UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Controllo robusto di un manipolatore industriale

Robust control of an industrial manipulator

Candidate:
Marco Petrosilli

Advisor:
Prof. Giuseppe Orlando

Coadvisor:
Prof. Gianluca Ippoliti

Academic Year 2023-2024

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Controllo robusto di un manipolatore industriale

Robust control of an industrial manipulator

Candidate:
Marco Petrosilli

Advisor:
Prof. Giuseppe Orlando

Coadvisor:
Prof. Gianluca Ippoliti

Academic Year 2023-2024

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione
Via Brecce Bianche – 60131 Ancona (AN), Italy

“ti diranno che è impossibile
tracciar la propria rotta come lo vuoi tu,
ma resteranno dietro,
nel tetro buio invisibile,
guardandoci brillare con la testa in su”
C.R. - “0602”

Sommario

Questa tesi si concentra sullo sviluppo e l'implementazione di un sistema di controllo robusto per un manipolatore industriale a due gradi di libertà, il Barras ERICC. L'obiettivo principale del lavoro è stato far sì che tramite la legge di controllo proposta, sliding mode control, si fosse in grado di far posizionare il manipolatore come voluto, e renderlo dunque in grado di raggiungere determinati pezzi designati come target. Le strategie di controllo proposte sono state dapprima testate in ambiente di simulazione, utilizzando MATLAB/Simulink, per avere un'idea della risposta del modello del robot alla legge di controllo applicata. I risultati delle simulazioni hanno mostrato delle prestazioni più che accettabili, lasciando dunque spazio aperto all'implementazione effettiva e le prove reali sul manipolatore. Successivamente, le soluzioni sviluppate sono state implementate sul robot, dopo aver adattato procedure e parametri ad un ambiente concreto, piuttosto che ad uno di simulazione. Inoltre si è arricchito il sistema con un software di visione, che allenato su un certo tipo di oggetti, è in grado di riconoscere quando uno di questi si trova di fronte al manipolatore, estrarne le coordinate della sua posizione cartesiana ed ottenere le posizioni necessarie per i giunti del robot per andare a raggiungerlo, andando a dare i segnali di riferimento al sistema di controllo. In conclusione, la tesi dimostra come la legge di controllo in sliding mode, appositamente affiancata dal sistema di visione, sia stata in grado di far compiere all'ERICC il task assegnato in principio.

Abstract

This thesis focuses on the development and implementation of a robust control system for a two-degree-of-freedom industrial manipulator, the Barras ERICC. The main aim of the work was to ensure that through the proposed control law, sliding mode control, it was possible to position the manipulator as desired, and therefore make it capable of reaching certain pieces designated as targets. The proposed control strategies were first tested in a simulation environment, using MATLAB/Simulink, to get an idea of the response of the robot model to the mentioned control law. The simulation results showed more than acceptable performance, leading us to the actual implementation and real tests on the manipulator. Subsequently, the developed solutions were implemented on the robot, after adapting procedures and parameters to a concrete environment, rather than to a simulation one. Furthermore, the system has been enriched with vision software, which, trained on a certain type of objects, is able to recognize when one of these is in front of the manipulator, extract the coordinates of its Cartesian position and obtain the positions necessary to the joints of the robot to reach it, giving the reference signals to the control system. In conclusion, the thesis demonstrates how the sliding mode control law, specifically supported by the vision system, was able to make the ERICC complete the task assigned in the beginning.

Ringraziamenti

Per il raggiungimento di questo obiettivo sento di voler ringraziare tutte quelle persone delle quali mi sono circondato durante questi tre anni e non solo, perchè si sa che coloro che ci circondano costituiscono gran parte di quello che siamo.
Dapprima voglio ringraziare i miei genitori, che da sempre fanno sacrifici per fornire i mezzi grazie ai quali è stato possibile affrontare questo percorso.
Ringrazio mio padre per avermi infuso l'amore per l'automazione, facendo crescere in me una forte passione per questa disciplina.
Ringrazio mia madre per avermi sempre affiancato e dato l'amore che serve a crescere, per guardare il mondo sotto occhi più adulti, realisti, ma sempre sognatori.
Ringrazio mio fratello, un compagno incondizionato dall'inizio dei giorni che mi conosce più di qualsiasi altra persona, al quale auguro il meglio, il benessere, e tutta la fortuna in questa vita, perchè una persona come lui merita il mondo.
In secondo luogo voglio ringraziare tutte quelle persone che ho conosciuto durante questo percorso, un cammino che abbiamo affrontato insieme collaborando e condividendo momenti di gioia e di frustrazione. I ringraziamenti vanno a Francesco, Alessio, Marco, Mattia, Matteo, Andrea, Simone, Alessandro e tutti coloro che mi sarò dimenticato di menzionare. Oltre questi vorrei ringraziare in particolar modo Giorgio, con il quale abbiamo sviluppato il progetto illustrato in questo elaborato, chiudendoci mesi in laboratorio e portando avanti ciò che amiamo; e Paolo, amico da prima dell'università, da prima della scuola, da prima di tutto, con il quale abbiamo affrontato insieme ogni esame di questa triennale, facendoci spalla a vicenda e cercando tra noi di influenzarci con i nostri pregi, e combattere i nostri difetti. Per ultimo, ma assolutamente non per importanza, voglio ringraziare tutti i miei amici e le persone che mi hanno circondato al di fuori dell'università, persone con cui sono cresciuto, ed il contributo delle quali è stato più determinante di quanto loro pensino, perchè i momenti di svago, spensieratezza e leggerezza che si alternando ai momenti di studio, sono anch'essi fondamentali per essere sereni ed ottenere buoni risultati accademici. Tra loro ringrazio Beni, Karrè, Monte, Pallot, Vasi, Cimbo, Petro, Salvuc, Daddi, Tardu, Luchè, Vale e Caterz.
Infine ringrazio me stesso, non per vanità, ma per ricordarmi che nonostante questo sia solo il primo traguardo raggiunto, tutto è possibile se ci si arma di tutta la propria dedizione.

Ancona, Settembre 2024

Marco Petrosilli

Indice dei contenuti

Chapter 1

Introduzione

1.1 Problema del controllo di un manipolatore planare

Nel corpo di questo elaborato si affronterà la tematica del controllo di un manipolatore planare a due gradi di libertà. Generalmente possiamo osservare un qualsiasi sistema come un qualcosa che può ricevere degli stimoli esterni ed è in grado di produrre un'uscita, dipendente dalle caratteristiche interne dello stesso. Chiaramente in funzione dei vari casi d'utilizzo si avrà il bisogno di conferire diverse caratteristiche ad un sistema, possono essere delle specifiche riguardanti la fedeltà di risposta più o meno accurata rispetto ad un segnale di riferimento, o ad esempio specifiche sulla durata e sulla qualità del comportamento transitorio, insomma si vogliono andare ad imporre delle peculiarità in modo tale che il sistema si comporti nel modo più consono alla situazione che ne prevede l'utilizzo. Per realizzare quanto detto pocanzi, il concetto di sistema di controllo riveste un ruolo centrale, si vuole infatti stabilire una legge che determini le modalità secondo le quali bisogna dare sollecitazioni al sistema in questione, per far sì che questo abbia nel tempo l'evoluzione desiderata. Per riuscire a stabilire il "come" bisogna dare questi stimoli bisogna poter conoscere lo stato attuale del sistema, questo viene misurato tremite dei sensori ed è proprio sulla base di ciò che viene visto da questi che il controllore calcolerà il segnale di controllo che andrà dato agli attuatori (dopo essere stato opportunamente amplificato). Come si vedrà in seguito in maniera più dettagliata, essendo nel caso di un manipolatore planare a due gradi di libertà, gli attuatori in questo caso sono due motori a collettore, e per conoscerne lo stato vengono utilizzati dei potenziometri (per la posizione angolare) e delle dinamo (per la velocità angolare). Lo stato misurato verrà poi interpretato per dar luogo ad un segnale di controllo tramite un controllore VSC (Variable Structure Control) che implementa il SMC (Sliding Mode Control).
Si illustrerà dapprima il manipolatore nel suo inisieme, proseguendo poi per la descrizione cinamatica e dinamica di quest'ultimo, si illustreranno le due componenti principali di questo lavoro, ossia il controllo e la computer vision, descrivendo poi i software realizzati per la simulazione e per l'implementazione reale del sistema di controllo, discutendo opportunamente i risultati ottenuti e traendo le dovute conclusioni.

Chapter 2

Manipolatore Barras ERICC

2.1 Descrizione

[1] Il manipolatore ERICC, prodotto da ABB Barras Provence, è il sistema fisico introdotto in precedenza. La catena cinematica dell'ERICC presenta nativamente una configurazione con 5 gradi di libertà: 3 per determinare la posizione dell'end-effector nello spazio di lavoro, che sfruttano dei motori a collettore mentre altri 2 per l'orientamento dell'end-effector, dotato di una pinza azionata da motori passo-passo, che consente di interagire con l'ambiente esterno afferrando e ruotando oggetti. Il corpo del manipolatore è costruito in alluminio e presenta un sistema di cinghie, riduttori e pulegge per la trasmissione del movimento, è proprio questo complesso che permette di fare sì che il polso mantenga sempre la stessa inclinazione rispetto al braccio. Originariamente il sistema prevede sì la struttura meccanica pocanzi illustrata, ma ingloba anche un unità di controllo propria con un terminale video ed un tastierino per un controllo manuale in real-time, tale unità di controllo è stata poi esclusa elettronicamente così da poter attuare l'asservimento con il VSC. Denotando con φ_i l'angolo di rotazione descritto dal giunto $i-$esimo, si osservi che le traiettorie che l'ERICC potrebbe percorrere data la sua struttura riguardano:

- φ_1 - Rotazione del tronco
- φ_2 - Rotazione della spalla
- φ_3 - Rotazione del gomito
- φ_4 - Rotazione del polso
- φ_5 - inclinazione del polso

Nel progetto in questione sono stati impiegati solo due dei giunti a disposizione, rispettivamente la spalla ed il gomito, questo per delle problematiche riguardanti l'hardware (trattate a seguire in maniera più esaustiva) che determinano l'impossibilità di controllare da PC gli altri attuatori. Viene riportata di seguito una tabella esplicativa dei limiti delle rotazioni permesse ad ogni giunto:

Giunto	Valore minimo [°]	Valore massimo [°]
φ_1	-135	135
φ_2	-45	90
φ_3	-45	90
φ_4	-180	180
φ_5	-90	90

