice Generale

Indice delle Figure

Indice delle Tabelle

Capitolo 1

Introduzione

L’arto superiore può essere descritto come un sistema il cui principale organo effettore è la mano, grazie alla quale l’uomo è in grado di compiere una varietà molto ampia di funzioni e abilità.

La progettazione di protesi di arto superiore è perciò un procedimento molto complesso, che ha avuto negli ultimi anni numerosi sviluppi e miglioramenti. Nonostante questo, restano ancora molti aspetti che possono essere migliorati e perfezionati in modo che l’utente possa avere il pieno controllo della protesi.

La funzione principale di una protesi è quella di mimare l’aspetto e sostituire la funzione dell’arto mancante. Idealmente questa dovrebbe essere semplice da controllare, confortevole da indossare ed esteticamente gradevole [1].

Un grosso problema nella realizzazione della protesi, è che essa debba muoversi allo stesso modo dell’arto mancante e nel caso particolare della mano, la complessità di progettazione diventa molto elevata. Tuttavia, il grande numero di diversi modelli esistenti, diventa limitato a un paio di soluzioni nella loro applicazione, a causa di problemi d’identificazione del segnale, di miniaturizzazione dei componenti e di altri problemi di progettazione; la ricerca futura deve chiudere questo divario tra 'ideale' e pragmatico.

Uno dei motivi per cui lo sviluppo delle protesi di arto superiore è cosi complesso è dato dall’elevato numero di gradi di libertà che questa possiede. Inoltre, i pazienti affetti da queste tipologie di amputazioni sono in numero molto inferiore rispetto alla totalità delle persone amputate; questo, fa sì che l’evoluzione nelle tecnologie utilizzate, proceda più lentamente rispetto ad altri tipi di protesi [2][3].

Ad oggi, il metodo più utilizzato per il controllo di una protesi attiva di arto superiore, è basato sull'EMG Pattern Recognition che permette di identificare il tipo di movimento che il soggetto è intenzionato a svolgere.

In questo modo, il soggetto deve contrarre il muscolo corrispondente al grado di libertà che vuole controllare; il risultato, è un controllo più intuitivo e una selezione rapida di ogni funzione. La non invasività del metodo lo rende apprezzato dai pazienti, anche se, durante i primi mesi dopo il montaggio della protesi, il controllo risulti molto innaturale e necessiti di un forte sforzo mentale.

Purtroppo, le protesi presenti in commercio, che utilizzano questi segnali, consentono solamente degli spostamenti molto robotici e lenti, che la maggior parte dei pazienti non gradisce. I segnali, inoltre, individuano i singoli movimenti del muscolo che deve essere mosso, ma non permettono alla protesi completa di intuire in modo immediato il tipo di movimento che dovrebbe effettuare.

Fino ad oggi, sono state effettuate diverse ricerche concernenti lo sviluppo di sistemi di controllo per protesi attive di mano, polso e gomito, basati sulla *Pattern Recognition* del segnale elettromiografico; limitata attenzione è stata dedicata al controllo di protesi per disarticolazione della spalla. Questo è dovuto al fatto che la maggior parte delle amputazioni ad arti superiori avviene a livello trans-radiale o trans-omerale.

Il lavoro di tesi effettuato da Scannella e Rivela [38] ha messo in evidenza che è possibile riconoscere con elevata precisione uno fra quattro movimenti acquisendo informazioni tramite gli sEMG dei muscoli della spalla. Questi segnali, sono inoltre, sempre presenti in qualsiasi tipo di amputazione subita agli arti superiori, rendendo, questo studio, fondamentale per la ricerca in campo medico e protesico.

Questo lavoro di tesi è stato svolto presso il Dipartimento di Elettronica, Informazione e Bioingegneria del Politecnico di Milano e si colloca all'interno di un più ampio progetto svolto in collaborazione con il Dipartimento di Meccanica del Politecnico di Milano.

Lo scopo finale del progetto è stato quello di realizzare un sistema completo, innovativo e a basso costo, che considerasse questo nuovo tipo di controllo e permettesse a una protesi di arto superiore di potersi muovere, in accordo ad informazioni provenienti da sensori, per poter raggiungere oggetti di fronte ad essa.

Il modello di protesi virtuale è stato sviluppato riproducendo la protesi progettata dal gruppo Meccanica dei Sistemi Uomo Macchina del Dipartimento di Meccanica del Politecnico di Milano. Essa è munita di: due gradi di libertà attivi a livello della spalla, che consentono di compiere i movimenti di flesso-estensione sul piano sagittale, di abduzione-adduzione sul piano frontale e loro combinazioni; un grado di libertà passivo a livello dell'omero, che impedisce danni alla protesi in caso di urto; un grado di libertà attivo in corrispondenza del gomito, che ne permette il movimento di flesso-estensione; una protesi di mano commerciale avente due gradi di libertà sul polso, più un grado di libertà per l'apertura e la chiusura delle falangi.

Per l'acquisizione di informazioni che riguardano l'ambiente, è stato utilizzato un sensore Kinect, per identificare oggetti di interesse al paziente ed eventuali ostacoli che potessero impedirne il movimento; mentre per l'acquisizione di informazioni sul tipo di movimento che il paziente vuole effettuare, sono stati utilizzati alcuni elettrodi collegati ai muscoli residui della spalla atti a captare il tipo di movimento in accordo con la volontà del paziente.

In questo lavoro di tesi, ho contribuito alla progettazione del codice di controllo della protesi virtuale e ho confrontato i vari risultati ottenuti dall'analisi delle varie parti del sistema, con l'obiettivo di verificare che questi soddisfino i requisiti prefissati.

Per motivi pratici e tempistici, è stato scelto di riprodurre la protesi fisica in un ambiente di simulazione dinamico che permettesse comunque di poter interagire con i vari sensori, in modo da poter creare un sistema il più possibilmente fedele a quello reale. E' per questo motivo che il codice prodotto per il controllo della protesi virtuale è stato implementato in modo tale che possa essere riutilizzato nel momento in cui si vorrà progettare il controllore della protesi fisica.

Alla fine di questo lavoro, si è riusciti a creare un sistema in grado di acquisire informazioni sul tipo di movimento, scegliere la traiettoria migliore per raggiungere un oggetto da afferrare e di muovere la protesi virtuale di conseguenza a queste.

La trattazione di questo studio di tesi è stata suddivisa nei seguenti capitoli:

Nel secondo capitolo, verrà dapprima effettuata una classificazione sui diversi tipi di protesi a seconda del grado di amputazione subita, successivamente verranno presentati gli attuali tipi di protesi per arto superiore oggi in commercio e i principali sistemi di controllo usati per comandarle.

Nel terzo capitolo verrà presentata la protesi fisica, utilizzata come riferimento e le sue funzionalità. Sarà presente, inoltre, una trattazione approfondita sul calcolo della cinematica diretta ed inversa di questa.

Nel quarto capitolo verranno mostrati i vari dispositivi e metodi utilizzati per realizzare l'intero sistema. In particolare, verranno descritti gli strumenti utilizzati per la rilevazione del segnale elettromiografico, il dispositivo e le librerie utili al sistema visivo e il simulatore utilizzato per la rappresentazione della protesi.

Il quinto capitolo sarà dedicato alla presentazione dell'architettura del sistema. In questo, verranno approfonditi, in particolar modo, i vari moduli software che la compongono.

Il sesto capitolo illustrerà le prove effettuate per testare la bontà della soluzione proposta e le valutazioni che da queste si possono trarre.

Infine, nel settimo capitolo, verranno presentate le conclusioni di questo lavoro di tesi e i possibili sviluppi futuri che da questa sono nati.

Capitolo 2

Protesi di arto superiore e loro comando

In questo capitolo verrà dapprima effettuata una classificazione dei diversi tipi di protesi secondo il livello di amputazione dato dalla norma europea EN ISO 9999, successivamente verranno presentati gli attuali tipi di protesi per arto superiore oggi in commercio e i principali sistemi di controllo usati per comandarle.

2.1 Classificazione delle protesi di arto superiore

La classificazione proposta sarà suddivisa in base al livello di amputazione e alle funzionalità fornite; a tale scopo, si farà riferimento ad una norma europea.

2.1.1 Norma EN ISO 9999

La norma EN ISO 9999 ha come scopo la classificazione degli ausili tecnici per disabili, per facilitare la trattazione degli stessi. Con ausilio tecnico, si identifica qualsiasi prodotto, strumento, attrezzatura o sistema tecnologico utilizzato da un disabile che prevenga, compensi, attenui o neutralizzi una menomazione, una disabilità o un handicap.

Innanzitutto è utile dare la definizione di protesi data dalla norma:

"Ausilio ortopedico che vicaria o sostituisce, seppur parzialmente l'arto mancante sia sotto l'aspetto funzionale che estetico".

Nella norma troviamo anche la seguente definizione di protesi di arto superiore:

"Una protesi di arto superiore è un insieme di componenti compatibili, solitamente prodotti da un singolo produttore e commercialmente disponibile. I componenti possono essere integrati con qualsiasi altro componente fabbricato individualmente, per produrre una gamma di differenti protesi di arto superiore."

Nella norma EN ISO 9999 le protesi vengono classificate in base ai gradi di amputazione, in particolare nella Tabella 1, vengono illustrati i gradi di amputazione per l'arto superiore.

Codice	***Descrizione***
06 18 03	Protesi di dito e amputazione di mano
06 18 06	Protesi per disarticolazione di polso
06 18 09	Protesi per amputazione transradiale
06 18 12	Protesi per disarticolazione di gomito
06 18 15	Protesi per amputazione transomerale
06 18 18	Protesi per disarticolazione di spalla
06 18 21	Protesi per amputazione interscapolotoracica

Tabella 1: Classificazione ISO per protesi di arto superiore

In questo lavoro di tesi ci focalizzeremo sull'ultimo tipo di protesi, quella identificata dal codice 06 18 21 - Protesi per amputazione interscapolotoracica.

Prima di andare avanti con la classificazione, è utile ricordare che non tutti i pazienti scelgono di usufruire di una protesi per sopperire alla mancanza dell'arto. Soggetti che hanno subito una perdita completa del braccio, o che sono nati con questa assenza, possono trovare l'utilizzo di una protesi un ostacolo insormontabile. I motivi principali di questo sono: l'iniziale inesperienza nell'uso della protesi, la mancanza di sensazione tattile e il fastidio nell'utilizzo di un corpo estraneo [4].

Per ottenere buoni risultati riabilitativi che permettano un completo reinserimento sociale e lavorativo a persone che necessitano di protesi di arto, occorre definire specifici e mirati programmi riabilitativi che si personalizzino e si differenzino in relazione al caso da trattare [5].

2.1.2 Classificazione funzionale delle protesi di arto superiore

Una volta individuati i diversi tipi di protesi rispetto l'amputazione subita, è utile riuscire a classificare queste, rispetto alle caratteristiche costruttive e funzionali. In questo modo, possiamo suddividere le protesi nei seguenti tipi: protesi passive/estetiche, protesi ad energia intracorporea/funzionali e protesi ad energia extracorporea/attive.

2.1.2.1 Protesi estetiche

Le protesi di arto superiore estetiche o cosiddette passive sono utilizzate da persone che considerano il loro aspetto esteriore molto importante o laddove le protesi attive che si avvalgono di fonti di energia corporea o extracorporea non possano essere utilizzate con successo.

Per alcuni utenti, infatti, il funzionamento attivo di una protesi di arto superiore ha minor importanza rispetto al design, all'aspetto estetico, al comfort, al peso ed al facile impiego. Il vantaggio nell'utilizzo di queste protesi è che si adattano a tutti i livelli di amputazione. Molti amputati utilizzano tuttavia la protesi, non solo come sostituzione visiva dell'arto mancante, ma anche come supporto alla mano sana o per portare oggetti leggeri. In questo caso, ci si serve delle cosiddette "mani sistema" o "mani passive" che possono essere aperte passivamente, aiutandosi con la mano sana e che si chiudono automaticamente, grazie ad un meccanismo a molla.

Le mani cosmetiche devono soddisfare gli standard estetici degli utenti che ritengono fondamentale recuperare l'aspetto estetico della mano ed un aspetto naturale. Normalmente le mani cosmetiche comprendono una mano interna e un guanto cosmetico. La forma, il colore e la struttura della superficie del guanto cosmetico lo rendono molto simile alla mano naturale, persino nei dettagli.

Figura 1 : Esempio di protesi estetica

I principali materiali usati in questo tipo di protesi sono: PVC rigido, lattice flessibile e silicone. Il vantaggio del PVC è dato dalla colorazione al momento della fabbricazione: poiché un guanto cosmetico in PVC, anche se graffiato, mantiene la propria tonalità e colore.

Il lattice è il materiale più comunemente utilizzato nelle protesi cosmetiche. Un guanto in lattice ha il vantaggio di poter essere adattato, in un secondo momento, alle esigenze del paziente aggiungendo, ad esempio, tratti caratteristici di questo, come le rughe, in modo da rendere più naturale l'aspetto. Questo materiale è leggero e poco costoso, ma si macchia facilmente. Il silicone, invece, è quello che dà migliori risultati estetici, ha una maggiore resistenza nel tempo e non si macchia facilmente. Gli unici svantaggi, però, rendono la protesi più pesante e il materiale più costoso rispetto agli altri.

2.1.2.2 Protesi attive ad energia corporea

L'idea base delle protesi ad energia corporea è quella di utilizzare i muscoli residui o muscoli di altre zone per muovere la protesi. Questa tecnica trova spazio nelle protesi cinematiche che sfruttano l'ausilio di bretellaggi: attraverso questi bendaggi a trazione il paziente può eseguire movimenti controllati servendosi dell'energia meccanica prodotta dal movimento della spalla o del moncone.

Le protesi cinematiche possono essere utilizzate praticamente per tutte le amputazioni fino al 3° medio transomerale, mentre nelle disarticolazioni di spalla la presenza di un moncone molto corto implica leve svantaggiose, tali da rendere difficoltoso l'azionamento della protesi [6].

A fronte di ottime caratteristiche costruttive, leggerezza, robustezza e affidabilità, tali protesi forniscono un basso livello di comfort, dovuto alla presenza di cinghie in tensione e ad un ridotto volume di lavoro. Queste, però, in alcuni casi, possono soddisfare sia esigenze cosmetiche che funzionali. Infatti, nella protesi di una mano, è racchiuso sia il meccanismo che permette il movimento, sia una mano interna che dà la forma e l'estetica necessaria alla protesi.

Il sistema della mano può essere di diversi tipi ma quelli più utilizzati sono quelli che sfruttano un tirante per l'apertura e la chiusura della mano. Inoltre, se la funzionalità della mano prevale sull'estetica, si può applicare un uncino o altri ausili specifici come strumenti da lavoro per una presa efficace. L'uncino della OttoBock, ad esempio, è utile in quelle attività in cui si richiede una certa precisione per afferrare piccoli oggetti, come le viti; questi tipi di strumenti vengono scelti, da molti pazienti, al posto di una mano cosmetica, data la sua utile funzionalità [3].

Figura 2: Protesi funzionale di arto superiore ad energia corporea

2.1.2.3 Protesi attive ad energia extracorporea

Vi sono stati diversi tentativi di realizzazione di protesi capaci di utilizzare fonti di energie alternative a quella corporea, ma il miglior risultato è stato ottenuto con l'uso di protesi elettromeccaniche. Queste impiegano l'energia fornita da accumulatori per azionare un motore elettrico in corrente continua, il cui verso di rotazione determina la chiusura o l'apertura della mano. Va poi specificato in che modo il paziente comanda gli azionamenti.

Le protesi ad energia extracorporea più diffuse in commercio sono quelle mioelettriche che non vengono controllate dalla forza muscolare dell'utente, ma mediante il comando mioelettrico.

Tali protesi, peraltro di introduzione relativamente recente, (le prime protesi installate in Italia risalgono al 1965), rappresentano lo stato dell'arte in termini tecnologici ed è in questo settore che si stanno avendo notevoli progressi grazie agli sviluppi dell'elettronica e della meccanica. Attualmente controllando fino a 3 motori in corrente continua si può determinare la chiusura o l'apertura della mano, la flesso-estensione del gomito e la prono-supinazione del polso.

Nelle protesi a comando mioelettrico occorre predisporre nell'invasatura una sede per ospitare gli elettrodi. Questi devono essere disposti su muscoli che forniscano un segnale elettromiografico sufficiente e in modo che non interferiscano tra di loro, inoltre, che non si attivino a seguito di altri movimenti, se non quelli desiderati.

Per attuare un movimento, ad esempio, l'apertura della mano, il paziente deve contrarre una fascia muscolare sulla quale è posto l'elettrodo, il quale rilevando una attività elettromiografica informa l'unità centrale della volontà di movimento espressa dal paziente.

Figura 3: Protesi attiva ad energia extracorporea

Tale attività, che ha origine dal meccanismo intrinseco di contrazione della fibra muscolare, ha un valore molto basso dell'ordine di decine di microvolt, ed una banda di frequenze situata al di sotto di 1KHz. E' per questo motivo che si ha bisogno di elaborare con estrema attenzione il segnale, per poter trovare un comando significativo per la protesi stessa. Per poter utilizzare questi segnali, perciò, ci sarà bisogno di amplificarli ed impiegare solo quelli che vengono generati in modo indipendente l'uno dall'altro, in modo da poter distinguere i singoli movimenti.

Le protesi funzionali a comando mioelettrico od elettronico possono essere utilizzate dai pazienti che abbiano subito amputazioni dalla disarticolazione di polso fino al livello di disarticolazione di spalla, se le condizioni sopra citate sono soddisfatte.

I vantaggi principali di queste sono l'elevata forza di presa e l'elevato grado di funzionalità anche con livelli di amputazione molto prossimali. Purtroppo queste non sempre possono essere applicate come nel caso in cui i segnali elettromiografici siano insufficienti o non controllabili indipendentemente oppure se la protesi ha un peso troppo elevato rispetto il paziente che la deve utilizzare, o infine, se la si usa per effettuare lavori troppo pesanti per cui la protesi potrebbe venir danneggiata.

Data la complessità della protesi, il paziente ha bisogno di un periodo di allenamento per poter imparare a servirsene, dato che i movimenti da effettuare sono molto robotici e innaturali. Un'altra problematica di queste protesi, è l'elevato costo commerciale, che rende questo tipo di tecnologia non accessibile a tutti.

E' in questo campo che si cerca di trovare soluzioni a basso costo alternative a quelle presenti in commercio per poter rendere questo prodotto più semplice da utilizzare e alla portata di tutti [7].

2.1.3 Struttura della protesi

La struttura di una protesi di arto superiore si può schematizzare in tre elementi principali: l'organo di presa, le parti di collegamento con eventuali articolazioni e l'invasatura.

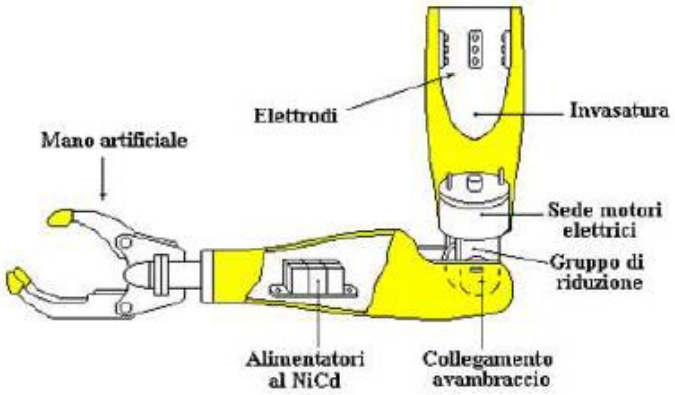


Figura 4: Struttura della protesi

L'invasatura è la parte di protesi che va a contatto con il moncone; questa deve essere personalizzata per ciascun paziente in modo da garantire una perfetta simbiosi con il moncone e deve essere costruita con materiali che non provochino allergie. Viene dapprima realizzata partendo dal modello di gesso del moncone, e successivamente deve essere provata per garantire il perfetto controllo della protesi, anche nel compiere le manovre più complesse.

L’organo di presa deve garantire la funzione di apertura e chiusura della mano. Questo movimento è realizzato tramite la chiusura del pollice sulla coppia indice, infatti l’unica articolazione presente è quella del metacarpo-falangea. Tale tipo di presa a pinza, viene chiamata “tridigitale” poiché partecipano tre dita alla presa, mentre le altre due hanno solo una funzione estetica.

Le parti di collegamento si interpongono tra l’invasatura e l’organo di presa e, a seconda del livello di amputazione del paziente, possono presentare articolazioni. Tali connessioni possono essere esoscheletriche, nel caso in cui le pareti esterne della protesi abbiano un ruolo strutturale, o endoscheletriche, se la componente esterna ha solo funzione estetica e il ruolo portante è svolto da una struttura modulare interna

Lo sviluppo di protesi per arto superiore è stato focalizzato su protesi attive per pazienti con amputazioni fino all’articolazione di gomito. Il fatto è dovuto al numero ridotto di pazienti con un grado di amputazione maggiore di esso. Per questa ragione attualmente non esistono in commercio protesi complete e definitive di giunti di spalla motorizzati.

Le protesi per arto superiore in commercio sono quindi dotate soltanto di tre gradi di libertà e permettono, oltre alla chiusura e apertura della mano, i movimenti relativi al gomito e al polso.

2.2 Comando

Per il controllo della protesi esistono diversi tipi di strategie di comando, ma la più diffusa è quella che utilizza il segnale elettromiografico. Il controllo di queste protesi si può effettuare in tre modi diversi [6]:

1. Comando digitale: tramite questo la velocità di contrazione rimane invariata anche in caso di segnali muscolari di diversa intensità. La forza di contrazione dipende dalla durata del segnale muscolare;
2. Comando DCM (Dynamic Mode Control): utilizza un solo elettrodo. L’intensità del segnale registrato funge da discriminante per il movimento che verrà eseguito dall’arto artificiale. Questo tipo di controllo viene

impiegato nelle amputazioni più prossimali, spesso bilaterali e con pochi muscoli a disposizione;

3. Multicanale: viene utilizzato qualora vi sia la necessità di attivare diverse componenti protesiche sfruttando una coppia di elettrodi a controllo proporzionale.

Altri tipi di controlli possibili sono quelli che utilizzano interruttori o comandi vocali [8], ma in entrambi i casi si hanno strategie poco intuitive e poco apprezzate dai pazienti che le utilizzano.

2.2.1 Segnale elettromiografico

L'elettromiografia è una tecnica diagnostica molto utilizzata in campo neurologico e ortopedico. Questa fornisce informazioni sia per quanto riguarda la funzionalità dei nervi periferici sia per quanto concerne i muscoli scheletrici.
Scopi principali dell'elettromiografia sono sia l'analisi dell'attività muscolare a riposo e durante l'attivazione volontaria, sia lo studio delle conduzioni nervose, motorie e sensitive.

Il segnale mioelettrico (EMG) può essere definito come una manifestazione di tipo elettrico derivante dall'attivazione muscolare associata alla contrazione e più precisamente dalle correnti ioniche che scorrono lungo le membrane delle fibre muscolari.

L'elettromiografia ha un ruolo molto importante nella riabilitazione dei muscoli che necessitano specifiche cure e terapie, dato che questa permette di valutare, in modo molto approfondito, la sede, l'entità e la tipologia della lesione.

Ad oggi, il metodo più utilizzato per il controllo di una protesi attiva di arto superiore è quello di rilevare i segnali elettromiografici che provengono dai muscoli residui dell'arto superiore per identificare i vari movimenti che il paziente vuole effettuare [9].

L'utilizzo degli EMG per l'aiuto delle persone con amputazioni è tuttora in fase di sviluppo, per fare in modo di poter riconoscere, non solo i tipi di movimenti voluti

dal paziente, ma anche per capire i vari tipi di approccio verso gli oggetti e, nel caso specifico, della presa di questi[10][11].

