UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA

—

DIPARTIMENTO DI INNOVAZIONE MECCANICA E GESTIONALE

—

LAUREA TRIENNALE IN INGEGNERIA BIOMEDICA

UTILIZZO DEL FEEDBACK MULTISENSORIALE IN NEURORIABILITAZIONE

RELATORE: CH.MO PROF. ING. ALDO ROSSI
CORRELATORE: CH.MO PROF. ING. GIULIO ROSATI

LAUREANDO: BENEDETTI GIULIO

ANNO ACCADEMICO 2009-2010

ai miei nonni Carlo e Ulisse

Indice

Sommario

Il mio lavoro è stato essenzialmente di tipo informatico e volto al'implementazione di un software di elaborazione dei dati per lo sviluppo futuro di nuova strumentazione. Questo software permette la riproduzione dei test effettuati con l'esoscheletro Pneu-WREX ed il contemporaneo invio di dati ad applicazione esterna via UDP. In secondo luogo mi sono occupato della realizzazione di un modulo in Simulink che permetta l'invio di dati in real time sempre attraverso attraverso porta UDP ad un'applicazione esterna.

Introduzione

Contesto

L'ictus è la terza causa di morte dopo le malattie cardiovascolari ed il cancro nel mondo occidentale e rappresenta la maggiore causa di disabilità articolare e motoria nei paesi più industrializzati [1]. Ogni anno negli Stati Uniti ed in Europa vi sono tra i 200 ed i 300 casi di ictus ogni 100000 abitanti. Approssimativamente il 30% dei soggetti che hanno contratto un ictus sopravvivono con gravi disabilità articolari e profonde limitazioni nello svolgimento delle normali attività quotidiane a causa di un difficile controllo della mobilità e di una perdita di abilità nell'uso delle braccia [1, 2]. A causa dell'aumento dell'età della popolazione nei paesi industrializzati, questa tendenza è destinata a salire nei prossimi decenni [2] ed il miglioramento delle attività riabilitazione motoria post ictus sta diventando uno degli obiettivi primari in ambito non solo medico.

Ad oggi i pazienti sono sottoposti, nella maggior parte dei casi, ad una terapia *hands-on,* ovvero a diretto contato con il terapista, per molti mesi per contrastare emiparesi e farli tornare quanto più possibile ad una vita indipendente. Risulta certamente incoraggiante per studi presenti e futuri la relazione ormai consolidata che si può riscontrare tra il numero di sedute a cui è sottoposto un paziente e l'aumento delle prestazioni raggiunte [3], [4], [5], [6], [7], anche se la quantità e la qualità delle terapie a cui un paziente si può sottoporre sono spesso limitate dai costi che queste comportano [8, 9, 10]. I pazienti potrebbero esercitarsi indipendentemente senza l'intervento del terapista, ma le limitate capacità motorie a cui sono costretti, a causa dell'ictus, rendono gli esercizi domestici scarsamente utili al fine della riabilitazione. Lo sviluppo di nuove tecniche riabilitative a basso

costo potrebbe permettere ai pazienti di svolgere le loro attività fisioterapiche per periodi più lunghi traendone quindi sicuro giovamento senza, magari, doversi nemmeno spostare da casa.

In questo contesto si inseriscono le numerose ricerche svolte negli ultimi due decenni che hanno tentato di sviluppare dispositivi robotici per la riabilitazione di persone con vari tipi di disabilità articolare. La maggior parte di questi studi si è concentrata sul trattamento di pazienti sopravvissuti ad ictus in quanto questi costituiscono un ampio target della casistica di disamobilità articolare. Solitamente un dispositivo robotico, come può essere un esoscheletro, viene usato per assistere fisicamente il paziente nel completamento di alcuni movimenti degli arti, presentati solitamante su di uno schermo quai come fossero dei giochi. Sono state sviluppate nel corso degli anni numerose strategie di controllo assistito che vanno da robot che muovono gli arti lungo uno schema preimpostato, a robot che assistono il paziente solo se la sua prestazione non rientra in predefiniti limiti spaziali e/o temporali, a robot che adattano il supporto fornito e seconda della debolezza o meno del paziente.

Robotica riabilitativa

Negli ultimi vent'anni, come già ricordato precedentemente, molti gruppi di ricerca si sono impegnati nello studio di dispositivi robotici che permettessero al paziente uscito dall'ictus, o affetto da altre malattie neurologiche, di eseguire esercizi che non necessitassero dell'intervento diretto del terapista.

I sistemi robotici utilizzati per la riabilitazione si possono distinguere in due categorie principali: le macchine operazionali e gli esoscheletri. Un esempio di robot del primo tipo può essere il MIT-Maunus sviluppato da Krebs et al [11, 12, 13] di figura 1, in cui l'interzione tra paziente e macchina avviene solo a livello dell'end-effector. Per gli esoscheletri, macchine che possono controllare l'interazione con l'utente a livello dell'intera articolazione , per esempio del braccio, sono da ricordare il Pneu-WREX [14] e l'Arm-in visibili in figura 2.

Alla luce anche di quanto detto precedentemente possiamo individuare tre principi fondamentali per il recupero dell'attività motoria: tempestività dell'in-

Figura 1: MIT-Manus[11, 12, 13]

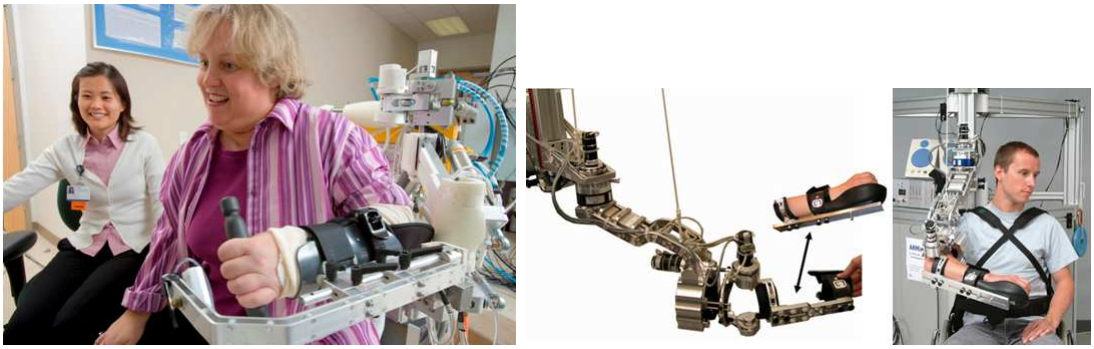


Figura 2: Pneu-WREX (sinistra)[14] e Arm-in (centro e destra)[15]

tervento, esercizi mirati e intensiva ripetizione dello sforzo [16]. Sotto questo punto di vista i dispositivi robotici forniscono ottime risposte in quanto possono automaticamente fornire al paziente ripetutamente l'esercizio da svolgere ed inoltre incentivare il soggetto invitandolo al raggiungimento di un obiettivo quasi si trattasse di un gioco da dover poratre a termine. La motivazione e la partecipazione del paziente diventano quindi altri fattori fondamentali.

In questo senso si sono sviluppate le ricerche negli ultimi anni, ovvero nel tentativo di realizzare una macchina che non svolgesse in maniera completamente autonoma l'esercizio proposto ma che incentivasse il più possibile la partecipazione del paziente.

Alcuni dispositivi, in particolare la protesi Lokomat (figura 3), per il miglioramento della camminata, ed il MIME (figura 3), per la riabilitazione degli arti superiori, utilizzano controlli relativamente rigidi per indirizzare il movimento di un arto o di un'estremità lungo una traiettoria desiderata. Tuttavia la rigida imposizione al robot di una determinata traiettoria da completare limita la

Figura 3: Lokomat (sinistra) [17] e MIME (destra) [18]

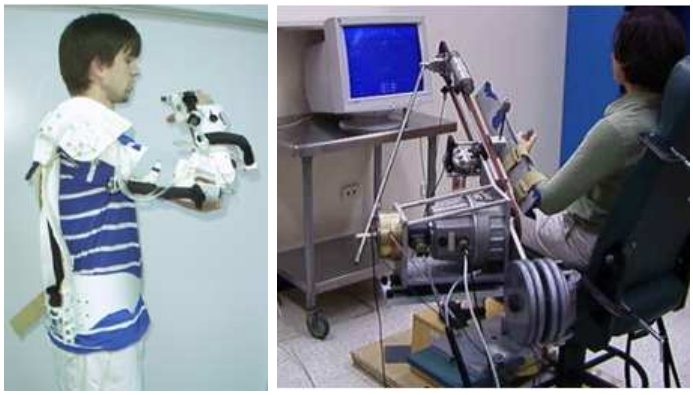


Figura 4: RUPERT (sinistra)[19] e ARM (destra) [20]

partecipazione e quindi lo sforzo da parte del paziente, facendo così perdere all'esercizio gran parte della sua utilità. In quest'ottica sono stati sviluppati, quindi, altri robot che seguono il principio "assistance-as-needed", ovvero dispositivi che entrano in funzione solamente dopo che il paziente ha provato autonomamente a compiere l'esercizio prestabilito. L'azionamento del robot può avvenire in modo automatico, o con l'intervento del terapista, dopo un certo lasso di tempo, oppure quando si nota una diminuzione delle prestazioni del paziente. Da ricordare in questo ambito i robot RUPERT (figura 4) e ARM (figura4).

Anche l'università di Padova si dimostra all'avanguardia nell'ambito della robotica riabilitativa con il NeReBot, visibile in figura 5, un robot a cavi per la riabilitazione degli arti superiori [21]. Tre cavi in nylon convertono il moto rotatorio di tre motori in c.c. in una traiettoria spaziale per l'arto del paziente [21]. Un software in real-time gestisce le fasi di acquisizione dei punti e ripetizione ciclica delle traiettorie spaziali ottenute per interpolazione dei punti acquisiti [21]. Il pri-

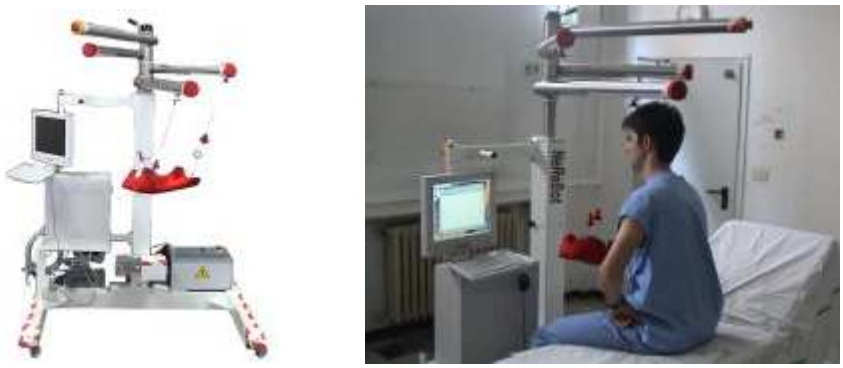


Figura 5: Il robot NeReBot sviluppato dal *dimeg* dell'università di Padova[21]

mo studio clinico su 30 pazienti ha dimostrato l'efficacia del robot nel trattamento riabilitativo post-ictus di pazienti in fase sub-acuta [21].

Reinkensmeyer ha ipotizzato che un dispositivo robotico che fornisce un qualche tipo di feedback interattivo può motivare maggiormente il paziente nello svolgimento degli esercizi assegnatigli [22]. A sostegno di questa ipotesi è stato mostrato come l'aggiunta di un qualche feedback visivo che misuri la partecipazione, come può essere la misura della forza applicata contro il robot, induca il paziente ad impegnarsi maggiormente nello svolgimento del task propostogli [23].

L'effettiva efficacia della robotica riabilitativa nei pazienti colpiti da ictus è, comunque, ancora oggi oggetto di discussione. Recenti letture del primo *Randomized Controlled Trials* (RCTs) sembrebbero mostrare che i pazienti che ricevono trattamenti fisioterapici robot-assistiti al braccio non hanno più probabilità rispetto agli altri di migliorare nello svolgimento delle loro normali attività quotidiane (*Activities of Daily Living, ADLs*), mentre potrebbero migliorare la funzione motoria e la forza dell'arto [24, 25]. In ogni caso questi risultati sono tutti da interpretare attentamente in quanto vi sono numerose differenze tra i vari test soprattutto per quanto riguarda la durata, il numero di prove, il tipo di trattamento e le caratteristiche del paziente.