Table 2.1: Valori massimi/minimi ammessi dai giunti

2.2 Attuatori

Come accennato precedentemente i tre giunti per il movimento di tronco, spalla e gomito vengono impiegati tre motori a collettore in corrente continua, prodotti dalla Portescap. A livello costruttivo questi dispositivi sono composti da:

- Statore: composto da magneti permanenti, che provvedono alla creazione del campo magnetico statorico
- Rotore: avvolgimento metallico, che percorso da corrente crea a sua volta un flusso di campo magnetico attraverso la spira

Il campo magnetico del rotore tende ad allinearsi con il campo statorico, andando a generale il movimento rotazionale, è poi presente un sistema di spazzole alla base del collettore, realizzato con una combinazione strisciante a tredici segmenti di rame e grafite. Quest'ultimo componente è proprio il responsabile della commutazione della corrente sulle spire rotoriche, che garantisce la continuità del movimento. Per diminuire quanto possile l'inerzia, l'asse di rotazione del rotore è stato realizzato in materiale non ferroso, rendendo il tutto più leggero. Gli attuatori in questione, lavorano ad una potenza nominale di 27 Watt, con una tensione tra ± 18 Volt. Nel caso di un motore in corrente continua con eccitazione costante (ricordando che parliamo di magneti permanenti) e controllato sulla tensione di armatura andiamo ad introdurre le seguenti relazioni matematiche:

$$V_a = R_a I_a + L_a \frac{d}{dt} I_a + E_m$$

$$E_m = K_e \omega$$

$$C_m = K_m I_a$$

Dove con V_a ed I_a si intendono la tensione di armatura, dunque applicata al rotore, e la corrente di armatura che vi scorre. R_a ed L_a sono invece parametri caratteristici che dipendono dalla costruzione del circuito di armatura, sono infatti la resistenza e l'induttanza di quest'ultimo. La E_m è invece la forza controelettromotrice che si

origina sul circuito rotorico, dovuta al fatto che il movimento rotatorio provoca una variazione del flusso di campo magnetico che attraversa la spira, questa grandezza è infatti proporzionale alla velocità di rotazione Ω tramite la costante K_e. Infine, la C_m è la coppia motrice erogabile dall'attuatore a meno della scatola di riduzione (che va a diminuire la velocità angolare, causando un conseguente aumento di coppia), questa grandezza è proporizionale alla I_a tramite la costante di coppia K_m.
I valori dei parametri utilizzati fanno però riferimento a quelli del motore Escap 28 DT-12, della stessa casa produttrice e assimilabile a quelli utilizzati per quanto riguarda dimensione e potenza, tutto ciò a causa di una scarsa disponibilità di informazioni sugli attuatori impiegati nell'ERICC.
Riportiamo qua di seguito i valori considerati per i parametri precedentemente illustrati:

R_a [Ω]	L_a [mH]	K_e $\left[\frac{V}{1000RPM}\right]$	K_m $\left[\frac{Nm}{A}\right]$
5.8	0.75	3.4	$13.4E-3$

Table 2.2: Valori dei parametri costruttivi del motore

2.3 Trasduttori

Per attuare il controllo del sistema dinamico considerato, abbiamo bisogno di ottenere un feedback da esso in termini di posizione e velocità angolare dei giunti, per la prima informazione si adoperano dei potenziometri, mentre delle dinamo tachimetriche per la seconda. Il potenziometro è un trasduttore di posizione, il concetto alla base del quale è impiegare il movimento meccanico ricevuto in input per influenzare una resistenza variabile, così facendo si va a variare la tensione data in output dal trasuttore. Il potenziometro viene posizione direttamente dull'asse di rotazione del giunto, nel caso in questione viene impiegato un potenziometro a strato a plastiche conduttrici, che garantisce una buona durata, affidabilità, resistenza ai disturbi ed un errore di linearita dello 0.05%. Si utilizza una costante di trasduzione di $44.4[mV/°]$.

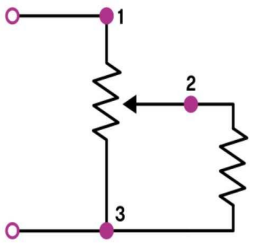


Figure 2.1: Schema circuitale di un potenziometro

Per quanto riguarda le dinamo tachimetriche, queste sono dei trasduttori di velocità,

che nel nostro caso vengono montate direttamente sull'asse di rotazione dei motori. Le dinamo altro non sono che dei motori a collettore usati in maniera inversa, normalmente si da in input dell'energia elettrica che viene trasformata dal motore in energia meccanica, con le dinamo si prende in ingresso una rotazione e si ottiene indietro la tensione di armatura risultante, questa si origina per il fatto che forzare una rotazione del rotore, immerso nel campo magnetico di eccitazione esterno, fa si che la variazione del flusso di quest'ultimo porti all'originarsi di una tensione ai capi del circuito d'armatura, che andiamo poi a leggere come dato in uscita dal trasduttore. Nella realizzazione di questi dispositivi, è importante avere una buona linearità e ridurre al minimo gli effetti indotti dall'isteresi magnetica e dalla temperatura.
L'equazione che rappresenta il funzionamento del potenziometro è:

$$V_u = K_t \cdot \omega$$

Dove K_t è la costante tachimetrica, ω è la velocità angolare del motore ed infine V_u è la tensione di uscita che si va a leggere dai morsetti di armatura della dinamo, nel caso in questione la costante tachimetrica è di 1Volt $\pm$ 2% ogni 1000 RPM.
Che questa sia una relazione valida solo in teoria è una precisazione importante, in primis perchè matematicamente assicura un perfetto rapporto di linearità tra la velocità angolare e la tensione in uscita dalla dinamo, quando in realta non è cosi e non si riesce davvero a raggiungere la linearità nel dispositivo reale, mentre in secondo luogo si riscontrano dei disturbi di carattere oscillatorio, causati dalla commutazione delle spazzole sull'armatura della dinamo.

2.4 Manipolatore, unità di controllo e schede di potenza

Come accennato in precedenza il manipolatore non viene fornito a sè, ma è presente un'unita di controllo nativa fornita anche di video terminale e un telecomando per il controllo da parte di un operatore. Il controllore nativo si presenta con una scheda madre sulla quale alloggiano le cinque schede di potenza, che hanno dunque il compito di trasformare i segnali logici di controllo in bassa tensione, in opportuni segnali che movimentino gli attuatori. Si era dunque già parlato della necessità di escludere tale logica di controllo, così da poter intervenire imponendo un'asservimento che partisse dal PC dedicato all'ERICC, per far ciò si è utilizzato un relè che permette di effettuare uno switch tra il controllo nativo e quello di seconda implementazione, andando fisicamente ed elettronicamente a scollegare uno dei due, il segnale di controllo a tale relè sarà poi comandabile dall'ambiente Simulink tramite un opportuna uscita configurata.
Il sistema sul quale si va a lavorare appare come segue:

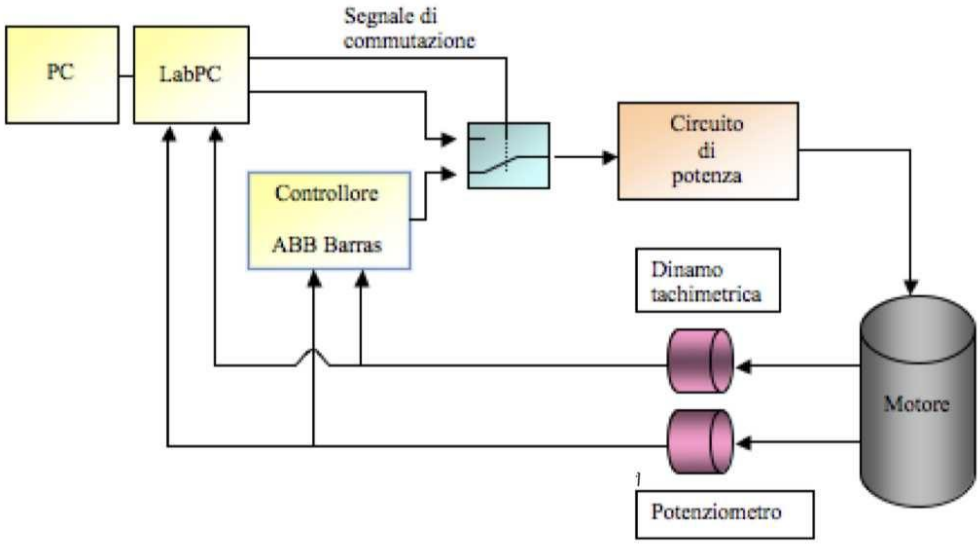


Figure 2.2: Schema di controllo del sistema

Per quanto riguarda l'interconnessione delle singole unità operative del nostro sistema vengono utilizzate le seguenti risorse.

2.4.1 Scheda di acquisizione dati

Per l'input e l'output di informazioni nel software di controllo viene impiegata una scheda di acquisizione dati PCI-6024E prodotta dalla National Instruments, montata sulla scheda madre del PC di laboratorio. Quest'organo hardware offre i seguenti ingressi ed uscite:

- 16 ingressi analogici, aventi frequenza di campionamento pari a 200 kH e conversione a 12 bit, di questi vengono utilizzati 4 ingressi analogici, 2 per il

campionamento delle posizioni dei giunti e 2 per campionamento indiretto delle correnti di armatura I_a ;

- 2 uscite analogiche, range di $\pm 10V$, entrambe utilizzate per inviare i segnali di controllo ai motori in corrente continua;
- 24 canali digitali configurabili come Input/Output raccolti in 3 BUS avendo dunque 8 canali per BUS, di questi ne vengono utilizzati 2 per comunicare con la scheda di interfaccia Robot-PC, questo perchè è proprio tramite queste uscite digitali che si vanno a dare i segnali di controllo ai relè, che permettono di escludere il controllore nativo lasciando strada libera al controllo da PC.

2.4.2 Scheda di interfacciamento

Questa scheda è l'organo hardware che permette realmente l'esclusione del controllore nativo, presenta infatti il relè del quale si è già parlato, così da permettere un dialogo con la scheda di acquisizione dati. Ma questa scheda di interfaccia non svolge solamente questo compito, questa ospita infatti anche un sistema di multiplexing che rende anche possibile il controllo di un terzo giunto. La scheda di acquisizione infatti avendo solamente due uscite analogiche non è adatta a comandare tre giunti, ma tramite questo sistema è possibile andare a fare il multiplexing dei segnali di controllo di due giunti su una singola uscita analogica ogni $T_c/2$. Non è però stata implementata questa variante, dato che come già spiegato andrebbe ad aggiungere della complessità che allontana dal concetto principale di questo progetto, e si rimane dunque sul modello di un manipolatore planare a due gradi di libertà.