Di seguito, verranno presentati i metodi per la rilevazione e per l'analisi del segnale:

2.2.2 Rilevazione del segnale

Tipicamente, per il rilevamento del segnale EMG, vengono impiegati elettrodi in configurazione bipolare seguiti da un amplificatore differenziale (Figura 5).

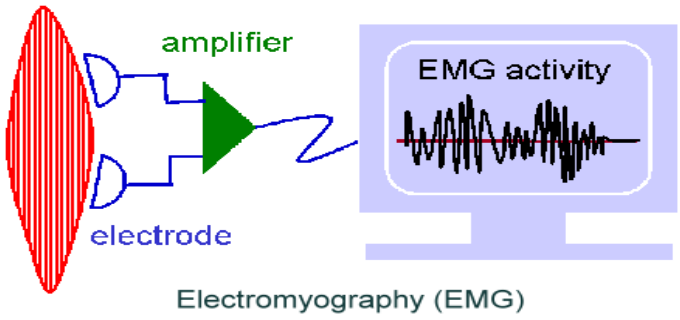


Figura 5: Esempio di acquisizione del segnale elettromiografico

Un elettrodo è un conduttore, di forma e natura opportuna, che adduce corrente o crea un campo elettrico in seno a un mezzo. Tramite questo conduttore possono essere misurati gli impulsi elettrici generati dai muscoli, ma dato che questi sono molto piccoli, variano da qualche μV fino a qualche mV, c'è bisogno di effettuare un'amplificazione del segnale per poterli valutare con una certa accuratezza.

La configurazione bipolare, utilizzata in tutti i sistemi elettromiografici, misura la differenza di tensione tra due elettrodi, posti vicini tra loro in zona contrattile.

Le metodologie per poter misurare i segnali elettromiografici possono essere di due tipi: una invasiva intramuscolare e un'altra superficiale.

La prima consiste nell'inserimento di un elettrodo ago o di un ago contenente due elettrodi a filo sottile, attraverso la pelle, nel tessuto muscolare in modo da poter misurare il segnale elettromiografico.

I vantaggi che offre l'impiego di elettrodi di questo tipo sono: alta sensibilità del segnale, possibilità di rilevare l'attività di una singola unità motoria, possibilità di accedere a muscoli profondi, basso volume di lavoro.

Gli svantaggi sono: richiesta di personale specializzato per l'applicazione, riposizionamento estremamente difficoltoso, costo, impossibilità di movimento durante l'utilizzo di questi, invasività, fastidio per il paziente.

La strategia che usa gli elettrodi superficiali, invece, consiste nel posizionamento di dischi metallici, di diametro inferiore al cm, sulla superficie della pelle in corrispondenza del muscolo da analizzare. Questi elettrodi devono, a differenza di quelli di profondità, essere equipaggiati con un amplificatore in modo da migliorare il segnale.

L'impiego di questi elettrodi offre i seguenti vantaggi: semplicità e rapidità di applicazione, nessun invasività, facile riposizionamento, non provoca fastidio al paziente.

Gli svantaggi, invece, sono: maggior volume di lavoro, minor selettività, coinvolgimento dei soli muscoli superficiali, posizionamento degli elettrodi non standardizzato, richiedono pulizia pelle, non adatti a muscoli piccoli o profondi, accuratezza delle informazioni dipende dai movimenti del soggetto.

Gli elettrodi superficiali più utilizzati sono quelli ad argento/argento cloruro dotati di gel, facili da applicare e monouso. Il gel utilizzato negli elettrodi commerciali può essere di due tipi: umido o adesivo. Il primo ha una maggiore conduzione e valori di impedenza tra elettrodo e pelle più bassi, mentre il secondo permette il riposizionamento in caso di errori.

2.2.3 Comando mediante segnale EMG

Il controllo della protesi basato sull'EMG Pattern Recognition si basa sull'assunzione che i pattern EMG debbano contenere informazioni consistenti sul movimento intenzionale dell'arto amputato.

Tramite una tecnica di classificazione del pattern, si può identificare il movimento che il soggetto è intenzionato a svolgere e viene, quindi, mandato un comando al controllore della protesi in modo che faccia eseguire il movimento.

In questo modo, il soggetto deve contrarre il muscolo corrispondente al grado di libertà che vuole controllare. Ne risulta così un controllo più intuitivo da parte del soggetto e una selezione rapida di ogni funzione. Ciò permette al soggetto di far muovere più facilmente la protesi con molteplici gradi di libertà, come richiesto nelle amputazioni di arto di alto livello.

I principali passi, per effettuare questo tipo di approccio, sono i seguenti:

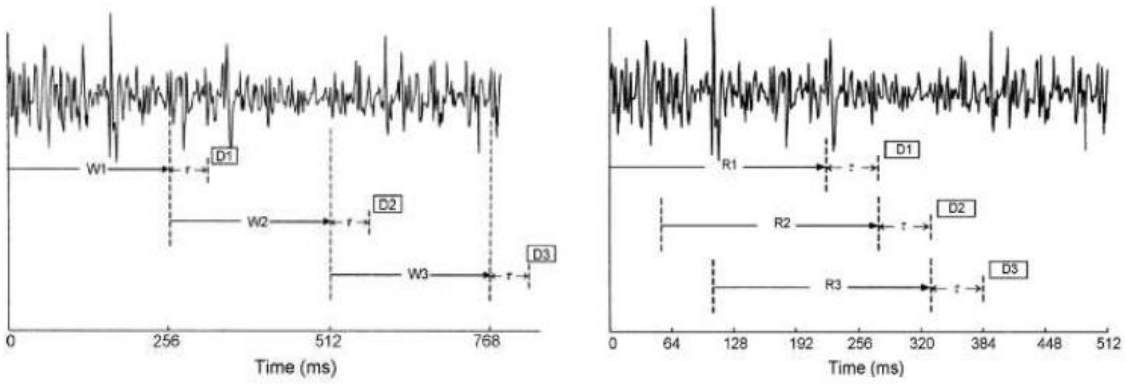
1. Acquisizione EMG multicanale
2. Segmentazione dei dati EMG
3. *Feature Rappresentation*
4. Classificazione delle caratteristiche
5. Controllo protesi

Inizialmente, vengono acquisiti i segnali mioelettrici facendo in modo di catturare informazioni sufficienti per la classificazione del movimento. In questa fase, bisogna porre una maggiore attenzione al numero di canali mioelettrici da utilizzare, questi dipenderanno dal numero di classi di movimento richieste e dal numero di muscoli che si andranno ad analizzare. Maggiore sarà il numero degli elettrodi maggiore sarà la complessità, il peso e il costo della protesi. Nonostante ciò, si è potuto notare, da studi precedenti [12], che all'aumentare del numero dei canali, l'errore di classificazione diminuisce. Per migliorare la qualità del segnale, è necessario, una volta catturato dagli elettrodi di superficie, eseguire un filtraggio attraverso un passa banda.

La frequenza di campionamento adottata in vari studi di controllo di protesi basato sull'EMG Pattern Recognition è intorno ai 1000 Hz [13][14].

Nella seconda fase, i vari segnali vengono segmentati in una serie di finestre di analisi che possono essere temporalmente sovrapposte o meno. Esistono due principali metodi per la finestratura dei dati: finestratura adiacente o sovrapposta. Nella prima, vengono creati dei segmenti disgiunti adiacenti di lunghezza fissa. In questo caso, per soddisfare il requisito di real-time è necessario che l'intervallo temporale dato dalla lunghezza della finestra e dal tempo di processing dei comandi di controllo siano uguali o inferiori a 300 ms. Nella finestratura sovrapposta invece, viene usato il tempo in cui il processore è inattivo in modo da massimizzare l'utilizzo delle risorse computazionali.

Farina e Merletti [15], hanno dimostrato che non sempre le finestre sovrapposte provocano un miglioramento nell'accuratezza della classificazione, ma determinano soltanto un aumento del tempo di processing. La lunghezza della finestra deve essere scelta in modo da rispettare il requisito di real-time, secondo cui il tempo di risposta della protesi non deve superare 300 ms. Oltre i 300 ms, infatti, il soggetto percepirebbe il ritardo [12].



(a) (b)

Figura 6: Finestre adiacenti (a) e sovrapposte (b)

La *feature rappresentation* ha un ruolo importante per l'accuratezza della classificazione del segnale perché permette di filtrare le informazioni nei dati EMG e di ridurre le informazioni ridondanti. Nella prima parte è effettuata la *feature extraction* che permette di estrarre informazioni utili nascoste all'interno del

segnale sEMG e di rimuovere le componenti indesiderate del segnale [16]. La scelta della *feature* è molto importante dato che alcune sono più robuste di altre e permettono di evitare complessi metodi di preprocessing. Nella seconda parte, invece, viene effettuata la *feature reduction* che permette di ridurre la complessità del problema attraverso il ridimensionamento delle *feature*. In questo modo, viene ridotta la quantità di memoria richiesta e viene aumentata la velocità del classificatore.

Diversi studi hanno cercato di valutare quali tipi di *feature* siano più adatti per la classificazione del segnale sEMG. Dai risultati di alcuni di questi [17], si è ricavato che la miglior performance si ottiene scegliendo la combinazione WPT/PCA/LDA.

	Elementi confrontati	**Combinazione per miglior performance (minor errore assoluto)**
Feature set	-TD -STFT -WT -WPT	WPT
Metodo di riduzione	-CS -PCA	PCA
Classificatore	-LDA -MLP	LDA

Tabella 2: Valutazioni delle performance nelle varie fasi

Nel caso in cui si voglia approfondire il funzionamento di queste tecniche, un appendice in fondo alla tesi sarà dedicato a questo.

Una volta che i dati sono stati elaborati, questi vengono inviati al classificatore, che si occuperà di predire la classe di movimento che permetterà, infine, di produrre lo spostamento della protesi desiderata. Un classificatore efficiente deve riuscire a classificare i pattern in poco tempo per rispondere ai vincoli specifici dell'applicazione real-time del dispositivo protesico.

I quattro migliori approcci utilizzati in letteratura nella Pattern Recognition sono: il template matching, la statistical classification, il syntactic o structural matching, e le neural networks.

Una breve descrizione riassuntiva di questi metodi è riportata nella Tabella 3:

Approccio	Rappresentazione	Funzione di riconoscimento	Criterio
Template matching	Campioni,Pixel, Curve	Correlazione, misura di distanza	Errore di classificazione
Statistical	Features	Funzioni di discriminazione	Errore di classificazione
Syntactic /Structural	Primitives	Regole, grammatica	Errore di accettazione
Neural Networks	Campioni,Pixel,Features	Funzione di rete	Errore quadratico medio

Tabella 3: Metodi di classificazione usati nella Pattern Recognition [19]

Per l'applicazione di nostro interesse è stato preso in considerazione l'approccio statistico, mentre gli altri sono stati esclusi.

2.2.4 Problemi nell'uso degli EMG

Nonostante questo tipo di approccio sia molto innovativo e da alcuni punti di vista consenta all'utente una maggiore interazione con la protesi, questo metodo presenta degli svantaggi che non devono essere trascurati.

Per prima cosa, bisogna ricordare che questo metodo non considera la possibilità che la protesi o il sistema invii segnali di feedback all'utente. Questo può essere molto importante, nel caso in cui, l'utente voglia interagire con la protesi, in modo da sentirla propria. Un'altra cosa da ricordare è il tempo di training che il soggetto deve impiegare per imparare a comandare correttamente la protesi. Questo fatto normalmente scoraggia chi la deve utilizzare poiché questi movimenti sono spesso robotici o comunque non naturali.

Infine, i movimenti possibili da effettuare attraverso la protesi sono ridotti. Questo permette di svolgere pochi tipi di attività, che comunque vengono effettuate, anche in questo caso, in modo innaturale.

Nonostante questi svantaggi, l'approccio si pone come una soluzione innovativa e dall'alto potenziale per la sua applicabilità nel campo delle protesi di arto superiore.

Capitolo 3

Modello di protesi di arto superiore

Si è visto nel capitolo precedente come i vari tipi di protesi possano essere classificate in base alla tecnologia usata, notando che quelle mioelettriche possono essere quelle di maggior interesse per i nostri studi e ricerche nel campo robotico, data la sua complessità di progettazione e le sue funzionalità.

Nel presente capitolo verrà presentata la protesi fisica utilizzata come riferimento in questo lavoro per la simulazione e le sue relative caratteristiche.

3.1 Protesi reale di arto superiore

La protesi analizzata in questo studio di tesi è stata progettata dal Dipartimento di Ingegneria Meccanica del Politecnico di Milano [32]. Questa possiede 3 gradi di libertà (due sulla spalla e uno sul gomito) a cui è collegata una mano commerciale dotata anch'essa di 3 gradi di libertà (due sul polso e uno sulla chiusura e apertura delle falangi).

La protesi può essere divisa in tre parti fondamentali: il meccanismo della spalla, il sistema gomito e quella della mano.

3.1.1 Meccanismo spalla

Il sistema spalla è costituito da un perno per il fissaggio al corpo del paziente, un blocco contenente i motori e i riduttori e infine un distanziale che ha la funzionalità di riprodurre la parte del braccio.

Il prototipo della protesi implementa un sistema differenziale composto da tre ruote coniche realizzate in materiale plastico.

Il meccanismo è formato da una coppia di motori brushless in corrente continua collegati rispettivamente a due riduttori.

Vengono utilizzati due motori Maxor Motor EC 45 che vengono alimentati ad una tensione pari a 12 V ed erogano 30 W di potenza. I riduttori utilizzati sono Harmonic Drive, HDC-010-Metric della Harmonic Drive Technologies, con rapporto di riduzione 1:80.

Per la trasmissione del moto al sistema differenziale è stato scelto un sistema a cinghia e pulegge dentate; di conseguenza anche i gruppi motoriduttori sono stati dotati di pulegge dentate. In questo modo l'ingombro viene gestito in modo migliore e il rapporto di riduzione 15:44 viene aumentato ulteriormente.

Il telaio è realizzato in materiale plastico Delrin, dotato di alta resistenza meccanica e durezza, comprendente di fori per l'alloggiamento dei gruppi motoriduttore.

Il sistema differenziale è costituito da due telai e da tre ruote coniche dentate (Figura 7).

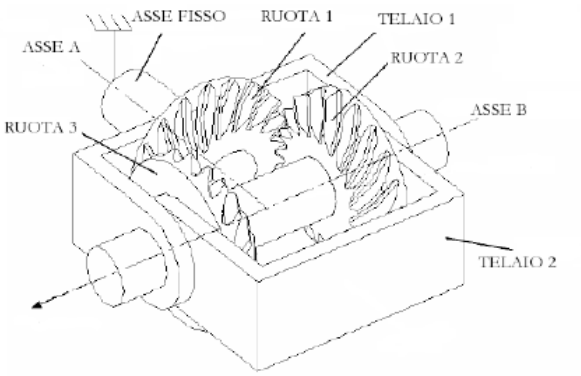


Figura 7: Sistema differenziale

La prima ruota conica (ASSE A) è fissata alla spalla, mentre le altre due sono libere di ruotare rispetto all'asse B. Il primo telaio (TELAIO 1) è libero di ruotare intorno all'asse A in modo da seguire l'asse B, mentre il secondo telaio (TELAIO 2) è libero e solidale al telaio del braccio. Il sistema differenziale consiste in movimenti intorno a questi due assi attraverso la combinazione delle rotazioni delle ruote dentate.

I movimenti possibili dipenderanno perciò dalle velocità angolari. Nel caso in cui si abbiano velocità angolari uguali in modulo, ma di segno opposto, si avrà una rotazione intorno all'asse A, producendo un movimento di flessione-estensione della spalla; velocità angolari uguali in modulo e in segno determineranno invece una rotazione intorno all'asse B, generando un movimento di abduzione-adduzione della spalla; infine, combinando velocità angolari diverse sia in modulo che in segno, si genera il movimento composto della spalla.

La protesi, inoltre, è dotata di un giunto a frizione realizzato con due dischi di materiale autolubrificante premuti mediante il serraggio di bulloni. La funzionalità di questo giunto è quella di avere un sistema di sicurezza contro eventuali urti, in modo da evitare che l'avambraccio effettui movimenti di rotazione assiale che escludano la sollecitazione della spalla. Attraverso questo giunto è stato possibile aggiungere un altro grado di libertà passivo che permette di migliorare il volume di lavoro della protesi.

3.1.2 Meccanismo gomito

Il sistema gomito è formato da un blocco motore-riduttore che collega il giunto a frizione, sopra descritto, all'avambraccio.

Il blocco motore-riduttore è simile a quello usato per la spalla, con la sola differenza che la puleggia è stata sostituita con un perno sagomato che si andrà ad accoppiare al componente dell'avambraccio.

L'avambraccio, che ha solo funzionalità estetiche e di sostegno della mano, è stato costruito in legno e presenta dei fori all'estremità in modo da permettere il posizionamento della mano.

3.1.3 Meccanismo mano

La protesi utilizzata per questo scopo è una mano commerciale fornita dall'azienda OttoBock [34].

Questa possiede due gradi di libertà: uno di prono-supinazione del polso e il secondo di apertura-chiusura della mano con ampiezza massima di 10 cm. La mano è dotata di una guaina che permette di riprodurre l'aspetto estetico della mano.

Figura 8: Protesi fisica di riferimento con relativo meccanismo mano

3.2 Analisi Cinematica

In questo paragrafo verrà analizzata la cinematica diretta e inversa della protesi di arto superiore utilizzata per questo studio di tesi. Questa, per il suo corretto funzionamento, è stata implementata in modo diverso da quella ricavata dal Dipartimento di Meccanica dato che si sono dovuti considerare ulteriori gradi di libertà.

La cinematica studia il legame fra le variabili indipendenti dei giunti e le posizioni (e orientamenti) cartesiane raggiunte dal robot [35].

Essa può essere suddivisa in: cinematica diretta e inversa.

Nella cinematica diretta si pone il problema di determinare la posizione e orientamento dell'end effector dati i valori dei giunti. In questo caso, data una configurazione dei vari giunti dell'arto si vuole individuare il punto nello spazio cartesiano raggiunto all'estremità di questo.

Nella cinematica inversa, invece, data una posizione dell'end-effector si vuole ricavare i valori dei giunti. Questo è utile nel caso in cui si conosca il punto che deve raggiungere l'estremità dell'arto ma non si sappia configurare correttamente i valori dei giunti.

Entrambi i problemi sono stati ampiamente discussi e analizzati in letteratura con il risultato che al giorno d'oggi esistono diverse soluzioni valide per la risoluzione di entrambi le questioni.

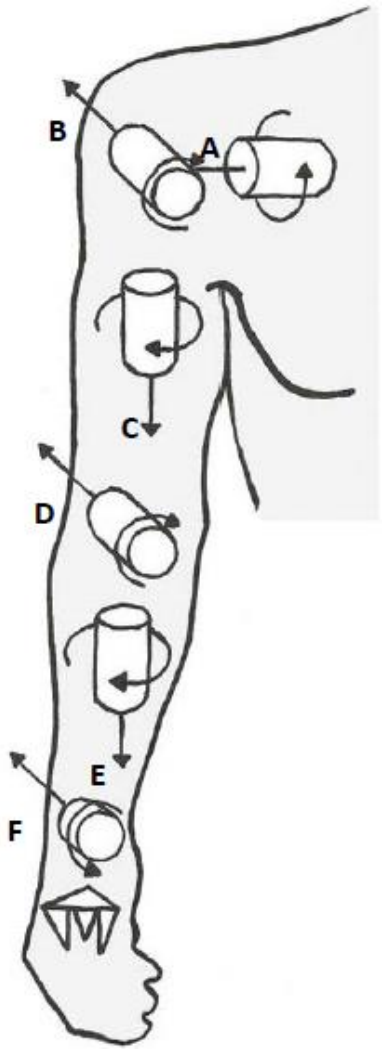
Per come è stato definito, il problema cinematico diretto ammette sempre una e una sola soluzione che viene ottenuta mediante il calcolo della matrice T del braccio. Nella cinematica inversa, invece, esistono diversi metodi che permettono di trovare la soluzione, anche se questa non è sempre garantita. E' per questo motivo che per lo studio della cinematica inversa bisogna effettuare un analisi più accurata e valutare caso per caso.

3.2.1 Cinematica diretta

Per ricavare la soluzione nel problema della cinematica diretta esiste una procedura generale e sempre valida per i manipolatori, questa si basa sulla rappresentazione di Denavit-Hartenberg.

Questa fornisce un metodo generale per la rappresentazione delle catene cinematiche aperte, e il fissaggio dei sistemi di riferimento sui giunti al fine di poterne determinare i parametri caratteristici.

Una volta messo il manipolatore in posizione zero possiamo passare ad assegnare i sistemi di riferimento ai giunti e link della protesi, secondo la convenzione D-H:



- Intorno all'asse A avviene il movimento di flesso-estensione della spalla, definita dalla grandezza angolare ϑ_1

- Intorno all'asse B avviene il movimento di abduzione della spalla, che diventa un movimento di flesso-estensione orizzontale della stessa nel caso l'asse A sia posto perpendicolarmente all'asse del corpo, definita dalla grandezza angolare ϑ_2

- Intorno all'asse C avviene il movimento di rotazione assiale del braccio (ottenuta tramite il giunto passivo), definita dalla grandezza angolare ϑ_3

- Intorno all'asse D avviene il movimento di flesso-estensione dell'avambraccio, definita dalla grandezza angolare ϑ_4

Figura 9: Schema cinematico della protesi

Per la mano invece abbiamo:

- Intorno all'asse E avviene il movimento di prono-supinazione del polso, definita dalla grandezza angolare ϑ_5
- Intorno all'asse F avviene il movimento di flesso-estensione del polso, definita dalla grandezza angolare ϑ_6

Una volta definiti i sistemi di riferimento non rimane che assegnare le costanti che rappresentano le lunghezze dei vari link:

- *a* la lunghezza della spalla
- *b* la lunghezza del braccio
- *c* la lunghezza dell'avambraccio
- *d* la lunghezza della mano (intesa come la distanza dall'attaccatura del polso al braccio fino al centro della mano)

Una volta definiti i parametri non rimane che calcolare le matrici di rototraslazione A_i che permettono di passare dal sistema di riferimento del giunto i-esimo a quello del giunto i+1-esimo. Queste matrici vengono costruite come somma tra la matrice di traslazione di quantità uguale alla lunghezza del link e la matrice di rotazione della quantità angolare di ciascun giunto:

$$DH_x = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & \cos\vartheta & -\sin\vartheta & d_y \\ 0 & \sin\vartheta & \cos\vartheta & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$DH_y = \begin{bmatrix} \cos\vartheta & 0 & \sin\vartheta & d_x \\ 0 & 1 & 0 & d_y \\ -\sin\vartheta & 0 & \cos\vartheta & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$DH_z = \begin{bmatrix} \cos\vartheta & -\sin\vartheta & 0 & d_x \\ \sin\vartheta & \cos\vartheta & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

dove ϑ rappresenta la rotazione rispetto l'asse considerato, mentre d_x, d_y, d_z le rispettive traslazioni lungo gli assi.

In questo modo si otterranno un numero di matrici uguali al numero di sistemi di riferimenti presenti nel sistema.

E' importante ricordare che nel nostro caso avremo bisogno di aggiungere un ulteriore matrice di rototraslazione per passare dal sistema di riferimento fisso (posizione del sensore Kinect) al sistema di riferimento del primo giunto.

Una volta calcolate le matrici A_i bisogna moltiplicarle tra loro in modo da trovare la matrice T del braccio. Questa matrice ci permetterà di passare dal sistema di riferimento fisso del Kinect al sistema finale della mano.

In questo modo si è riusciti a trovare un metodo valido per passare dai valori dei giunti alla posizione finale dell'end-effector nello spazio cartesiano.