Un campo che è stato ancora poco esplorato è quello del trattamento della fase più acuta della malattia [26]. Secondo un recente studio condotto in Europa nei centri di riabilitazione per pazienti affetti da ictus [27], durante la fase più acuta i pazienti spendono più del 72% del loro tempo in attività non terapeutiche, mentre questo, da un punto di vista plastico, sarebbe il momento migliore per la

riabilitazione [28]. Come riportato da Mehrholz et al [24] la pratica robot assistita nella fase più acuta , ovvero entro i primi tre mesi dall'attacco, è quella che produce i maggiori risultati dal punto di vista del recupero delle *ADLs* rispetto alla terapia nella fase successiva.

D'altra parte, come ricordato in precedenza, il continuo allenamento produce dei miglioramenti e quindi riscuote comunque un grande interesse lo sviluppo di sistemi che permettano al paziente di poter portare avanti una terapia personale e personalizzata per periodi più lunghi magari anche fuori dal'ospedale.

Un altro passo che permetterebbe di compiere una decisa evoluzione alla robotica riabilitativa consisterebbe nel trovare il modo di insegnare ai pazienti movimenti complessi che si avvicinano il più possibile a quelli della vita reale, in quanto i dispositivi attuali permettono perlopiù solo semplici movimenti schematici e ripetitivi.

Capitolo 1

Neuroriabilitazione e feedback multisensoriale

Come ipotizzato da Reinkensmeyer, l'aggiunta al dispositivo di un qualche tipo di feedback, come quello visivo, può fungere da stimolo per il paziente a portare a compimento l'esercizio richiesto nel modo più soddisfacente possibile. In quest'ottica sono allo studio e sono stati sviluppati alcuni sistemi di feedback tipo visivo e acustico per la riabilitazione in pazienti affetti da patologie neurologiche che comportano disfunzionalità motorie. Dal punto di vista visivo è già stato visto come ad esempio la misura della forza applicata dal paziente al robot può fungere da stimolo al soggetto per migliorare le proprie prestazioni [23].

Per quanto riguarda il suono sono già stati effettuati esperimenti che, grazie allo studio di immagini che fotografano l'attività cerebrale [29], hanno dimostrato un aumento dell'attività neurale in varie zone del cervello, un aumento dell'attenzione, della memoria e delle funzioni motorie in associazione all'ascolto di musica da parte del paziente [30, 31]. A supporto di questo vi è anche uno studio di Särkämö et al.[32] che suggerisce che l'ascolto di musica migliora il recupero delle facoltà cognitive e dell'umore nei pazienti post ictus.

Alcuni studi hanno dimostrato che un feedback uditivo ha prodotto dei miglioramenti nella distribuzione del peso nel sedersi e durante la camminata in pazienti affetti da malattie neurologiche come può essere il Parkinson [33], la sclerosi multipla [34] ed emiparesi dovute ad ictus [35]. In un *RCT* Schauer e Mauritz [36],

confrontando una terapia per il miglioramneto dell'andatura tradizionale con una che utilizza feedback audio, hanno mostrato che la presenza del feedback migliora la camminata in accordo con alcuni parametri come la velocità della camminata, la simmetria, la lunghezza del passo ed il movimento del piede. Questo perchè probabilmente il feedback audio diventa una sorta di segnale esterno che serve al paziente come riferimento per bilanciare il proprio movimento.

L'aumento delle prestazione dovuto all'introduzione di un feedback acustico è stato riscontrato anche in una serie di esperimenti svolti sull'esoscheletro Pneu-WREX. Al paziente veniva chiesto di seguire un target su di uno schermo e, contemporaneamente, di identificare un distrattore che compariva in un angolo dello schermo. Mentre la presenza del distrattore aumentava l'errore di posizione del paziente rispetto al target, l'introduzione di un feedback acustico che segnalava il superamento di un certo margine di errore, riportava la prestazione ai livelli precedenti l'introduzione del distrattore [37].

Un feedback di tipo acustico o visivo necessita, quindi, di una serie di sensori preposti alla registrazione di determinati aspetti dello stato del sistema, di una funzione di feedback che converta i segnali provenienti da suddetti sensori in parametri acustici o visivi e di un sistema di resa audio o video controllato dalla funzione di feedback.

Su queste basi, su quali siano i parametri di cui tener conto per la definizione di una efficace funzione di feedback e su il modo migliore per rendere visivamente e/o acusticamente i dati raccolti per aumentare la partecipazione del paziente si concentra, oggi, il lavoro di molti gruppi di ricerca.

1.1 Obiettivi

L'obiettivo finale del lavoro di ricerca, di cui il mio elaborato costituisce una piccola parte, è quello di modulare in maniera opportuna il feedback audio di cui dotare il robot Pneu-WREX. Si tratta infatti di trovare i giusti parametri su cui basare la funzione di feedback, come possono essere gli errori di velocità tra paziente e target, e di capire in che modo deve essere realizzato il feedback perchè sia il più utile possibile per il lavoro che deve svolgere il paziente. In questo

contesto si inserisce il mio lavoro, di stampo essenzialmente informatico, nel quale mi sono occupato della realizzazione di un software di elaborazione di dati già raccolti in test precedenti e di un modulo real time per l'invio di dati via UDP ad un'applicazione esterna responsabile della generazione del suono.

1.1.1 Software di elaborazione dati

Il primo obiettivo del mio elaborato è stato la creazione di un software di analisi dei dati. I dati in questione sono quelli relativi ai test effettuati sull'esoscheletro Pneu-WREX alla University of California su pazienti affetti ictus In particolare questo software deve portare a compimento simultaneamente due diverse operazioni. In primo luogo deve permettere la riproduzione dei test completa di target, end-effector, distrattore ed individuazione del distrattore; in secondo luogo deve inviare via UDP al software Pure Data le posizioni sui tre assi di end effector e target e le differenze di velocità tra questi due lungo i tre assi x, y e z. Utilizzando le differenze di velocità come base per la generazione di un qualche tipo di suono, si potrebbe quindi osservare la variazione dello stesso punto per punto. Questo può servire come base di test per la creazione di un feedback che utilizzi come parametro le differenze di velocità tra target e paziente e da inserire su un nuovo dispositivo robotico per la riabilitazione. In questo caso si è scelto di inviare al software preposto alla generazione del suono le differenze di velocità, ma il software è facilmente adattabile a tutti i tipi di dati raccolti nei test svolti con il Pneu-WREX.

1.1.2 Interfacciamento con il robot: modulo real time

La seconda parte del mio lavoro ha avuto come fine l'implementazione di un blocco Simulink da inserire nello schema dell'esoscheletro che permetta l'invio in real time, sempre via UDP, di dati di posizione o velocità al software desiderato. Il blocco, quindi, non verrebbe utilizzato per il trattamento di dati già acquisiti in altri test, come è per la GUI, ma servirebbe per inviare i dati presi in tempo reale dal paziente e da utilizzare per la generazione del suono a seconda del criterio scelto.

Capitolo 2

Sistema robotico

2.1 Esoscheletro Pneu-WREX

L'esoscheletro Pneu-Wrex è un'ortesi a quattro gradi di libertà azionata pneumaticamente, inizialmente progettata da Robert Sanchez. L'intento era quello di realizzare un dispositivo che potesse tranquillamente seguire il paziente in tutti i movimernti del braccio. In più il dispositivo avrebbe dovuto essere in grado di sostenere un'ampia gamma di esercizi terapeutici per la riabilitazione di pazienti con disabilità motorie.

L'ortesi è una versione modificata di un supporto passivo anti gravità per il braccio che usa fasce elastiche per alleggerire il peso del braccio dell'utente. Questo dispositivo risulta essere l'evoluzione dell'esoscheletro T-WREX a cui sono stati aggiunti degli attuatori pneumatici. Tuttavia, il sistema di controbilanciamento elastico del peso è stato mantenuto per fare in modo che il robot non si afflosci su sè stesso qualora venga spento e perchè questo aumenta il range di forze utilizzabili dagli attuatori pneumatici che così non si devono sobbarcare il peso del dispositivo. In seguito sono stati scelti attuatori pneumatici perchè sono leggeri, possono generare grandi forze e hanno il potenziale per un buon controllo di queste ultime. D'altro canto sono, però, difficili da controllare perchè presentano un comportamento non lineare.

Il Pneu-WREX è un diretto discendente dell'esoscheletro WREX (Wilmington Robotic Exoskeleton), che era stato originariamente disegnato per aiutare

Figura 2.1: Pneu-WREX[38]

i bambini con particolari problemi di debolezze alle articolazioni superiori nello svolgimento delle normali attività quotidiane. Il WREX è un'ortesi passiva a cinque gradi di libertà che utilizza fasce elastiche, avvolte intorno a due meccanismi four-bar (visibili in figura 2.2), per controbilanciare il peso del braccio. Questo è stato poi modificato per creare una nuova versione dimensionata per un adulto e dotata di strumentazione, chiamata Training WREX, o T-WREX. Il T-REX è stato poi accoppiato con un computer ed utilizzato per fini fisioterapici. Il computer permette di sfruttare quella che viene chiamata Java Therapy, ovvero una serie di esercizi che vengono presentati al paziente sullo schermo del computer e che devono essere completati dallo stesso. L'aggiunta di attuatori pneumatici porta alla definitiva conformazione di quello che prende il nome di Pneu-WREX, visibile in figura 2.1. Anche le fasce elastiche vengono sostituite con delle molle ed un'ulteriore molla viene aggiunta per bilanciare il peso dell'esoscheletro, in questo modo gli attuatori pneumatici vengono utilizzati per compensare i diversi pesi delle braccia dei pazienti.

Rispetto all'originale WREX sono state eseguite anche altre sostanziali modifiche. Innanzitutto è stato eliminato un grado di libertà bloccando un giunto

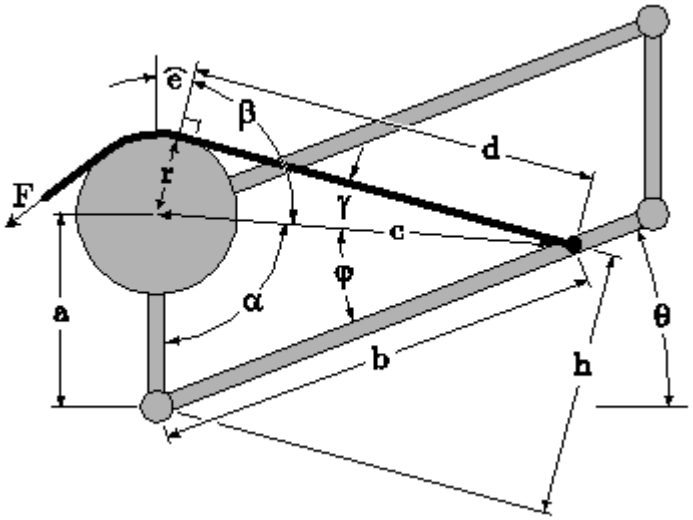


Figura 2.2: Meccanismo four-bar con puleggia e cavi[38]

di rotazione interno/esterno nella posizione zero e rimuovendo il corrispondente attuatore. Il giunto in questione è quello finale della catena che permette la rotazione interna/esterna della spalla. La modifica è stata necessaria per motivi di sicurezza in quanto lo spazio di azione dell'ortesi era superiore a quello effettivo del braccio. Quando il giunto del gomito dell'ortesi è completamente esteso, questo giunto finale può eseguire una rotazione ancora maggiore mentre il braccio non può. Se questo non è un problema per un'ortesi passiva, come il T-WREX, lo diventa invece per il Pneu-WREX che potrebbe esercitare una tensione eccessiva sul gomito del paziente rischiando di risultare dannosa per la salute del soggetto stesso. Questa modifica, necessaria ai fini della sicurezza del paziente, limita alcuni movimenti costringendo, ad esempio, il soggetto a tenere l'avambraccio parallelo al suolo.