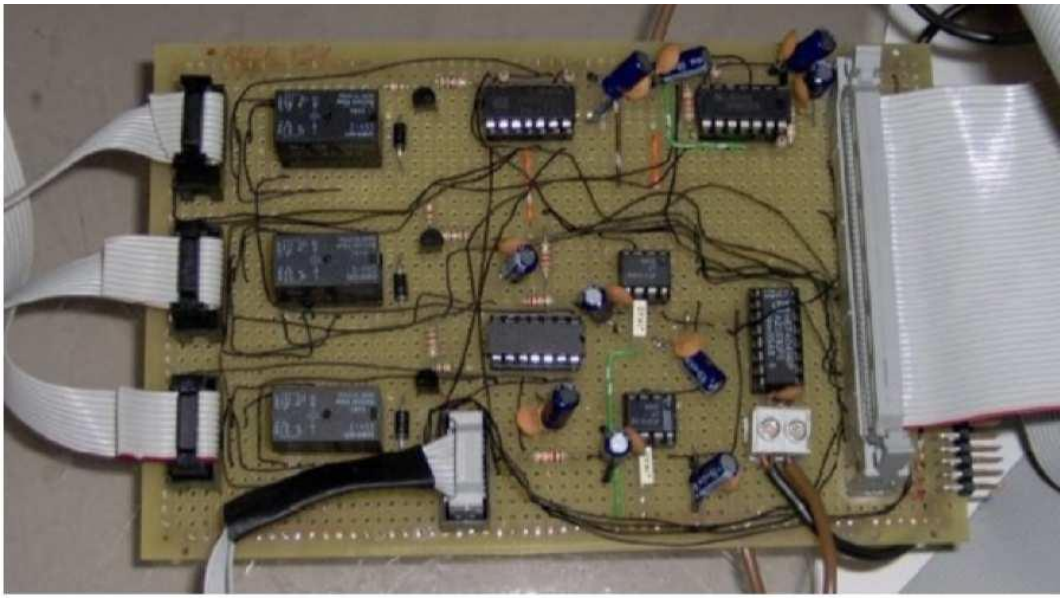


Figure 2.3: Scheda di interfaccia PC-Robot

2.4.3 Scheda aggiuntiva

La scheda aggiuntiva [2] si interpone tra scheda di interfacciamento PC-Robot e la scheda di acquisizione, il suo scopo è permettere di ottenere le correnti di armatura I_a ed inviare tali informazioni direttamente alla PCI-NI6024E. Ciò avviene grazie a dei resistori piazzati su questa scheda, uno per ogni giunto. Questo componente viene posto in una posizione nel circuito tale che i resistori risultano in serie al motori, così facendo si riesce a risalire all'informazione della corrente che scorre, calcolandola come $I_a = V_R/R$. Proprio per questo principio di funzionamento bisogna però precisare che la scheda consente unicamente l'acquisizione di tensioni single-ended, cioè con valori che si riferiscono a massa.

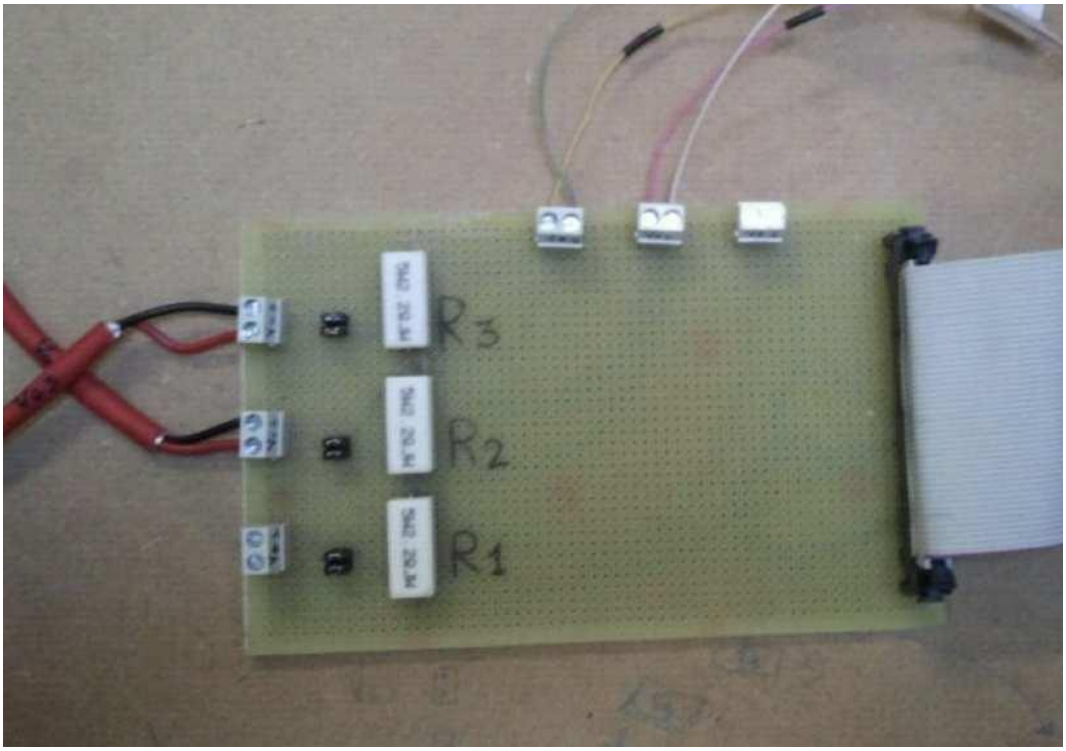


Figure 2.4: Scheda aggiuntiva per ottenere le correnti di armatura

2.4.4 Scheda di adattamento

Nel collegamento tra scheda di interfaccia PC-Robot e scheda di acquisizione dati bisogno fre i conti una problematica di natura hardware, ossia la prima menzionata fa Input/Output su un cavo flat da 50 linee, mentre la seconda ne usa uno da 68. Dunque questa scheda di adattamento altro non è che l'unione di due morsettiere a sè stanti, che vengono unite collegando tra loro i diversi pin secondo un maappatura ben precisa che verr riportata poi di seguito. Quest'approccio lascia anche il sistema aperto a cambiare scheda di acquisizione in base alle più svariate necessità, questo perchè nel caso si debba procedere in questo modo bisogna andare ad aggiornare solo la morsettiera che riferimento alla DAQ, ricollegando ovviamente i pin in maniera opportuna ma evitando di dover cambiare interamente l'apparato di adattamento.

Segnale	Scheda PC− Robot	PCI-NI6024E
posizione giunto 2	ACH1 = 2	AI1 = 33
posizione giunto 3	ACH = 3	AI2 = 65
velocità giunto 2	ACH = 5	AI4 = 28
velocità giunto 3	ACH = 6	AI5 = 60
tensione giunto 2	DAC1OUT = 12	AO1 = 21
tensione giunto 3	DAC0OUT = 10	AO = 22
tensione V_{R2}^{+}		AI12 = 61
tensione V_{R2}^{-}		AI13 = 26
tensione V_{R3}^{+}		AI14 = 58
tensione V_{R3}^{-}	PB1 = 23	AI1 = 23
abilitazione giunto 2	PB2 = 24	PO1 = 17
abilitazione giunto 3	AIGND = 9	AISENSE = 62
masse dei segnali analogici e digitali	AISENSE = 9	AIGND = 64
	AGND = 11	AOGND = 55, 54
	DGND = 13, 50	DGND = 18

Table 2.3: Mappatura pin Scheda di interfaccia-Scheda di acquisizione

Si riporta invece qui di seguito uno schema che riassume da un punto di vista visivo quali sono gli organi hardware dei quali si è parlato pocanzi, e come questi sono tra loro interconnessi.

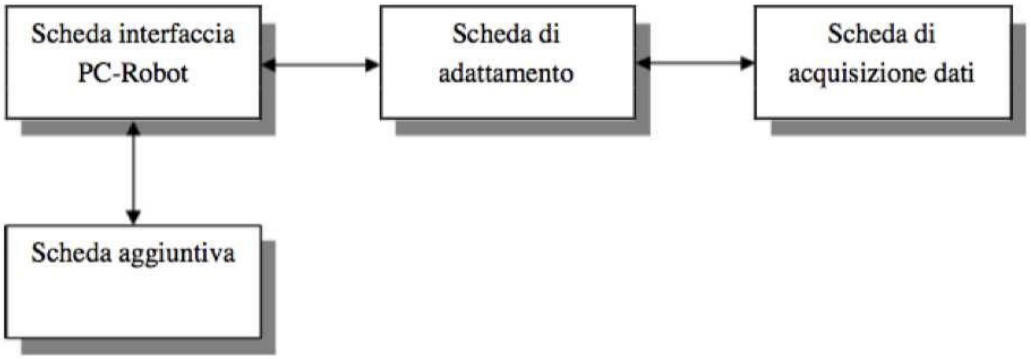


Figure 2.5: Schema di connessione del sistema

Chapter 3

Cinematica e dinamica del manipolatore

3.1 Introduzione al problema

Dopo aver caratterizzato in maniera qualitativa l'ERICC ed altri componenti che lavorano a stretto contatto con esso, bisogna invece capire come si rappresenta un manipolatore dal punto di vista matematico, sia per quanto riguarda la cinematica che per la sua dinamica. La cinematica di un manipolatore si preoccupa di studiare analiticamente lo spostamento nello spazio di quest'ultimo, dunque il suo moto ed i movimenti che esso può intraprendere, prescindendo dalle forze applicate per la sua movimentazione. Generalmente quello che interessa nello studio cinematico, è determinare e/o sfruttare un modello che metta in relazione lo spazio dei giunti e lo spazio di lavoro. Con "spazio dei giunti" si intende lo spazio matematico definito dalle coordinate dei giunti del robot. Ogni giunto rappresenta un grado di libertà e può essere descritto da un angolo (giunto rotoidale) o da una posizione lineare (giunto prismatico). Mentre con "spazio di lavoro" si intende l'insieme di tutti i punti nello spazio che l'end-effector del robot può raggiungere, è lo spazio fisico in cui opera. Stabilire un collegamento analitico tra gli elementi di questi due insiemi è di cruciale importanza perchè permette di poter risolvere le due tipologie di problemi cinematici:

- Cinematica diretta: permette di determinare posizione cartesiana ed orientamento dello spazio dell'organo terminale del manipolatore, a partire dall'informazione del valore delle variabili di giunto $[q_1(t), q_2(t), \ldots, q_n(t)]^T$ (n è il numero di gradi di libertà del sistema) e dei parametri geometrici del modello.
- Cinematica inversa: a partire da posizione/orientamento desiderato per l'end effector e dai parametri geometrici del modello, va a ricavare i valori necessari delle variabili di giunto per ottenere tale configurazione. La soluzione ricercata può non essere unica, può non esistere per nulla o esistere sotto forma di numero complesso (in questi ultimi due casi il significato concreto è che non esistono dei valori per lo spazio dei giunti che riescano a portare il manipolatore in quel determinato)

Per quanto riguarda invece la dinamica di un manipolatore robotico, questa si

occupa di studiare il moto conseguente all'applicazione di forze o coppie ai giunti. Lo studio della dinamica di un robot è fondamentale per quanto riguarda la modellizzazione del processo fisico da controllare, che permette poi l'implementazione di un software di simulazione, tappa immancabile prima di applicare il controllo al modello reale. Per approcciarsi allo studio della dinamica di un robot si può procedere in due modalità

- Metodo di Lagrange: Il metodo di Lagrange si basa sulla differenza tra energia cinetica e energia potenziale del sistema, utilizza le equazioni di Lagrange per derivare il modello dinamico. È concettualmente semplice perché consente di ottenere il modello in una forma chiusa, cioè tutte le equazioni sono espresse in termini delle coordinate generalizzate (ad esempio, angoli dei giunti) e delle loro derivate. Tuttavia può essere computazionalmente oneroso per manipolatori con molti gradi di libertà, questo perchè viene richiesta l'esecuzione di calcoli con matrici di grandi dimensioni
- Metodo di Newton-Eulero: Il metodo di Newton-Eulero si basa sulle leggi del moto di Newton e sui principi di Eulero per rotazioni. Questo metodo sviluppa il modello dinamico utilizzando equazioni differenziali che descrivono le forze e i momenti in ogni giunto e link del robot. È computazionalmente più efficiente perché sfrutta la struttura a catena aperta del manipolatore, permettendo una risoluzione ricorsiva delle equazioni. Questo rende il calcolo più veloce e adatto per l'implementazione in tempo reale. Purtroppo però proprio questo vantaggio in termini computazionali si porta dietro una maggiore complessità e il bisogno di più attenzione ai dettagli per la derivazione delle equazioni.

Nel corso di questo lavoro, si farà riferimento ad un modello dinamico ottenuto mediante la formulazione di Lagrange.

3.2 Parametrizzazione di Denavit-Hartenberg

[3] Fornisce un approccio sistematico e generalizzato che sfrutta le matrici algebriche per rappresentare matematicamente la struttura del braccio e risolvere il problema della cinematica diretta (soluzione valida nei casi di catene cinematiche aperte). Questo metodo utilizza una matrice di trasformazione omogenea 4x4 per descrivere la relazione spaziale fra due segmenti meccanici rigidi adiacenti e sfruttando il fatto che è possibile adoperare un operatore di composizione per iterare tale ragionamento su n segmenti consecutivi si arriva a:

$$\boldsymbol{T}_n^0(\boldsymbol{q}) = \boldsymbol{A}_1^0\left(q_1\right)\boldsymbol{A}_2^1\left(q_2\right)\ldots\boldsymbol{A}_n^{n-1}\left(q_n\right)$$

(con n che è il numero dei gradi di libertà) dunque si riduce il problema della cinematica diretta alla ricerca di una equivalente matrice di trasformazione. Secondo l'approccio di Denavit-Hartenberg si numerano da 1 ad n i giunti del manipolatore,

mentre da 0 ad n i link (segmenti meccanici del robot) che in numero sono $n+1$, prendendo come link 0 quello adiacente alla base del manipolatore, mentre il link $n-esimo$ corrisponde con l'end effector. Nella parametrizzazione, facendo riferimento al giunto $i-esimo$ si considera quello che unisci i link i ed $i+1$. Detto ciò, la prima cosa da effettuare è la determinazione degli assi di giunto e di corpo. Per un giunto prismatico, l'asse di giunto è la retta parallela alle generatrici del prisma e passante per il baricentro. Per un giunto rotazionale, l'asse di giunto coincide con l'asse di rotazione. L'asse di corpo è definito come la normale comune agli assi di giunto adiacenti. A questo punto si fissa il sistema di coordinate del giunto $i-esimo$ con la seguente convenzione:

- L'asse z_i si sceglie lungo l'asse del giunto $i+1$, il verso è arbitrario,
- L'origine O_i del sistema di riferimento $i-esimo$ viene posta lungo z_i, nell'intersezione tra l'asse stesso e la normale comune tra z_i-1 e z_i, analogamente fissiamo O_i' sull'intersezione situata lungo l'asse z_i-1.
- L'asse x_i si prende lungo la normale comune tra gli assi z_i-1 e z_i con direzione dal giunto i al giunto $i+1$,
- Si prende infine y_i in maniera tale da formare una terna destrorsa con gli altri assi;

chiarita la convenzione per la scelta dei sistemi di riferimento solidali ai giunti del manipolatore, si introducono ora i parametri che descrivolono la geometria di quest'ultimo:

- a_i: distanza tra O_i e O_i'
- d_i: distanza tra coordinata di O_i' lungo z_i-1
- α_i: angolo compreso tra gli assi z_i-1 e z_i attorno a x_i, considerato positivo quando la rotazione è fatta in senso antiorario
- θ_i: angolo compreso tra gli assi x_i-1 e x_i attorno a z_i, considerato positivo quando la rotazione è fatta in senso antiorario

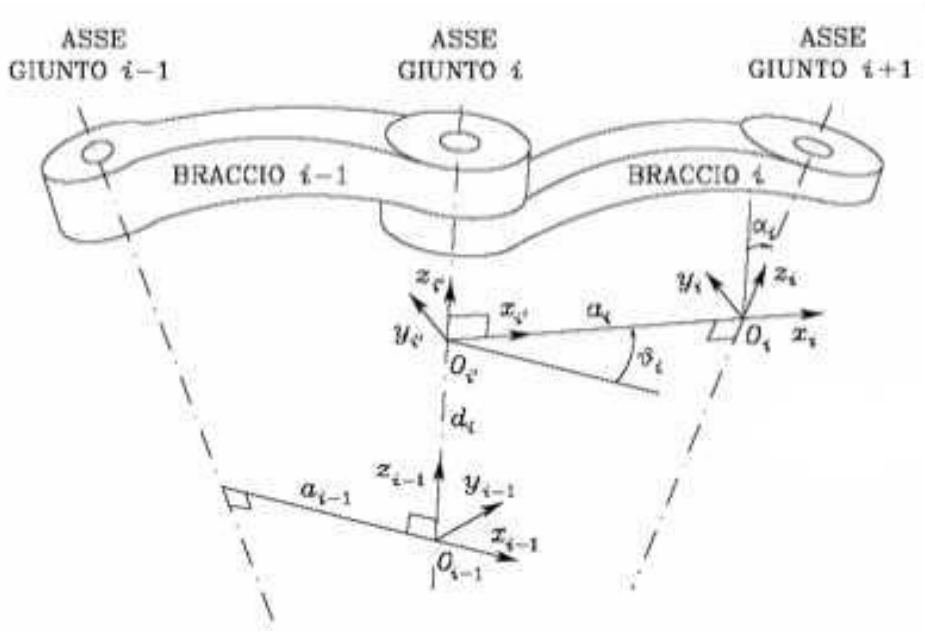


Figure 3.1: Sistemi di riferimento dei giunti e parametri di D.H.

Per unificare la definizione analitica per le variabili di giunto, si sfruttanp le coordinate generalizzate $q_i, i = 1, .., n$, definite come segue:

$$q_i = (1 - \sigma_i)\theta_i + \sigma_i d_i$$

dove $\sigma_i = 1$ se il giunto $i - esimo$ è prismatico e $\sigma_i = 0$ se è rotazionale.
Inoltre, si sfrutta la nozione di forze generalizzate Q_i, che in funzione della natura del giunto sarà:

- τ_i: coppia applicata, per i giunti rotoidali;
- f_i: forza applicata, per i giunti prismatici;

Infine, è possibile sfruttare gli elementi precedentemente introdotti per ottenere una matrice di trasformazione che permetta il trasferimento di coordinate tra i sistemi di riferimento $(i-1) - esimo$ ed $i - esimo$:

$$A_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.3 Equazioni di Lagrange

Come già accennato, il modello dinamico di un manipolatore fornisce una descrizione della relazione esistente tra la coppia (o la forza) fornita dagli attuatori

dei giunti ed il movimento della struttura meccanica. La determinazione del modello tramite l'utilizzo della formulazione di Lagrange, permette di giungere ad una soluzione in maniera sistematica. Le equazoni che si ottengono poi, comprendono vari conributi che vanno ad avere effetto sul movimento del manipolatore, come il carico inerziale, forza centrifuga, di Coriolis e la gravità; ovviemente tutti questi fattori dipendono sia da specifiche configurazioni e posizioni assunte dal robot, che da parametri che caratterizzano quest'ultimo. La formulazione parte dall'introdurre la Lagrangiana, definita come segue:

$$L = T - U \tag{3.1}$$

dove dunque si va ad identificare questa quantità come differenza tra:

- T: Energia cinetica associata al movimento delle parti del manipolatore. Dipende dalla velocità e dalla massa di ciascun elemento del sistema.
- U: Energia potenziale associata alla posizione delle parti del manipolatore rispetto a un riferimento.

Andiamo poi ad espriremere le equazioni di Lagrange per il giunto i-esimo come :

$$Q_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \quad i = 1, \ldots, n \tag{3.2}$$

dove Q_i è la forza generalizzata associata al giunto i-esimo, chiaramente parleremo di forza nel caso di un giunto prismatico, e di coppia nel caso rotoidale.