3.2.2 Cinematica inversa

Come già visto, il compito della cinematica inversa è quello di determinare i valori dei giunti data la posizione raggiunta dall'end-effector. Purtroppo, diversamente che per la cinematica diretta, nel caso del problema cinematico inverso generalmente non è garantita né l'esistenza né l'unicità della soluzione.

In generale, se il punto da raggiungere è nello spazio di lavoro e il robot a 6 gradi di libertà allora esiste una soluzione. Questa proprietà vale sicuramente perché con 6 gradi di libertà è possibile raggiungere qualsiasi posizione con qualsiasi orientamento.

Tra i metodi di analisi più utilizzati per la risoluzione della cinematica inversa troviamo:

- Metodi geometrici: che consistono nel ridurre il problema in sottoproblemi a due dimensioni, in modo da semplificare la risoluzione.
- Metodi algebrici: metodi di soluzione basati su espressioni aritmetiche o sulla soluzione di polinomi di grado 4 o inferiore che non richiedono calcoli iterativi.

- Metodi iterativi: soluzioni numeriche approssimate.

I metodi geometrici sono facilmente applicabili in problemi a pochi gradi di libertà o in quei problemi che possono riportarsi a casi più semplici.

I metodi algebrici permettono di trovare una soluzione esatta, ma non sempre è possibile effettuarne il calcolo.

I metodi iterativi non consentono di trovare una soluzione in un tempo noto a priori e inoltre non garantiscono di trovare tutte le soluzioni della cinematica inversa. Questi sono raramente utilizzati in robot industriali ma utili da adoperare quando gli altri metodi falliscono.

Il metodo scelto per questo studio è quello geometrico, dato che la soluzione del problema può essere suddiviso in due sottoproblemi:

1. Calcolo della cinematica inversa relativa al solo braccio (tre gradi di libertà),

2. Calcolo della cinematica inversa relativa alla sola mano (due gradi di libertà).

Una volta risolti, possiamo unirli per ricavare la soluzione della cinematica inversa del sistema a 5 gradi di libertà.

3.2.2.1 Cinematica inversa del braccio

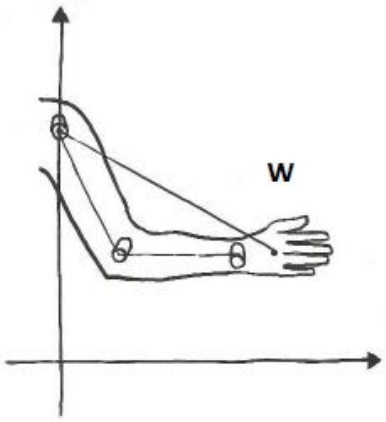


Figura 10: Braccio vista laterale

Per prima cosa viene calcolata la distanza tra il punto da raggiugere P(*x*, *y*, *z*) e l'origine della terna di riferimento associata al secondo giunto (punto di connessione tra la spalla e il braccio)

$$w = \sqrt{(x - d_1)^2 + y^2 + (z - d_0)^2}$$

Applicando il teorema di Carnot al triangolo formato dal vettore spalla-mano (distanza *w*), braccio e avambraccio, possiamo calcolare l'angolo ϑ_k, inteso come l'angolo tra il vettore w e il segmento formato dal braccio:

$$\cos\vartheta_k = -\frac{{d_5}^2 - w^2 - {d_3}^2}{2 \cdot w \cdot d_3}$$

$$\sin\vartheta_k = \sqrt{1 - \cos{\vartheta_k}^2}$$

$$\vartheta_k = Atan2(\sin\vartheta_k, \cos\vartheta_k)$$

L'utilizzo della funzione trigonometrica Atan2(*y*, *x*) è stata necessaria poiché restituisce sempre un angolo nell'intervallo (-π,+ π), al contrario della funzione atan(*y*, *x*) che ritorna valori non validi negli estremi (-π/2,+ π/2).

Dato che queste funzioni restituiscono angoli espressi in radianti diviene utile per una semplice interpretazione convertirli in gradi.

Per ricavare l'angolo ϑ_3:

$$\cos\vartheta_3 = \frac{w \cdot \cos\vartheta_k - d_3}{d_5}$$

$$\sin\vartheta_3 = \sqrt{1 - \cos\vartheta_3{}^2}$$

$$\vartheta_3 = Atan2(\sin\vartheta_3, \cos\vartheta_3)$$

A questo punto è utile verificare se il punto sia raggiungibile, cioè se la distanza w è minore della lunghezza del braccio completamente esteso (d_3+d_5).

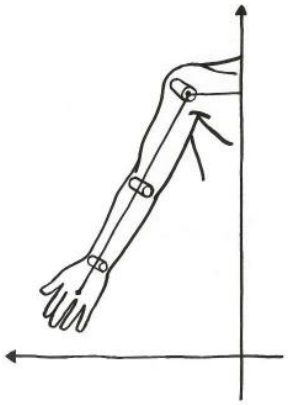


Figura 11: Braccio vista frontale

Nel caso in cui l'oggetto non sia raggiungibile (w > d_3+d_5) possiamo calcolare ϑ_2 nel modo seguente:

$$\sin\vartheta_2 = x - d_1$$

$$\cos\vartheta_2 = \sqrt{(z - d_0)^2 + y^2}$$

$$\vartheta_2 = Atan2(\sin\vartheta_2, \cos\vartheta_2)$$

Mentre nel caso contrario (w < d_3+d_5) :

$$\sin\vartheta_2 = \frac{x - d_1}{d_3 + d_5 \cdot \cos\vartheta_3}$$

$$\cos\vartheta_2 = \sqrt{1 - \sin\vartheta_2{}^2}$$

$$\vartheta_2 = Atan2(\sin\vartheta_2, \cos\vartheta_2)$$

Rimane solo da calcolare l'angolo ϑ_1, ma prima bisogna calcolare ϑ_w, inteso come l'angolo tra l'asse z e il vettore w:

$$\cos \vartheta_w = y$$

$$\sin \vartheta_w = z - d_0$$

$$\vartheta_w = Atan2(\sin \vartheta_w , \cos \vartheta_w) + \frac{\pi}{2}$$

Incrementando ϑ_2 si può notare come questo produca una variazione sull'asse delle z del punto P, come se fosse stato incrementato l'angolo ϑ_1, per modellizzare ciò è utile introdurre l'angolo ϑ_0:

$$\sin \vartheta_0 = \frac{(1 - \cos \vartheta_2) \cdot \cos \vartheta_w}{w}$$

$$\cos \vartheta_0 = \sqrt{1 - \sin \vartheta_0{}^2}$$

$$\vartheta_0 = Atan2(\sin \vartheta_0 , \cos \vartheta_0)$$

Nel caso in cui il punto P si trovi sopra il sensore Kinect ($z - d_0 > 0$), ϑ_0 viene posto uguale al suo reciproco.

Una volta effettuati i vari calcoli si può passare al calcolo di ϑ_1:

$$\vartheta_1 = \vartheta_w - \vartheta_k - \vartheta_0$$

Calcolato ϑ_1, si hanno tutte le informazioni necessarie delle variabili di giunto relative al solo braccio.

3.2.2.2 Cinematica inversa della mano

Per prima cosa è importante capire il tipo di approccio che si vuole effettuare per afferrare l'oggetto. Questo può essere di tre tipi: un approccio frontale, uno laterale e uno dall'alto.

Il tipo di approccio è dato dalle informazioni acquisite tramite i segnali EMG.

Una volta conosciuto il tipo di approccio e il punto P, che identifica il baricentro dell'oggetto da afferrare, posso calcolare il punto I.

Questo nuovo punto rappresenta idealmente il punto in cui dovrà arrivare l'estremità dell'avambraccio per far si che la mano possa afferrare l'oggetto con il tipo di presa deciso in precedenza, tenendo conto dei vincoli fisici di rotazione del polso:

1. Approccio laterale:
    - $\vartheta_4 = 0$
    - $\vartheta_5 = \frac{\pi}{4}$
2. Approccio superiore:
    - $\vartheta_4 = \frac{\pi}{4}$
    - $\vartheta_5 = \frac{\pi}{2}$
3. Approccio frontale:
    - $\vartheta_4 = 0$
    - $\vartheta_5 = 0$

Il punto I (*x*, *y*, *z*) è calcolato come:

$$\begin{cases} x_i = x_p + d_7 \cdot \sin\vartheta_5 \cdot \cos\vartheta_4 \\ y_i = y_p + d_7 \cdot \cos\vartheta_4 \\ z_i = z_p + d_7 \cdot \sin\vartheta_5 \cdot \sin\vartheta_4 \end{cases}$$

Usando le formule della cinematica inversa illustrate prima troviamo le variabili di giunto relative al solo braccio affinché l'estremità dell'avambraccio raggiunga il punto I. Ora bisogna calcolare con che orientamento la mano arriva in questo punto, per farlo applichiamo le formule di cinematica diretta illustrate precedentemente impostando le variabili di giunto ancora ignote (ϑ_4 e ϑ_5) a 0, troviamo il punto che identifica l'estremità del polso W(*w*, *y*, *z*) e il punto che identifica il centro della mano H(*x*, *y*, *z*) nello spazio cartesiano rispetto alla terna di riferimento fissa.

A questo punto è possibile calcolare la distanza tra questi due punti e il punto rappresentante l'oggetto da afferrare P (*x*, *y*, *z*):

$$w = \sqrt{(x_p - x_w)^2 + (y_p - y_w)^2 + (z_p - z_w)^2}$$

$$h = \sqrt{(x_p - x_h)^2 + (y_p - y_h)^2 + (z_p - z_h)^2}$$

Ora è possibile calcolare l'angolo ϑ_5 usando il teorema di Carnot applicato al triangolo formato dalla lunghezza polso-mano, e i vettori trovati *w* e *h*:

$$\sin\vartheta_5 = -\frac{w^2 - {d_7}^2 - h^2}{2 \cdot d_7 \cdot h}$$

$$\cos\vartheta_5 = \sqrt{1 - \sin{\vartheta_5}^2}$$

$$\vartheta_5 = Atan2(\sin\vartheta_5, \cos\vartheta_5)$$

Per il calcolo dell'ultima variabile di giunto rimasta ϑ_4 bisogna ripetere la procedura appena vista: bisogna ricalcolare i punti W e H utilizzando nella formula della cinematica diretta l'angolo $\vartheta_5$sopra trovato, calcolare i vettori w e h e infine calcolare l'angolo ϑ_4 con la stessa formula usata per l'angolo ϑ_5.

Nel caso di approccio dall'alto ϑ_4 dovrà essere posto uguale al suo reciproco.

Ultimato questo ultimo calcolo abbiamo tutte le informazioni necessarie per poter posizionare il centro della mano della nostra protesi in corrispondenza del punto indicante l'oggetto d'afferrare.

Capitolo 4

Dispositivi e metodi

In questo capitolo, verranno presentati i dispositivi e i metodi impiegati per la creazione del sistema di controllo tramite EMG per la protesi di arto superiore. Inizialmente, verrà introdotta l'intera struttura del sistema, per poi passare a descrivere i vari dispositivi che lo compongono e come questi vengano utilizzati.

4.1 Struttura generale

L'intero sistema è strutturato in tre parti principali: gli elettrodi, il sensore di profondità e la protesi di arto superiore.

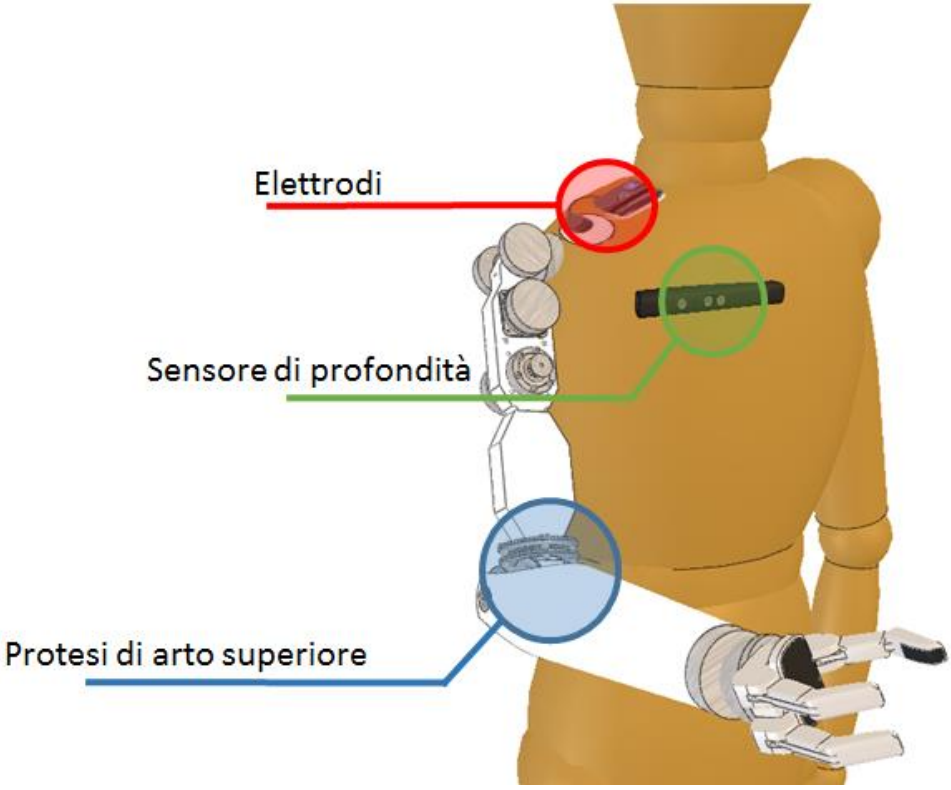


Figura 12: Struttura dell'intero sistema

Gli elettrodi forniscono indicazioni sul tipo di movimento che l'utente vuole effettuare, mentre il sensore di profondità permette di analizzare l'ambiente alla ricerca di eventuali oggetti/ostacoli.

La protesi di arto superiore utilizzata è la modellizzazione della protesi fisica sviluppata dal Dipartimento di Meccanica del Politecnico di Milano. Essa possiede, a differenza di quella reale, 2 gdl nella spalla (flesso/estensione e adduzione/abduzione), 1 gdl nel gomito (flesso/estensione) e due gdl nel polso (rotazione e flesso/estensione), per un totale di 5 gdl.

Nei seguenti paragrafi verranno introdotti i dispositivi e i metodi utilizzati per la gestione del segnale elettromiografico, per il sensore di visione e per la riproduzione della protesi fisica.

4.2 Analisi segnale elettromiografico

L'analisi dei segnali EMG è stato effettuato dal Dipartimento di Bioingegneria del Politecnico di Milano [38]. E' stato necessario l'utilizzo di un set up sperimentale per l'acquisizione dei segnali elettromiografici utilizzati per svolgere l'analisi off-line necessaria per l'individuazione dei pattern da dare in ingresso al classificatore. Le componenti principali del set up sperimentale sono:

- Elettromiografo BTS FREEEMG 300®;
- Elettrodi per la rilevazione del segnale sEMG;
- Sistema optoelettronico SMART di analisi del movimento con 8 telecamere.

Il segnale elettromiografico è stato rilevato tramite il BTS FREEEMG 300® (Figura 13). Il modello utilizzato è basato su tecnologie wireless, ed impiega, per il prelievo e la trasmissione del segnale EMG, sonde miniaturizzate che amplificano il segnale, lo convertono in formato digitale e lo trasmettono ad un'unità ricevente che può essere indossata dal paziente, posata sul tavolo o tenuta dal medico.

L'unità ricevente può inviare i dati ad una Workstation, su cui è installato il software sviluppato per l'acquisizione, la visualizzazione e una prima elaborazione dei segnali elettromiografici.

Figura 13: BTS FREEEMG 300®

Per il prelevamento del segnale EMG sono state utilizzate sonde wireless miniaturizzate ad elettrodi attivi dal peso inferiore ai 9 grammi. Per la loro applicazione è stato sufficiente agganciarle, attraverso delle clip, direttamente agli elettrodi.

Gli elettrodi utilizzati nella fase di sperimentale sono degli elettrodi standard per elettrostimolazione neonatale, ECG per neonatologia, EMG e biofeedback, circolari con diametro di 26mm, in FOAM, a clip in Ag/AgCl, pregellati, monouso, prodotti da BIELSAN®. Questi sono stati scelti di dimensioni abbastanza contenute, per fare in modo da poter analizzare il segnale elettromiografico nel modo più accurato possibile.

La strumentazione utilizzata viene gestita in remoto da un PC tramite i software della famiglia SMART dedicati all'analisi del movimento, con i quali è stato possibile acquisire e processare i segnali elettromiografici in maniera integrata e sincrona ai segnali del sistema optoelettronico connesso

Il sistema optoelettronico SMART è la soluzione tecnologica più diffusa per l'analisi del movimento umano. Questo utilizza marcatori ottici, posizionati sul corpo del soggetto in particolari punti di repere anatomici, e di un numero variabili di telecamere operanti nell'infrarosso.

I marcatori ottici utilizzati sono di forma sferica e realizzati in materiale catarifrangente in modo che la luce infrarossa emessa dai quattro anelli di LED disposti su corone circolari coassiali con la telecamera venga riflessa e possa quindi essere rilevata dal sistema di lenti e sensori matriciali indirizzabili CCD.

Figura 14: Telecamera del sistema SMART-e (BTS)

Il sistema consta di 8 telecamere disposte in modo da coprire l'intero volume di lavoro.

4.2.1 Metodi

Fino ad oggi, sono state svolte molte ricerche relativamente allo sviluppo di sistemi di controllo per protesi attive di mano, polso e gomito, basati sulla *Pattern Recognition* del segnale elettromiografico. Poca attenzione è, invece, stata dedicata al controllo di protesi per disarticolazione della spalla. Questo è dovuto al fatto che la maggior parte delle amputazioni avviene a livello trans-radiale o trans-omerale.

Il lavoro di tesi svolto da Rivela e Scannella ha preso in considerazione la disarticolazione della spalla [38]. I tipi di movimenti analizzati nella fase di sperimentazione sono stati i seguenti: la flesso/estensione nel piano sagittale, l'ab/adduzione nel piano frontale e una loro combinazione nel piano inclinato di 45° rispetto al piano frontale.

Per l'acquisizione del segnale elettromiografico si è scelta una frequenza di campionamento di 1000 Hz, mentre per l'acquisizione delle informazioni cinematiche è stato impiegato il software SMART, scegliendo una frequenza di campionamento di 120 Hz. L'elaborazione dei dati, invece, è stato svolto attraverso il software MATLAB 2013a.

Dopo diverse analisi, si è potuto osservare come la scelta della segmentazione incida sulla bontà della classificazione. In particolare, a prescindere dal metodo di elaborazione utilizzato, la segmentazione migliore, prendendo in considerazione diverse lunghezze della finestra e diversi valori di incremento tra finestre è risultata essere la combinazione data dalla finestra lunga 500 ms con l'incremento pari a 62 ms.

Questa tipologia di segmentazione permette di avere un flusso decisionale ogni 62 ms, tempo massimo in cui il processore montato sulla protesi deve elaborare i segnali elettromiografici acquisiti da ciascuno degli otto canali e fornire in uscita i comandi alla protesi.

Come metodo di *feature reduction*, è stata adottata la tecnica della PCA, in quanto è adatta ad applicazioni real-time ed è considerata la migliore tra tutte le altre forme di riduzione per la maggior parte dei feature set estratti [12].

In fase di classificazione, si è utilizzato il classificatore basato sulla LDA, che ha il vantaggio di essere un metodo di classificazione semplice da implementare, veloce da allenare, computazionalmente efficiente per le operazioni real-time e non richiede alcun parametro da settare [18].

Gli sperimenti effettuati da Rivela e Scannella hanno messo in evidenza che è possibile ottenere un'accuratezza della classificazione pari al 92.56% nel discriminare un movimento all'interno di un set di nove classi di movimenti separati, pari al 97.39% se il set è costituito da cinque classi e pari al 100% se la scelta è limitata a quattro classi [38].

Questi promettenti risultati hanno permesso di presumere che il sistema di acquisizione e classificazione sEMG possa essere implementato come un sistema stand-alone (usabile in svariate applicazioni) da interfacciare con il controllore della protesi.

4.3 Sensore di Visione

In questo paragrafo verranno introdotti il sensore Microsoft Kinect, le varie librerie utilizzate per la gestione delle informazioni visive e il tipo di classificatore usato per questo lavoro.

4.3.1 Microsoft Kinect

Microsoft Kinect, inizialmente conosciuto con il nome Project Natal, è un accessorio originariamente pensato per Xbox 360, sensibile al movimento del corpo umano. Questo a differenza di altri accessori di gioco, consente al giocatore il controllo del sistema senza la necessità di indossare o impugnare alcunché. A febbraio 2012, Microsoft ha rilasciato il software necessario per potere utilizzare il Kinect anche nei computer con sistema operativo Microsoft Windows.

La tecnologia Kinect sta diventando una risorsa molto importante grazie al basso costo, all'affidabilità e alla velocità di misurazione e pongono il dispositivo, per la misura del 3D, tra i più importanti in ambito robotico laddove si ha bisogno di riconoscere a real-time ambienti e oggetti [21].

Il sensore si presenta all'utente come una barra orizzontale connessa a una base in grado di inclinarsi automaticamente e progettata per essere posta sopra o sotto lo schermo utilizzato dall'Xbox. Kinect è dotato di telecamera RGB, doppio sensore di profondità a raggi infrarossi composto da uno scanner laser a infrarossi e da una telecamera sensibile alla stessa banda.

Figura 15: Descrizione del sensore Microsoft Kinect

La telecamera RGB ha una risoluzione con cavo VGA di 640 × 480 pixel con precisione 8 bit e opera alla frequenza di 30 Hz, mentre quella ad infrarossi usa cavo VGA con una matrice di 640 × 480 pixel con precisione ad 11 bit. Kinect dispone anche di un array di microfoni utilizzato dal sistema per la calibrazione dell'ambiente in cui ci si trova, mediante l'analisi della riflessione del suono sulle pareti e sull'arredamento. In tal modo, il rumore di fondo e i suoni del gioco vengono eliminati ed è possibile riconoscere correttamente i comandi vocali. L'array è composto da 4 microfoni ognuno dei quali processa audio a 16 bit con frequenza di campionamento di 16kHz.

La barra del Kinect è motorizzata lungo l'asse verticale e segue i movimenti dei giocatori, orientandosi nella posizione migliore per il riconoscimento dei movimenti ed è capace di inclinare il sensore su o giù di 27 gradi ma, poiché la motorizzazione per il meccanismo di inclinazione richiede più potenza di quella che supporta una porta USB, viene utilizzato dalla Microsoft un connettore proprietario combinando un cavo comunicazione USB con uno standard per la presa a muro di corrente elettrica.

Il sensore di profondità consta di un proiettore a laser infrarosso combinato con un sensore CMOS monocromatico, che cattura dati video in 3D sotto ogni condizione di luce ambientale.

Per raccogliere i dati in maniera ottimale, gli oggetti d'interesse devono essere posti a una distanza dalla telecamera compresa tra 1.2m e 3.5m; tuttavia il dispositivo riesce a rilevare con discreta precisione oggetti anche a distanze inferiori, fino a 60cm.

Di fatto, la periferica permette all'utente di interagire con la console senza l'uso di alcun controller da impugnare, ma solo attraverso i movimenti del corpo, i comandi vocali o attraverso gli oggetti presenti nell'ambiente.

Uno dei sogni, che negli anni ha portato molte sfide a ricercatori ed ingegneri, è quello di costruire un computer in grado di riconoscere e comprendere gli scenari come l'uomo. Questo diventa sempre più un obiettivo possibile grazie all'emergere di nuove tecnologie visive [22].

Questo tipo di dispositivo, non solo permette il suo utilizzo nel campo dei videogiochi ma può essere utile anche in altri come quelli robotici e medici, come nel caso specifico dell'aiuto alla riabilitazione [23].