2.1.1 Cinematica

Le coordinate di partenza, necessarie per l'analisi cinematica, sono riportate in 2.3 [38].

dove le variabili *ua* e *v* definiscono definiscono, rispettivamente, la lunghezza dell'avambraccio e la distanza tra il giunto superiore della spalla e la spalla in pollici, come si può vedere in figura 2.4.

$$[\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4, \mathbf{q}_5, \mathbf{q}_{5b}] = \begin{bmatrix} 0 & -3.91 & -3.91 & -3.91 & 0.09 & -0.016 \\ 0 & 0 & 1.4 & 1.4+ua & 2.95+ua & 1.25+ua \\ 0 & 0 & -v-3 & -v-3 & -v-6.375 & -v-6.375 \end{bmatrix}$$

$$[\mathbf{q}_{c1b}, \mathbf{q}_{c2b}, \mathbf{q}_{c3b}, \mathbf{q}_{c4b}] = \begin{bmatrix} -2.125 & -5.351 & -6.474 & -6.574 \\ -12.9 & -20.75 & 1.4 & -6.8 \\ 0 & 0 & -v-1.5 & -v-8.2 \end{bmatrix}$$

$$[\mathbf{q}_{c1r}, \mathbf{q}_{c2r}, \mathbf{q}_{c3r}, \mathbf{q}_{c4r}] = \begin{bmatrix} -2.15 & -5.377 & -6.474 & -6.29 \\ 0 & 1.467 & 12.051 & 3.454+ua \\ 0 & 0 & -v-3 & -v-7.692 \end{bmatrix}$$

$$\mathbf{q}_{arm} = \mathbf{q}_5 + \mathbf{hand}$$

Figure 2.3: Equazioni di base

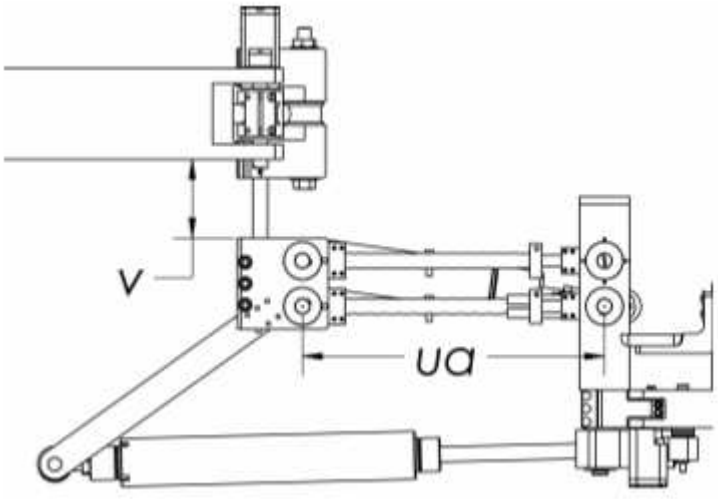


Figura 2.4: Definizioni di lunghezza dell'avambraccio, *ua*, e altezza della spalla, *v*.[38]

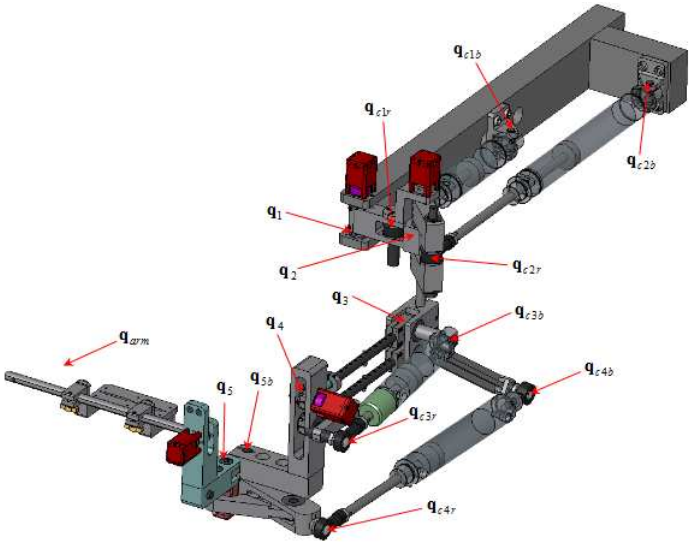


Figura 2.5: Posizioni a riposo del Pneu-WREX per le equazioni cinematiche[38]

Inoltre, sempre con riferimento alla 2.3, $\mathbf{q}_i$sono i punti sugli assi di giunzione, $\mathbf{q}_{cib}$sono i punti sulla base dei cilindri, $\mathbf{q}_{cir}$sono i punti sugli assi dei cilindri, $\mathbf{q}_{arm}$ rappresenta l'interfaccia tra il Pneu-WREX e l'avambraccio del soggetto ed infine la terna $\mathbf{q}_{arm} = \begin{bmatrix} arm_x & arm_y & arm_z \end{bmatrix}^T$ [38] è la posizione della mano rispetto all'ultimo asse $\mathbf{q}_5$. Le posizioni dei punti citati sono visibili in figura 2.5.

Il dispositivo deve essere mantenuto nella posizione di partenza mostrata in figura 2.5 in modo da calibrare i sensori di posizione. La struttura del Pneu-WREX comprende numerosi buchi di allineamento che, quando vi è inserito un asse, fissano l'orientazione dell'esoscheletro nella posizione di base. Gli angoli degli assi di rotazione del dispositivo sono sei e sono definiti secondo la 2.1, [38]

$$\boldsymbol{\theta}_{full} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_{5b} \end{bmatrix}^T \tag{2.1}$$

Gli assi di rotazione sono invece definiti dalla 2.2 [38], dove $\boldsymbol{\omega}_i$ rappresenta il vettore unitario lungo ogni asse di rotazione. Il posizionamento degli assi, invece, è visibile in figura 2.6.

$$\begin{bmatrix} \boldsymbol{\omega}_1 & \boldsymbol{\omega}_2 & \boldsymbol{\omega}_3 & \boldsymbol{\omega}_4 & \boldsymbol{\omega}_5 & \boldsymbol{\omega}_{5b} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{2.2}$$

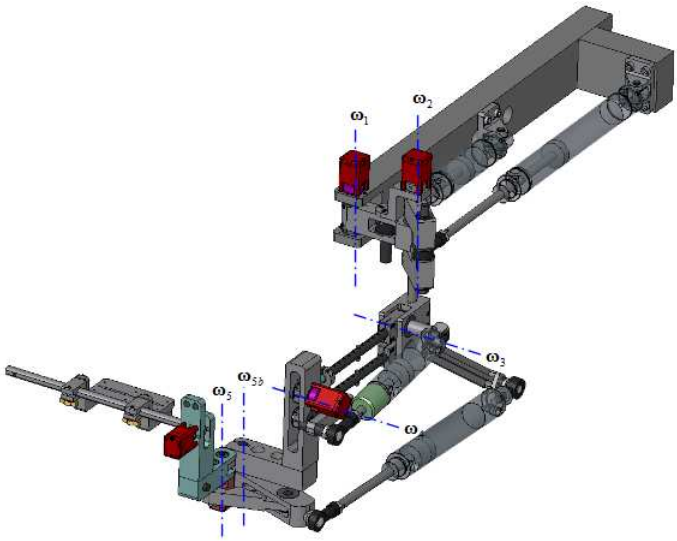


Figura 2.6: Assi di rotazione dell'esoscheletro Pneu-WREX[38].

2.1.2 Cinematica del braccio

Definiti i necessari $\boldsymbol{q}$ e $\boldsymbol{\omega}$, e rispettando i limiti spaziali, può essere trovato un punto omogeneo per il posizionamento dei punti di montaggio dei cilindri. Questi punti vengono trovati utilizzando le definizioni per la torsione, $\boldsymbol{\xi}$, e per l'operatore $\hat{}$, date in [39].

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\omega} \times \boldsymbol{q} \\ \boldsymbol{q} \end{bmatrix} \quad \hat{\boldsymbol{\xi}} = \begin{bmatrix} \hat{\boldsymbol{\omega}} & -\boldsymbol{\omega} \times \boldsymbol{q} \\ 0 & 0 \end{bmatrix} \quad \hat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -\boldsymbol{\omega} \\ \boldsymbol{\omega} & 0 \end{bmatrix} \tag{2.3}$$

[38]

Il punto omogeneo $\boldsymbol{r}_{arm}$ che descrive la posizione della mano nello spazio consentito è quindi

$$\boldsymbol{r}_{arm} = e^{\hat{\xi}_1\theta_1} e^{\hat{\xi}_2\theta_2} e^{\hat{\xi}_3\theta_3} e^{\hat{\xi}_4\theta_4} e^{\hat{\xi}_5\theta_5} \begin{bmatrix} \boldsymbol{q}_{arm}^T & 1 \end{bmatrix}^T \tag{2.4}$$

2.2 Esoscheletro Pneu-WREX: software

2.2.1 Ambiente di controllo Real-Time

Il controllo in Real-Time dell'esoscheletro Pneu-Wrex è stato sviluppato utilizzando il software Simulink prodotto dalla *The Mathworks inc.* ed eseguito utilizzando il Toolbox xPC Target. L'utilizzo di xPC Target richiede la presenza di

Figura 2.7: Schede di acquisizione dati dell'esoscheletro Pneu-WREX. Pneu-WREX utilizza tre schede PCIM-DAS16JR/16 PCI (sinistra) e una scheda PCIM-DAS16JR/16 PCI (destra) [38]

due computer, connessi tra loro, che vengono chiamati host e target. I controlli sono sviluppati sul computer host nell'ambiente grafico di programmazione di Simulink. Il toolbox xPC Target fornisce blocchi di I/O specifici per le schede di acquisizione dati visibili in figura 2.7 . Il codice viene realizzato utilizzando il toolbox Real-Time Workshop e scaricato sul computer target in real time dal kernel di xPC Target che controlla il Pneu-Wrex.

2.2.2 Caratteristiche tecniche dei computer host e target

Il computer host è dotato di un processore Intel P4 3.0 GHz installato su una scheda madre Intel D865GBFL FSB800/533 che utilizza una RAM da 512MB di tipo PC3200 DDR400 [38]. Il computer target, invece, è dotato di un preocessore Intel P4 2.4 GHz installato su una scheda madre Intel D865GBFL FSB800/533 che utilizza una RAM da 512MB di tipo PC3200 DDR400 [38]. La scheda madre Intel è stata scelta in quanto dotata di 6 PCI slot, che permettono l'installazione di schede A/D e D/A per un numero totale di 4 e di una scheda xPC Target compatibile per l'interfaccia di rete [38]. La restante slot può essere utilizzata per future espansioni.

2.2.3 Schede PCI A/D e D/A

L'acquisizione dei dati è ottenuta con tre schede PCI input/output analogiche (modello PCIM-DAS16JR/16) [38]. Queste schede forniscono ognuna 8 input A/D

differenziali a 16-bit e due output D/A a 16-bit anch'ess [38]i. I segnali di controllo analogici delle valvole sono generati da una scheda PCI modello PCI-DDA08/16 che fornisce 8 output analogici a 16-bit. Schede a cavi e viti isolati sono utilizzate per minimizzare il rumore elettrico [38].

2.2.4 Codice di controllo Simulink

Il controllo in real-time per il Pneu-WREX è programmato in Simulink, compilato da xPC Target e scaricato sul computer target che contiene i necessari A/D e D/A per agire sulla protesi robotica. Simulink è un liguaggio di programmazione grafica che ha prevede un flusso di dati da e verso blocchi che svlogono determinati blocchi codificati. Il programmatore ha accesso a centinaia di blocchi precompilati per numerosi tipi di operazioni e ad una serie di blocchi personalizzabili per compiere operazioni più complesse o per definire diverse strutture dati altrimenti non determinabili.

Il codice di controllo per il Pneu-WREX è diviso in numerosi sottosistemi, ognuno dei quali provvede a diverse funzioni del controllore. Questi sottosistemi sono organizzati in un livello superiore di Simulink il cui schema è visibile in figura 2.8 [38].

Comunicazione con l'interfaccia grafica

Gli esercizi da sottoporre al paziente sono realizzati in Visual Basic, che comunica con il computer target utilizzando la librearia xPC Target COM API. La creazione di una libreria COM API è un opzione presente nei settaggi del compilatore di Simulink. Il compilatore crea una libreria COM API basandosi sulle *xPC tags* inserite nel codice Simulink dal programmatore. Ci sono due diversi tipi di xPC Tags: blocchi sorgente e segnali. I blocchi sorgente sono creati inserendo il codice (nella forma xPCTag(1) = nome_variabile;) nel campo *Description* del menù *General* tra le proprietà del blocco. I segnali, invece, sono realizzati inserendo il codice (nella forma xPCTag(1,2,..n) = nome_segnale1,nome_segnale2... nome_segnalen;) nel campo *Description* del menù *Documentation* tra le proprietà del segnale.