- La prima parte dell'equazione $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right)$ rappresenta il cambiamento nel tempo della quantità di moto coniugata associata a q_i. Questa quantità di moto è legata alla velocità del manipolatore e alla sua inerzia.
- La seconda parte, $\frac{\partial L}{\partial q_i}$ rappresenta le forze generalizzate che agiscono sul sistema, derivanti dall'energia potenziale.

Alle forze generalizzate viene aggiunto il contributo delle forze non conservative agenti sul sistema, coppia generata dagli attuatori, coppia di attrito degli attuatori e coppia indotta sui giunti dall'interazione tra l'end effector e l'ambiente circostante. Grazie al risultato raggiunto si è in grado di risalire al modello dinamico del manipolatore, partendo unicamente dall'analisi da un punto di vista meccanico del sistema, andando a determinare l'energia cinetica e l'energia potenziale. Rappresentando le equazioni in forma matriciale più compatta, si noti come a valle di tutti i conti si arrivi a tale equazione:

$$B(q(t)) \cdot \ddot{q}(t) + C(q(t), \dot{q}(t)) \cdot \dot{q}(t) + F_v \cdot \dot{q}(t) + F_S \cdot \mathrm{sgn}(\dot{q}(t)) + g(q(t)) = Q(t) \tag{3.3}$$

Considerando:

- $Q(.) \in \Re^n$ è il vettore delle forze generalizzate applicate ai giunti $i = 1, 2, \ldots, n$;
- $q(.) \in \Re^n$ è il vettore delle posizioni generalizzate;
- $\dot{q}(.) \in \Re^n$ è il vettore delle velocità generalizzate;
- $\ddot{q}(.) \in \Re^n$ è il vettore della accelerazioni generalizzate;
- $B(q(.))$ è la *matrice d'inerzia* $(n \times n)$ che risulta essere simmetrica, definita positiva e dipendente dalla configurazione del sistema.
- $C(q(.), \dot{q}(.))$ è una matrice $(n \times n)$ che descrive gli effetti centrifughi e di Coriolis, dipendente dalle posizioni e dalle velocità generalizzate.
- F_v è la matrice diagonale $(n \times n)$ dei coefficienti di attrito viscoso;
- F_s è la matrice diagonale $(n \times n)$ che contribuisce a ottenere un modello semplificato dell'attrito statico mediante la determinazione delle coppie di attrito coulumbiano $f_s = F_s \operatorname{sgn}(\dot{q})$ in cui $\operatorname{sgn}(\dot{q})$ indica il vettore $(n \times 1)$ le cui componenti sono date dalle funzioni segno delle velocità dei singoli
- $g(q(.)) \in \Re^n$ è il vettore che tiene conto della forza di gravità.

Le matrici precedentemente introdotte avranno la seguente struttura:

$$B(q) = \begin{bmatrix} b_{11}(q_3) & b_{12}(q_3) \\ b_{21}(q_3) & b_{22} \end{bmatrix}$$

$$C(q, \dot{q}) = h(q_3) \begin{bmatrix} \dot{q}_3 & (\dot{q}_2 + \dot{q}_3) \\ -\dot{q}_2 & 0 \end{bmatrix}$$

$$G(q) = \begin{bmatrix} G_1(q_2, q_3) & G_2(q_2, q_3) \end{bmatrix}^T$$

Dove i singoli elementi vengono calcolati come:

- $b_{11}(q_3) = I_{\ell_2} + m_{\ell_2}\ell_2^2 + k_{r_2}^2 I_{m_2} + I_{\ell_3} + m_{\ell_3}\left(a_2^2 \quad + \ell_3^2 + 2a_2\ell_3\cos(q_3)\right) + I_{m_3} + m_{m_3}a_2^2$
- $b_{12}(q_3) = b_{21}(q_3) = I_{\ell_3} + m_{\ell_3}\left(\ell_3^2 + a_2\ell_3\cos(q_3)\right) + k_{r_3}I_{m_3}$
- $b_{22} = I_{\ell_3} + m_{\ell_3}\ell_3^2 + k_{r_3}^2 I_{m_3}$
- $h(q_3) = -m_{\ell_3}a_2\ell_3\sin(q_3)$
- $G_1(q_2, q_3) = (m_{\ell_2}\ell_2 + m_{\ell_3}a_2)\, g\cos(q_2) + m_{\ell_3}\ell_3 g\cos(q_2 + q_3)$
- $G_2(q_2, q_3) = m_{\ell_3}\ell_3 g\cos(q_2 + q_3)$

dove g è l'accelerazione di gravità, l_i, $i = 2, 3$ è la distanza dal giunto al centro di massa del link $i - esimo$, a_i, $i = 2, 3$ è la lunghezza del link $i - esimo$, m_{l_i}, $i = 2, 3$ è la massa del link $i - esimo$, I_{l_i}, $i = 2, 3$ è il momento d'inerzia del link $i - esimo$, I_{m_i}, $i = 2, 3$ è l'inerzia del motore del giunto $i - esimo$ ed infine k_{r_i}, $i = 2, 3$ è il rapporto di trasmissione del giunto $i - esimo$. L'equazione matriciale della dinamica del manipolatore sarà di fondamentale importanza nell'attuazione del controllo, perchè come si mostrerà in seguito e proprio lì che andrà imposta la condizione che assicura la convergenza dell'errore tra riferimento e traiettoria reale.

Chapter 4

Interfacciamento tra i PC

4.1 Perchè un altro PC

Nel progetto sviluppato, l'utilizzo della computer vision per l'ottenimento della posizione finale voluta per il manipolatore, è un obiettivo divenuto chiaro sin da subito. Per questo motivo, essendo gli algoritmi di CV piuttosto pesanti, è stato immediato comprendere che questi non potessero girare sul computer dell'ERICC, dovendo dunque essere ospitati su un computer esterno. Quest'ultimo deve essere in grado di comunicare con quello presente in laboratorio, scambiando informazioni durante l'esecuzione del programma.

4.2 Collegamento fisico e setup della comunicazione ethernet

L'interconnessione viene effettuata tramite un cavo ethernet, che nel retro del PC dell'ERICC va connesso alla porta ethernet situata nella parte più sopraelevata, in quanto direttamente collegata alla scheda di rete.

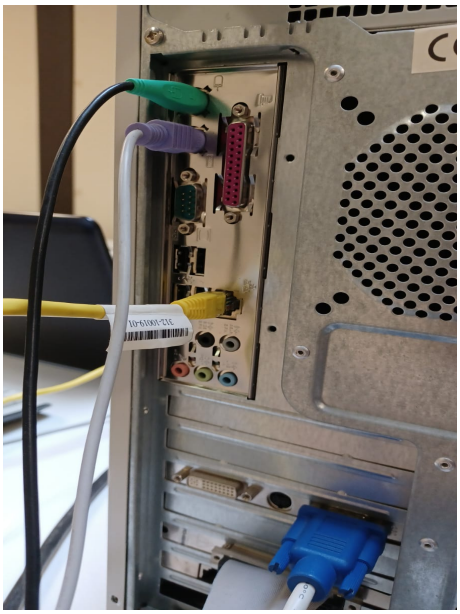


Figure 4.1: Cavo ethernet, lato PC dell'ERICC

La procedura che verrà introdotta successivamente permette di effettuare il setup della connessione (si precisa che nel caso in questione è veritiero per quanto riguarda Windows 10, utilizzato nello sviluppo di questo progetto). Dalle impostazioni: stato/centro e connessioni di rete/impostazioni di condivisione avanzate, abilitare poi la possibilità di condividere file e stampanti, e disattivare la necessità di inserire una password per connettersi alla rete. Nell'interfaccia della corrente pagina delle impostazioni il risultato che bisogna ottenere è il seguente:

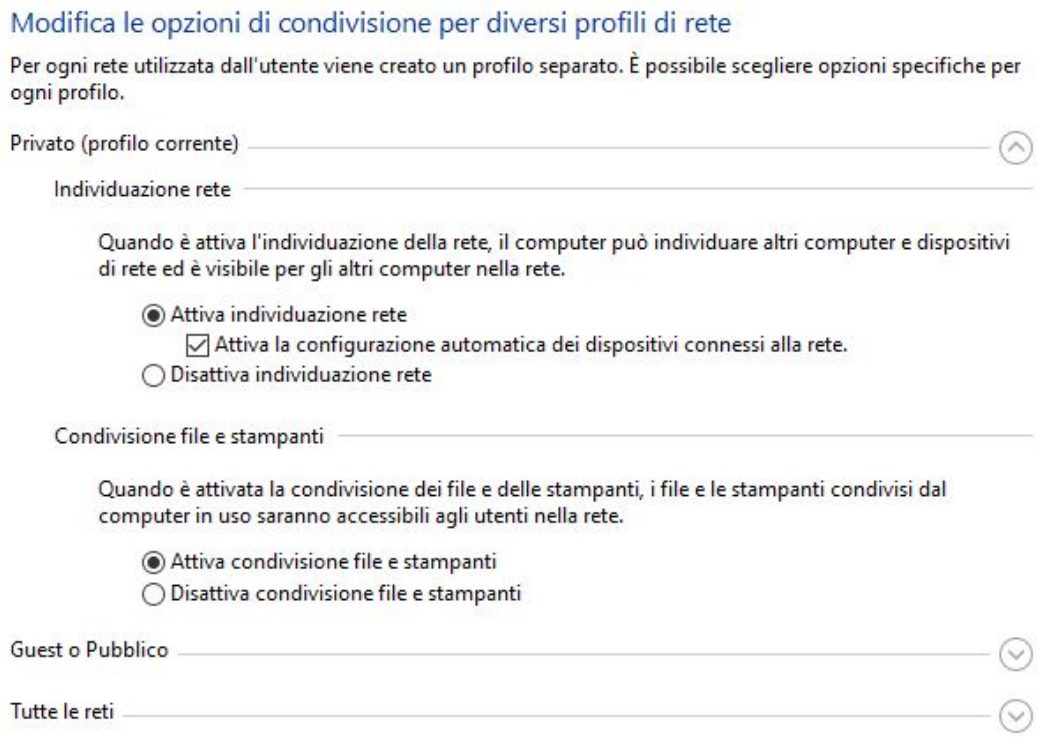


Figure 4.2: Interfaccia con modifiche per permettere la connessione

Come ultimo step, bisogna posizionarsi all'interno di: stato/modifica opzioni scheda/ethernet/proprietà/tcp-ip/ipv4/proprietà. Nell'interfaccia, bisogna riempire i campi con le informazioni riportate a seguire.