In questo lavoro di tesi, il Kinect, sarà utile nel calcolo degli ostacoli [24], per fare in modo che la protesi esegua un movimento continuo senza collisioni.

In particolare, verranno citati e utilizzati i driver rilasciati a dicembre 2010 dall'azienda israeliana PrimeSense.

4.3.2 Point Cloud Library

Figura 16: Simbolo della Point Cloud Library

La Point Cloud Library (PCL) è un progetto open source a grande scala per immagini 2D/3D e per l'elaborazione di nuvole di punti [25].

Il fulcro fondamentale in PCL nasce dal concetto di nuvola di punti. Questa può essere definita come una struttura dati impiegata per rappresentare un insieme di punti multi-dimensionali ed è comunemente applicata per rappresentare i dati tridimensionali. In una nuvola di punti 3D, i punti rappresentano di solito le coordinate geometriche X, Y, e Z di una superficie campionata. Nuvole di punti possono essere acquisiti da sensori hardware come telecamere stereo, scanner 3D, o fotocamere tempo di volo, oppure generati da un programma per computer sinteticamente.

Il framework PCL contiene numerosi algoritmi che includono filtraggio, ricostruzioni di superfici, registrazione, stima di caratteristiche e modelli di raccordo e segmentazione. Questi algoritmi possono essere usati, ad esempio, per filtrare i valori anomali di dati rumorosi, per rilevare parti rilevanti in una scena, per riconoscere gli oggetti nel mondo in base al loro aspetto geometrico e per creare superfici a partire da nuvole di punti.

PCL è rilasciato sotto i termini della licenza BSD 3-clause ed è un software open source. E' gratuito per uso commerciale e di ricerca.

In questo lavoro di tesi, la libreria PCL sarà utilizzata per acquisire ed elaborare la mappa di profondità proveniente dal Microsoft Kinect. In essa, ogni punto è identificato da una terna (*u, d, v*) dove *u* e *v* sono le coordinate dei pixel identificati

dalla telecamera e *d* è la distanza rilevata tra i pixel in oggetto e il Kinect. Tale mappa viene poi convertita nel sistema di riferimento cartesiano in modo che possano essere manipolati dalla libreria PCL.

4.3.3 Bullet Physics Library

Bullet Physics è una libreria professionale open source per il rilevamento di collisioni 3D e per il calcolo delle dinamiche dei corpi rigidi e deformabili [26]. E' utilizzata particolarmente nei giochi, e negli effetti visivi nei film.

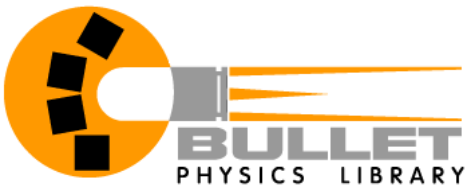


Figura 17: Logo Bullet Physics Library

La libreria è gratuita e per uso commerciale, ed è distribuita sotto licenza Zlib.

Le principali caratteristiche di questa sono le seguenti: gestione in modo discreto o continuo di rilevamento delle collisioni, gestione automatica delle dinamiche tra oggetti rigidi, oggetti deformabili come corde o tessuti e veicoli pre-implementati nel sistema, l'integrazione con Blender e la possibilità di importare\esportare modelli fisici tramite COLLADA, un formato file di interscambio informazioni tra applicazioni 3D.

Il pacchetto è fornito con alcuni esempi di codice sui quali l'utente deve basarsi per comprendere come utilizzare le funzioni che permettono l'interazione del codice C++ con la libreria. Ciò nonostante il suo utilizzo risulta sufficientemente intuitivo da giustificare l'assenza di una documentazione dettagliata.

La scelta di questa libreria è dovuta alla sua leggerezza e compattezza in termini di quantità di righe di codice necessarie al suo utilizzo, dato che non sono richieste particolari funzionalità avanzate, ma bensì funzioni basilari per il rilevamento di collisioni, utilizzabili in modo semplice e diretto.

4.3.4 Classificatore

Il classificatore utilizzato in questo lavoro è stato sviluppato presso il Dipartimento di Elettronica e Informazione del Politecnico di Milano [27].

Questo software, è utile al nostro lavoro perché permette di stimare l'orientamento di un oggetto posto di fronte al sensore visivo, in modo da poterne calcolare successivamente i punti di approccio.

Dato che una semplice telecamera non può fornire una rappresentazione tridimensionale della scena ripresa, ma si limita a rappresentare tramite un'immagine bidimensionale i dati raccolti, vi è necessità di un sensore visivo che possa aggiungere le coordinate di profondità all'immagine RGB captata dalla telecamera. Per questo motivo è stato impiegato il sensore Kinect di Microsoft.

Il lavoro che deve effettuare il classificatore può essere diviso in diverse fasi:

Nella prima, il sensore Kinect deve inviare al classificatore una mappa di profondità dello spazio percepito, in modo che questo possa creare un file contenente una nuvola di punti che rappresenta la scena di fronte al sensore.

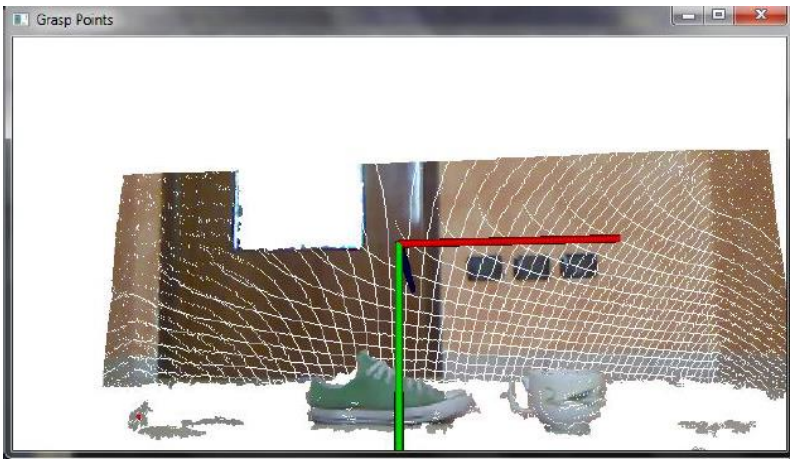


Figura 18: Esempio di creazione della nuvola di punti

Originariamente, per effettuare l'inizializzazione di tale operazione, il classificatore impiegava all'incirca 8 secondi, tempo inaccettabile per un sistema che si prefigge di essere real-time. Infatti il tempo trascorso tra il rilevamento del segnale EMG e l'effettivo movimento della protesi deve essere minimo.

Per ovviare a questo problema, l'inizializzazione è stata spostata all'esterno del classificatore ed eseguita da un thread, in modo da rimanere in attesa di essere attivato soltanto quando si ha la necessità di una nuova nuvola di punti della scena.

Una volta creata la nuvola di punti, è possibile passare alla fase di clustering. Questa consiste nell'estrapolazione dei punti appartenenti agli oggetti in primo piano. Prima della fase di clustering, è utile però eseguire un'operazione di downsampling, con dimensione di foglia un centimetro, per fare in modo di ridurre il dataset iniziale che potrebbe risultare particolarmente oneroso da analizzare. Una volta fatto questo è possibile processare la nuvola di punti per ricercare i cluster. Anche in questo caso è utile semplificare ulteriormente l'operazione eliminando i punti appartenenti al tavolo su cui poggiano gli oggetti.

Per questo viene utilizzato RANSAC (RANdom SAmple Consensus), un algoritmo iterativo in grado di stimare i parametri di un modello matematico da un insieme osservato di dati contenente outliers. Esso analizza l'immagine ed etichetta come inliers tutti i punti appartenenti al piano e come outliers i restanti punti.

Terminata l'analisi, tutti gli inliers vengono eliminati dalla nuvola semplificando significativamente la fase di clustering. Una volta effettuato questo è possibile dividere l'immagine nei vari cluster che la compongono sfruttando le funzioni presenti nella libreria.

Nel nostro caso, ogni cluster rappresenterà un oggetto che la protesi può afferrare, presente nell'ambiente. Per ridurre il numero di computazioni è stato scelto di tenere in memoria solo i cluster che non superino una soglia di distanza, in modo da scartare quelli esterni allo spazio di destrezza del braccio dato che essi non sarebbero in ogni caso raggiungibili.

Una volta che si sono campionati i vari cluster, è possibile passare alla terza fase che consiste nella classificazione di questi. Classificare un cluster vuol dire assegnare ad ognuno di questi una tra le possibili etichette apprese in fase di training. Per la ricerca del modello più appropriato da associare all'oggetto in analisi viene utilizzata la libreria FLANN, scelta per la sua efficienza nel migliorare il tempo per effettuare una query.

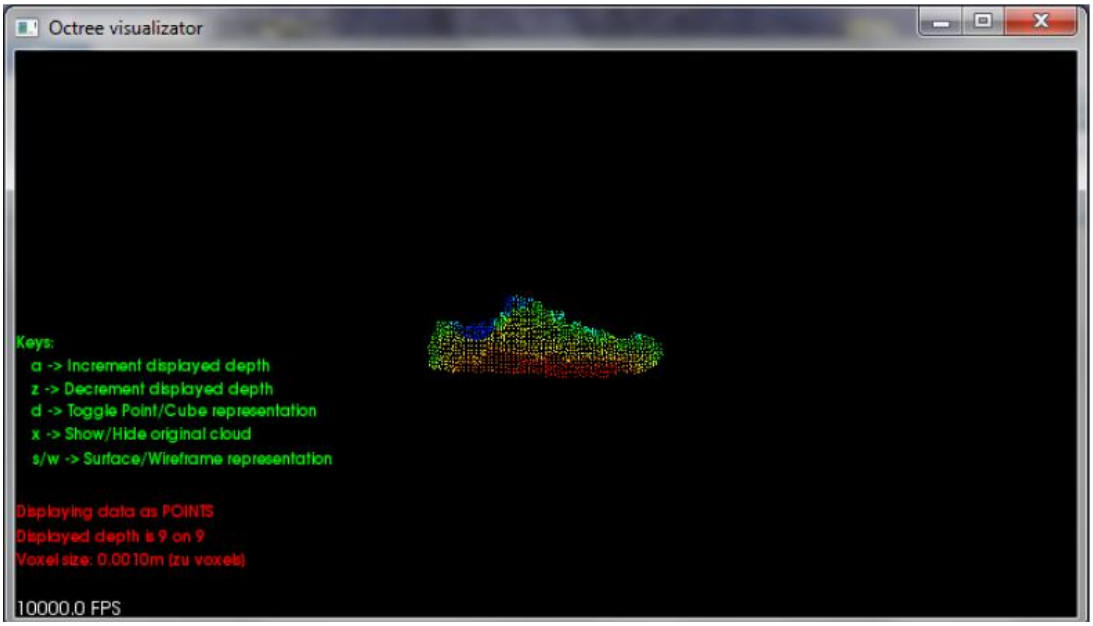


Figura 19: Esempio di cluster di scarpa

Una volta effettuata la classificazione degli oggetti presenti sul tavolo e assegnate le informazioni riguardanti classe di appartenenza e orientamento, è possibile passare alla fase di scelta dell'oggetto da afferrare. Allo stato attuale, per effettuare quest'operazione, l'utente deve inserire a terminale l'identificativo dell'oggetto prescelto.

Questa metodologia è stata scelta dopo aver analizzato altre possibili soluzioni che si sono rivelate inadatte per questo lavoro. Le principali alternative analizzate sono state un sistema di comando watch target bastato su piattaforma inerziale e alcuni tipologie di occhiali eye-tracking.

La prima soluzione presa in considerazione è la piattaforma inerziale 3DM-GX1. Questa è prodotta dalla Microstrain Inc. ed è composta da un sistema integrato di sensori che permette di misurare le tre rotazioni attorno ai tre assi cartesiani di riferimento sia in ambienti dinamici che statici. Il sistema di riferimento assoluto è ottenuto tramite tre magnetometri che posizionano l'asse Z diretto verso il centro del pianeta, l'asse X diretto verso nord e l'asse Y ricavato in modo da avere una terna destrorsa. Questa soluzione permette di rappresentare l'orientamento attraverso matrici, quaternioni o tramite gli angoli di Eulero. Per questa misura il

dispositivo può ricorrere, se richiesto, all'ausilio di giroscopi che ne stabilizzano il risultato rispetto ad accelerazioni lineari e interferenze magnetiche che porterebbero errori di misura.

Nel nostro caso, per determinare la posizione dell'oggetto da afferrare è quindi necessario posizionare la piattaforma inerziale sulla testa del paziente. Questa soluzione ci permette di ricavare il vettore visivo in modo immediato ma lascia diverse possibili soluzioni per la gestione della profondità. Un altro problema è la necessità di convertire le rotazioni dal riferimento assoluto a quello locale. Per fare questo si dovrebbe impostare una seconda piattaforma inerziale, posta sulla spalla del paziente, che permette di avere un riferimento solidale con il corpo.

Questa soluzione non è stata presa in considerazione perché risulta antiestetica e poco ergonomica per il paziente, a causa dell'ingombro che si andrebbe a creare, oltre che dalle protesi, anche dai due dispositivi montati rispettivamente sulla spalla e sulla testa del paziente.

La seconda soluzione è quella che considera l'introduzione di un paio di occhiali capaci di seguire il movimento delle pupille. Questa soluzione è sembrata molto utile per il nostro scopo, infatti percependo la direzione dello sguardo del paziente, l'occhiale può dare le informazioni utili per selezionare automaticamente l'oggetto da afferrare. Questa soluzione inoltre è molto semplice da utilizzare dato che può essere indossato come un comune paio di occhiali.

Lo svantaggio di questo tipo di soluzione però è rappresentato dal prezzo elevato che raggiunge le migliaia di euro. Infatti i vari modelli analizzati (*SMI Eye Tracking Glasses* [28], *SR research eye link II* [29] e *Tracksys Eye Tracking Glasses* [30]*)* hanno prestazioni molto superiori a quelle richieste dal nostro sistema e perciò risultano fuori dalla portata di questo lavoro di tesi che si prefigge di trovare una soluzione a basso costo a portata dell'utente medio.

Figura 20: SMI Eye Tracking Glasses

L'unica fonte, tra quelle analizzate, che propone un dispositivo che risponda alle caratteristiche necessarie è l'articolo che tratta i *GT3D Glasses* [31]. Questo speciale tipo di occhiali è prodotto a un costo estremamente ridotto, attorno ai 60$, e permette di muovere il puntatore del mouse utilizzando solamente lo sguardo. Purtroppo non è stato possibile verificare se il dispositivo potesse essere utilizzato per il nostro lavoro dato il progetto è tuttora in fase di sviluppo.

Per questi motivi, si è preferito lasciare in sospeso questa parte di sistema, in attesa di sviluppi futuri. La scelta dell'oggetto da afferrare è perciò effettuata via shell inserendo il valore intero corrispondente al oggetto che si vuole afferrare.

4.4 Modello di protesi di arto superiore

In questo paragrafo verrà presentato il sistema di simulazione utilizzato per ricreare la protesi fisica di riferimento. Dopo una breve introduzione sul simulatore verrà presentata il modello di protesi di arto superiore.

4.4.1 Simulatore Robotico v-rep

Figura 21: Logo V-REP

V-REP è un simulatore di robot per uso generale che fornisce un ambiente di sviluppo integrato basato su un'architettura di controllo distribuita [33].

La scelta di usare un programma di simulazione, è dovuta alla necessità di progettare l'intero sistema, evitando di impiegare la protesi fisica, in modo da avere una maggiore flessibilità e una diminuzione del tempo d'implementazione del codice.

Sensori, robot, meccanismi e sistemi integrati, possono essere modellati e simulati in vari modi. Questo rende V-REP molto versatile e ideale per le applicazioni multi-robot.

L'uso di V-REP diventa molto importante nei seguenti casi: creazione di prototipi che devono essere collaudati velocemente, sviluppo di algoritmi in modo rapido, educazione correlata alla robotica, monitoraggio remoto, controllo hardware, simulazione di sistemi per automazione industriale, monitoraggio della sicurezza, presentazione di un prodotto, ecc.

Il simulatore si basa su tre elementi principali che sono gli oggetti di scena, i moduli di calcolo e i meccanismi di controllo.

4.4.1.1 Oggetti di scena

I principali elementi di V-REP che sono utilizzati per la costruzione di una scena di simulazione sono gli oggetti di scena. Questi sono visibili nella gerarchia scena e nella vista scena. Nella vista scena, gli oggetti hanno una rappresentazione tridimensionale, mentre nell'altra si ha solo l'indicazione del nome e del ruolo che esso ha rispetto gli altri.

Gli oggetti di scena principali sono i seguenti:

- *forme geometriche*: possono essere, ad esempio, forme piane, cubiche, sferiche, cilindriche che possono essere elaborate in seguito;
- *giunti:* possono essere rotoidali, prismatici, a vite o sferici;
- *luci:* oggetti che permettono di illuminare la scena in punti particolari;
- *oggetti elaborati:* come ad esempio sedie, tavoli, robot, manichini o specchi;
- *grafici:* utilizzati per registrare e visualizzare i dati di simulazione;
- *tracciati:* oggetto che definisce un percorso o traiettoria nello spazio;
- *sensori*: come ad esempio sensori di prossimità, di visione o di forza;
- *camere*: permettono di ottenere una particolare vista dello scenario per poterla registrare.

4.4.1.2 Moduli di calcolo

I moduli di calcolo presenti in V-REP permettono:

- *rilevamento delle collisioni*: il modulo consente di monitorare, registrare e visualizzare le collisioni che potrebbero verificarsi tra qualsiasi entità;
- *calcoli di dinamica:* consente di simulare le interazioni tra gli oggetti che sono vicino a interazioni tra oggetti reali, permettendo a questi di cadere, di scontrarsi, di rimbalzare, ma anche a un manipolatore di afferrare gli

oggetti, un nastro trasportatore di guidare le parti in avanti, o a un veicolo di rotolare in modo realistico su terreni irregolari;

- *calcolo di distanza minima*: misura la distanza minima tra due entità misurabili in maniera molto flessibile;
- *path planning*: esegue calcoli di pianificazione del percorso per gli oggetti in 2-6 dimensioni. Permette di risolvere problemi con vincoli olonomici e non;
- *calcolo delle cinematiche*: consente di risolvere in modo molto efficiente qualsiasi tipo di problema cinematico.

A differenza di molti altri pacchetti software di simulazione, V-REP non è un simulatore di dinamica pura, infatti, può essere visto come un simulatore ibrido che combina cinematica e dinamica al fine di ottenere le migliori prestazioni per i vari scenari di simulazione.

4.4.1.3 Meccanismi di controllo

Ogni aspetto di una simulazione può essere personalizzato; ciò è consentito attraverso un elaborato Application Programming Interface (API) che consente di supportare sei diversi approcci, ciascuno con vantaggi particolari (e ovviamente anche degli svantaggi) rispetto agli altri, reciprocamente compatibili. Gli approcci possono essere del seguente tipo: uno script incorporato, un add-on, un plugin, un nodo ROS, un client API remoto, o una soluzione personalizzata client/server.

L'approccio impiegato in questo lavoro è stato quello delle API remote che permettono di controllare una simulazione (o il simulatore stesso) da un'applicazione esterna o un hardware remoto (ad esempio un robot reale, computer remoto, ecc.). Le API remote sono circa un centinaio di funzioni che possono essere richiamate da un'applicazione C/C++, uno script Python, un'applicazione Java, un programma Matlab, o uno script Urbi.

4.4.1.4 Modello della protesi di arto superiore

Per creare il modello della protesi nell'ambiente v-rep, si è cercato di rappresentare la controparte fisica il più fedelmente possibile, questo per fare in modo che le risposte che si sarebbero ricevute dalla protesi simulata sarebbero state simili a quelle date dalla protesi fisica.

In questa maniera, è reso molto più semplice il passaggio dalla protesi simulata a quella fisica che richiederebbe minimi cambiamenti al modello, poiché conoscendo già il comportamento della protesi, il sistema richiederebbe calibrazioni marginali.

Nell'ambiente di simulazione i vari elementi che compongono il modello sono collegati tra loro secondo uno schema gerarchico nel quale i punti di separazione tra le varie parti sono i giunti, i quali simulano i vari attuatori.

E' stato possibile modellizzare le parti della protesi mantenendo le stesse dimensioni e lo stesso ingombro della loro controparte fisica; il peso, invece, è differente a causa dell'impossibilità di smontare e pesare singolarmente i vari elementi della protesi fisica. Nel modello perciò avendo a disposizione il peso totale del braccio e della mano, si è suddiviso questo fra i vari elementi secondo le proporzioni che statisticamente rappresentano il peso di un braccio umano. [36]

Avendo il peso totale del braccio di 1459 g è stato calcolato un peso del solo avambraccio di 935 g, per il braccio un peso di 524 g, mentre per la mano 453 g.

Nella figura seguente è stata rappresentata la struttura gerarchica del modello della protesi:

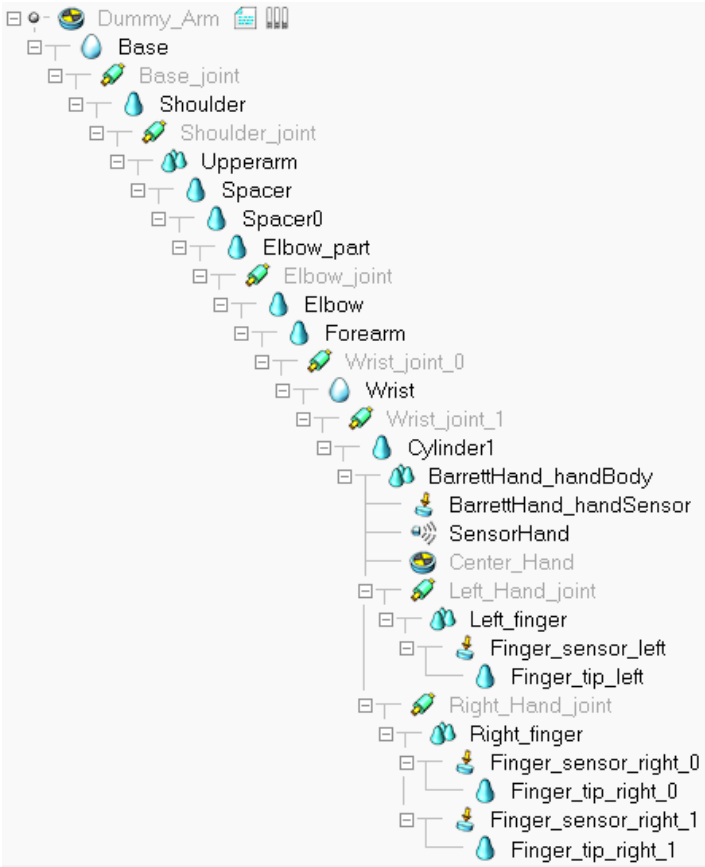


Figura 22: Lista gerarchica del modello della protesi

Poiché nella protesi fisica è sempre stato usato lo stesso motore, anche le caratteristiche dinamiche dei vari giunti sono identiche tra loro, in particolare partendo dalle specifiche dei motori e dei riduttori usati nella protesi reale, si è concluso che la coppia massima ottenibile a 5000 rpm è 3,52 Nm, mentre la velocità massima arriva a 78,12 rpm.