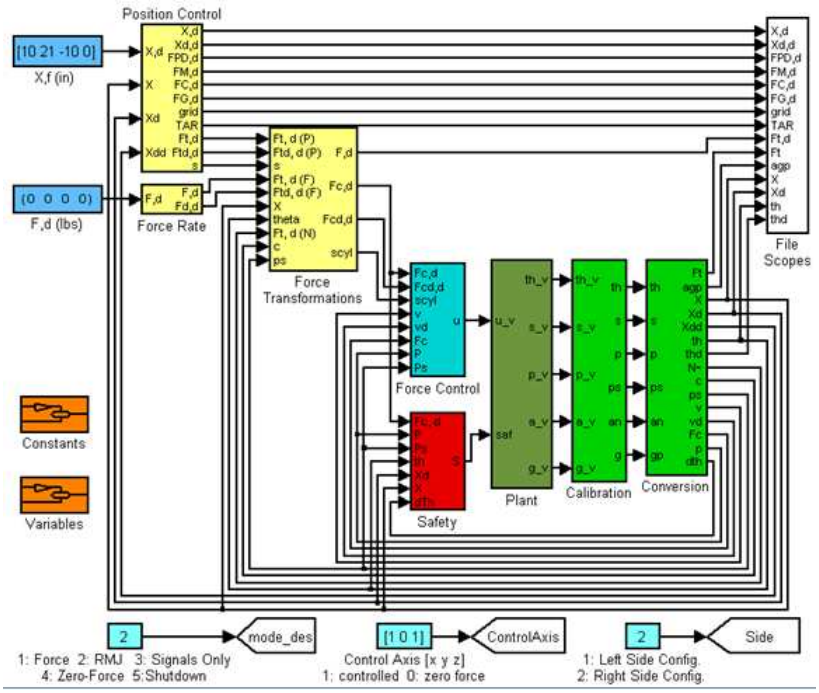


Figura 2.8: Schema Simulink del livello superiore dell'esoscheletro Pneu-WREX[38]

Salvataggio e lettura dei dati

I file relativi ai test effettuati sui pazienti post ictus con il Pneu-WREX vengono salvati in diverse cartelle in base al tipo di test effettuato. Per ogni paziente se ne possono distinguere cinque:

A - test senza distrattore e senza feedback audio;

B - test con il solo utilizzo del distrattore;

C - test con distrattore e feedback audio

D - test con solo feedback audio;

E - test con solo distrattore e con il braccio del paziente a peso morto.

Per ogni tipo di test sono salvati sei diversi tipi di file secondo lo schema di figura 2.9.L'accesso ai dati contenuti nei file presenti in ogni cartella è possibile grazie al metodo *readxpcfile* appartenente alle utilities di xPC Target. Questo metodo converte il contenuto del file, salvato in formato xPC Target, in dati di

tipo double. In particolare converte il file in questione in una struttura dati di tipo *struct1x1* con i seguenti campi (e.g. file 99HF2956.B):

file =

version: 0

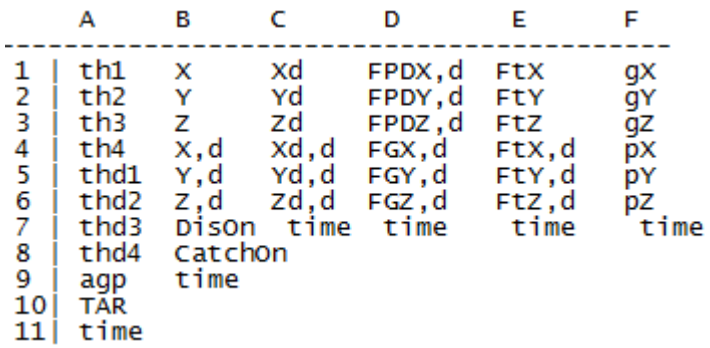
headersize: 512

sector: 0

numSignals: 9

data: [28599x9 double]

Il campo in cui sono contenuti i dati utili al fine di successive elaborazioni è *data* , nel quale è contenuta una matrice la cui dimensione dipende dal tipo di file letto. Nello specifico i file che ho utilizzato per le mie elaborazioni sono stati i B e i C. Il campo *data* dei file di tipo B contiene le informazioni relative alle posizioni rispetto ai tre assi dell'end-effector e del target, espresse in pollici (in), al distrattore, al "catch" del paziente ed infine il tempo, espresso in secondi. Il campo *data* dei file di tipo C, invece, contiene i dati sulle velocità dell'end-effector e del target lungo i tre assi espressi in pollici/secondo (in/s) ed il tempo espresso in secondi. Il campionamento di tutti i dati i dati avviene ogni 5ms, cioè a 200Hz. Nonostante i file di una stessa cartella appartengano tutti al medesimo test i dati non sono perfettamente sincronizzati ma vi è uno sfasamento di qualche millesimo di secondo tra l'uno e l'altro dovuto, probabilmente,a tempi tecnici di trasmissione tra registrazione del dato e scrittura nel file.



	A	B	C	D	E	F
1	th1	X	Xd	FPDX,d	FtX	gX
2	th2	Y	Yd	FPDY,d	FtY	gY
3	th3	Z	Zd	FPDZ,d	FtZ	gZ
4	th4	X,d	Xd,d	FGX,d	FtX,d	pX
5	thd1	Y,d	Yd,d	FGY,d	FtY,d	pY
6	thd2	Z,d	Zd,d	FGZ,d	FtZ,d	pZ
7	thd3	DisOn	time	time	time	time
8	thd4	CatchOn				
9	agp	time				
10	TAR					
11	time					

Figura 2.9: Descrizione dei diversi file salvati nelle cartelle relative ai diversi task. Le lettere maiuscole in alto indicano il tipo di file, i campi di testo sottostanti il tipo di dato che si andrà a leggere mentre i numeri sulla sinistra la corrispondente colonna della matrice del campo *file.data*. Incrociando i numeri con i campi di testo si ottiene la posizione del tipo di informazione che si vuole leggere.

2.2.5 Test eseguiti con il Pneu-WREX

Ai pazienti è stato chiesto di seguire, con la massima accuratezza possibile, un target su un schermo, identificabile con un pallino rosso, con il movimento orizzontale del braccio sinistro rappresentato invece da un pallino verde. Il distrattore visivo è costituito da una gamma di otto simboli casualmente disposti sullo schermo tra quali il paziente deve sceglierne due. Questo è il risultato di una combinazione di tre diversi parametri: il colore, che può essere rosso o verde, una piccola barra gialla, che può essere posizionata sopra o sotto il pallino, ed il posizionamento a destra o sinistra dello schermo. Il distrattore, negli esercizi in cui è prevista la sua presenza, appare per *2s* ad un intervallo di tempo variabile tra *1s* e *5s*. Al paziente è stato chiesto di cliccare, con la mano libera, il tasto sinistro di un mouse ogni volta che uno dei due distrattori prescelti appariva sullo schermo. L'intento era quello di simulare una situazione in cui il partecipante rivolge la sua attenzione visiva ad un leggero distrattore che appare nel suo ambiente durante l'esercitazione.

In alcuni test è stato inoltre aggiunto un feedback acustico nella forma di *beep* della durata di *0.1s* a frequenza che varia proporzionalmente alla grandezza del vettore relativo errore di posizionamento, con una zona morta di *1cm* di raggio

intorno al target.

Protocollo

Ad ogni soggetto è stato chiesto di portare a termine cinque diversi tipi di esercizio con l'ausilio dell'esoscheletro:

A - seguire il target senza la presenza di distrattori visivi e senza feedback acustico;

B - seguire il target con la presenza di ditrattori visivi e senza feedback audio

C -seguire il target con distrattore visivo e con feedback audio

D - seguire il target senza distrattore visivo e con la presenza di feedback;

E - come il task A, ma al paziente deve lasciare il braccio completamente rilassato.

Ogni tipologia di esercizio consiste nella ripetizione di 20 movimenti sinsitra-destra-sinistra, della durata di circa 6 secondi l'uno, per un totale di *120s.* Il profilo di velocità del target è stato scelto secondo la legge di *minimum jerk* sviluppata da Hogan, dove per *jerk* si intende la derivata prima dell'accelerazione secondo il tempo.

Capitolo 3

Sintesi del suono

Il suono che costituirà il feedback audio del dispositivo robotico verrà generato dal software Pure Data sfruttando i dati che gli vengono trasmessi via UDP. Per rendere un'idea questo feedback potrebbe essere costituito da un suono il cui volume aumenta o diminuisce a seconda che la differenza di velocità tra paziente e target aumenti o diminusca. Oppure con lo stesso principio ci si potrebbe basare sull'errore di posizione tra l'uno e l'altro. Pure Data sfrutta un protocollo di invio dati che si chiama OSC (Open Sound Control) le cui specifiche verranno spiegate a breve.

3.1 Protocollo OSC

Open Sound Control è un protocollo di trasmissione dati, ideato e tutt'ora sviluppato all'UC Berkeley Center for New Music and Audio Technology (CNMAT), che permette a computer ed in generale a dispositivi multimediali di scambiare quello che i creatori semplificano in "music performance data" in tempo reale attraverso una semplice rete interna o internet.

Le differenze rispetto al MIDI, di cui rappresenta tutto sommato l'evoluzione, sono sostanziali in quanto l'OSC permette, ad esempio, una velocità di trasmissione dei dati decisamente maggiore ed una risoluzione che va ben oltre gli 8bit permessi dal MIDI.

3.1.1 Sintassi

Tipi di dati fondamentali

Int32: interi a 32bit in complemento a 2 memorizzati in formato *big endian.*

OSC-timetag: numero a virgola fissa a 64bit memorizzati in formato *big endian.*

Float32: numeri a virgola mobile a 32bit rappresentabili in formato IEEE754 memorizzati secondo il formato *big endian.*

Stringhe: una sequenza di caratteri tradotti in codice ASCII al cui termine c'è un byte di 0 seguiti da 1-3 byte di 0 per rendere il numero totale di bit multiplo di 32 e quindi il numero di byte multiplo di 4. Se noi, ad esempio, volessimo rappresentare la stringa "dimeg" dovremmo farlo nel seguente modo:

d	i	m	e	g	/0	/0	/0

OSC-blob: un int32 che rappresenta la dimensione, seguito da un numero arbitrario di byte di dati in formato binario, seguiti a loro volta da 0-3 byte di zero per rendere il numero totale di bit multiplo di 32.

3.1.2 OSC Packets

I pacchetti sono l'unità base di trasmissione del protocollo OSC. L'applicazione che invia i pacchetti si definisce client mentre l'applicazione che li riceve si chiama server. Questi sono costituiti sia dai dati in esso contenuti, scritti in formato binario, sia dalla dimensione di questi dati, ovvero il numero di byte che occupano che, come già visto in precedenza, deve essere sempre un multiplo di 4. La rete che trasporta i pacchetti deve tener conto sia delle informazioni relative al contenuto sia di quelle relative alla dimensione che devono essere entrambe trasmesse. In un protocollo di trasmissione dei dati, come può essere il TCP, il flusso dovrebbe cominciare con un intero a 32 bit che definisce la dimensione, seguito dai dati veri propri e così via anche per i pacchetti successivi.

I dati che vengono trasmessi in un pacchetto OSC si dividono in due tipologie: messaggio (message) o fascio (bundle). Il primo byte di dati permette di definire se il contenuto del pacchetto è del primo o del secondo tipo.

OSC TypeTag	OSC Arguments
i	int32
f	float32
s	stringa
b	OSC-blob

Tabella 3.1: Type Tag[40]

OSC Messages

Un OSC Message è costituito da un OSC Address Pattern, ovvero una stringa che comincia con carattere "/", seguito da un OSC Type Tag String e da eventuali altri OSC Arguments.

Un OSC Type Tag String è una stringa che comincia con il carattere "," seguito da una serie di caratteri che rappresentano il tipo di dato (OSC Arguments) a cui corrispondono. Nella tabella 3.1 sono riportati i caratteri con il corrispondente tipo di dato.

Alcune applicazioni comunicano tra di loro con altri Type Tag addizionali che sono riportati nella tabella3.2 . Qualsiasi altra stringa contenete caratteri non riconosciuti dal protocollo OSC viene automaticamente scartata dal sistema.

OSC Arguments

Una sequenza di OSC Arguments è rappresentata dalle serie delle rappresentazioni in codice binario di ogni singolo argomento.

OSC Bundles

Un fascio è formato dalla stringa "#bundle" seguita da una Time Tag cui seguono zero o più Bundle Elements. La Time Tag è una timetag a 64 bit a virgola fissa il cui significato verrà spiegato in seguito.

Un Bundle Element consiste della sua dimensione e del suo contenuto: la dimensione viene indicata da un int32, multiplo di 4, che rappresenta il numero di byte del contenuto, mentre il contenuto può essere un Messagge o un altro Bundle.