IP	192.168 .1 .1
Subnet mask	255.255 .255 .0
DNS	8.8 .8 .8
DNS alternativo	8.8 .4 .4

Table 4.1: Valori da inserire, lato PC esterno

4.3 Abilitazione SMB1 sul PC esterno

SMB1 è la prima versione del protocollo Server Message Block (SMB), un protocollo di rete utilizzato per la condivisione di file, stampanti e altre risorse tra computer all'interno di una rete locale. Nelle moderne versioni di windows, Microsoft ha iniziato a disabilitare di default SMB1, indirizzando l'utilizzo di SMB2 o SMB3, che garantiscono migliori prestazioni, affidabilità e sicurezza. Si deve perciò provvedere a riabilitare SMB1, dato che il computer dell'ERICC ospita Windows XP che ovviamente non supporta le versioni più moderne del protocollo. Dunque, dal pannello di controllo selezionare visualizzazione per categoria e si procede come segue: Programmi/Programmi e funzionalità/Attiva o disattiva funzionalità di Windows, da qui individuare "Supporto per condivisione file SMB 1.0/CIFS" e mettere la spunta sulle voci "Client SMB 1.0/CIFS" e "Server SMB 1.0/CIFS", effettuare infine il riavvio della macchina. Successivamente entrare nella sezione Servizi, e impostare l'avvio del servizio "Pubblicazione risorse per individuazione" in modalità automatica. Si può a questo punto, tramite Esplora file/Rete, assicurarsi che effettivamente la risorsa di rete in questione sia visibile. Se tutto è visibile e senza alcun tipo di errori, bisogna poi impostare sulla rete il nome del WORKGROUP uguale su entrambi i computer. Successivamente, sul computer dell'ERICC dalla sezione Rete bisogna creare la rete domestica, e seguendo la procedura guidata sarà proprio quì che andrà inserito il nome del workgroup del quale si è parlato prima. Riempire quanto richiesto con le informazioni riportate di seguito:

IP	192.168 .1 .2
Subnet mask	255.255 .255 .0
DNS	8.8 .8 .8
DNS alternativo	88.8 .4 .4

Table 4.2: Valori da inserire, lato PC dell'ERICC

4.4 Remote Desktop Protocol

Nelle fasi finali del progetto, al fine di rendere l'esecuzione del sistema di controllo su due dispositivi il più unificato possibile, si è voluto instaurare un controllo remoto del PC dell'ERICC da parte del PC esterno, cosicchè quest'ultimo possa essere determinante anche per l'avvio di script MATLAB, eliminando il bisogno di mettere le mani su entrambi i computer per far lavorare il sistema. Per adempiere anche a questo scopo, si deve accedere come amministratori al PC dell'ERICC e successivamente: Risorse del computer/Proprietà/Remote/Abilitare desktop remoto/Consenti agli utenti di connettersi in remoto a questo computer, e dopodichè

esplicitare conferma per gli utenti abilitati. Così facendo dal Pc esterno (che a questo punto diventa l'organo centrale del sistema) accedere a "Connessione desktop remoto" ed inserire le credenziali del pc di laboratorio del robot.

4.5 Accesso alla memoria negli script MATLAB

La connessione creata non solo ci permette di far scambiare tra i computer le informazioni riguardanti la visione, che vengono estrapolate dal PC portatile e passate al PC fisso, ma bensì è di cruciale importanza anche per il recupero degli output di esecuzione, che vengono salvati localmente sul PC dell'ERICC tramite appositi comandi come "save" e "load". Grazie alla comunicazione tra i due calcolatori possiamo infatti accedere alla memoria del computer dell'ERICC e recuperare le informazioni che si vogliono visionare.
In tutti i programmi al fine di consentire lo scambio dei valori (relativamente a valori di posizione, output dell'esecuzione del movimento del manipolatore, ecc.), all'interno dei programmi matlab sono presenti spesso funzioni 'save' e 'load' utilizzate in maniera specifica al fine di salvare e leggere dati nella e dalla memoria del PC collegato al manipolatore sfruttando la connesione creata.

Chapter 5

Legge di controllo

5.1 Introduzione alla stabilità di Lyapunov

Per comprendere a fondo la natura del Variable Structure Control (VSC), è utile introdurre dapprima il concetto della stabilità di Lyapunov, quest'ultimo infatti sarà proprio quello strumento che ci assicurerà il funzionamento della tecnica di controllo oggetto di questo capitolo. La stabilità di Lyapunov è una nozione fondamentale nella teoria dei sistemi dinamici, utilizzato per analizzare il comportamento di un sistema in prossimità di un punto di equilibrio. Essa fornisce un metodo per determinare se, e in che misura, le perturbazioni iniziali si attenueranno o cresceranno nel tempo. Consideriamo un sistema del tipo:

$$\dot{x}(t) = f(x(t)), \quad x(t) \in \mathbb{R}^n$$

dove $x(t)$ rappresenta lo stato attuale al tempo t, e $f(x(t))$ descrive la dinamica del sistema. Un punto x^* è un punto di equilibrio per il sistema se e solo se $f\,(x^*) = 0$.
Nel caso in cui si vada a lavorare con sistemi dove $\dot{x}(t) = f(x(t))$ è un insieme di relazioni lineari, questo può essere rappresentato in forma matriciale come: $\dot{x} = Ax$.
Si noti che nei sistemi che hanno questa descrizione matematica l'origine $x = 0$ è un punto di equilibrio per $\dot{x} = Ax$.
Si introducono di seguito a funzione di Lyapunov $V(x)$, e le condizioni che questa deve rispettare.

Teorema

Sia $V(x) : \mathbb{R}^n \Rightarrow \mathbb{R}$ una funzione continua e con derivata prima continua, tale che:

$$V(0) = 0 \tag{5.1}$$

$$V(x) > 0, \quad \forall x \neq 0 \tag{5.2}$$

$$\dot{V}(x) < 0, \quad \forall x \neq 0 \tag{5.3}$$

Allora $x = 0$ è un punto di equilibrio asintoticamente stabile.
Per capire il significato dietro questa definizione matematica, bisogna pensare al fatto che la $V(x)$ restituisce in output uno scalare, dunque una quantità, e che

quest'ultima è funzione dello stato del sistema. Dunque il fatto che la derivata $\dot{V}(x)$ sia strettamente negativa significa che la quantità della quale si parlava pocanzi, tenderà sempre a descrescere fino ad arrivare a 0, dunque viene da sé intuire che lo spazio di stato si porterà nel punto $x = 0$, ed è facile notare come per i sistemi del tipo $\dot{x} = Ax$ questo rappresenti sempre un punto di equilibrio in quanto, $\dot{x} = A \cdot 0 \Rightarrow \dot{x} = 0$, e dato che l'evoluzione in spazio di stato tende verso quel punto, si ha che quest'ultimo è un punto di equilibrio asintoticamente stabile (dunque attrattivo). Per quanto invece riguarda i sistemi non lineari, il punto $x = 0$ non è necessariamente un punto di equilibrio, tuttavia se le condizioni espresse dal teorema di Lyapunov sono rispettate, si ha comunque la garanzia che il sistema sia asintoticamente stabile. Se nella terza condizione da verificare non si ha che $\dot{V}(x) < 0$, ma bensì $\dot{V}(x) \leq 0$, ciò sta a significare che esistono punti in cui la derivata si eguaglia a 0, e dunque la $V(x)$ non tende a decrescere ma si mantiene costante, suggerendo quindi la non asintotica stabilità del sistema, che invece sarà semplicemente stabile dato che la traiettoria nello spazio di stato si mantiene limitata.

5.2 VSC e SMC

[4] Il VSC, ossia il $VariableStructureControl$, è una tecnica che lavora in retroazione dallo stato nella quale il tratto distintivo è una continuità a tratti nella variabile di controllo. Infatti ogni componente di quest'ultima, nel corso dell'evoluzione nello spazio di stato, può assumere due forme differenti entrambe funzioni continue dello stato, il passaggio da una all'altra prende il nome di *switching*, che a tutti gli effetti rappresenta un cambiamento della struttura di controllo. In funzione dei momenti nei quali avviene questa commutazione, si andranno ad individuare delle discontinuità per la variabile di controllo, ed è proprio la posizione e la frequenza delle commutazioni che determinano la "logica" con la quale si va ad effettuare lo switching. I punti di discontinuità sono posizionati lungo un numero n di superfici nello spazio di stato dove n è proprio la dimensione del controllo che si sta attuando, ogni superficie è infatti relazionata ad una specifica componente del controllo, quest'ultima subisce il cambio strutturale nel momento in cui lo stato va ad intersecarsi con la superficie ad essa corrispondente. Il concetto alla base dell'utilizzo di questa tecnica di controllo, è il voler far sì che oltrepassato un transitorio iniziale di stabilizzazione lungo le superfici, lo stato raggiunga una coincidenza più o meno precisa con esse ed una volta raggiunte vi rimanga per il resto dell'evoluzione del sistema. Dunque scegliendo la forma matematica di queste superfici in maniera opportuna, si è in grado di andare ad imporre al sistema le specifiche desiderate. Questa modalità di movimento nello spazio di stato, prende il nome di SMC, ossia *Sliding Mode Control*, tecnica molto utilizzata all'interno della più ampia famiglia delle tecniche di controllo a struttura variabile.

Alcuni vantaggi dell'utilizzo dell'approccio in sliding mode sono:

- Robustezza rispetto alle perturbazioni e alle incertezze di modello, cruciale in questo caso specifico di applicazione.
- Semplicità di implementazione, nonostante la discontinuità nella legge di controllo, il controllo a sliding mode può essere relativamente semplice da implementare rispetto ad altre tecniche non lineari.

Mentre alcuni svantaggi possono essere:

- Chattering, oscillazioni ad alta frequenza che possono verificarsi quando il sistema cerca di mantenersi sulla superficie di sliding. Questo può causare problemi pratici, come l'usura nei sistemi meccanici.
- Disturbi ad alta frequenza, il SMC spesso introduce componenti ad alta frequenza nel segnale di controllo per compensare le incertezze e i disturbi. Tuttavia, qualora i disturbi reali fossero presenti anche ad alta frequenza, possono sovrapporsi al segnale di controllo rendendo difficile la distinzione tra il disturbo e la parte utile del segnale, andando a rendere ancora più complessa la progettazione del controllore.