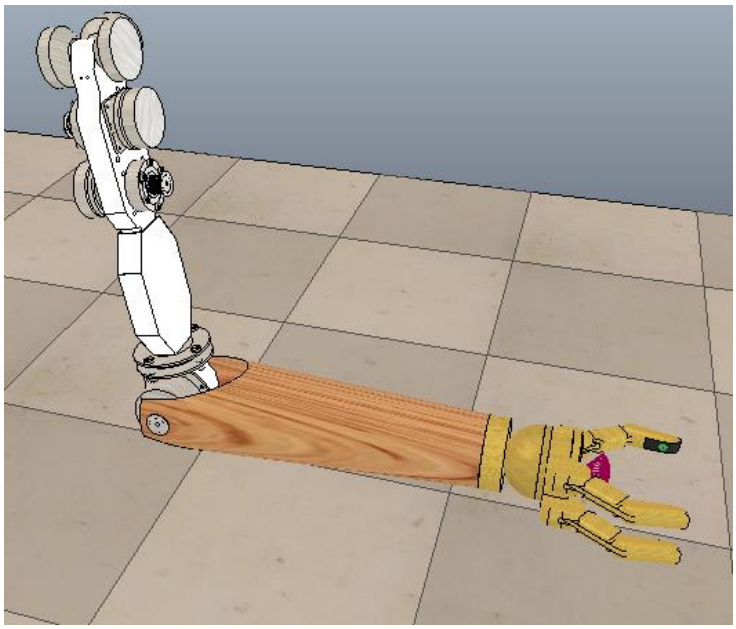


Figura 23: Modello della protesi in v-rep

I modelli del braccio e avambraccio, come si può vedere dalla Figura 23, sono stati riprodotti in modo fedele tramite modelli CAD della protesi fisica; mentre per la mano questi non erano presenti, ed è stata ricreata da modelli base.

Nella mano, inoltre, sono stati inseriti diversi sensori: un sensore di forza in ogni polpastrello e un sensore di prossimità al centro del palmo.

Al momento, il solo sensore di prossimità è utilizzato nella chiusura della mano; sviluppi futuri potrebbero migliorare la presa dell'oggetto gestendo i rimanenti sensori di forza.

4.4.1.5 Scenario in V-REP

Lo scenario utilizzato per la modellazione include la protesi di arto superiore, un manichino, che ha il ruolo di sorreggere la protesi e un tavolo con sopra un bicchiere. Questa scena è stata scelta data la sua semplice riproducibilità in un ambiente reale. Su questo semplice esempio sono stati svolti le varie prove utili a verificare l'efficienza dell'intero sistema.

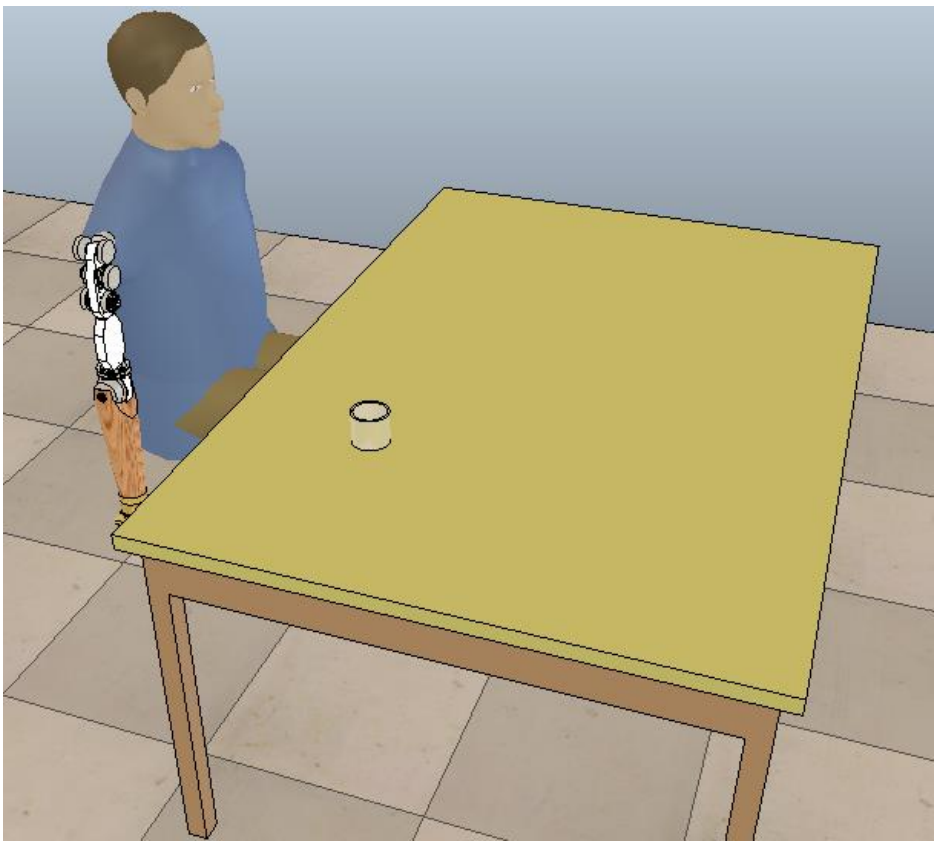


Figura 24: Scenario in V-REP

Capitolo 5

Architettura del sistema

In questo capitolo sarà presentata l'architettura, dal punto di vista software, del sistema frutto di questo lavoro di tesi. Inizialmente, sarà esposto uno schema generale che riassumerà le interazioni tra i vari moduli; in seguito, saranno spiegate le varie parti che compongono l'intera architettura, approfondendo con maggior dettaglio la parte riguardante la simulazione della protesi.

5.1 Moduli dell'architettura

In questo secondo paragrafo saranno presentati i vari moduli che compongono il sistema. L'intera architettura è formata da quattro moduli: il modulo elettromiografico, il modulo di visione, il modulo di calcolo della traiettoria e per ultimo il modulo che si occupa dei movimenti dei giunti. I quattro moduli sono collegati tra loro come indicato dalla Figura 25.

Il modulo elettromiografico, sviluppato presso il Dipartimento di Bioingegneria del Politecnico di Milano [38], si occupa di rilevare i segnali EMG provenienti dagli elettrodi posti sul corpo dell'utente e di analizzarli in modo da restituire la tipologia di movimento che l'utente vuole compiere con la protesi. I movimenti, presi in considerazione per la spalla, sono quelli di flesso/estensione nel piano sagittale e quello di ab/adduzione sul piano frontale. I dati, una volta acquisiti, sono stati elaborati tramite il software Matlab, in modo da indicare il tipo di movimento compiere al modulo di calcolo della traiettoria.

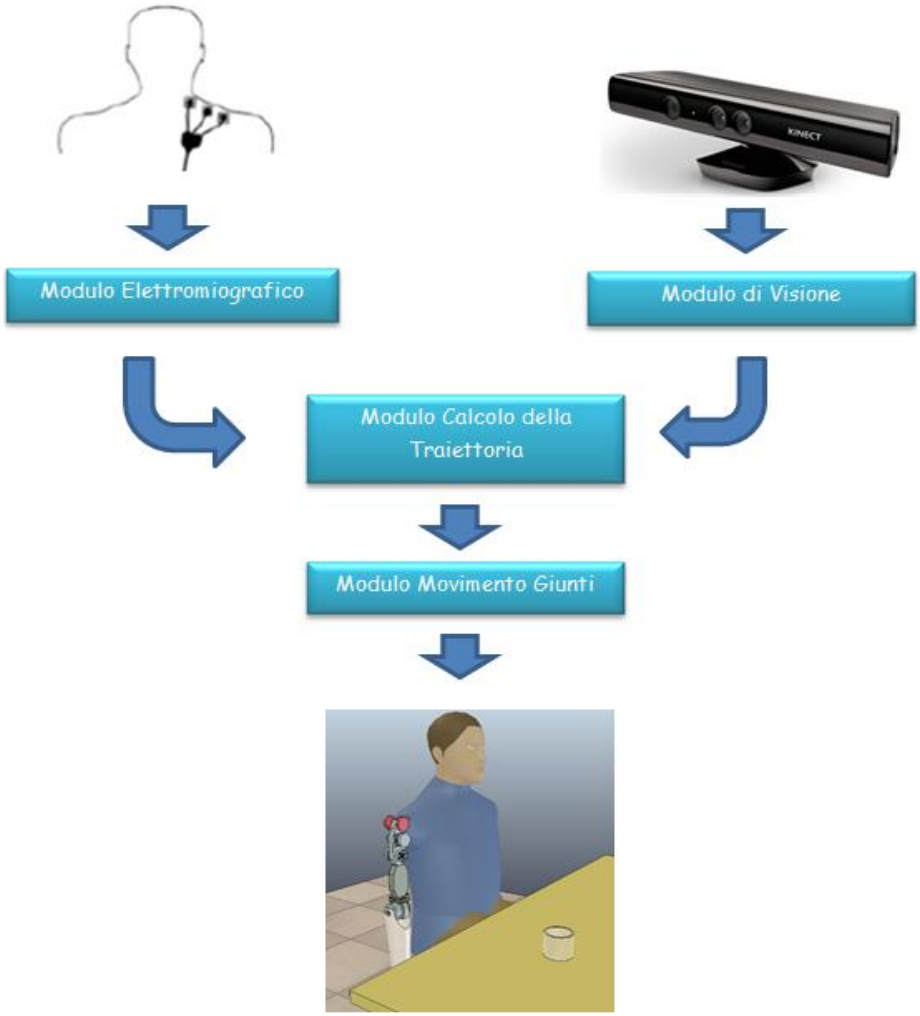


Figura 25: Moduli dell'intero sistema

Il modulo di visione deve analizzare attraverso il sensore Kinect l'ambiente posto di fronte a questo, per identificare la posizione dell'oggetto da afferrare ed eventuali ostacoli da tenere in considerazione. Alla fine di questo rilevamento deve ricercare la migliore traiettoria verso l'oggetto da afferrare e restituire i punti di via che la compongono.

Il modulo di calcolo della traiettoria riceve i risultati dei precedenti moduli, li interpreta in modo da capire che tipo di movimento si vuole effettuare e calcola il valore che deve assumere i vari giunti in modo che si raggiunga il movimento desiderato dall'utente. Questo modulo esegue un controllo ciclico durante lo spostamento in modo da poter percepire eventuali variazioni durante il movimento, in modo da permettere all'utente di avere pieno controllo della protesi.

Il modulo movimento giunti riceve i valori dei giunti calcolati dal modulo precedente ed effettua il movimento di questi nell'ambiente di simulazione. Questo modulo è costituito dalle API per il controllo del modello della protesi in V-REP.

I moduli calcolo della traiettoria e movimento dei giunti saranno ulteriormente approfonditi in questo capitolo, in modo da capire meglio il loro funzionamento.

5.1.1 Modulo calcolo traiettoria

Questo modulo riceve dal modulo elettromiografico le informazioni sul tipo di movimento che l'utente vuole compiere (abduzione o flesso-estensione orizzontale della spalla) e dal modulo di visione le varie informazioni sull'ambiente. In particolare, se la parte di controllo Kinect rileva un possibile oggetto da afferrare, restituisce a questo modulo i punti di via necessari per raggiungere tale oggetto.

Una volta apprese tali informazioni il calcolo della traiettoria da seguire avviene nel modo seguente:

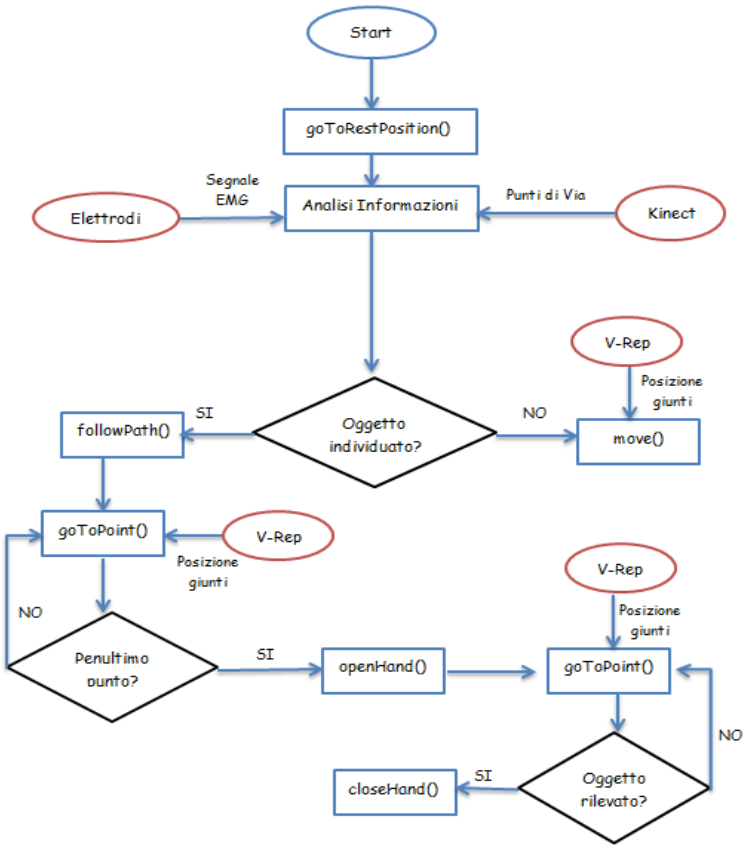


Figura 26: Schema di funzionamento del modulo calcolo traiettoria

Per semplificare la struttura del modulo, questo è stato suddiviso in più classi, in base alle funzionalità implementate.

Nei seguenti paragrafi saranno introdotte le varie classi che lo compongono, spiegando i vari collegamenti tra queste.

L'intero codice non sarà riportato di seguito, ma solo le righe di codice che permettono di comprendere il lavoro svolto.

Nell'appendice, è possibile consultare il codice sorgente di questo lavoro nel caso in cui si vogliano approfondire le varie classi.

5.1.1.1 InverseKinematics.cpp

Questa classe implementa tutte le funzionalità necessarie per il calcolo della cinematica inversa. Dato che quest'argomento è già stato approfondito nel capitolo precedente, saranno presentate le funzioni principali di questa classe, in modo da renderla di più facile comprensione.

```
invKinResult invKin(Point p);
```

La funzione *invKin* contiene i calcoli relativi alla cinematica inversa, riceve in ingresso un unico parametro di tipo *Point* e restituisce un valore di tipo *invKinResult*. L'oggetto *invKinResult* è definito nel seguente modo:

```
typedef struct {

	bool reachable;

	double teta1;
	double teta2;
	double teta3;


} invKinResult;
```

essa contiene perciò un valore booleano che indica se il punto è raggiungibile e tre variabili numeriche che indicano i rispettivi valori dei giunti calcolati.

L'oggetto *Point* è semplicemente definito come una struttura di un punto cartesiano a tre dimensioni e implementa i metodi principali utilizzabili su di essa.

5.1.1.2 Arm.cpp

La classe *Arm* contiene tutte le funzionalità di alto livello per il movimento della protesi. Di seguito saranno introdotte le principali funzioni della classe:

```
Arm(int clientID, int baseJoint, int shoulderJoint, int elbowJoint, int
wristJoint_0, int wristjoint_1, int right_Hand_Joint, int
left_Hand_Joint, int proximity_Sensor);
```

Questa funzione definisce il costruttore della classe *Arm* che riceve in ingresso parametri utili alla comunicazione con il simulatore V-REP: come il codice identificativo della connessione (`clientID`) e interi che rappresentano i vari giunti e il sensore di prossimità, posto sulla mano, nel modello simulato. Tutti questi dati devono essere prelevati dal simulatore V-REP quando si crea la connessione.

```
void goToRestPosition(vector<Point> path);
```

Questa funzione si occupa di portare la protesi nella posizione di riposo che coincide con l'avere tutti i giunti del braccio a un angolo di zero gradi. Questa è molto utile nella fase d'inizializzazione nella quale si vuole preparare la protesi a compiere un movimento. Inoltre, è stata implementata in modo che dopo aver eseguito un movimento, questa possa riportare la protesi alla posizione iniziale passando dai punti calcolati in precedenza, in modo da evitare ostacoli.

Nel caso la posizione non sia corretta la funzione provvede a muovere i giunti al fine di raggiungerla.

```
void goToPoint(Point point);
```

Questa funzione permette di eseguire il movimento dei giunti del braccio (3 gradi di libertà), in modo da raggiungere il punto dato in ingresso. Per fare questo, è chiamata la funzione `invKin (Point p)` in modo da trovare i valori dei giunti da applicare alla protesi. Una volta avuti i vari valori, questi possono essere applicati attraverso l'utilizzo delle API di V-REP. In questo modo, avviene il movimento della protesi, lasciando i giunti della mano inalterati. Durante questo è fatto un controllo ciclico in modo che la protesi raggiunga effettivamente quei valori.

Una volta che il sistema riconosce di aver raggiunto il penultimo dei punti di via utili per raggiungere il punto indicante l'oggetto da afferrare, questa esegue il movimento riguardante i giunti della mano. In questo modo è possibile collocare il centro della mano in modo coerente col tipo di approccio scelto, in corrispondenza dell'oggetto da afferrare.

```
void followPath(vector<Point> path, int tipoApproccio);
```

Questa funzione contiene al suo interno le funzionalità appena illustrate in modo da gestire efficientemente le informazioni provenienti dal sensore Kinect e dagli elettrodi. In ingresso riceve un vettore di punti di via (`path`) che formano il percorso che il braccio deve percorrere e il tipo di approccio per afferrare l'oggetto (`tipoApproccio`).

Per compiere il movimento del braccio è chiamata ciclicamente la funzione `goToPoint`. Prima di raggiungere l'ultimo punto, la mano è aperta in modo da poter afferrare l'oggetto, una volta raggiunto l'ultimo punto, la mano viene chiusa con la sua rispettiva funzione.

All'inizio di ogni iterazione del ciclo è controllato se è stato ricevuto un nuovo comando dall'utente attraverso il modulo EMG, per assicurare a quest'ultimo di poter cambiar traiettoria, il tipo di movimento e l'azione da compiere alla protesi a ogni istante.

Nel caso in cui un nuovo segnale è ricevuto, il ciclo è interrotto e la protesi è lasciata nell'ultima posizione raggiunta. In questo modo il nuovo segnale può essere processato attraverso un nuovo ciclo.

```
void move(int val, float degree);
```

Questa funzione implementa la possibilità di processare un segnale ricevuto attraverso gli elettrodi nel caso in cui il sensore Kinect non identifichi nessun oggetto possibile da manipolare.

In questo modo sono resi possibili quei movimenti che non consistono nell'approccio verso un oggetto, ma possono essere semplicemente movimenti come gesto di saluto.

I parametri in ingresso identificano il giunto da muovere (`val`) e l'ampiezza di movimento di questo (`degree`). Il giunto da muovere può essere 0 nel caso di movimento di flesso/estensione della spalla o 1 nel caso di add/abduzione di questa; nel primo caso l'approccio sarà frontale, mentre nel secondo sarà dall'alto.

```
void setSpeedForce(float speed, float force);
```

Questa funzione permette di impostare i valori di forza e velocità di ogni giunto uguali a quelli passati come parametri al simulatore V-REP. Il simulatore prenderà questi come riferimenti ideali, cercando di portare i giunti a quei valori.

```
void openHand();
```

Questa funzione si occupa dell'apertura delle falangi della mano fino alla loro massima ampiezza.

```
void closeHand ();
```

Questa funzione permette la chiusura delle falangi della mano. Questa funzione è molto importante nella fase di approccio, infatti, una volta identificato l'oggetto da afferrare tramite il sensore di prossimità posto al centro del palmo, chiude lentamente le falangi, afferrando in modo appropriato l'oggetto.

5.1.1.3 ArmClient.cpp

Questa classe implementa il main dell'intera applicazione gestendo i controlli ad alto livello.

Questa permette la comunicazione tra il simulatore V-REP e gli altri moduli, richiamando in base ai segnali ricevuti le diverse funzionalità descritte in precedenza.

Nella prima parte sono dichiarate le variabili di tipo intero che identificano i vari elementi del modello simulato della protesi:

```
int baseJoint;            // Gestore base joint
int shoulderJoint;        // Gestore shoulder joint
int elbowJoint;           // Gestore elbow joint
int wristJoint_0;         // Gestore wrist joint_0
int wristJoint_1;         // Gestore wrist joint_1
```

```
int right_Hand_Joint;      // Gestore right hand joint
int left_Hand_Joint;       // Gestore left hand joint
int proximity_Sensor;      // Gestore del sensore di prossimità
```

In seguito sono dichiarate le variabili utili al collegamento Client-Server con V-REP:

```
int portNb = 0;            // Porta connessione V-REP
int clientID;              // ID client V-REP
```

Questi tipi di variabili, sia quelle del modello sia quelle della connessione, sono acquisiti all'esecuzione della simulazione in V-REP perché passate tramite parametri dalla funzione `int main (int argc, char* argv[])`.

Una volta acquisiti questi tipi d'informazioni, è possibile inizializzare gli altri moduli che si occupano dell'interfacciamento con i sensori EMG e Kinect oltre a quello del movimento del braccio.

```
Emg emgCtr = Emg();
Arm armCtr = Arm(clientID, baseJoint, shoulderJoint, elbowJoint,
wristJoint_0, wristJoint_1, right_Hand_Joint, left_Hand_Joint,
proximity_Sensor);
Command cmd;
```

A questo punto è utile muovere la protesi in posizione iniziale, nel caso in cui non lo sia:

```
armCtr.goToRestPosition ();
```

Una volta in quella posizione, può partire il ciclo di controllo della protesi:

```
while (simxGetConnectionId (clientID)!=-1){
```

All'inizio, si controlla tramite un ciclo `while` che la connessione Client-Server con il simulatore sia presente, in questo modo si assicura che non siano effettuati processi di calcolo se non è possibile poi compiere i movimenti.

In seguito, è richiamata la funzione del modulo EMG che mette in attesa di un segnale elettromiografico il programma.

```
cmd = emgCtr.getRandomEmgSignal ();
```

Una volta acquisito il segnale, questo deve essere analizzato in modo da capire che tipo di movimento sia:

```
if(cmd.getMovementType() == Command::FLT_EST){
```

Nel caso in cui questo sia un commando di flesso/estensione della spalla, sarà dapprima richiamato il sensore Kinect per analizzare l'ambiente in modo da percepire eventuali oggetti da afferrare:

```
path = classifier(0, NULL);
if(!path.empty()){
```

Nel caso in cui la funzione Kinect trova un oggetto nello spazio di lavoro della protesi, questa restituirà un insieme di punti di via per raggiungerlo.

A questo punto è possibile chiamare la funzione per seguire questi punti:

```
armCtr.followPath (path, 0);
```

A questa funzione è passato il vettore contenente i punti di via (`path`) e un intero che indica il tipo di approccio voluto. In questo caso il valore `0` indica un approccio frontale dato che il movimento della spalla è di flesso/estensione.

Nel caso in cui la funzione Kinect non restituisca nessun insieme di punti di via, sarà richiamata la funzione `move` dato che non ha oggetti da afferrare, ma l'unica cosa possibile da fare è eseguire un semplice movimento:

```
else{
	armCtr.move(0, cmd.getWidth());
}
```

Ora è eseguita la stessa identica procedura per l'altro tipo di movimento possibile, cioè l'abduzione della spalla:

```
else if(cmd.getMovementType() == Command::ABDUCTION){
```

In questo caso se il sensore Kinect rileva un possibile oggetto da afferrare, il movimento suggerito sarà diverso dal precedente.

```
path = classifier(1, NULL);
if(!path.empty()){
	armCtr.followPath(path, 1);
}
```

Questo movimento avverrà dall'alto in base all'altezza dell'oggetto rispetto la protesi.

Come prima, nel caso in cui non ci siano oggetti da afferrare, il movimento sarà di semplice abduzione della spalla:

```
else{
	armCtr.move(1, cmd.getWidth());
}
```

Nel caso in cui non si avranno più movimenti da compiere, la comunicazione col server può essere terminata e il simulatore interrotto:

```
simxFinish (clientID);
```

5.1.2 Modulo movimento dei giunti

Questo modulo contiene tutte le funzionalità necessarie per eseguire il controllo dei giunti del modello in V-REP.

Queste funzionalità sono rese disponibili dall'ambiente di simulazione, per fare in modo che questo possa essere utilizzato anche da remoto.