OSC Type String	OSC Arguments
h	intero a 64 bit in complemento a due in modalità big endian
t	OSC-timetag
d	double a 64 bit IEEE754 a virgola mobile
S	rappresentazione alternativa di una stringa
c	carattere ASCII a 32 bit
r	colore RGBA a 32 bit
m	messaggio MIDI a 4 byte
T	*true,* vi sono byte allocati nell'argument data
F	*false*, non vi sono byte allocati nell'argument data
N	*nil*, non vi sono byte allocati nell'argument data
I	*infinitum,* non vi sono byte allocati nell'argument data
[	inizio di un array di dati
]	fine di un array di dati

Tabella 3.2: Type Tag addizionali [40]

Data	Dimensione	Scopo
Stringa "#bundle"	8 byte	Riconosce il tipo *bundle*
OSC timetag	8 byte	Si applica all'intero *bundle*
Dimensione primo elemento	int 32 = 4 byte	Primo elemento
Contenuto primo elemento	Il numero di byte indicato dalla rispettiva dimensione	
Dimensione secondo elemento	int 32 = 4 byte	Secondo elemento
Cintenuto secondo elemento	Il numero di byte indicato dalla rispettiva dimensione	
ecc	ecc	ecc

Tabella 3.3: Struttura di un bundle a due o più elementi[40]

La tabella 3.3mostra le parti di un Bundle a due o più elementi e la dimensione di ogni parte.

3.1.3 OSC Semantics

OSC Address e OSC Addresses Spaces

Ogni OSC server, cioè l'applicazione che riceve il pacchetto di dati, ha un set di metodi (OSC Methods). I metodi sono le potenziali destinazioni degli OSC messages ricevuti dal server e corrispondono ad ognuno dei punti di controllo che l'applicazione rende disponibile. L'invocazione di un OSC method è la stessa cosa di una chiamata a procedura: significa cioè fornire al metodo gli argomenti ed eseguire il metodo.

I metodi dei server sono organizzati secondo una struttura dati dinamica ad albero chiamata OSC Address Space, le cui foglie sono appunto i metodi e i rami

Carettere	Codice ASCII
“	32
#	35
*	42
,	44
/	47
?	63
[	91
]	93
{	123
}	125

Tabella 3.4: Caratteri non utilizzabili [40]

gli OSC Containers. Ogni Method ed ogni Container, apparte la radice, ha un nome definito da una stringa in codifica ASCII di cui però non possono far parte i caratteri visibili nella tebella 3.4.

L' OSC Address è un nome simbolico dato ad un metodo per identificarlo all'interno della struttura ad albero. Partendo dalla radice si separa ogni ramo che si incontra, cioè ogni OSC Container, con il carattere “/” fino ad arrivare al metodo interessato il cui nome deve essere anch'esso riportato. Questa sintassi è stata scelta perché uguale alla sintassi degli URL.

Spedizione dei messaggi e pattern matching

- Quando un server riceve un messaggio, deve invocare l'appropriato metodo per accedere ad esso basandosi sull'indirizzo del messaggio. Questo processo è chiamato dispatching, spedizione. Una volta ricevuto dal server, il messaggio deve essere inviato a tutti i metodi il cui indirizzo corrisponde a quello del messaggio. In pratica viene inviato a tutti i metodi il cui OSC Address corrisponde all'OSC Address Pattern del messaggio. I due indirizzi corrispondono se contengono lo stesso numero di parti e se e se tutte le parti corrispondono carattere per carattere. In questo contesto si distinguono dei

caratteri speciali:

- "?" nell'OSC Address Pattern corrisponde positivamente ad ogni carattere dell'address del metodo
- "*" nell'OSC Address Pattern corrisponde positivamente ad ogni 0 o sequenza di zero dell'Address del metodo
- una stringa di caratteri all'interno di parentesi quadre (e.g. "[testo]") nell'OSC Address Pattern corrisponde positivamente ad ogni carattere della stringa. All'interno delle parentesi i caratteri "-" e "!" hanno significati diversi:
- Una lista di stringhe chiuse all'interno di parentesi graffe e separate tra loro da una virgola indica la corrispondenza di tute le stringhe nella lista

Temporal Semantics e OSC Time Tags

Il protocollo OSC non prevede alcuna tipologia di clock, mentre un server OSC necessita di un qualche meccanismo di questo tipo. Quando il server riceve un pacchetto contenente un unico OSC Message deve semplicemente invocare il corrispondente metodo nel minor tempo possibile. Se invece il pacchetto contiene un OSC Bundle diventa necessario un sistema che scandisca l'andamento temporale per sincronizzare ogni elemento dell'OSC Bundle con il corrispondente metodo; questo sistema di scansione temporale è costituito dalle Time Tags. Se il tempo rappresentato dalla Time Tag è minore o uguale al tempo corrente,il server deve invocare il metodo immediatamente a meno che l'utente non abbia impostato lo stesso in modo da rifiutare i messaggi che arrivano troppo tardi. Oppure le Time Tag possono rappresentare un tempo nel futuro e il server deve salvare l'OSC Bundle fino al tempo specificato ed invocare, quindi, il corretto metodo.

Le Time Tag sono rappresentate da un numero a 64 bit a virgola fissa. I primi 32 bit rappresentano il numero di secondi dal 1/1/1900, mentre gli altri 32 bit rappresentano frazioni di secondo fino ad una precisione di 200 ps. La Time Tag rappresentata da una serie di zero ed un 1 nel bit meno significa che il metodo deve essere invocato immediatamente.

Quando un OSC Address Pattern corrisponde all'Address di più di un metodo, l'ordine con cui i metodi vengono invocati non è importante. Quando, invece, un OSC Bundle contiene più di un messaggio, i set di metodi corrispondenti ai vari messaggi devono essere invocato nello stesso ordine con cui i messaggi compaiono nel pacchetto. Se, infine, un Bundle contiene altri Bundle, la Time Tag del contenuto deve ovviamente essere maggiore di quella del contenitore in quanto questi viene analizzato per primo.

3.1.4 Il nostro caso

Analizziamo ora come è stata preparata la stringa OSC per l'invio dei dati a Pure Data.

Innanzitutto viene definito il dato da inviare che, nell'esempio seguente, è costituito dalle differenze di velocità sui tre assi tra l'end effector ed il target mentre nel caso reale vengono inviati anche le posizioni di end effector e target sempre lungo i tre assi. I dati vengono salvati in un vettore di dimensioni 1x3 dove *counter* è l'indice e vengono prima convertiti in *single* in quanto i numeri a singola precisione hanno una codifica a 32bit che rispetta la dichiarazione di tipo *float a 32bit* dell'OSC Messagge che vedremo a breve:

data=[single(diffX(counter)) single(diffY(counter)) single(diffZ(counter))];

Successivamente si utilizza il metodo *[attr datastring]=cstruct(data)* che scompone il vettore *data* in un vettore di *uint8.*. Il vettore in questione è la rappresentazione in virgola mobile del dato trasmesso in modalità *big endian*, cioè passando per primo il byte più significativo. Un dato di tipo *single*, che è formato da 4 byte, viene scomposto in un vettore di 4 *uint8* che rappresentano i suoi byte. In figura 3.1 si può vedere il risultato della funzione *cstruct* applicata al dato *single(10).* *datastring* è appunto questo vettore mentre *attr* è una structure che contiene i campo *size*, che determina quanti byte sono richiesti per salvare il dato, e *align*, che determina l'allineamento dei byte. I dati vengono convertiti in *single* prima di essere passati a *cstruct* in quanto in questo modo viene rispettato l'*OSC Arguments* che richiede un numero a virgola mobile a 32 bit e l'allineamento di 4 byte

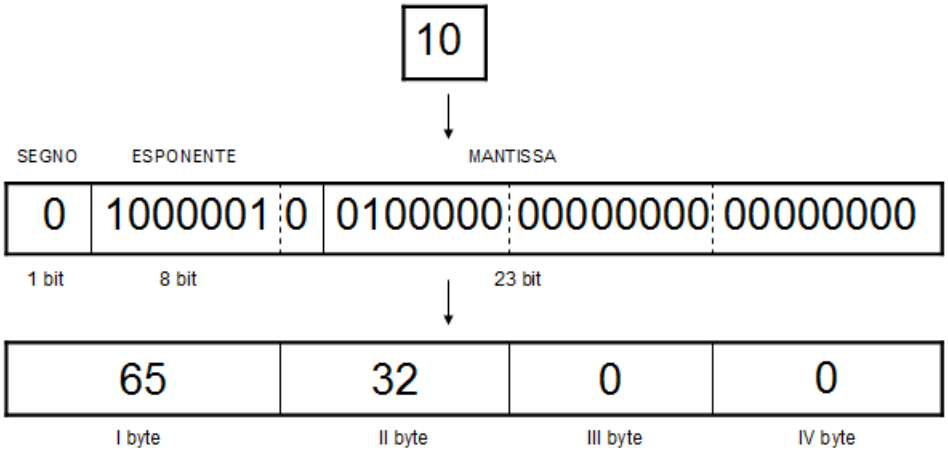


Figura 3.1: Risultato di *cstruct* per il numero *10* a singola precisione

47 (/)	100 (d)	97 (a)	116 (t)	
97 (a)	0 (NULL)	0 (NULL)	0 (NULL)	
44 (,)	102 (f)	102 (f)	102 (f)	
63 (?)	88 (X)	105 (i)	129 (non rappr.)	diffX(counter)
190 ($\frac{3}{4}$)	223 (ß)	233 (é)	97 (a)	diffY(counter)
189 ($\frac{1}{2}$)	233 (é)	240 (ð)	182 (¶)	diffZ(counter)

Tabella 3.5: Esempio di struttura messaggio OSC per il file 99HF1550.C, dopo che è stato riconvertito in uint8 per mostarrne meglio la struttura, nell'intervallo aprossimativo di 60-80 s con counter=11 (tra parentesi il corrispondente carattere)

previsto dal protocollo OSC. Lo stesso metodo viene utilizzato per predisporre i byte di 0 necessari per rendere sempre il numero totale di byte multiplo di 4.

La stringa da inviare a Pure Data, ovvero l'OSC Server, è quindi la seguente:

```
string=char(['/data' zerostring(1:3) ',fff' datastring]);
```

dove le tre "f" indicano i tre float che rappresentano i valori delle tre differenze di velocità. Nella tabella 3.5 è evidenziata la struttura del messaggio.

Le modalità di composizione dell'OSC Messagge nel blocco Simulink sono diverse per l'impossibilità di utilizzare il metodo *cstruct* e verranno spiegate in seguito.

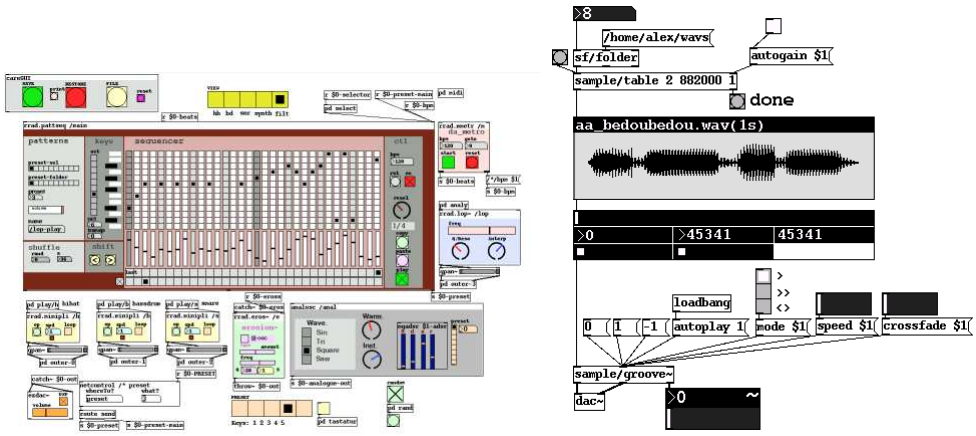


Figura 3.2: Esempi di alcuni progrmammi realizzati con Pure Data [42], [43]

3.2 Pure Data

Pure Data è un ambiente grafico di programmazione real-time per processi audio, video e grafici. Sviluppato da Miller Puckette negli anni '90, PD è un progetto open source ideato per la creazione di musica interattiva con il computer per lavori multimediali [41].

PD è dotato di un'interfaccia grafica che permette di collegare vari blocchi funzionali tra loro e di personalizzarli con una grande gamma di forme e colori.

Nell'immagine 3.2 alcuni esempi di interfacce realizzate con PD.

3.2.1 Il nostro caso

Nel nostro caso il software Pure Data rappresenta l'applicazione server, responsabile della generazione del suono, a cui inviare i dati relativi alle posizioni rispetto agli assi x, y, z e le differenze di velocità tra paziente ed esoscheletro. Nell'immagine 3.3 l'interfaccia di prova che ho utilizzato per testare l'efficacia della GUI dello schema Simulink.