5.3 Sintesi di un controllore

[5] La sintesi di una legge di controllo che sfrutti quanto detto fin'ora, si può dividere in due fasi:

- Inizialmente si deve scegliere una regione di spazio di dimensione $n \times m$ sulla quale avverrà lo scivolamento. Successivamente si va a definire matematicamente tale superficie (che come detto avrà tante componenti quanto è la dimensione dello stato):

$$s(x) = 0 \tag{5.4}$$

  con $s(x) : \mathbb{R}^n \Rightarrow \mathbb{R}^m$, dove n è la dimensione dello spazio di stato, ed m la dimensione dello sforzo di controllo.

- In seguito, si vanno a determinare le m componenti di controllo, quindi quelle funzioni $u_i(x,t)$ con $i = 1, \ldots, m$ in modo tale che la (5.1) da un certo istante t in poi possa considerarsi verificata. Come accennato in principio nella descrizione di queste tecnica, le funzioni in questioni si presentano come segue:

$$u_i(x,t) = \begin{cases} u_i^+(x,t) & \text{se} \quad s_i(x) > 0 \\ u_i^-(x,t) & \text{se} \quad s_i(x) < 0 \end{cases} \tag{5.5}$$

  $i = 1, \ldots, m$ con $u_i^+(x,t), u_i^-(x,t)$ funzioni continue, $u_i^+(x,t) \neq u_i^-(x,t)$ su $s_i(x) = 0$.

Per far sì che il sistema di controllo in questione funzioni, e si riesca dunque ad avere un moto di scivolamento nell'intorno della superficie definita alla (5.2), bisogna controllare che siano rispettate le *condizioni di esistenza dello sliding mode*. Per comprendere inizialmente il concetto dietro a tali condizioni, si consideri una $s(x)$ con $m = 1$, $s(x) : \mathbb{R}^n \Rightarrow \mathbb{R}^m$, dunque per sistemi aventi uno scalare come sforzo di controllo. Nel caso descitto, ledisuguaglianze che descrivono le condizioni di esistenza sono:

$$\lim_{s\to 0^-} \dot{s} > 0, \quad \lim_{s\to 0^+} \dot{s} < 0 \tag{5.6}$$

Che in altra via, possono essere riassunte con:

$$\lim_{s\to 0} s\dot{s} < 0 \tag{5.7}$$

Analizzando il caso con $x \in \mathbb{R}^2$ è possibile apprezzare un'interpretazione geometrica in maniera chiara. Si noti che è possibile esprimere $\dot{s}$ come:

$$\dot{s} = \nabla s \cdot \dot{x} \tag{5.8}$$

ossia il prodotto scalare tra il vettore velocità $\dot{x}$, tangente alla traiettoria di s, ed il gradiente di quest'ultima ∇s. Dunque in termini concreti si intende che, quando $s(x)$ si avvicina alla superficie ∇s e $\dot{x}$ devono essere in una precisa posizione reciproca, andando a vincolare il segno del loro prodotto scalare:

- $s < 0$: ∇s e $\dot{x}$ devono formare un angolo acuto
- $s > 0$: ∇s e $\dot{x}$ devono formare un angolo ottuso

dunque le traiettorie devono essere dirette verso la superficie $s(x) = 0$, come si può osservare di seguito.

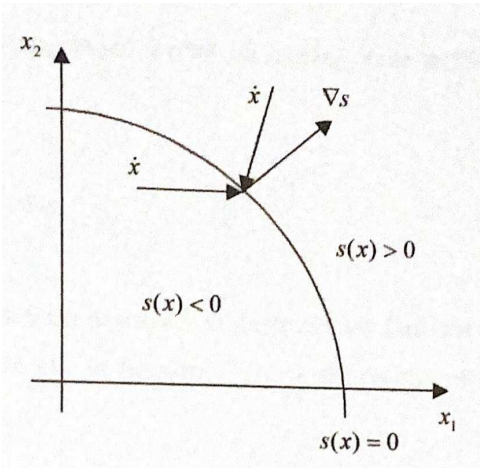


Figure 5.1: Interpretazione geometrica delle condizioni d'esistenza dello sliding mode

5.4 Applicazione del principio al caso specifico

Sia q_r la traiettoria di riferimento nello spazio dei giunti, e siano $\dot{q_r}$, $\ddot{q_r}$ velocità e accelerazione di riferimento. Considerando che le grandezze precedentemente introdotte sono vettori 2×1, dato che come si è già chiarito si andrà ad implementare il controllo sul secondo ed il terzo giunto dell'ERICC. Si definisca l'*errore di tracking* $e(t)$: $e(t) = q(t) - q_r(t)$.
E perchè il controllo funzioni, si vuole:

$$\lim_{t\to\infty} e(t) = 0 \Rightarrow q_r(t) = q(t); \quad t \to \infty$$

Si introduce ora la *superficie di sliding*:

$$s(t) := \dot{e}(t) + \Lambda e(t); \quad con \quad \lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}; \lambda_1, \lambda_2 > 0$$

Si noti che se $s(t) = 0$ si ha:

$$\begin{cases} \dot{e}_1(t) + \lambda_1 e_1(t) = 0 \\ \dot{e}_2(t) + \lambda_2 e_2(t) = 0 \end{cases}$$

E risolvendo entrambe le equazioni differenziali, si può notare che $e(t) \to 0$ come $e^{-\lambda t}$, dunque il fatto che ci si trovi sulla superficie $s(t) = 0$ assicura la convergenza a 0 dell'errore. Il problema si sposta ora su un altro fonte, ossia si deve individuare uno sforzo di controllo, che possa portare al verificarsi della condizione $s(t) = 0$.
Partiamo dall'analizzare la stabilità di Lyapunov per quanto riguarda l'evoluzione nel tempo di $s(t)$, si prende:

$$V(s) = \frac{1}{2} s(t)^\top s(t)$$

e si vogliono verificare le condizioni (5.1), (5.2), (5.3).
Le prime due sono rispettate di per sè, per come è stata scelta la funzione $V(s)$, mentre venendo alla (5.3), in sostanza si sta imponendo che $\frac{1}{2}\|s(t)\|^2$ decresca sempre, e dunque che:

$$\|s(t)\| \to 0 \quad \Rightarrow \quad s_1(t), s_2(t) \to 0; \qquad t \to \infty$$

Dunque si ha:

$$\dot{V}(s) < 0 \Leftrightarrow \frac{1}{2} s^\top(t)\dot{s}(t) + \frac{1}{2}\dot{s}^\top(t)s(t) < 0 \Leftrightarrow s^\top(t)\dot{s}(t) < 0$$

Bisogna capire come è fatta $\dot{s}(t)$:

$$\dot{s}(t) = \ddot{e}(t) + \Lambda\dot{e}(t) = \ddot{q}_r(t) - \ddot{q}(t) + \Lambda\dot{e}(t)$$

E se dalla (3.3) viene esplicitata $\ddot{q}$ possiamo riscrivere $\dot{s}(t)$ come:

$$\dot{s}(t) = B(q)^{-1}\left(-C(q,\dot{q})\dot{q} - F_v\dot{q}(t) - G(q) + u(t)\right) + d(t,q,\dot{q}) - \ddot{q}_r(t) - \Lambda\dot{e}(t) \quad (5.9)$$

E ricordando che stiamo considerando $s(t) = 0$, possiamo prendere $u(t)$ come:

$$u(t) = C(q,\dot{q})\dot{q} + F_v\dot{q} + G(q) + B(q)\left(\ddot{q}_r(t) - \Lambda\dot{e}(t) + v(t)\right) \quad (5.10)$$

con $v(t)$ ingresso ausiliario, la forma del quale è da determinarsi affinche sia vera la condizione $\dot{V}(s) < 0$. Quindi si arriva a:

$$s(t)^{\mathrm{T}}\dot{s}(t) = s(t)^{\mathrm{T}}(v(t) + d(t,q,\dot{q})) = s_1(t)(v_1(t) + d_1(t,q,\dot{q})) + s_2(t)(v_2(t) + d_2(t,q,\dot{q})) = \\ = \sum_{i=1}^{2} s_i(t)(v_i(t) + d_i(t,q,\dot{q}))$$

Imporre: $s_i(t)(v_i(t) + d_i(t,q,\dot{q})) < 0; i = 1,2$ equivale ad imporre $s(t)^{\mathrm{T}}\dot{s}(t) < 0$, cioè $\|\mathbf{s(t)}\|$ sempre decrescente, dunque $s(t) \to 0$. Successivamente se si procede come segue si può notare che:

$$s_i(t)(v_i(t)+d_i(t,q,\dot{q})) \leq s_i(t)v_i(t)+|s_i(t)|\,|d_i(t,q,\dot{q})| < s_i(t)(v_i(t)+|s_i(t)|\,\rho_i; \quad i = 1,2$$

Per concludere, si sceglie $v_i = -\rho_i\,\mathrm{sign}(s_i)$ e si ha che la quantità $s_i(t)v_i(t) + |s_i(t)|\,\rho_i = 0$. Quindi si è dimostrato che $s_i(t)(v_i(t) + d_i(t,q,\dot{q})) < 0; i = 1,2$. Dunque ricapitolando, si va ad attuare la seguente la seguente legge di controllo:

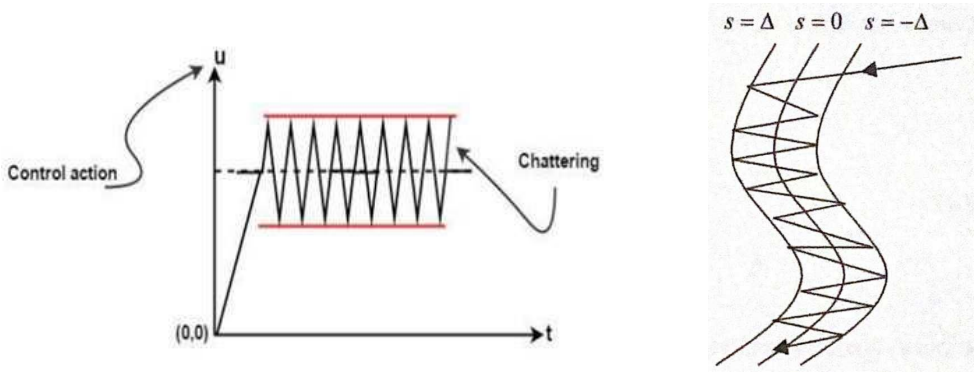
$$u(t) = C(q,\dot{q})\dot{q} + F_v\dot{q} + G(q) + B(q)\left(\ddot{q}_r(t) - \Lambda\dot{e}(t) + v(t)\right)$$

con l'opportuno ingresso ausiliario ora determinato:

$$v(t) = \begin{pmatrix} -\rho_1\,\mathrm{sign}(s_1(t)) \\ -\rho_2\,\mathrm{sign}(s_2(t)) \end{pmatrix}$$