Due importanti parametri che sono utilizzati nelle varie funzioni sono clientID e operationMode. Il primo parametro serve a identificare il client connesso al server V-REP, poiché questo può soddisfare richieste da diversi utenti. Il parametro operationMode indica il metodo di comunicazione tra il client e il server.

Una cosa importante da ricordare è che tutte le funzioni restituiscono un intero simxInt che indica eventuali messaggi di errori di ritorno dalla funzione.

Di seguito saranno presentate le funzioni utilizzate per la comunicazione con il simulatore e il modello:

```
simxInt simxGetJointPosition(simxInt clientID, simxInt jointHandle,
simxFloat* position, simxInt operationMode)
```

Questa funzione restituisce la posizione (in radianti) del giunto identificato dalla variabile jointHandle, tale informazione è salvata nella variabile *position.

```
simxInt simxSetJointPosition(simxInt clientID, simxInt jointHandle,
simxFloat position, simxInt operationMode)
```

Questa funzione muove il giunto identificato dalla variabile `jointHandle` nella posizione (definita in radianti) contenuta nella variabile position.

```
simxInt simxSetJointTargetVelocity(simxInt clientID,simxInt
jointHandle,simxFloat targetVelocity,simxInt operationMode)
```

Questa funzione imposta la velocità desiderata che si vorrebbe raggiungere, attraverso la variabile `targetVelocity`, del giunto identificato dalla variabile `jointHandle`.

```
simxInt simxSetJointForce(simxInt clientID,simxInt jointHandle,simxFloat
force,simxInt operationMode)
```

Questa funzione imposta la coppia desiderata che si vorrebbe raggiungere, passata tramite la variabile force, del giunto identificato dalla variabile jointHandle.

```
simxInt simxReadVisionSensor(simxInt clientID,simxInt
sensorHandle,simxChar* detectionState,simxFloat** auxValues,simxInt**
auxValuesCount,simxInt operationMode)
```

Questa funzione interroga il sensore di prossimità identificato dalla variabile sensorHandle e ne salva il risultato nella variabile detectionState, nel caso il sensore abbia individuato qualcosa, la variabile è posta a 1, altrimenti è lasciata a 0.

Capitolo 6

Risultati Ottenuti

In questo capitolo verranno illustrate le prove effettuate per testare la bontà della soluzione proposta e le valutazioni che da queste si possono trarre.

La prima cosa da controllare è stata quella di verificare che i vari moduli svolgano la propria funzione correttamente, dato che è inutile il loro collegamento se uno di loro non esegue il proprio lavoro correttamente. Per fare questo, sono state create alcune funzioni interne ai moduli per testare la bontà dei risultati.

Una volta fatto questo, si è passato a verificare la corretta comunicazione tra i vari moduli del sistema, in modo che l'architettura funzioni correttamente nel suo complesso. Per soddisfare ciò si è dovuto raggiungere l'obiettivo finale: permettere a un paziente di afferrare un oggetto posto di fronte ad esso.

In questo lavoro di tesi, le prove effettuate hanno preso in esame il modello della protesi realizzato all'interno dell'ambiente di simulazione v-rep.

In particolare inizialmente sono state create delle classi test per il Kinect e i segnali EMG, in modo da rendere autonoma la simulazione della protesi dai vari dispositivi. Successivamente sono state implementate le funzioni per poter comunicare correttamente con i vari sensori.

I test effettuati hanno considerato il solo interfacciamento con il sensore Kinect, lasciando la ricezione dei segnali elettromiografici simulata, dato che lo studio effettuato su di essi ha analizzato le modalità per classificare i vari movimenti, senza però produrre un codice applicativo da integrare all'intero sistema.

I test sull'intero sistema hanno riguardato:

- la precisone di posizione dei singoli giunti quando essi effettuano dei semplici movimenti
- la precisione di posizionamento dell'end-effector rispetto al punto individuato, e comunicato, dal dispositivo Kinect
- il tempo richiesto dal sistema per completare le operazioni di movimento della protesi e di conseguenza il tempo richiesto per afferrare un oggetto passando attraverso diversi punti di via.

Nelle prove effettuate la protesi è sempre stata fatta partire dalla sua posizione di riposo (coincidete con l'ampiezza di tutti i giunti posta a 0°), la traiettoria fornita dal modulo Kinect era formata da quattro punti distinti di cui l'ultimo identificava la posizione dell'oggetto target; in ogni prova il modello di protesi ha sempre correttamente afferrato l'oggetto target, evitando collisioni con gli eventuali ostacoli presenti sulla scena.

Questo risultato è sicuramente riconducibile alla precisione e alla correttezza dei punti di via forniti: infatti il punto finale identificava correttamente la posizione dell'oggetto target e i restanti punti di via erano posti, rispetto agli ostacoli, a una distanza che ne evitava eventuali collisioni.

Prendendo in esame il sistema nella sua interezza si può affermare che tutti gli obiettivi posti siano stati raggiunti con successo.

6.1 Test sulla precisione di posizionamento dei giunti

Per effettuare questo tipo di test è stata impiegata la funzione `move()`, precedentemente presentata, la quale permette di muovere i giunti senza il bisogno di afferrare un oggetto.

Per il calcolo della precisione di posizionamento sono stati spostati i vari giunti in posizioni specifiche, identificate dall'ampiezza in gradi del giunto, in questo modo, una volta raggiunta la posizione, si è potuto calcolare l'errore tra la posizione attesa e quella raggiunta.

I risultati ottenuti si possono osservare nella Tabella 4, sotto illustrata. Sono state effettuate tredici prove muovendo i giunti in posizioni diverse. Dai risultati ottenuti si può notare come non ci siano grandi variazioni tra la posizione attesa e quella effettiva, infatti l'errore massimo è di 0,57° e si ha una media di 0,12°.

N° prova	*Variabile di giunto*	*Posizione attesa*	*Posizione effettiva*	*|Errore|*	*% |Errore|*
1	ϑ_1	0	0,000004	0,000004	0,00%
	ϑ_2	0	0,000006	0,000006	0,00%
	ϑ_3	0	0,000004	0,000004	0,00%
2	ϑ_1	0	0,000004	0,000004	0,00%
	ϑ_2	70	69,627432	0,372568	0,41%
	ϑ_3	0	0,000090	0,000090	0,00%
3	ϑ_1	79	78,485501	0,514499	0,29%
	ϑ_2	11	11,272558	0,272558	0,30%
	ϑ_3	0	0,369182	0,369182	0,41%
4	ϑ_1	78	78,498911	0,498911	0,28%
	ϑ_2	37	37,055854	0,055854	0,06%
	ϑ_3	0	0,243907	0,243907	0,27%
5	ϑ_1	44	44,250227	0,250227	0,14%
	ϑ_2	66	65,999969	0,000031	0,00%
	ϑ_3	12	12,022167	0,022167	0,02%
6	ϑ_1	151	150,999991	0,000009	0,00%
	ϑ_2	30	29,928803	0,071197	0,08%
	ϑ_3	46	46,079854	0,079854	0,09%
7	ϑ_1	78	78,250970	0,250970	0,14%

	ϑ_2	76	75,999831	0,000169	0,00%
	ϑ_3	39	38,999877	0,000123	0,00%
8	ϑ_1	178	177,994424	0,005576	0,00%
	ϑ_2	35	34,979246	0,020754	0,02%
	ϑ_3	23	22,808558	0,191442	0,21%
9	ϑ_1	0	0,000073	0,000073	0,00%
	ϑ_2	50	49,856923	0,143077	0,16%
	ϑ_3	36	35,916280	0,083720	0,09%
10	ϑ_1	129	128,999977	0,000023	0,00%
	ϑ_2	78	77,776482	0,223518	0,25%
	ϑ_3	90	89,850401	0,149599	0,17%
11	ϑ_1	170	169,428072	0,571928	0,32%
	ϑ_2	27	26,999994	0,000006	0,00%
	ϑ_3	65	64,821091	0,178909	0,20%
12	ϑ_1	120	119,999991	0,000009	0,00%
	ϑ_2	90	89,936395	0,063605	0,07%
	ϑ_3	89	89,071281	0,071281	0,08%
13	ϑ_1	90	89,881156	0,118844	0,07%
	ϑ_2	30	29,998434	0,001566	0,00%
	ϑ_3	0	0,000010	0,000010	0,00%
			Media:	**0,123751**	**0,11%**

Tabella 4: Errore di posizionamento dei giunti

La percentuale di errore media, calcolata come la media delle percentuali di errore di ogni singolo giunto rispetto all'ampiezza massima raggiungibile dal giunto stesso è dello 0,11%, quindi è possibile calcolare la precisione di posizionamento dei giunti secondo la seguente formula:

$$1 - \frac{\sum_{i=1}^{n} \%|Errore|_i}{n}$$

La precisione ottenuta tramite questo calcolo è del 99,89%, che risulta pienamente soddisfacente rispetto agli obiettivi prefissati.

6.2 Test sulla precisione di posizionamento dell'end effector

Questo test ha considerato la precisione del posizionamento dell'end effector, dato un punto nel piano cartesiano che questo deve raggiungere.

Per fare questo è stata utilizzata la funzione goToPoint() per il movimento della protesi verso un dato punto.

Come si può notare dalla Tabella 5, sono state svolte 13 prove e sono state considerate le coordinate attese ed effettive raggiunte dall'end-effector nel movimento.

N° prova	*Coordinata*	*Valore atteso (m)*	*Valore effettivo (m)*	*|Errore| (m)*	*|Errore distanza|(m)*
1	X	0,0850	0,0850	0	0
	Y	0	0	0	
	Z	0,8725	0,8725	0	
2	X	0,6747	0,6475	0,0272	0,0278
	Y	0	0	0	
	Z	1,2854	1,2911	0,0057	
3	X	0,2047	0,2077	0,0030	0,0081
	Y	0,6025	0,6034	0,0009	
	Z	1,3719	1,3794	0,0075	
4	X	0,4626	0,4466	0,0160	0,0280
	Y	0,4902	0,4695	0,0207	
	Z	1,3958	1,4059	0,0101	
5	X	0,6513	0,6513	0	0,0013
	Y	0,2271	0,2278	0,0007	
	Z	1,3688	1,3699	0,0011	
6	X	0,3457	0,3450	0,0007	0,0008
	Y	0,0003	0	0,0003	
	Z	2,0161	2,0163	0,0002	
7	X	0,6187	0,6187	0	0,0011
	Y	0,1756	0,1748	0,0008	
	Z	1,6862	1,687	0,0008	
8	X	0,4291	0,4292	0,0001	0,0012
	Y	-0,1185	-0,1174	0,0011	
	Z	1,9958	1,9963	0,0005	
9	X	0,5149	0,5142	0,0292	0,0016
	Y	0,2043	0,2038	0,0005	
	Z	1,1393	1,138	0,0013	

10	X	0,3589	0,3595	0,0006	0,0013
	Y	-0,1734	-0,1725	0,0009	
	Z	1,8067	1,8075	0,0008	
11	X	0,2788	0,2792	0,0004	0,0059
	Y	-0,2441	-0,2392	0,0049	
	Z	1,9292	1,9324	0,0032	
12	X	0,3711	0,3706	0,0005	0,0006
	Y	-0,1737	-0,1735	0,0002	
	Z	1,8009	1,8011	0,0002	
13	X	0,3987	0,3987	0	0,0011
	Y	0,5434	0,5434	0	
	Z	1,5000	1,4989	0,0011	
			Asse X:	**0,0060**	
		Media:	**Asse Y:**	**0,0024**	**0,0061**
			Asse Z:	**0,0025**	

Tabella 5: Errore di posizionamento dell'end effector

Infine, è stato calcolato l'errore tra le due posizioni trovate.

Dai risultati si può notare come l'errore di posizionamento sia molto piccolo, infatti valutando quest'errore per ogni asse cartesiano si ha:

- 6 mm sull'asse delle X;
- 2,4 mm sull'asse delle Y;
- 2,5 mm sull'asse Z.

Da questi risultati, si può concludere che l'errore maggiore si ha sull'asse delle X, dove avviene il movimento di abduzione della spalla, su tale asse è stato registrato anche il valore massimo di errore sulla singola coordinata pari a 27,2 mm.

L'errore medio rilevato, riguardante la distanza tra il punto atteso e quello raggiunto, è stato di 6,1 mm.

Questi errori sono pienamente compatibili con gli obiettivi proposti e con gli scenari in cui tale sistema verrà a trovarsi: infatti un errore nell'ordine di 10^1 mm può considerarsi trascurabile per la stragrande maggioranza delle attività che una persona comune compie nel corso della sua quotidianità.

6.3 Test sui tempi di risposta del sistema

Questo test ha permesso di valutare i tempi di risposta del sistema, in modo da verificare che questi soddisfino gli obiettivi di avere un movimento immediato nella protesi.

Per fare questo, è stato utilizzato lo scenario costruito nel simulatore in cui si è posto un manichino seduto di fronte a un tavolo, sul quale è stato posto un bicchiere a 35 cm dal manichino.

N° prova	*Tempo produzione punti di via (s)*	*Tempo movimento protesi (s)*	*Tempo totale di movimento*
1	7,75	18,81	22,25
2	7,78	20,73	22,96
3	7,89	19,14	20,68
4	7,45	18,86	22,24
5	7,98	18,70	22,07
6	7,99	18,69	22,22
7	7,68	18,95	21,09
8	7,82	18,71	21,25
9	7,79	18,98	21,44
10	7,83	18,72	21,68
Media:	**7,80**	**19,03**	**26,82**

Tabella 6: Tempo di risposta del sistema

Nelle analisi non è stato effettuato il calcolo del tempo impiegato dal modulo elettromiografico per individuare il tipo di movimento, dato che questo dipende soprattutto dal sistema e dalle risorse computazionali impiegate.

In questo caso sono state effettuate dieci prove, riavviando ogni volta la simulazione e calcolando i tempi impiegati dal modulo Kinect nel calcolo dei punti di via e dal simulatore per effettuare il movimento di presa.

Dalle prove effettuate, purtroppo, non si hanno avuto i risultati sperati: come si può notare dalla Tabella 6, i test hanno rivelato che, in media, sono necessari circa 27 secondi per afferrare un oggetto posto a circa 35 cm dal paziente.

In particolare più di 7 secondi sono necessari al modulo Kinect per analizzare la scena e produrre i punti di via atti a portare la protesi in corrispondenza dell'oggetto da afferrare.

La colpa di questi risultati però non è solo da imputare al sensore Kinect, dato che il tempo medio richiesto dalla protesi per spostarsi raggiungendo tre punti di via è di circa 19 secondi.

Questi tempi sono ben distanti da quelli richiesti da una protesi fisica, in quanto il movimento per compiere lo stesso gesto da una persona normodotata è di circa 5 secondi, molto inferiore rispetto i tempi ottenuti.

Inoltre, una protesi di questo tipo risulterebbe particolarmente scomoda per una persona, dato che dovrebbe attendere all'incirca tra i 20-30 secondi ogni volta che vuole compiere un movimento atto ad afferrare un'oggetto.

Per valutare meglio i risultati ottenuti è stata effettuato un successivo test per verificare se questo ritardo dipendesse dal fatto che il sistema ogni volta venisse ripristinato da capo. Per fare ciò sono state eseguite altre dieci prove senza interrompere la simulazione, effettuando dei movimenti uno dietro l'altro.

In questo caso, la protesi partendo dalla posizione iniziale si muove passando attraverso i tre punti di via per afferrare l'oggetto; una volta fatto ciò questo lo rilascia ritornando alla posizione di partenza ricominciando il ciclo con un altro movimento.

N° prova	*Tempo produzione punti di via (s)*	*Tempo movimento protesi (s)*	*Tempo totale di movimento (s)*
1	7,87	14,38	22,25
2	7,89	15,07	22,96
3	7,79	12,89	20,68
4	7,82	14,42	22,24
5	7,73	14,34	22,07
6	7,78	14,44	22,22
7	7,80	13,29	21,09
8	7,84	13,41	21,25
9	7,72	13,72	21,44
10	7,71	13,97	21,68
Media:	**7,80**	**13,99**	**21,79**

Tabella 7: Tempo di risposta del sistema nella seconda prova

Da questi risultati si può notare che il tempo per la produzione dei punti di via rimane invariato, circa 7,8 secondi. Questo indica come il ritardo dato dal modulo Kinect non sia influenzato dal sistema di simulazione, ma dipende strettamente dall'algoritmo che gestisce questo modulo e dalle capacità di calcolo messi a disposizione.

Il tempo di movimento della protesi dalla ricezione dei punti di via fino alla presa dell'oggetto è invece variato, infatti, si è passato dai precedenti 19,03 agli attuali 13,99 secondi in media. Questo risultato, di circa 5 secondi in meno, è un miglioramento significativo ma non ancora sufficiente per un utente medio di protesi.

Dopo aver analizzato questi risultati si può dire che:

- l'ambiente di simulazione effettui più operazioni duranti i primi istanti in cui si fa partire la simulazione rispetto al resto del tempo, infatti, il primo movimento richiede circa il 26% del tempo in più rispetto a tutti gli altri, anche se il tipo di movimento risulta identico.

- il tempo di risposta della protesi è dovuto in buona parte dalla politica di movimento adottata dall'algoritmo che calcola i punti di via: infatti l'algoritmo privilegia le traiettorie sicure e prive di collisioni con altri oggetti, piuttosto che una traiettoria più breve. Nonostante si abbia il beneficio di passare a una distanza di sicurezza da eventuali oggetti, il tragitto percorso dalla protesi risulta più lungo rispetto a quello percorso da una persona comune, allungando i tempi di esecuzione.

Si può concludere che il sistema risulta affidabile, preciso e robusto, in quanto gli errori di posizionamento e di movimento della protesi presentano valori più che accettabili per le operazioni quotidiane per la quale è stato pensato. L'unico svantaggio lo si trova nei risultati ottenuti sui tempi di esecuzione, che risultano ancora troppo lunghi e non adatti ad una protesi reale.

Capitolo 7

Conclusioni

In questo capitolo verranno discusse le conclusioni tratte da questo lavoro di tesi e verranno presentati gli scenari rimasti aperti e i possibili sviluppi futuri che da questo studio sono nati.

Il metodo di controllo usato per la protesi si è basato sulle innovative tecniche di Pattern Recognition che considerassero i segnali EMG provenienti da muscoli residui della spalla. Questi segnali, sono inoltre, sempre presenti in qualsiasi tipo di amputazione subita agli arti superiori, rendendo, questo studio, fondamentale per la ricerca in campo medico e protesico.

Lo scopo finale del lavoro è stato quello di realizzare un sistema completo, innovativo e a basso costo, che considerasse questo nuovo tipo di controllo e permettesse a una protesi di arto superiore di potersi muovere, in accordo ad informazioni provenienti da sensori, per poter raggiungere oggetti di fronte ad essa.

Il modello di protesi virtuale è stato sviluppato riproducendo la protesi progettata dal gruppo Meccanica dei Sistemi Uomo Macchina del Dipartimento di Meccanica del Politecnico di Milano. Essa è munita di: due gradi di libertà attivi a livello della spalla, che consentono di compiere i movimenti di flesso-estensione sul piano sagittale, di abduzione-adduzione sul piano frontale e loro combinazioni; un grado di libertà passivo a livello dell'omero, che impedisce danni alla protesi in caso di urto; un grado di libertà attivo in corrispondenza del gomito, che ne permette il movimento di flesso-estensione; una protesi di mano commerciale avente due gradi di libertà sul polso più un grado di libertà per l'apertura e la chiusura delle falangi.

I sensori impiegati per l'acquisizione delle informazioni propedeutiche al movimento sono stati:

- Un insieme di elettrodi posti sulla spalla del paziente il cui segnale processato è stato usato per creare una stringa identificante il tipo di movimento della spalla che il paziente aveva intenzione di compiere;

- Un dispositivo Kinect montato sul petto del paziente, che analizzava la scena posta di fronte ad esso, individuava i potenziali oggetti che potevano essere oggetto di presa da parte della protesi, e successivamente produceva un vettore contenente i punti di via formanti la traiettoria da seguire per poter afferrare, senza collidere con eventuali ostacoli, l'oggetto individuato dal paziente tra quelli presenti.

La protesi è stata riprodotta fedelmente in un ambiente virtuale di simulazione dinamica (V-REP), e attraverso questo è stato testato il funzionamento completo del sistema di controllo sviluppato ricevendo risultati simili a quelli che si avrebbero avuti impiegando la protesi fisica.

Il mio ruolo in questo progetto è stato quello di contribuire alla progettazione del codice di controllo della protesi virtuale e confrontare i vari risultati ottenuti dall'analisi delle varie parti del sistema, con l'obiettivo di verificare che questi soddisfino i requisiti prefissati.

Dai test effettuati è emerso che il sistema così sviluppato è preciso, robusto ed affidabile: la precisione di posizionamento dei giunti è stata del 99,89% con un errore medio di posizionamento nello spazio cartesiano di 6,1mm; inoltre in tutti i test effettuati nei quali l'obiettivo era afferrare un oggetto, la protesi ha sempre centrato l'obbiettivo afferrando il target nel 100% dei casi.

Meno incoraggianti sono stati i risultati riguardanti i tempi di risposta e di movimento del sistema: il tempo richiesto dalla protesi per muoversi risulta particolarmente elevato, di circa 4 volte superiore al tempo richiesto a una persona per eseguire lo stesso tipo di movimento. La causa di questo è dato sia dall'elevato tempo richiesto dal modulo di visione per calcolare i punti di via e dalle ampie traiettorie da questo prodotte.

Un altro motivo di questo, è dato dal codice generato sia per la simulazione, sia per il calcolo dei punti di via, che non è ottimizzato per un architettura particolare ma è stato implementato in modo tale che l'intero lavoro potesse funzionare su diversi sistemi. Infatti, la progettazione così fatta permette una facile migrazione del sistema di controllo dalla protesi simulata alla protesi fisica; difatti, basterà sostituire le funzioni presenti nel modulo di movimento (che ora comandano la protesi virtuale) con le rispettive funzioni del firmware della protesi fisica, nessun altro cambiamento nel codice è necessario.

E' lasciato, a sviluppi futuri, l'ottimizzazione di questo codice che sicuramente permetterebbe di diminuire in modo rilevante i tempi di risposta del sistema.

Un altro problema riscontrato è stato il diverso livello di astrazione utilizzato tra la protesi fisica e quella virtuale. La prima, permetteva il movimento attraverso l'utilizzo di motori, mentre il simulatore permette soltanto di effettuare modelli robotici a livello di giunti. Nonostante questo, il passaggio da un modello all'altro è possibile e non deve essere considerato un grosso ostacolo.

Una questione molto importante, che conviene ricordare, è che l'interazione tra corpo-protesi, qui poco considerata, non è facile da modellizzare, dato che ogni soggetto ha le proprie caratteristiche. Infatti, le protesi sono create su misura, rispetto al livello di amputazione subito e ad alcune caratteristiche fisiche del soggetto, come il peso, altezza, lunghezza braccia, ecc.

Un'altra importante questione, che è stata fondamentale per la scelta della protesi mioelettrica, è stata quella di lasciare al soggetto il maggior controllo possibile sulla protesi. Per fare questo, oltre ai due tipi di movimenti scelti, è stato considerato un segnale di stop, che consentisse all'utente di cambiare tipo di movimento o di ritornare alla posizione di riposo.

Purtroppo, il calcolo della traiettoria è stata implementato considerando il caso pessimo, in cui un tavolo sia posto di fronte al soggetto; questo fa sì che il movimento iniziale sia effettuato lateralmente e molto ampio. In questo modo, il movimento della protesi diventa molto robotico e innaturale e potrebbe infastidire il soggetto che la usa.