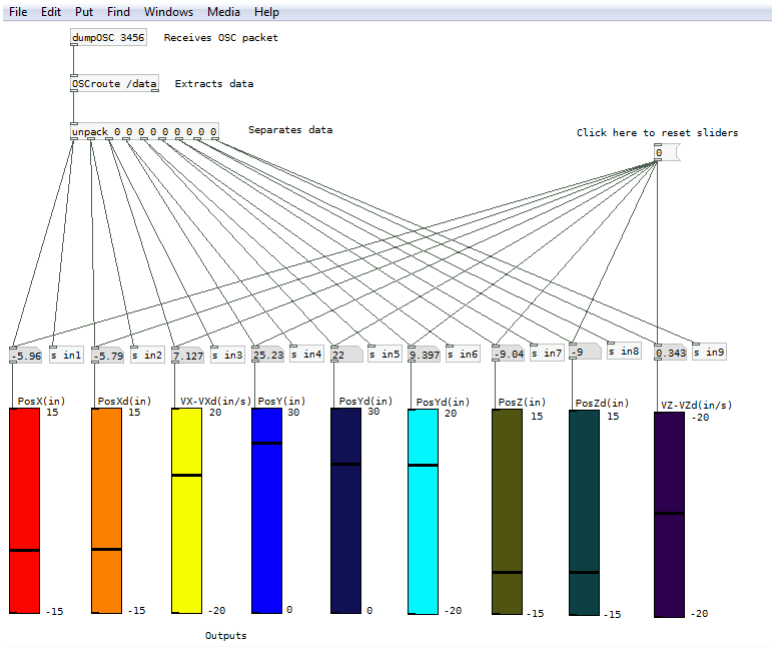


Figura 3.3: L'interfaccia di Pure Data usata per testare il funzionamento della GUI e dello schema Simulink. I nomi relativi agli indicatori colorati sono puramente indicativi e non corrispondono all'effettivo dato che rappresentano.

Capitolo 4

Software sviluppati

4.1 Implementazione interfaccia

Lo scopo dell'interfaccia da me realizzata utlizzando le GUI di Matlab e visibile in figura 4.1 ha lo scopo di visualizzare, alla frequenza di 25Hz i dati relativi agli spostamenti lungo gli assi x e z compiuti dall'esoscheletro *Pneu-WREX* e dal paziente e, contemporaneamente, di inviare, alla frequanza di 100Hz, via UDP a Pure Data i dati relativi alle posizioni e le differenze di velocità tra paziente ed esoscheletro lungo i tre assi. La procedura di utilizzo della GUI è la seguente:

- scelta del task;
- scelta del soggetto;
- scelta dell'intervallo temporale su cui si vuole operare;
- visualizzazione dei dati di posizione degli assi x e z alla frequenza di 25Hz ed invio delle differenze di velocità alla frequenza di 100Hz.

4.1.1 Istruzioni per l'utilizzo del software

Checkbox A, B, C, D, E - permettono la scelta del task.

Load_data - permette la scelta del paziente.

Select - permette la scelta dell'intervallo temporale interessato.

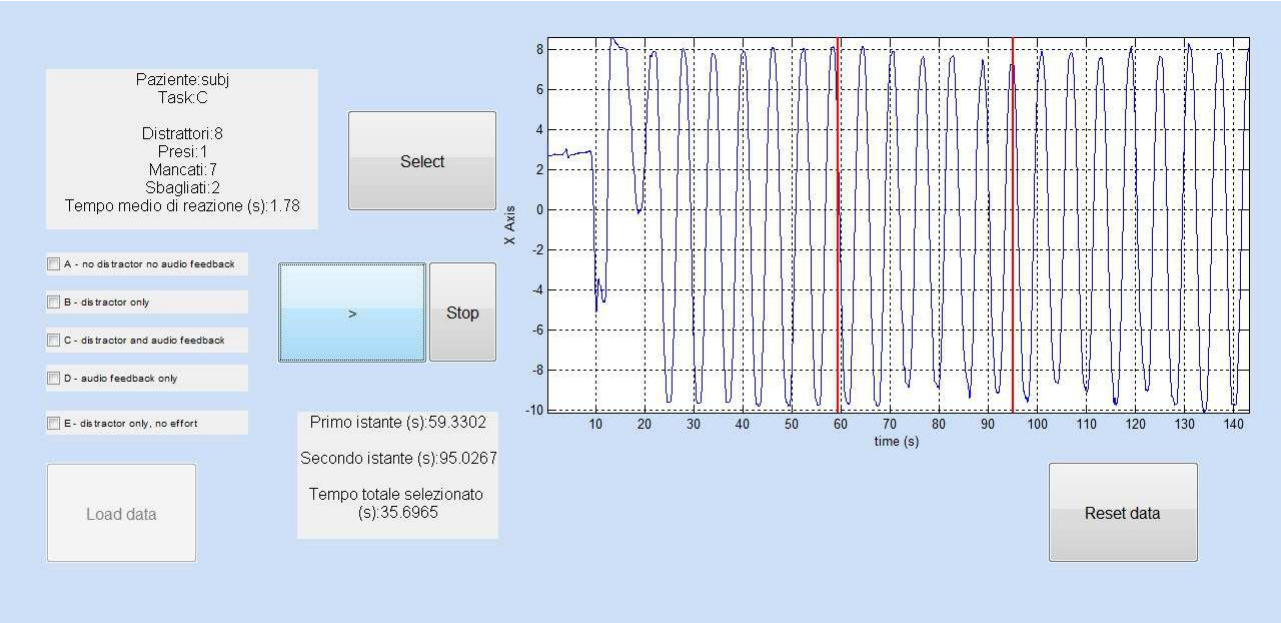


Figura 4.1: Schermata principale del software di analisi dei dati

Play/Pause (>/||) - avvia la simulazione e l'invio dei dati a Pure Data. Permette di bloccare il processo nel punto desiderato e di farla poi ripartire dal medesimo punto.

Stop - stoppa la simulazione e azzera il contatore in modo che premendo successivamente play il processo ricominci dall'inizio.

Reset_data - azzera tutte le variabile ed il contatore facendo così tornare l'intero software allo stato iniziale.

4.1.2 Scelta del task e del soggetto

La scelta del task avviene spuntando uno dei checkbox che si trovano sul lato sinistro della GUI. La scelta può essere effettuata tra:

A - test senza distrattore e senza feedback audio;

B - test con il solo utilizzo del distrattore;

C - test con distrattore e feedback audio

D - test con solo feedback audio;

Figura 4.2: Scelta del task e del paziente di cui caricare i dati e tabella riassuntiva relativa ai distrattori visivi

E - test con solo distrattore e con il braccio del paziente a peso morto.

Per la scelta del paziente, invece, grazie al metodo *uigetdir*, si apre una finestra che permette rapidamente di decidere il soggetto su cui si vuole lavorare.

La casella di testo presente sopra i checkbox mostra nelle prime due righe il soggetto ed il task desiderati mentre nelle altre i dati relativi al numero di distrattori visivi comparsi nel test, quanti di questi sono stati "catturati", quanti sono stati mancati, quanti sono stati gli errori ed il tempo medio di reazione del paziente. Nella figura4.2 si può vedere il dettaglio della GUI.

L'elaborazione dei dati relativi ai distrattori è stata eseguita sull'intera durata del test eseguito dal paziente. Poichè la durata del tempo di "catch" è il più delle volte maggiore della singola unità temporale con cui sono salvati i dati (5 ms) è stato necessario inserire un controllo che evitasse di contare come due catch due successive celle con valore 1. Per fare questo è stato è stato sufficiente imporre al programma di incrementare il numero dei catch solo nel caso in cui sono uguali a 1 la cella relativa al distrattore, quella corrispondente del "catch" ed uguale a 0 la cella del catch precedente a quella in questione. Di seguito la porzione di codice Matlab relativa al controllo in questione dove *handles.dison* è il vettore relativo al distrattore e *handles.catchon* quello relativo al catch:

if handles.dison(i)==1 && handles.catchon(i)==1 && handles.catchon(i-

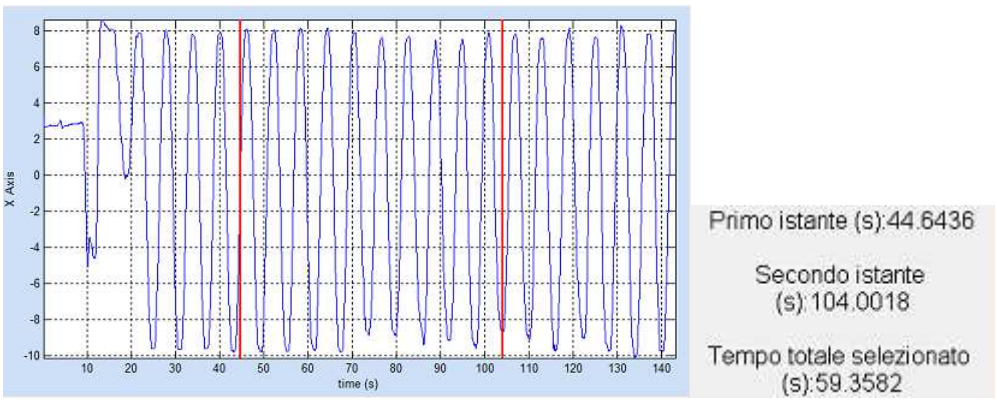


Figura 4.3: Grafico (t, x) e tabella riassuntiva

1)==0

...

end

4.1.3 Scelta dell'intervallo temporale

Sull'area grafica posta in alto a destra viene visualizzato l'andamento temporale, relativi al solo asse x, degli spostamenti eseguiti dal paziente. Con il tasto *Select* si può accedere al grafico e scegliere tremite clic del mouse scegliere i due istanti che delimitano l'arco temporale desiderato. Due linee rosse in corrispondenza dei clic permettono una comprensione più immediata del periodo selezionato mentre la casella di testo sottostante riporta, con precisione al decimo di millesimo di secondo, i due istanti selezionati e la durata dell'intervallo. La figura 4.3 mostra il dettaglio della GUI.

A questo punto, prima di poter dare inizio alla simulazione, è necessario ridurre il numero di campioni dell'intervallo selezionato per evitare di intasare e rallentare il sistema con un numero eccessivo di dati. Ho quindi dovuto effettuare un resampling su base temporale di 100Hz per i dati da inviare a Pure Data ed uno sulla base di 25 Hz per quelli da visualizzare nella simulazione. Questa operazione è possibile grazie al metodo $yi = interp1(x,Y,xi)$ che esegue un'interpolazione lineare della funzione Y nei punti presenti nel vettore xi e dove Y deve essere della stessa lunghezza di x. Nel mio caso Y è il vettore contenete di volta in volta i dati relativi alle posizioni, al distrattore e alle velocità nell'intervallo, x è il vet-

tore contenente il periodo selezionato e *xi* è un vettore contenete le nuove scale temporali. Infatti, per rispettare le due diverse frequenze a cui abbiamo deciso di lavorare, i campioni saranno separati di 0.04s e 0.01s rispettivamente per i dati di posizione e velocità.

Per i dati relativi all'individuazione del distrattore da parte del paziente è stato necessario lavorare in maniera diversa, in quanto l'operazione di interpolazione avrebbe escluso dal vettore risultante la quasi totalità delle celle uguali a 1 risultando un vettore costituito da soli 0. Per sopperire a questo probema ho dovuto costruire un vettore vuoto lungo quanto il nuovo vettore dei tempi e, in corrispondenza degli istanti di "catch" nel vettore originale, porre uguale a 1 le celle del vettore vuoto più le dieci seguenti per permettere una chiara visualizzazione sul grafico. Per una maggiore chiarezza riporto il codice relativo a quanto appena spiegato, dove *handles.catchres* è il nuovo vettore relativo agli istanti di catch, *handles.catchon* è il vettore originale, *handles.tres* è il nuovo vettore dei tempi per la visualizzazione a 25Hz e *handle.time* è il vettore dei tempi originale.

```
handles.catchres=zeros(1, length(handles.tres));
for i=1:length(handles.catchon)
    if (handles.catchon(i)==1 && handles.catchon(i-1)==0)
        j=find(handles.tres>handles.time(i), 1, 'first');
        handles.catchres(j:j+9)=1;
    end
end
```

4.2 Visualizzazione ed invio dati

4.2.1 Pnet

La "pnet" è una libreria di I/O per la comunicazione attraverso porte UDP e/o TCP tra Matlab ed altri software sviluppato Peter Rydesäter della Mid Sweden University.

Differenze tra UDP e TCP

Per la comunicazione tra Matlab ed il software Pure Data è stato scelto di utilizzare una connessione con protocollo UDP (User Datagram Protocol), la quale si differenzia principalmente da quella con protocollo TCP (Transmission Control Protocol) per i seguenti motivi:

- UDP non offre alcuna garanzia né sull'arrivo dei dati né sul loro ordine di arrivo a differenza del protocollo TCP al costo, però, di un maggiore overhead. Per overhead si intendono tutte quelle richieste accessorie che permettono di soddiasfare le prerogative del protocollo TCP rispetto all'UDP[44];
- TCP è un protocollo orientato alla connessione, pertanto, per stabilire, mantenere e concludere una connessione, necessita di pacchetti di servizio i quali aumentano notevolmente l'overhead. UDP, al contrario, invia solo i dati necessari all'applicazione [44];
- l'oggetto della comunicazione di tipo TCP è il flusso di byte, mentre quello della comunicazione UDP è il singolo dato [44].

In sostanza il protocollo TCP viene utilizzato quando si vogliono garanzie riguardo all'arrivo ed all'ordine di arrivo dei vari segmenti di dati, mentre la connessione UDP viene utilizzata quando vi sono vincoli sulla velocità di trasmissione dei dati o sull'economia di risorse della rete in quanto più "leggero" del precedente.

Il nostro caso

Per le comunicazioni tramite protocollo UDP ci siamo serviti di alcuni metodi della *pnet*. Prima di tutto è stato necessario creare il socket per la trasmissione dati ed impostare la porta sorgente.

```
mysock = pnet('udpsocket', 3455);
```

Dove *'udpsocket'* indica appunto che il socket creato è per la trasmissione via UDP mentre*3455* è il numero che identifica la porta sorgente. Quindi impostare l'indirizzo IP e la porta di destinazione, identificati grazie al software *SocketSniff*.

```
pnet(mysock, 'udpconnect', '127.0.0.1', 3456);
```

dove *3456* è la porta che riceve il socket. Dopo aver preparato il dato da inviare, in accordo alle norme del protocollo OSC, ho quindi potuto potuto scrivere il dato nel buffer del socket ed inviarlo.

```
pnet(sock, 'write', string);
pnet(sock, 'writepacket'); .
```

4.2.2 Impostazione del timer

Per consentire la contemporaneità delle due azioni, pur a diverse frequenze di lavoro, e per fare in modo che la loro durata fosse il più possibile corrispondente all'effettiva durata dell'arco temporale precedentemente selezionato, è stato scelto di utilizzare un oggetto di tipo timer che richiamasse una funzione atta sia al plottaggio dei dati di posizione sia all'invio di tutti i dati a Pure Data. Il timer viene creato con le seguenti proprietà:

'Period': 0.01;

'TasksToExecute': (intervallo selezionato)/0.01;

'ExecutionMode': 'fixedRate'

Come detto in precedenza la funzione richiamata dal timer deve contemporaneamente svolgere due azioni a due frequenza diverse. Per ottenere questo risultato l'invio di dati a Pure Data avviene ad ogni chiamata del timer mentre il plottaggio dei dati relativi alle posizioni avviene ogni quattro chiamate. In questo modo viene rispettata sia la frequenza di 100Hz con cui viene richiamata la funzione impostata nel timer determinata dal campo *'Period'* del timer sia quella di 25Hz che è banalmente un quarto della prima.

4.2.3 Resa finale

Il risultato finale del plottaggio dei dati è quello che si può vedere in figura 4.4. I colori dello sfondo e dei pallini sono stati scelti per similitudine a quelli effettivamente utilizzati per i test sui pazienti. Per quanto riguarda invece i dati inviati a Pure Data il riferimanto è alla figura 4.7.

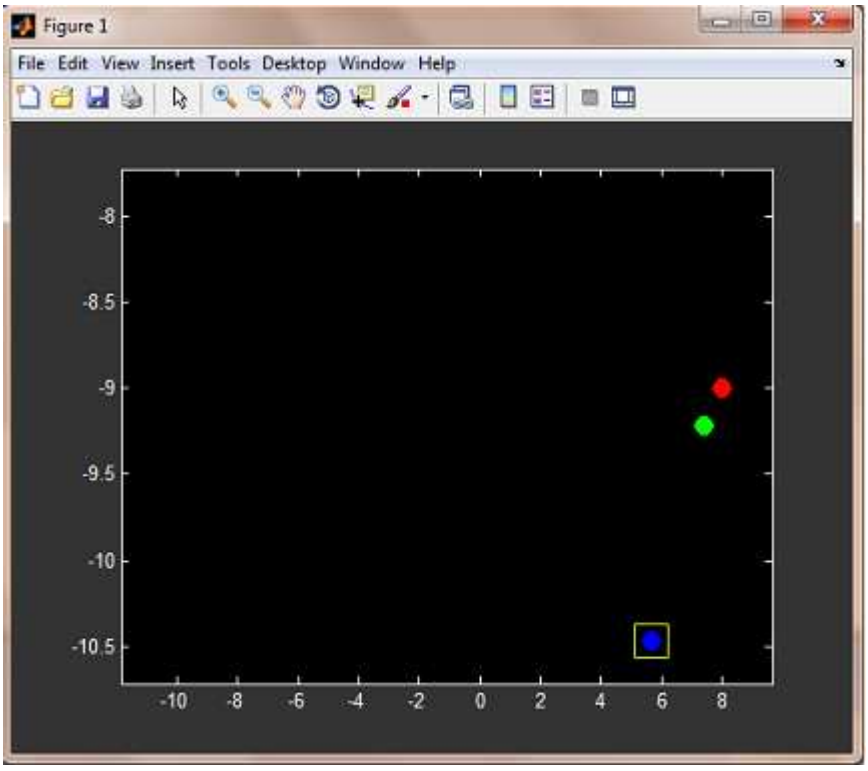


Figura 4.4: Plottaggio dei dati relativi alle posizioni lungo gli assi x e z, al distrattore e al catch. Il pallino rosso rappresenta il target, il pallino verde l'end-effector dell'esoscheletro comandato dal paziente, quello blu il distrattore ed il quadrato giallo il catch.

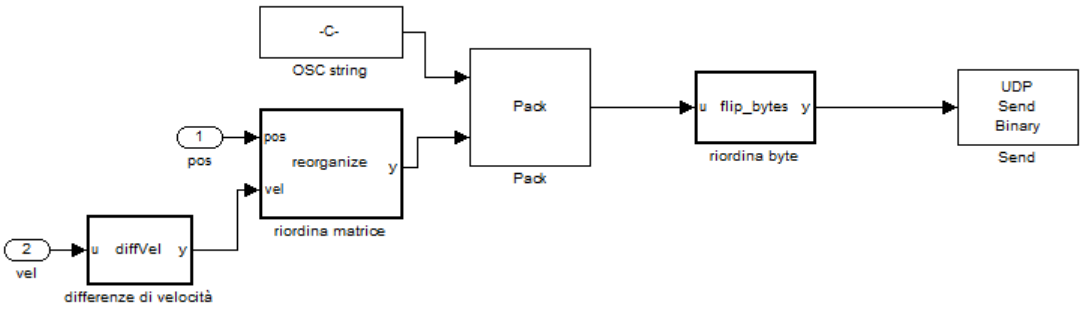


Figura 4.5: Blocco *Send data* per l'invio dati.

4.3 Interfacciacciamento con il robot: modulo real time

Lo schema Simulink che ho realizzato è composto da due sottosistemi: uno responsabile dell'acquisizione dei dati ed uno responsabile dell'invio via UDP di questi ultimi.

4.3.1 Blocco di acquisizione (Load data)

Il blocco di acquisizione, per quanto riguarda il mio elaborato, serve solo per acquisire un file di posizioni e uno di velocità da utilizzare per verificare il corretto funzionamento del blocco responsabile dell'invio dei dati via UDP. L'invio dei dati avverrà in real time all'acquisizione quindi il blocco veramente utile e meritevole di attenzione è solamente il secondo. Questo, infatti è semplicemente costituito da due blocchi *From workspace* che acquisiscono il dato convertito da *double* a *single*. Quando i dati verranno acquisiti in real time basterà aggiungere due blocchi per convertire il dato in single, sempre che questo non avvenga già in qualche passaggio precedente dello schema. e verrà successivamente eliminato. La conversione Per la mia simulazione ho utilizzato i file 99HF2956.B e 99HF2956.C.

4.3.2 Blocco di invio dati (Send data)

Il blocco di invio dati è quello che poi verrà inserito nello schema del robot Sophia3. Il suo schema è visibile in figura 4.5.

Il principio per la realizzazione del blocco è lo stesso che ho seguito per la realizzazione della GUI: definzione del dato, composizione della stringa secondo protocollo OSC ed invio del dato al server. A differenza di quanto è stato fatto per la GUI le modalità sono state diverse in quanto, per problemi riscontrati in varie fasi della costruzione del programma, non ho potuto usare alcune funzioni, come ad esempio *cstruct*, e non mi sono potuto servire della *pnet.*

I primi due blocchi, *diffVel* e *reorganize*, sono due *Embedded Matlab function* e servono semplicemente per definire i tre errori di velocità sui tre assi e per ricomporre il dato da inviare in modo che a Pure Data arrivino, per ogni asse, rispettivamente la posizione dell'end effector manovrato dal paziente, la posizione del target e l'errore di velocità. Una volta definito il dato bisogna quindi comporre la stringa OSC da mandare a Pure Data, che viene formata con le stesse modalità previste dal protocollo OSC utilizzate per la GUI. Nel blocco costante viene inserita la stringa di char

char(['/data' 0 0 0 ',fffffffff' 0 0])

dove *'/data'* definisce L'OSC Address Pattern e *',fffffffff'* rappresenta i nove numeri di tipo float a 32-bit che ci accingiamo ad inviare. Il tutto viene completato da 0, che poi diventeranno byte vuoti, che servono per rendere il numero totale di byte sempre multiplo di 4. Prima di passare allo step successivo la stringa, sempre nel medesimo blocco, viene covertita in *uint8* in quanto il blocco *Pack* non accetta in ingresso dati di tipo char. A questo punto il blocco *Pack* combina la stringa, convertita in *uint8*, con il vettore preparato in precedenza, creando così il messaggio OSC per Pure Data, e converte tutto in *uint8,* che è il formato previsto per l'invio dei dati. Il dato che esce dal blocco è quindi un vettore di *uint8* in cui ad ogni carattere della stringa corrisponde un byte, ed un elemento del vettore, e ad ogni numero del vettore di dati corrispondono 4 byte, cioè 36byte in tutto, per un totale di 16+36=52 byte, cioè 52 elementi del vettore.

Il blocco flip_byte

Nella GUI la conversione ad *uint8* della stringa OSC avviene grazie al metodo *cstruct.* In Simulink questo metodo non può essere utilizzato in quanto crea

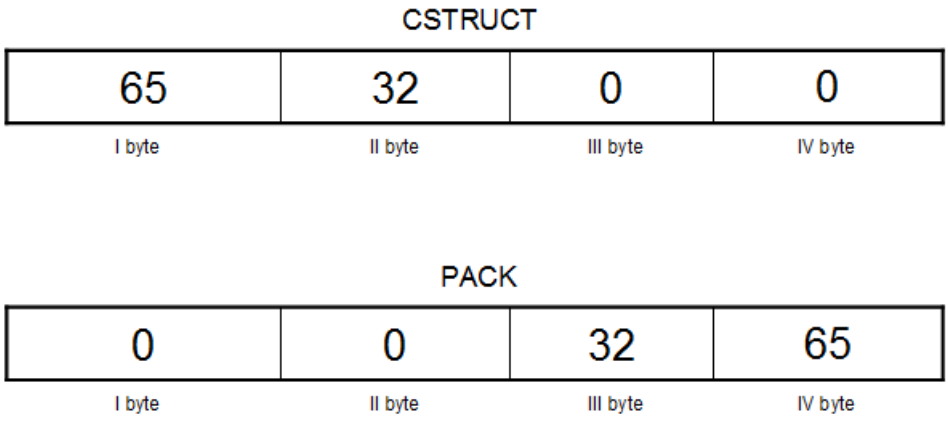


Figura 4.6: Confronto tra *cstruct* (big endian) e *Pack* (little endian) per il numero *10* in singola precisione

problemi in fase di compilazione. Tuttavia la procedura precedentemente descritta supplisce all'uso di *cstruct* ma con una differenza. Quando il vettore esce dal blocco *Pack*, è corretto nella parte riguardante l'Address Pattern e la definizione del tipo e del numero dei dati, ma i byte dei dati relativi ai test non sono trasmessi nell'ordine corretto ma sono invertiti: il primo byte diventa il quarto, il secondo diventa il terzo e viceversa. In sostanza mentre da *cstruct* i byte uscivano in modalità *big endian*, qua escono in formato *little endian*, cioè il primo byte a passare nel blocco *send* è il meno significativo. Di seguito possiamo vedere il codice del blocco.

```
function y = flip_bytes(u)
y = u([1:20 24:-1:21 28:-1:25 32:-1:29 36:-1:33 40:-1:37 44:-1:41
48:-1:45 52:-1:49 56:-1:53]);
```

dove *y* è il vettore in ingresso e *u* quello in uscita. I primi 20 elemento che rappresentano L'*Address Pattern* e l'*OSC Arguments* rimangono inviariate mentre di quattro in quattro viene scambiato l'ordine dei successivi.