L’uso dei segnali elettromiografici ha permesso di utilizzare una nuova metodologia di controllo tramite Pattern Recognition. Quest’analisi, è stata effettuata solo i muscoli della spalla, tralasciando al calcolo della traiettoria la scelta di movimento del gomito e della mano. Questo però, ha permesso di avere dei buoni risultati, che potrebbero in futuro essere migliorati nel caso in cui si considerino altri tipi di protesi per amputazioni a livelli diversi. In quei casi, si avrebbero un numero maggiore di informazioni elettromiografiche e si otterrebbero migliori risultati.

Questo lavoro apre ampi spazi per possibili sviluppi futuri, in primis l’implementazione del sistema all’interno della protesi fisica, in modo da verificare se le risposte date dal sistema sono positive al pari di quelle fornite dalla controparte virtuale.

Un altro fattore, che potrebbe essere migliorato, è la scelta dell’oggetto da afferrare. Alla fine di questo lavoro, la scelta di questo è effettuata tramite l’inserimento di una stringa tramite shell. Potrebbe essere molto interessante poter valutare un nuovo sistema visivo che permetta la scelta dell’oggetto attraverso lo sguardo del soggetto. Inoltre, l’uso di un sensore di profondità più preciso e dalla maggiore risoluzione permetterebbe al modulo di visione di calcolare traiettorie più corte e meno ampie.

Questo lavoro, potrà essere sfruttato, inoltre, in quelle applicazioni che hanno come obiettivo la riabilitazione dell’arto superiore, come gli esoscheletri o l’addestramento all’uso delle protesi. Infatti, grazie a questo sistema il soggetto può semplicemente allenarsi all’impiego di queste, attraverso il sistema simulato, evitando di danneggiare la protesi fisica o l’esoscheletro.

Infine, nel caso i test effettuati sulla protesi fisica dovessero dare i risultati sperati, si potranno iniziare i test clinici di sperimentazione sui pazienti.

Questo lavoro di tesi, visti i risultati ottenuti, può essere considerato un significativo passo in avanti rispetto al problema che si prefiggeva di risolvere, ponendolo come un aiuto importante al problema di restituire funzionalità ed indipendenza ai soggetti interessati da amputazioni totali di braccia.

Bibliografia

[1] J.C.K. Lai, M.P. Schoen, A. Perez Gracia, D.S. Naidu and S.W. Leung, "*Prosthetic devices: challenges and implications of robotic implants and biological interfaces*", Proc. IMechE, Vol. 221, p. 173-183, 2007

[2] Gregory-Dean, *"Amputations: statistics and trends",* Annals of the Royal College of Surgeons of England vol. 73, p. 137-142, 1991

[3] E.J. Davies, B.R. Friz and F.W. Clippinger, *"Amputees and their Prostheses",* Artificial Limbs, Vol. 14, No 2, p. 19-48, Autumn 1970

[4] C. Lake and R. Dodson, "*Progressive Upper Limb Prosthetics*", Physical Medicine and Rehabilitation Clinics of North America, 2006

[5] R. Sacchetti and A. Davalli, "*Le protesi di arto*", 2009

[6] E. Biotto, "*Le protesi di arto superiore*" , Università Cattolica del Sacro Cuore, 2009

[7] C. Toledo, L. Leija, R. Munos, A. Vera and A. Ramirez, *"Upper limb prostheses for amputation above elbow",* Health Care Exchanges, PAHCE Pan American, p. 104-108, 2009

[8] E. Mainardi and A. Davalli, "*Controlling a prosthetic arm with a throat microphone*", Engineering in Medicine and Biology Society, p. 3035-3039, EMBS 2007

[9] A.A. Ali, *"EMG signals detection technique in voluntary muscle movement",* Information Science and Service Science and Data Mining (ISSDM), 6th International Conference on New Trends, p. 738 – 742, 2012

[10] C. Castellini, E. Gruppioni, A. Davalli and G. Sandini, *"Fine detection of grasp force and posture by amputees via surface electromyography"*, Journal of Physiology, p. 255-262, 2009

[11] N. Fligge, H. Urbanek, P. van der Smagt, *"Relation between object properties and EMG during reaching to grasp",* Journal of Electromyography and Kinesiology, 2012

[12] K. Englehart, B. Hudgins, P. Parker. "*A wavelet-based continuous classification scheme for multifunction myoelectic control"*. 2001, p. 302-310.

[13] R.F. Weir, A.B. Ajiboye. "*A heuristic fuzzy logic approach to EMG pattern recognition for multifunctional prosthesis control*" . IEEE Trans. Neural Syst. Rehabil. Eng. 2005, Vol.13, p. 280 -291.

[14] L. J. Hargrove, K. Englehart and B. Hudgins. "*A comparison of surface and intramuscular myoelectric signal classification*." IEEETrans. Biomed. Eng. 2007, Vol. 54, p. 847 -853.

[15] D. Farina, R. Merletti. "*Comparison of algorithms for estimation of EMG variables during voluntary isometric contractions."* Journal of Electromyography and Kinesiology. 2000, Vol. 10, 5, p. 337–349.

[16] Moradi, R. Boostani and M. H. "*Evaluation of the forearm EMG signal features for the control of a prosthetic hand."* Physiol. Meas. 2003, Vol. 24, p. 309 -319.

[17] K. Englehart, B. Hudgins, P. Parker, M. Stevenson. "*Classification of the myoelectic signal using time-frequency based representation.*" Med. Eng. Phys. 1999, Vol. 21, p. 431-438.

[18] A. Phinyomark, A. Nuidod, P. Phukpattaranont, C. Limsakul,. *Feature reduction and selection for EMG signal classification*. Expert Systems and Applications. 2012c, p.7020-7431.

[19] A. J. Jain, R. P. W. Duin and J. Mao. *"Statistical pattern recognition: A review.*" IEEE Transactions on Pattern Analysis and Machine Intelligence. 2000, Vol. 22, 1, p. 4 -37.

[20] X. Xi and Q. Zhang, *"Research and Development: Two-freedom Electromyography Prosthetic Hand",* Information Science and Engineering, p. 453-456, 2010

[21] J. Smisek, M. Jancosek and T. Pajdla, *"3D with Kinect"*, IEEE International Conference on Computer Vision Workshops, p. 1154-1160, 2011

[22] J. Han, L. Shao and D. Xu, *"Enhanced Computer Vision with Microsoft Kinect Sensor: A Review",* Cybernetics, IEEE Transactions on, 2013

[23] N. Kitsunezaki, E. Adachi, T. Masuda and J. Mizusawa, *"Kinect applications for the physical rehabilitation ",* Medical Measurements and Applications Proceedings, p. 294-299, 2013

[24] P. Rakprayoon, M. Ruchanurucks and A. Coundoul, *"Kinect-based Obstacle Detection for Manipulator"*, System Integration, IEEE/SICE International Symposium on, p. 68-73, 2011

[25] R. Bogdan, R. and S. Cousins, *"3D is here: Point Cloud Library (PCL)",* Robotics and Automation (ICRA), IEEE International Conference on, 2011

[26] Bullet Physics Library project. *http:// bulletphysics.org*

[27] E. Franchi, Tesi di laurea magistrale: *Object pose and grasping points estimation using a depth map generated by the Kinect camera*. Politecnico di Milano, 2012.

[28] SMI eye tracking glasses project. *http://www.eyetracking-glasses.com*

[29] SR research eye link II project. *http://www.sr-research.com/EL_II.html*

[30] Tracksys eye tracking glasses project. *http://www.tracksys.co.uk/product-details.php?id=24*

[31] GT3D glasses. *http://usabilitygeek.com/gt3d-eye-tracker-makes-life-easier-for-physically-impaired-users*

[32] F. Valerio, Tesi di dottorato: *Sistema di controllo per una protesi di arto superiore.* Politecnico di Milano, 2011

[33] v-rep by Coppelia Robotics. *http://www.coppeliarobotics.com*

[34] Otto Bock Corporation. *http://www.ottobock.com*

[35] G. Gini e V. Caglioti, Robotica, Bologna: Zanichelli, 2003

[36] R. Drillis, R. Contini and M. Bluestein, "*Body Segment Parameters: A Survey of Measurement Techniques*", Artificial Limbs, vol. 8, n. 1, p. 44-66, 1964.

[37] L. Cavazzana, Tesi di laurea magistrale : "*Integrating an EMG signal classifier and a hand rehabilitation device: early signal recognition and real-time performances*", 2011

[38] D. Rivela and A. Scanella, Tesi di laurea magistrale : "*Classificazione del segnale sEMG tramite Pattern Recognition per il controllo del giunto della spalla di una protesi attiva di arto superiore*", 2013

.

Appendice A

Codice Sorgente

In questa appendice viene riportato per intero il codice sorgente dei moduli di controllo della traiettoria e di movimento dei giunti illustrati nei capitoli precedenti.

Arm.h

```
/* Classe Arm gestisce il braccio */

#pragma once

#define _USE_MATH_DEFINES

#include "Kinematics.h"
#include <windows.h>
#include "Emg.h"

extern "C" {
    #include "extApi.h"
}

class Arm
{
private:
    int baseJoint;                // Gestore base joint
    int shoulderJoint;            // Gestore shoulder joint
    int elbowJoint;               // Gestore elbow joint
    int wristJoint_0;             // Gestore wrist joint_0
    int wristJoint_1;             // Gestore wrist joint_1
    int right_Hand_Joint;         // Gestore right hand joint
    int left_Hand_Joint;          // Gestore left hand joint
    int proximity_Sensor;         // Gestore del sensore di prossimità
    float teta1, teta2, teta3, teta4, teta5;
    float delta;
    int clientID;
    int error;
    float speed;
    float force;
    float tetaLeft, tetaRight;

public:
```

```
        Arm(int clientID, int baseJoint, int shoulderJoint, int elbowJoint,
int wristJoint_0, int wristjoint_1, int right_Hand_Joint, int
left_Hand_Joint, int proximity_Sensor);
        ~Arm(void);
        void goToRestPosition(vector<Point> path);
        void goToPoint(Point point);
        void followPath(vector<Point> path, int tipoApproccio);
        void move(int val, float degree);
        void setSpeedForce(float speed, float force);
        void openHand();
        void closeHand();
};
```

Arm.cpp

```
#include "Arm.h"
#include "KinematicModel\KinematicModelTypes.h"

invKinResult invKin(Point p, bool recursive);

Arm::Arm(int clientID, int baseJoint, int shoulderJoint, int elbowJoint, int
wristJoint_0, int wristJoint_1, int right_Hand_Joint, int left_Hand_Joint,
int proximity_Sensor)
{
        this->clientID = clientID;
        this->baseJoint = baseJoint;
        this->shoulderJoint = shoulderJoint;
        this->elbowJoint = elbowJoint;
        this->wristJoint_0 = wristJoint_0;
        this->wristJoint_1 = wristJoint_1;
        this->right_Hand_Joint = right_Hand_Joint;
        this->left_Hand_Joint = left_Hand_Joint;
        this->proximity_Sensor = proximity_Sensor;
        this->teta1 = 0.f;
        this->teta2 = 0.f;
        this->teta3 = 0.f;
        this->teta4 = 0.f;
        this->teta5 = 0.f;
        this->tetaLeft = 0.f;
        this->tetaRight = 0.f;
        this->error = 0.f;
        this->delta = 10.f / 180 * M_PI;
        this->speed = 100;
        this->force = 100;
}


Arm::~Arm(void)
{
}

/* Funzione che porta il braccio nella posizione di riposo */
void Arm::goToRestPosition(vector<Point> path)
{
        printf("FUNCTION: goToRestPosition\n");

        error = simxGetJointPosition(clientID, baseJoint, &teta1,
simx_opmode_streaming );
```

```
	error = simxGetJointPosition(clientID, shoulderJoint, &teta2,
simx_opmode_streaming );
	error = simxGetJointPosition(clientID, elbowJoint, &teta3,
simx_opmode_streaming );
	error = simxGetJointPosition(clientID, wristJoint_0, &teta4,
simx_opmode_streaming );
	error = simxGetJointPosition(clientID, wristJoint_1, &teta5,
simx_opmode_streaming );
	error = simxGetJointPosition(clientID, left_Hand_Joint, &tetaLeft,
simx_opmode_streaming );
	error = simxGetJointPosition(clientID, right_Hand_Joint, &tetaRight,
simx_opmode_streaming );
	printf("Sono in teta1: %f\n", teta1*180/M_PI);
	printf("Sono in teta2: %f\n", teta2*180/M_PI);
	printf("Sono in teta3: %f\n", teta3*180/M_PI);
	printf("Sono in teta4: %f\n", teta4*180/M_PI);
	printf("Sono in teta5: %f\n", teta5*180/M_PI);

	setSpeedForce(this->speed, this->force);

	if(!((path.at(0).getValue(Point::X) == 0) &&
(path.at(0).getValue(Point::Y) == 0) && (path.at(0).getValue(Point::Z) ==
0))){
		for(int i = path.size()-1; i >= 0; i--)
		{
			goToPoint(path[i]);
		}
	}
	error = simxSetJointTargetPosition(clientID, baseJoint, 0,
simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, shoulderJoint, 0,
simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, elbowJoint, 0,
simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, wristJoint_0, 0,
simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, wristJoint_1, 0,
simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, left_Hand_Joint, 0,
simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, right_Hand_Joint, 0,
simx_opmode_oneshot);
	printf("Vado in posizione start\n");

	while(!(teta1 <= delta && teta1 >= -delta && teta2 <= delta && teta2
>= -delta && teta3 <= delta && teta3 >= -delta && teta4 <= delta && teta4 >=
-delta && teta5 <= delta && teta5 >= -delta))
	{
		Sleep(2000);
		error = simxGetJointPosition(clientID, baseJoint, &teta1,
simx_opmode_buffer );
		error = simxGetJointPosition(clientID, shoulderJoint, &teta2,
simx_opmode_buffer );
		error = simxGetJointPosition(clientID, elbowJoint, &teta3,
simx_opmode_buffer );
		error = simxGetJointPosition(clientID, wristJoint_0, &teta4,
simx_opmode_buffer );
		error = simxGetJointPosition(clientID, wristJoint_1, &teta5,
simx_opmode_buffer );
	}
		printf("Sono in posizione start\n\n");
```

```
}

/* Funzione che porta il braccio verso un punto */
void Arm::goToPoint(Point p)
{
	printf("FUNCTION: goToPoint %f, %f, %f\n", p.getValue(Point::X),
p.getValue(Point::Y), p.getValue(Point::Z));

	float angle_base, angle_shoulder, angle_elbow, angle_wrist_0,
angle_wrist_1;
	vector<float> joints;

	error = simxGetJointPosition(clientID, baseJoint, &teta1,
simx_opmode_buffer);
	error = simxGetJointPosition(clientID, shoulderJoint, &teta2,
simx_opmode_buffer);
	error = simxGetJointPosition(clientID, elbowJoint, &teta3,
simx_opmode_buffer);
	error = simxGetJointPosition(clientID, wristJoint_0, &teta4,
simx_opmode_buffer);
	error = simxGetJointPosition(clientID, wristJoint_1, &teta5,
simx_opmode_buffer);
	printf("Sono in teta1: %f\n", teta1*180/M_PI);
	printf("Sono in teta2: %f\n", teta2*180/M_PI);
	printf("Sono in teta3: %f\n", teta3*180/M_PI);
	printf("Sono in teta4: %f\n", teta4*180/M_PI);
	printf("Sono in teta5: %f\n", teta5*180/M_PI);


	invKinResult ikr = invKin(p, true);
	joints.clear();
	joints.push_back(ikr.teta1);
	joints.push_back(ikr.teta2);
	joints.push_back(ikr.teta3);

	angle_base = joints.at(0)/180*M_PI;
	angle_shoulder = joints.at(1)/180*M_PI;
	angle_elbow = joints.at(2)/180*M_PI;
	angle_wrist_0 = 0.f;
	angle_wrist_1 = 0.f;

	setSpeedForce(this->speed, this->force);

	error = simxSetJointTargetPosition(clientID, baseJoint, angle_base,
simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, shoulderJoint,
angle_shoulder, simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, elbowJoint, angle_elbow,
simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, wristJoint_0,
angle_wrist_0, simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, wristJoint_1,
angle_wrist_1, simx_opmode_oneshot);
	printf("Vado in teta1: %f\n", angle_base*180/M_PI);
	printf("Vado in teta2: %f\n", angle_shoulder*180/M_PI);
	printf("Vado in teta3: %f\n", angle_elbow*180/M_PI);


	while(!((teta1<=angle_base+delta)&&(teta1>=angle_base-
delta)&&(teta2<=angle_shoulder+delta)&&(teta2>=angle_shoulder-
delta)&&(teta3<=angle_elbow+delta)&&(teta3>=angle_elbow-delta)))
```

```
	{
		Sleep(2000);
		error = simxGetJointPosition(clientID, baseJoint, &teta1,
simx_opmode_buffer);
		error = simxGetJointPosition(clientID, shoulderJoint, &teta2,
simx_opmode_buffer);
		error = simxGetJointPosition(clientID, elbowJoint, &teta3,
simx_opmode_buffer);
		printf("Sono in teta1: %f e devo andare in %f\n",
teta1*180/M_PI, angle_base*180/M_PI);
		printf("Sono in teta2: %f e devo andare in %f\n",
teta2*180/M_PI, angle_shoulder*180/M_PI);
		printf("Sono in teta3: %f e devo andare in %f\n",
teta3*180/M_PI, angle_elbow*180/M_PI);
	}
}


/* Funzione che gestisce il movimento del braccio dati dei punti di via e un
tipo di approccio */
void Arm::followPath(vector<Point> p, int tipoApproccio)
{
		printf("FUNCTION: followPath\n");
		Emg emgCtr = Emg();

		for(int i = 0; i <= p.size()-1; i++)
		{
			if(!emgCtr.getStop()){
				if(i < p.size()-1){
					goToPoint(p[i]);
				}
				else{
					openHand();
					Sleep(2000);
					goToPoint(p[i]);
					closeHand();
				}
			}
			else{
				setSpeedForce(0, force);
				printf("\n");
				return;
			}
		}
		printf("\n");
		return;
}

/* Funzione che gestisce un movimento casuale nel caso non si abbiano punti
di via */
void Arm::move(int J, float degree)
{
	printf("FUNCTION: move\n");

	setSpeedForce(this->speed, this->force);

	int joint = 0;
	float teta = 0;

	error = simxGetJointPosition(clientID, baseJoint, &teta1,
simx_opmode_buffer);
```

```
	error = simxGetJointPosition(clientID, shoulderJoint, &teta2,
simx_opmode_buffer);
	error = simxGetJointPosition(clientID, elbowJoint, &teta3,
simx_opmode_buffer);

	printf("Sono in teta1: %f\n", teta1*180/M_PI);
	printf("Sono in teta2: %f\n", teta2*180/M_PI);
	printf("Sono in teta3: %f\n", teta3*180/M_PI);

	switch(J){
		case 0:
			joint = baseJoint;
			teta = teta1;
			break;
		case 1:
			joint = shoulderJoint;
			teta = teta2;
			break;
		case 2:
			joint = elbowJoint;
			teta = teta3;
			break;
		case 3:
			joint = wristJoint_0;
			teta = teta4;
			break;
		case 4:
			joint = wristJoint_1;
			teta = teta4;
			break;
		default:
			return;
	}
	printf("Muovo joint %d di %f rad\n\n", joint, teta*180/M_PI);
	error = simxSetJointTargetPosition(clientID, joint,
teta+(degree/180.f*M_PI), simx_opmode_oneshot);
	printf("devo andare in %f gradi\n\n",
(teta+(degree/180.f*M_PI))*180/M_PI);
	float teta_arr = teta+(degree/180.f*M_PI);
	while(!((teta<=teta_arr+delta)&&(teta>=teta_arr-delta)))
		{
			Sleep(2000);
			error = simxGetJointPosition(clientID, joint, &teta,
simx_opmode_buffer);
			printf("Sono in teta: %f e devo andare in %f\n",
teta*180/M_PI, teta_arr*180/M_PI);
	}
}

/* Funzione che setta la forza e la velocità a cui il braccio si deve
muovere */
void Arm::setSpeedForce(float speed, float force){

	error = simxSetJointTargetVelocity(clientID, baseJoint, speed,
simx_opmode_oneshot);
	error = simxSetJointTargetVelocity(clientID, shoulderJoint, speed,
simx_opmode_oneshot);
	error = simxSetJointTargetVelocity(clientID, elbowJoint, speed,
simx_opmode_oneshot);
	error = simxSetJointTargetVelocity(clientID, wristJoint_0, speed,
simx_opmode_oneshot);
```

```
	error = simxSetJointTargetVelocity(clientID, wristJoint_1, speed,
simx_opmode_oneshot);
	error = simxSetJointTargetVelocity(clientID, right_Hand_Joint, speed,
simx_opmode_oneshot);
	error = simxSetJointTargetVelocity(clientID, left_Hand_Joint, speed,
simx_opmode_oneshot);

	error = simxSetJointForce(clientID, baseJoint, force,
simx_opmode_oneshot);
	error = simxSetJointForce(clientID, shoulderJoint, force,
simx_opmode_oneshot);
	error = simxSetJointForce(clientID, elbowJoint, force,
simx_opmode_oneshot);
	error = simxSetJointForce(clientID, wristJoint_0, force,
simx_opmode_oneshot);
	error = simxSetJointForce(clientID, wristJoint_1, force,
simx_opmode_oneshot);
	error = simxSetJointForce(clientID, right_Hand_Joint, force,
simx_opmode_oneshot);
	error = simxSetJointForce(clientID, left_Hand_Joint, force,
simx_opmode_oneshot);
}

/* Funzione che gestisce l'apertura della mano */
void Arm::openHand(){
	printf("apro la mani\n");
	error = simxSetJointTargetPosition(clientID, left_Hand_Joint,
45*M_PI/180 , simx_opmode_oneshot);
	error = simxSetJointTargetPosition(clientID, right_Hand_Joint,
45*M_PI/180, simx_opmode_oneshot);

	error = simxGetJointPosition(clientID, left_Hand_Joint, &tetaLeft,
simx_opmode_streaming);
	error = simxGetJointPosition(clientID, right_Hand_Joint, &tetaRight,
simx_opmode_streaming);
}

/* Funzione che gestisce la chiusura della mano */
void Arm::closeHand(){
	printf("Chiudo la mano\n");
	char detection;
	int conta=1;

	simxGetObjectHandle(clientID, "SensorHand", &proximity_Sensor,
simx_opmode_oneshot_wait);

	error = simxGetJointPosition(clientID, left_Hand_Joint, &tetaLeft,
simx_opmode_buffer);
	error = simxGetJointPosition(clientID, right_Hand_Joint, &tetaRight,
simx_opmode_buffer);

	simxReadProximitySensor(clientID, proximity_Sensor,
&detection,NULL,NULL,NULL,simx_opmode_streaming);
	printf("HAND: leftAngle: %f, rightAngle: %f\n", tetaLeft*M_PI/180,
tetaRight*M_PI/180);
	//while(!detection){
	Funzionamento non corretto del sensore
	while(tetaLeft > 0.001f && tetaRight > 0.001){
		if(conta < 3){
		error = simxSetJointTargetPosition(clientID, left_Hand_Joint,
tetaLeft - 5*M_PI/180 , simx_opmode_oneshot);
```

```
                error = simxSetJointTargetPosition(clientID, right_Hand_Joint,
tetaRight - 5*M_PI/180, simx_opmode_oneshot);
                Sleep(1000);
                error = simxGetJointPosition(clientID, left_Hand_Joint,
&tetaLeft, simx_opmode_buffer);
                error = simxGetJointPosition(clientID, right_Hand_Joint,
&tetaRight, simx_opmode_buffer);
                printf("HAND: leftAngle: %f, rightAngle: %f\n",
tetaLeft*M_PI/180, tetaRight*M_PI/180);

                simxReadProximitySensor(clientID, proximity_Sensor,
&detection,NULL,NULL,NULL,simx_opmode_buffer);
                printf("detection: %d\n", detection);
                conta++;
                }
                else{break;}
        }
}
```

Command.h

```
/* Classe Command indica i tipi di comandi che un segnale EMG può produrre
*/

#pragma once
class Command
{
public:
        enum mov_type {FLT_EST, ABDUCTION, NOTHING};

private:
                mov_type movement;
                int width;
                int speed;

public:
        Command();
        Command(mov_type movement, int width, int speed);
        ~Command(void);
        mov_type Command::getMovementType();
        void setCommand(mov_type movement, int width, int speed);
        int getWidth();
        int getSpeed();
};
```

Command.cpp

```
#include "Command.h"

Command::Command()
{
}

Command::Command(mov_type movement, int width, int speed)
{
```

```
	this->movement = movement;
	this->width = width;
	this->speed = speed;
}


Command::~Command(void)
{
}

void Command::setCommand(mov_type movement, int width, int speed)
{
	this->movement = movement;
	this->width = width;
	this->speed = speed;
}


Command::mov_type Command::getMovementType()
{
	return this->movement;
}

int Command::getWidth()
{
	return this->width;
}

int Command::getSpeed()
{
	return this->speed;
}
```

EMG.h

```
/* Classe EMG gestisce i segnali EMG */

#include "Command.h"
#include <stdlib.h>
#include <time.h>
#include <stdio.h>

#pragma once

class Emg
{
public:
	Emg(void);
	~Emg(void);
	Command getRandomEmgSignal();
	Command getRealEmgSignal();
	bool Emg::getStop();
	bool Emg::getRealStop();
};
```

EMG.cpp

```
#include "Emg.h"

Emg::Emg(void)
{
}

Emg::~Emg(void)
{
}

/* Questa funzione produce un segnale EMG casuale di flessione o abduzione
*/
Command Emg::getRandomEmgSignal()
{
	srand(time(NULL));
	Command cmd = Command();
	printf("Sto producendo un segnale EMG: ");
	int type = rand()%2;
	switch(type){
		case 0:
			cmd.setCommand(Command::FLT_EST, (rand()%180+1),
rand()%10+2);
			printf("Comando FLT_EST\n");
			break;
		case 1:
			cmd.setCommand(Command::ABDUCTION, (rand()%90+1),
rand()%10+2);
			printf("Comando ABDUCTION\n");
			break;
		default:
			cmd.setCommand(Command::NOTHING, 0, 0);
			printf("Comando NOTHING-DEFAULT\n");
	}
	printf("Il comando e' di tipo: %d, velocità: %d, ampiezza: %d \n\n",
cmd.getMovementType(), cmd.getSpeed(), cmd.getWidth());
	return cmd;
}
/* Questa funzione genera un segnale di STOP casuale gestito nella
FollowPath */
bool Emg::getStop(){
	srand(time(NULL));
	int type = rand()%9;
	if (type == 9){
		printf("Sto producendo un segnale EMG di STOP\n\n");
		return true;
	}
	else {
		return false;
	}
}
```

ArmClient.cpp

```
/*Classe principale per gestire il braccio con V-REP

 Una volta compilato controllare che sia presente armClient.exe in V-
REP_PRO_EDU/
I modelli di test devono essere presenti in V-REP_PRO_EDU/ se usati

Nel caso si abbia il Kinect e si voglia testare il sistema togliere i
commenti dalle parti segnate e commentare il resto      */


#include "Classifier\OpenNIScreenCapturer.h"
#include "Arm.h"
//#include "KinectDemo.h"   Se si usa il Kinect o i modelli già creati non
serve
#include "KinematicModel\KinematicModelTypes.h"

using namespace std;

vector<Point> classifier(int approach, OpenNIScreenCapturer* o);
invKinResult invKin(Point p, bool recursive);

void task(OpenNIScreenCapturer* o){
	o->run();
}


int main(int argc,char* argv[]){

	int portNb = 0;		// Porta connessione V-REP
	int clientID;		// ID client V-REP
	int baseJoint;		// Gestore base joint
	int shoulderJoint;	// Gestore shoulder joint
	int elbowJoint;		// Gestore elbow joint
	int wristJoint_0;	// Gestore wrist joint_0
	int wristJoint_1;	// Gestore wrist joint_1
	int right_Hand_Joint;	// Gestore right hand joint
	int left_Hand_Joint;	// Gestore left hand joint
	int proximity_Sensor;	// Gestore del sensore di prossimità

	if (argc>=10){
		portNb=atoi(argv[1]);
		baseJoint=atoi(argv[2]);
		shoulderJoint=atoi(argv[3]);
		elbowJoint=atoi(argv[4]);
		wristJoint_0=atoi(argv[5]);
		wristJoint_1=atoi(argv[6]);
		right_Hand_Joint=atoi(argv[7]);
		left_Hand_Joint=atoi(argv[8]);
		proximity_Sensor = atoi(argv[9]);
	}
	else{
		printf("Indica i seguenti argomenti: 'portNumber baseJoint
shoulderJoint elbowJoint wristJoint_0 wristJoint_1 right_hand_Joint
left_hand_Joint proximity_Sensor'!\n");
		extApi_sleepMs(5000);
		return 0;
	}
```

```
	/* Parte riservata all'avvio del Kinect, da utilizzare se Kinect
collegato al pc
		OpenNIScreenCapturer o;
		boost::thread t = boost::thread(task, &o);
		std::cout << "Initializing Screen Capturer" << std::endl;
		//Sleep(5000);
		std::cout << "Screen Capturer Initialized" << std::endl;
	*/

	clientID = simxStart("127.0.0.1",portNb,true,true,2000,5);
	if (clientID != -1) {

		Emg emgCtr = Emg();
		Arm armCtr = Arm(clientID, baseJoint, shoulderJoint,
elbowJoint, wristJoint_0, wristJoint_1, right_Hand_Joint, left_Hand_Joint,
proximity_Sensor);
		Command cmd;
		vector<Point> path;
		path.push_back(Point(0,0,0));
		armCtr.goToRestPosition(path);
		path.clear();
		Sleep(3000);

		while (simxGetConnectionId(clientID)!=-1){

			/* Simulazione */

			cmd = emgCtr.getRandomEmgSignal();
			if(cmd.getMovementType() == Command::FLT_EST){
				path = classifier(0, NULL);
// Se Kinect usare path = classifier(0, o);
				if(!path.empty()){
					armCtr.followPath(path, 0);
				}
				else{
					armCtr.move(0, cmd.getWidth());
				}
			}
			else if(cmd.getMovementType() == Command::ABDUCTION){
				path = classifier(1, NULL);
	// Se Kinect usare path = classifier(1, o);
				if(!path.empty()){
					armCtr.followPath(path, 1);
				}
				else{
					armCtr.move(1, cmd.getWidth());
				}
			}
			armCtr.goToRestPosition(path);
			Sleep(5000);
		}
		simxFinish(clientID);
	}
	return(0);
}
```

KinematicsModels.h

```
#ifndef       KINEMATIC_MODEL_TYPES_H
#define KINEMATIC_MODEL_TYPES_H
#include "Point.h"

typedef struct {

	bool reachable;

	double teta1;
	double teta2;
	double teta3;


} invKinResult;

typedef struct {

	Point shoulder;
	Point elbow;
	Point wrist;
	Point handCenter;
	Point graspingPoint1;
	Point graspingPoint2;
	Point fingerPoint1;
	Point fingerPoint2;


} dirKinResult;

#endif
```

Point.h

```
#pragma once
class Point
{
private:
	double x, y, z;

public:
	enum coordinate {X, Y, Z};

	Point(void);
	Point(double x, double y, double z);
	~Point(void);
	double distance(void);
	double distance(Point p);
	double getValue(coordinate c);
	void setX(double x);
	void setY(double y);
	void setZ(double z);
	void scaling(double scaling);
	void cwToStandard();
	void standardToCw();
};
```

Point.cpp

```
#include "Point.h"
#include <math.h>


Point::Point(void)
{
}

Point::Point(double x, double y, double z)
{
	this->x = x;
	this->y = y;
	this->z = z;
}

Point::~Point(void)
{
}

double Point::distance(void)
{
	return sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
}

double Point::distance(Point p)
{
	return sqrt(pow(this->x - p.getValue(Point::X), 2) + pow(this->y -
p.getValue(Point::Y), 2) + pow(this->z - p.getValue(Point::Z), 2));
}

double Point::getValue(coordinate c)
{
	if(c == X) return this->x;
	else if(c == Y) return this->y;
	else if(c == Z) return this->z;
	else return -1;
}

void Point::setX(double x){
	this->x = x;
}

void Point::setY(double y){
	this->y = y;
}

void Point::setZ(double z){
	this->z = z;
}

void Point::scaling(double scaling){
	this->x = this->x * scaling;
	this->y = this->y * scaling;
	this->z = this->z * scaling;
}

void Point::cwToStandard(){
	double aux;
```

```
        aux = this->y;
        this->y = this->z;
        this->z = -aux;
}

void Point::standardToCw(){
        double aux;
        aux = this->y;
        this->y = -this->z;
        this->z = aux;
}
```

InverseKinematics.cpp

```
#include <iostream>
#include <Eigen/Dense>
#include <math.h>

#include "ArmDimensions.h"
#include "KinematicModelTypes.h"

dirKinResult dirKin(double teta1, double teta2, double teta3, double teta4);

invKinResult invKin(Point p);

invKinResult invKin(Point p, bool recursive){

        double x = p.getValue(Point::X);
        double y = p.getValue(Point::Y);
        double z = p.getValue(Point::Z);


        invKinResult res;

        res.reachable = false;

        double d0 = SHOULDERZ;
        double d1 = SHOULDERX;
        double a0 = SHOULDERY;
        double a2 = FIRST_LINK_LENGTH;
        double a3 = SECOND_LINK_LENGTH + HAND_HEIGHT + HAND_CENTER_OFFSET;

        double w = sqrt( (x-d1)*(x-d1) + (y-a0)*(y-a0) + (z-d0)*(z-d0) );


        double ctetak = -(a3*a3 - (w*w) - (a2*a2)) / (2.0*w*a2);
        double stetak = sqrt(1 - (ctetak*ctetak));
        double tetak = atan2(stetak, ctetak);
        double tetakd = tetak*180.0/M_PI;


        double cteta3 = ((w*ctetak)-a2)/a3;
        double steta3 = sqrt(1 - (cteta3*cteta3));
        double teta3 = atan2(steta3, cteta3);
        double teta3d = teta3*180/M_PI;
```

```
	double steta2;
	double cteta2;
	if( w > (a2+a3) ){

		steta2 = x-d1;
		cteta2 = sqrt( (z-d0)*(z-d0) + y*y );

	}
	else{

		steta2 = (x-d1)/(a2+(a3*cteta3));
		cteta2 = sqrt( 1 - steta2*steta2);

	}

	double teta2 = atan2(steta2, cteta2);
	double teta2d = teta2*180.0/M_PI;


	double ctetaw = y;
	double stetaw = (z-d0);
	double tetaw = atan2(stetaw, ctetaw);
	double k = w*(1-cos(teta2))*(ctetaw/w);
	double steta0 = k/w;
	double cteta0 = sqrt(1-(steta0*steta0));


	double teta0 = atan2(steta0, cteta0);
	if( (z-d0)>=0 )
		teta0 = -teta0;

	double teta0s = teta0*180/M_PI;
	stetaw = z-d0;

	tetaw = atan2(stetaw, ctetaw) + M_PI/2;
	double tetawd = tetaw*180/M_PI;



	double teta1 = tetaw - tetak - teta0;
	double teta1d = teta1*180/M_PI;

	//Vincoli (per ora quelli del corpo umano)
	if(teta1d >= -30 && teta1d <= 180   &&   teta2d >= -3 && teta2d <=
180   &&   teta3d >= 0 && teta3d <= 130   && w <= a2+a3){

		res.teta1 = teta1d;
		res.teta2 = teta2d;
		res.teta3 = teta3d;


		res.reachable = true;

	}
	else{


		if(recursive){

			p.setX(p.getValue(Point::X) + 0.002);
			if(res.reachable == false)
```

```
                        res = invKin(p, false);


                    p.setX(p.getValue(Point::X) - 0.004);
                    if(res.reachable == false)
                        res = invKin(p, false);


                    p.setX(p.getValue(Point::X) + 0.002);
                    p.setY(p.getValue(Point::Y) + 0.002);
                    if(res.reachable == false)
                        res = invKin(p, false);


                    p.setY(p.getValue(Point::X) - 0.004);
                    if(res.reachable == false)
                        res = invKin(p, false);


                    p.setY(p.getValue(Point::Y) + 0.002);
                    p.setZ(p.getValue(Point::Z) + 0.002);
                    if(res.reachable == false)
                        res = invKin(p, false);


                    p.setZ(p.getValue(Point::Z) - 0.004);
                    if(res.reachable == false)
                        res = invKin(p, false);

            }
        }


        if(res.reachable == false && recursive == true)
                    std::cout << "\nINVKIN: Grasping point (" << x << ", "
<< y << ", " << z << ") unreachable!" << std::endl;

        return res;
}



invKinResult invKin(Point p){

        return invKin(p, true);

}
```

Appendice B

Metodi EMG

In questo appendice verranno approfonditi i metodi utilizzati per l'analisi dei segnali EMG, tratti dal lavoro di tesi di Rivela e Scanella [38]. In particolare, verranno presentati il metodo per la riduzione Principal Component Analysis (PCA) e quello per la classificazione Linear Discriminant Analysis (LDA).

Principal Component Analysis (PCA)

L'analisi delle componenti principali è un metodo di estrazione di feature basata sul criterio dell'errore quadratico medio. È una tecnica non supervisionata e non è quindi espressamente finalizzata ad ottimizzare l'accuratezza in un problema di classificazione ma è invece un approccio classico alla riduzione della dimensionalità. Essa esprime un dato segnale come somma di componenti ortonormali e scorrelate ed è applicabile sia a vettori aleatori (dimensione finita) che a processi aleatori; questo metodo statistico identifica le proiezioni lineari delle feature che corrispondono alle variazioni principali nei dati.

Sia $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$ un data set di n campioni, modellizzati come realizzazioni indipendenti di un vettore aleatorio $\boldsymbol{x}$. Se si assume che le n feature che compongono tale vettore siano combinazioni lineari di m sorgenti $s_1, s_2, \dots, s_m$ $(m \leq n)$, supposte indipendenti (e quindi anche scorrelate), si ha un problema di *Blind Source Separation*. La PCA sfrutta l'informazione di scorrelazione tra le sorgenti e genera un insieme di, al massimo, n feature scorrelate $\mathrm{y}_1, \mathrm{y}_2, \dots, \mathrm{y}_\mathrm{n}$, come stime delle sorgenti, a partire dalle n feature generiche $x_1, x_2, \dots, x_n$.

La matrice di covarianza nello spazio trasformato è diagonale:

$$COV\{\mathbf{y}\} = E\{(\mathbf{y} - E\{\mathbf{y}\})(\mathbf{y} - E\{\mathbf{y}\})^t\} = \\ = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$$

In particolare, l'elemento diagonale i-esimo λ_i della covarianza nello spazio trasformato è la varianza della feature y_i: $VAR\{y_i\} = \lambda_i$, per $i = \{1,2,\dots,n\}$.
La decorrelazione delle feature date si ottiene mediante una trasformazione lineare definita da una matrice di dimensioni $n \times n, T$: $\mathbf{y} = T\boldsymbol{x}$.

La matrice T si ottiene diagonalizzando la matrice di covarianza di $\boldsymbol{x}$: $S = COV\{\boldsymbol{x}\}$.
S è una matrice $n \times n$ simmetrica, semidefinita positiva, ed ammette quindi n autovalori $\lambda_1, \lambda_2, \dots, \lambda_n \geq 0$ ed n autovettori ortonormali $\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_n$. La matrice T adottata dalla PCA è, per definizione, data dalla giustapposizione per righe degli autovettori della matrice di covarianza, ed è quindi una matrice ortogonale :

$$T = \begin{bmatrix} e_1^t \\ \vdots \\ e_n^t \end{bmatrix}$$

Le variabili aleatorie $y_1, y_2, \dots, y_n$ (ossia le proiezioni di $\boldsymbol{x}$ lungo gli assi $\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_n$) si dicono componenti principali. Generalmente, si ordinano gli n autovalori di S in ordine decrescente ($\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$). Quindi, l'asse $\boldsymbol{e}_1$ è la direzione lungo cui si ha la massima dispersione ed $\boldsymbol{e}_n$ è la direzione lungo cui la dispersione è più bassa. La PCA fornisce quindi intrinsecamente un ordinamento delle sorgenti estratte. Se si assume che il potere informativo di una feature sia legato alla sua varianza e se gli ultimi autovalori sono molto piccoli, le feature trasformate corrispondenti si possono considerare poco significative.
La matrice di covarianza S del vettore aleatorio $\boldsymbol{x}$ è solitamente sconosciuta e va stimata a partire dal dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$. Si assume che $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$ sia un insieme di n vettori aleatori indipendenti ed identicamente distribuiti con distribuzione uguale a quella di $\boldsymbol{x}$. Sotto tali ipotesi, una stima non polarizzata e consistente della matrice di covarianza S è data dalla covarianza-campione: $\hat{S} = \frac{1}{n-1}\sum_{k=1}^{n}(\boldsymbol{x}_k - \hat{\mu})(\boldsymbol{x}_k - \hat{\mu})^t$, dove $\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n}\boldsymbol{x}_k$ è la media-campione, stima non polarizzata e consistente della media $\mu = E\{\boldsymbol{x}\}$ del vettore aleatorio $\boldsymbol{x}$.
Operativamente, la PCA consiste nei seguenti step:

1. Stima della media m e della matrice di covarianza S in termini di media-campione e covarianza-campione del data set;
2. Calcolo degli autovalori della covarianza-campione ed i corrispondenti autovettori;
3. Ordinamento degli autovalori in ordine decrescente;
4. Costruzione della matrice di trasformazione T giustapponendo per righe gli autovettori.

La PCA rappresenta la riduzione lineare ottima delle feature rispetto al criterio dell'errore quadratico medio.

Linear Discriminant Analysis (LDA)

L'LDA è un metodo supervisionato che calcola la trasformazione ottima che massimizza il rapporto della distanza tra le classi e la distanza entro la classe, ottenendo la massima discriminazione. Nella discriminant analysis la separazione tra le classi è enfatizzata sostituendo la matrice di covarianza totale della PCA con una misura di separabilità generale, come il criterio di Fisher, il che si traduce nel trovare gli autovettori di $S_w^{-1}S_b$ (il prodotto dell'inverso della scatter matrix dentro la classe, S_w^{-1}, e la scatter matrix tra le classi, S_b).

Data una matrice di dati $A \in \mathbb{R}^{m\times n}$, la *discriminat analysis* classica calcola una trasformazione lineare $G \in \mathbb{R}^{m\times l}$ che mappa ogni colonna a_i di A nello spazio m-dimensionale, in un vettore y_i nello spazio l-dimensionale: $G: a_i \in \mathbb{R}^m \rightarrow y_i = G^T a_i \in \mathbb{R}^l (l < m)$. L'obiettivo dell'LDA è di trovare la trasformazione G in modo che la struttura dello spazio originale ad alta dimensione sia mantenuta nello spazio a dimensione ridotta.

Sia la matrice di dati A partizionata in k classi $A = [A_1, A_2, .., A_k]$, dove $A_i \in \mathbb{R}^{m\times n_i}$, e $\sum_{i=1}^{k} n_i = n$, nell' LDA , vengono definite tre matrici di scatter chiamate *within-class* (S_w), *between-class* (S_b) e *total* (S_t) *scatter matrix*, definite come:

$$S_w = \frac{1}{n}\sum_{i=1}^{k}\sum_{x\epsilon A_i}\left(x - c^{(i)}\right)\left(x - c^{(i)}\right)^T,$$

$$S_b = \frac{1}{n}\sum_{i=1}^{k}\sum_{x\epsilon A_i}\left(c^{(i)} - c\right)\left(c^{(i)} - c\right)^T = \frac{1}{n}\sum_{i=1}^{k} n_i\left(c^{(i)} - c\right)\left(c^{(i)} - c\right)^T,$$

$$S_t = \frac{1}{n}\sum_{j=1}^{n}\left(a_j - c\right)\left(a_j - c\right)^T,$$

dove il centroide $c^{(i)}$ dell' i-esima classe è definito come $c^{(i)} = \frac{1}{n_i}A_i e^{(i)}$ con $e^{(i)} = (1,1,\dots 1)^T \in \mathbb{R}^{n_i}$, e il centroide globale c è definito come $c = \frac{1}{n}Ae$, con $e = (1,1,\dots,1)^T \in \mathbb{R}^n$. È facilmente verificabile che $S_t = S_b + S_w$.

Definendo le matrici H_w, H_b, H_t:

$$H_w = \frac{1}{\sqrt{n}}[A_1 - c^{(1)}(e^{(1)})^T, \dots, A_k - c^{(k)}(e^{(k)})^T],$$

$$H_b = \frac{1}{\sqrt{n}}[\sqrt{n_1}(c^{(1)} - c), \dots, \sqrt{n_k}(c^{(k)} - c)],$$

$$H_t = \frac{1}{\sqrt{n}}(A - ce^T).$$

Quindi $S_t, S_b,$ e S_w possono essere espresse come: $S_w = H_w {H_w}^T$, $S_b = H_b {H_b}^T$, $S_t = H_t {H_t}^T$. Le tracce delle due matrici di scatter S_w ed S_b possono essere calcolate come segue:

$$trace(S_w) = \frac{1}{n} \sum_{i=1}^{k} \sum_{x \epsilon A_i} \left(x - c^{(i)}\right)^T \left(x - c^{(i)}\right) = \frac{1}{n} \sum_{i=1}^{k} \sum_{x \epsilon A_i} \left\| \left(x - c^{(i)}\right) \right\|^2 ,$$

$$trace(S_b) = \frac{1}{n} \sum_{i=1}^{k} n_i \left(c^{(i)} - c\right)^T \left(c^{(i)} - c\right) = \frac{1}{n} \sum_{i=1}^{k} n_i \left\| \left(c^{(i)} - c\right) \right\|^2 .$$

Quindi, $trace(S_w)$ misura la coesione within-class, mentre $trace(S_b)$ misura la separazione between-class.

Nello spazio a dimensione ridotta risultante dalla trasformazione lineare G, le matrici di scatter diventano: $S_w^L = G^T S_w G$; $S_b^L = G^T S_b G$; $S_t^L = G^T S_t G$.

Una trasformazione ottima G massimizza $trace(S_b^L)$ e minimizza $trace(S_w^L)$ simultaneamente, il che è equivalente a massimizzare $trace(S_b^L)$ e minimizzare $trace(S_t^L)$ simultaneamente, poiché $S_t^L = S_w^L + S_b^L$. Un'ottimizzazione comune nella discriminat analysis classica è

$$G = argmax_G \{trace(S_t^L)^{-1}(S_b^L)\}. \qquad [\S]$$

La soluzione può essere ottenuta applicando una decomposizione in autovettori della matrice $S_t^{-1} S_b$, se S_t è non singolare. Ci sono al massimo $k - 1$ autovettori corrispondenti ad autovalori diversi da zero, poiché il rango della matrice S_b è limitato superiormente da $k - 1$. Quindi, la dimensione ridotta dall'LDA classica è, al massimo, $k - 1$. Un modo stabile per risolvere questo problema di decomposizione è applicare la *Singular Value Decomposition* (SVD) sulle matrici di scatter.

Una limitazione di questo approccio è la necessità di avere una matrice di scatter totale, S_t, non singolare, il che può non essere ottenuto in casi di sottocampionamento; questo è noto come problema del sottocampionamento o problema di singolarità.