Nelle tabelle **??** possiamo vedere a confronto il risultato di *cstruct* ed il segnale che esce da *Pack* per il numero *single(10)* utilizzato anche per l'esempio precedente. Il vettore risultante viene comunque inviato dal blocco *UDP Send Binary*, e ricevuto da Pure Data che però non riconosce alcun numero in quanto si aspetta

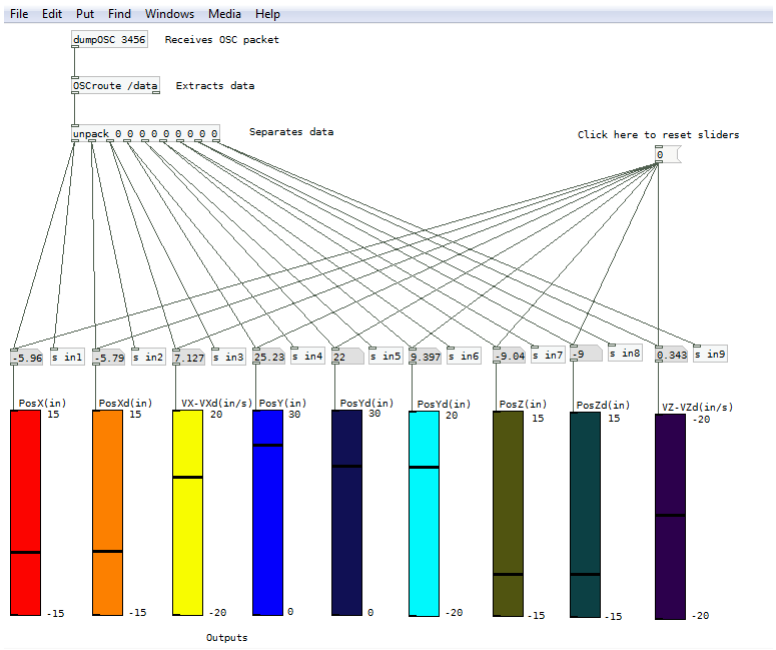


Figura 4.7: Schema di Pure Data utilizzato per le prove con il software di analisi e con il modulo real time. Gli indicatori vanno letti a gruppi di tre: posizione dell'end effector, posizione del target ed errore di velocità dell'asse considerato, rispettivamente x, y e z.

di ricevere il dato in formato *big endian*, come previsto dal protocollo OSC. Il blocco *flip_byte*, realizzato con una *Embedded function*, quindi serve proprio a ristabilire la modalità *big endian,* invertendo l'ordine dei byte, e permettere così una ricezione corretta da parte di Pure Data. In figura 4.7 lo schema di Pure Data utilizzato per le prove con Simulink.

Conclusioni

La GUI ha dimostrato di funzionare in maniera soddisfacente rispettando quelle che erano le previsioni di lavoro iniziali. I dati vengono plottati correttamente permettendo una chiara riproduzione dei test eseguiti sui pazienti con l'esoscheletro Pneu-WREX. Anche l'invio dei dati a Pure Data avviene senza problemi permettendo così all'utente di avere istante per istante la posizione del target e dell'end effector e le relative differenze di velocità.

Il blocco Simulink sembra, almeno a livello teorico, funzionare in maniera corretta. Tuttavia solo l'inserimento nello schema finale del robot dimostrerà o meno la bontà del lavoro svolto.

Sia la GUI che il blocco Simulink sono stati creati per lavorare su certi tipi di dati che devono salvati ed ordinati secondo un certo schema quindi, specialmente per quanto riguarda il blocco, i dati devono essere per forza passati nel formato xPC Target e rispettare lo schema su cui ho lavorato io. Tuttavia mi sento di poter dire che, operando qualche modifica ai codici, i software si potrebbero adattare senza eccessive difficoltà a diversi tipi di formato. Soprattutto il blocco Simulink, a mio avviso, con poche modifiche può essere utilizzato per inviare il numero ed il tipo prescelto di dati senza eccessivi problemi; ricordando comunque nuovamente che il suo effettivo funzionamento al momento resta solo teorico.

Bibliografia

[1] W. Rosamond, K. Flegal, and et al, "Heart disease and stroke statistics-2007 update: A report from the american heart association statisticc committee and stroke statistics subcommittee," *Circulation*, vol. 115.

[2] T. Truelsena, M. Ekmanb, and G. Boysena, "Cost of stroke in europe," *European Journal of Neurology*, vol. 12.

[3] H. Feys, W. D. Weerdt, and et al, "Effect of therapeutic intervention for the hemiplegic upper limb in the acute phase after stroke: a single-blind, randomized, controlled multicenter trial," *Stroke*, vol. 29, pp. 785–792, 1998.

[4] , "Early and repetitive stimulation of the arm can substantially improve the long term outcome after stroke: a 5 year follow-up study of randomized trial," *Stroke*, vol. 35, pp. 924–929, 2004.

[5] G. Kwakkel, R. V. Pappen, and et al, "Effect of augmented exercise therapy after stroke: a 5-year follow-up study of randomized trial," *Stroke*, vol. 35, pp. 2529–2539, 2004.

[6] G. Kwakell, R. C. Wagenaar, J. W. Twisk, G. J. Lankhorst, and J. C. Koetsier, "Intensity of leg and arm training after primary middle-cerebral-artery stroke: a randomised trial," *Lancet*, vol. 354, pp. 191–196, 1999.

[7] S. L. Wolf and C. J. Winstein, "Effect of constraint-induced movement therapy on upper extremity function 3 to 9 months after stroke: the excite randomized clinical trial," *JAMA*, vol. 296, pp. 2095–2104, 2006.

[8] K. J. Ottenbacher, P. M. Smith, S. B. Illig, R. T. Linn, G. V. Ostir, and C. V. Granger, "Trends in length of stay, living setting, functional outcome and

mortality following medical rehabilitation," *JAMA*, vol. 292, pp. 1687–1695, 2004.

[9] S. M. Schmidt, L. Guo, and S. J. Scheer, "Changes on the status of hospitalized stroke patients since inception of the prospective payment system in 1983," *Arch Phys Med Rehabil*, vol. 83, pp. 894–898, 2002.

[10] R. K. Bode, A. W. Heinemann, P. Semik, and T. Mallinson, "Relative importance of rehabilitation therapy characteristcs on functional outcomes for persons with stroke," *Stroke*, vol. 35, pp. 2537–2542, 2004.

[11] M. L. Aisen, H. I. Krebs, N. Hogan, and B. Volpe, "The effect of robot-assisted therapy and rehabilitative trianing on motor recovery following stroke," *Arch. Neurol*, vol. 54, pp. 443–446, 1997.

[12] B. Volpe, H. Krebs, N. Hogan, L. Edelsteinn, C. Diels, and M. Aisen, "Robot training enhanced motor outcome in patients with stroke manteined over 3 years," *Neurology*, vol. 53, pp. 1874–1876, 1999.

[13] H. Krebs and J. P. et al, "Rehabilitation robotics: performance-based oregressive robot-assisted therapy," *Auto. Rob.*, vol. 15, pp. 7–20, 2003.

[14] E. T. Wolbrecht, V. Chan, D. J. Reinkensmeyer, and J. E. Bobrow, "Optimizing compliant, model-based robotic assistance to promote neurorehabilitation," *IEEE Transactions on Neural Systems and Reahabilitation Engineering*, vol. 16, no. 3.

[15] Nef, T, and R. Riener, "Armin-design of a novel arm rehabilitation robot," *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on:*, pp. 57–60, 2005.

[16] F. Malouin, C. Richards, B. McFayden, and J. Doyon, "New perspectives of locomotor rehabilitation after stroke," *Med sci*, no. 19, pp. 994–998, 2003.

[17] Jezernik, S, G. Colombo, and et al, "Robotic orthosis lokomat: A rehabilitation and research tool," *Neuromodulation 6*, vol. 2, pp. 108–115, 2003.

[18] P. Lum, C. Burgar, and et al, "Mime robotic device for upper-limb neurorehabilitation in subacute stroke subjects: A follow up study," *Journal of Rehabilitation Research and Development*, vol. 5, no. 43.

[19] J. He, E. J. Koeneman, and et al, "Rupert: a device for robotic upper extremity repetitive therapy," *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual Internatinal Conference of the:*, pp. 6844–6847, 2005.

[20] D. J. Reinkensmeyer, L. E. Kahn, M. Averbuch, A. McKenna-Cole, B. D. Schmit, and W. Z. Rymer, "Understanding and treating arm movement impairment after chronic brain injury: Progress with the ARM Guide," *Journal of Rehabilitation Research and Development*, vol. 37, no. 6, pp. 653–662, 2000.

[21]

[22] A. RodÃ , F. Avanzini, S. Masiero, and G. Rosati, "Auditory feedback in robot-asissted neuro-rehabilitation: state of the art and future prospects," p. 2.

[23] R. Riener, L. Lunenburger, S. Jezernik, M. Anderschitz, G. Colombo, and V. Dietz, "Pazient-cooperative strategies for robot-aided treadmill traininga. first experimental results," *IEEE Trans Neural Sys and Rehab Eng.*, vol. 13, pp. 380–394, 2005.

[24] J. Mehrholz, T. P. T, J. Kugler, and M. Pohl, "Electromechanical and robot-assisted arm training for improving arm function and activities of daily living after stroke (review)," *Cochrane Database of Systematic Reviews*, no. 4, 2008.

[25] G. Kwakkel, B. J. Kollen, and H. I. Kreb, "Effects of robot-assisted therapy on upper limb recovery after stroke: A sistematic review," *Neurorehabil Neural Repair*, no. 22, pp. 111–121, 2008.

[26] W. S. Harwin, J. L. Patton, and V. R. Edgerton, "Challenges and opportunities for robot-mediated neurorehabilitation," *Proceedings of the IEEE*, vol. 94, no. 9, pp. 1717–1726, 2006.

[27] L. D. Wit and et al, "Use of time by stroke patients: A comparison of four europena rehabilitation centers," *Stroke*, vol. 36, pp. 1977–1983, 2005.

[28] S. H. Kreisel, H. Bazner, and M. G. Hennerici, "Pathohysiology of stroke rehabilitation: temporal aspects of neuro-functional recovery," *Cerebrovasc Dis*, vol. 21, pp. 6–17, 2006.

[29] I. Peretz and R. J. Zatorre, "Brain organization for music processing," *Annu Rev Psychol*, vol. 56, pp. 89–114, 2008.

[30] S. Koelsch, E. Kasper, D. Sammler, K. Schulze, T. G. T, and A. D. Friederici, "Music, language and meaning: brain signatures of semantic processing," *Nat Neurosci*, vol. 7, pp. 302–307, 2004.

[31] M. Popescu, A. Otsuka, and A. A. Ioannides, "Dynamics of brain activity in motor frontal cortical areas during music listening: a magnetoencephalographic study," *Neuroimage*, vol. 21, no. 9, pp. 1622–1638, 2004.

[32] Sarkamo and et al, "Music listening enhances cognitive recovery and mood after middle cerebral artery stroke," *Brain*, vol. 131, pp. 866–876, 2008.

[33] J. P. Azulay and et al, "Visual control of locomotation in parkinson's disease," *Brain*, vol. 122, pp. 111–120, 1999.

[34] Y. Baram and A. Miller, "Auditory feedback control for improvement of gait in patients with multiple sclerosis," *J Neurol Sci.*, vol. 254, pp. 90–94, 2007.

[35] M. H. Thaut, G. C. MacIntosh, and R. R. Rice, "Rythmic facilitation of gait training in hemiparetic stroke rehabilitation," *Clin rehabil*, vol. 17, pp. 713–722, 2003.

[36] M. Schauer and K. H. Mauritz, "Musical motor feedback (mmf) in walking hemiparetic stroke patients: randomized trials of gait improvement," *Clin Rehabil*, vol. 17, pp. 713–722, 2003.

[37] R. Secoli, G. Rosati, and D. J. Reinkensmeyer, "Using sound feedback to counteract visual distractor during robot-assisted movement training," 2009.

[38] E. T. Wolbrecht, "Adaptive, assist-as-needed control of a pneumatic orthosis for optimizing robotic movement therapy following stroke," Ph.D. dissertation, University of California Irvine, 2007.

[39] R. Murray, Z. Li, and et al, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[40]

[41]

[42]

[43]

[44]