5.5 Chattering e Soluzione di Slotine e Sastry

Già precedentemente è stato introdotta la problematica del *chattering*, ossia un fenomeno indesiderato che si manifesta come un'oscillazione rapida e continua della variabile di controllo vicino alla superficie di scorrimento. Questo fenomeno può causare usura prematura dei componenti del sistema, rumore e instabilità, in quanto porta alla necessità di applicazione di forti coppie in maniera improvvisa e bruschi movimenti nello spazio



(a) Immagine A (b) Immagine B

Figure 5.2: Effetto del chattering su u(t) (imm. A) e su s(x) (imm. B)

La proposta elaborata Slotine e Sastry per sopperire a questo problema consiste nel rinunciare ad uno scivolamento perfetto sulla superficie, e nell'andare invece ad imporre che il movimento avvenga entro un certo intorno di $s(x) = 0$, stabilito a priori. Per rendere l'idea conviene considerare il caso di $s(x)$ scalare e osservare:

$$|s(x)| \leq \epsilon; \quad \epsilon > 0 \tag{5.11}$$

Nel caso di $x(t) \in \mathbb{R}^2$ si può visualizzare quanto inteso:

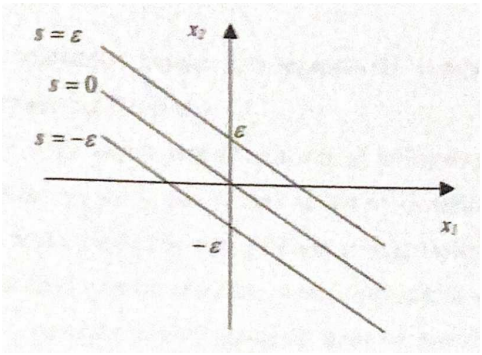


Figure 5.3: Sugli assi $e(t)$ errore di tracking ed $\dot{e}(t)$, boundary layer disegnato

5.6 Implementazione in Matlab del controllore

Il concetto descritto precedentemente per via matematica, è stato implementato a livello di codice come segue:

```
function [u,s_slid] = controllore(q,qp,qr,qrp,qrpp,Gq,Cqqp,Bq,Kp1,
Kp2, Kd1, Kd2, lam,ro1, ro2, eps1, eps2, Fv)

e=q-qr;
```

```
ep=qp-qrp;
s_slid=ep+lam*e;
s_slid1=s_slid(1);
s_slid2=s_slid(2);

b11q3=Il2+ml2*l2*l2+kr2*kr2*Im2+Il3+ml3*(a2*a2+l3*l3+2*a2*l3*cos(q3))+
+Im3+mm3*a2*a2;
b12q3=Il3+ml3*(l3*l3+a2*l3*cos(q3))+kr3*Im3;
b22=Il3+ml3*l3*l3+kr3*kr3*Im3;

hq3=-ml3*a2*l3*sin(q3);

G1q2q3=(ml2*l2+ml3*a2)*g*cos(q2)+ml3*l3*g*cos(q2+q3);
G2q2q3=ml3*l3*g*cos(q2+q3);

Bq=[b11q3 b12q3;b12q3 b22];
IBq=inv(Bq);
Gq=[G1q2q3;G2q2q3];
Cqqp=hq3*[q3p (q2p+q3p);-q2p 0];

ueq=Gq+(Cqqp+Fv)*qp+Bq*(qrpp+lam*ep);

if abs(s_slid1)>eps1
    un1=-ro1*sign(s_slid1);
else
    un1=-ro1*s_slid1/eps1;
end

if abs(s_slid2)>eps2
    un2=-ro2*sign(s_slid2);
else
    un2=-ro2*s_slid2/eps2;
end

u=ueq+Bq*[un1;un2];
```

Inizialmente si va a definire l'errore di tracking $e(t)$ e la sua derivata $\dot{e}(t)$. Successivamente si va a creare la $s(t)$ e si separano le due componenti sulle variabili s_slid1 e s_slid2.

Si procede poi calcolando le matrici necessarie per poter arrivare al U_{eq}, queste sono:

- $G(q)$
- $B(q)$
- $C(q, \dot{q})$

Si va in seguito a calcolare la componente principale dello sforzo di controllo, ossia la U_{eq}. Dopo aver calcolato quest'ultima quantità, si possono andare a calcolare le componenti degli ingressi ausiliari, delle quali si è parlato nella sezione 5.4 di questo capitolo, si noti come sia stata applicata la soluzione di Slotine e Sastry, andando ad introdurre i parametri $eps1$ ed $eps2$, che rappresentano le ampiezze degli intorni entro i quali vogliamo che si mantengano rispettivamente la s_slid1 e la s_slid2. Il tutto è stato racchiuso all'interno di una MATLAB function, che riceve come input informazioni riguardanti:

- $q(t)$, $\dot{q}(t)$, entrambe ottenute dalla retroazione,
- $q_r(t)$, $\dot{q}_r(t)$,$\ddot{q}_r(t)$ ossia i riferimenti, in termini di posizioni, velocità ed accelerazione dei giunti.

mentre manda in output:

- lo sforzo di controllo $u(t)$,
- le superfici di sliding $s_1(x)$ ed $s_2(x)$, entrambe racchiuse in un vettore 2×1;

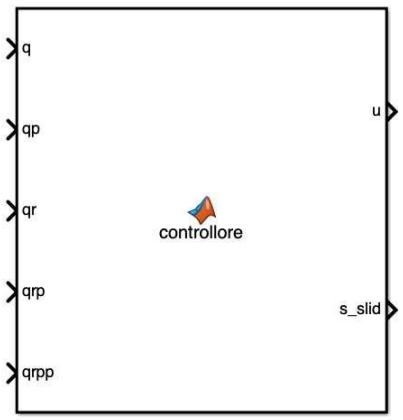


Figure 5.4: Blocco della MATLAB function che rappresenta il VSC

Chapter 6

Visione

6.1 Configurazione dell'ambiente

Per quanto riguarda il lato di computer vision si è dovuto predisporre un ambiente che permettesse l'acquisizione di una foto che identificasse la posizione di un eventuale pezzo nello spazio di lavoro del robot. Dunque si è posizionata una webcam sul telaio in metallo situato di fronte al manipolatore, e sulla piattaforma dove è poggiato quest'ultimo è stata indicato con il nastro adesivo il piano nel quale può agire il manipolatore, il tutto è mostrato in figura:

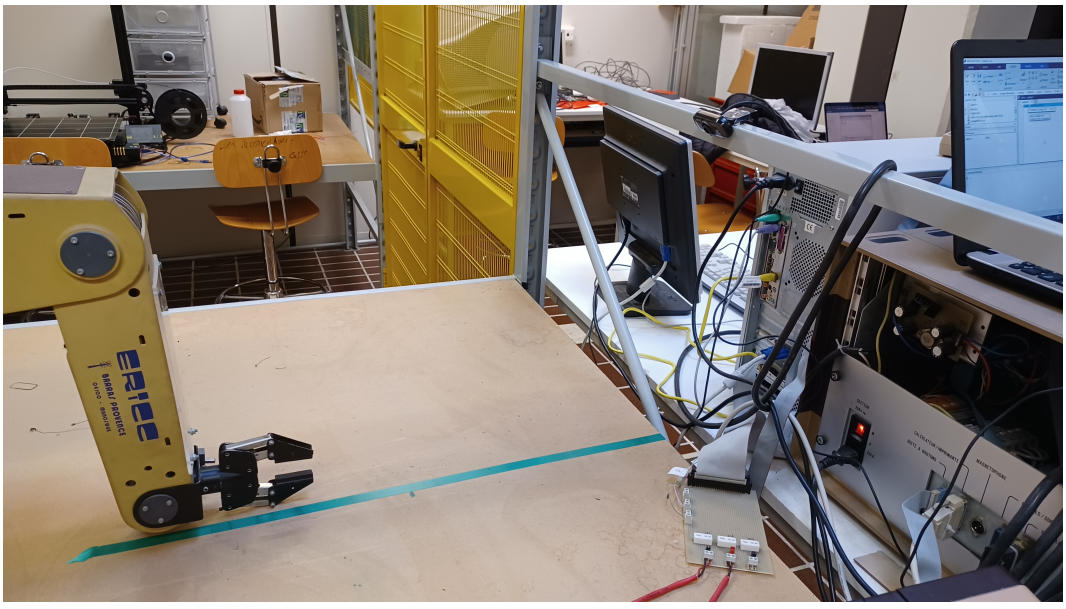


Figure 6.1: Webcam di fronte al manipolatore, e spazio di lavoro evidenziato

successivamente sono poi state prese le misure dell'effettivo campo di operatività del manipolatore, e sono stati posti due pezzi di nastro adesivo in orizzontale che delimitassero la zona, in modo da evitare errori di posizionamento del pezzo. La webcam utilizzata è una Logitech PC Webcam C210, ma quanto spiegato nelle sezioni successive è replicabile con una qualsiasi webcam.

6.2 Creazione del dataset

Come primo step per la creazione del software di visione, si è dovuto creare un dataset sul quale poter allenare il detector, in maniera tale che potesse riuscire a riconoscere la presenza di un pezzo ed eventualmente estrarne anche le posizioni cartesiane, di cruciale importanza per poter effettuare i calcoli di cinematica inversa. Per creare un dataset, bisogna dapprima raccogliere un insieme di immagini, suddivise in due categorie:

- Immagini positive: immagini contenenti gli oggetti che si vogliono riconoscere,
- Immagini negative: immagini che non ritraggono gli oggetti che devono essere riconosciuti dal sistema

Ai fini di questo progetto sono state acquisite 150 foto positive e circa 80 negative, ad ogni modo la quantità di immagini varia di caso in caso in base al livello di precisione che si vuole ottenere nel riconoscimento. Le immagini positive devono essere etichettate, per capire meglio di cosa si parla bisogna pensare al fatto che come detto sopra, lo scopo di questa catregoria di immagini è far capire al detector qual'è l'oggetto da riconoscere nelle foto, e per far ciò bisogna durante la fase di training bisogna "spiegare" al modello qual'è l'oggetto nelle varie immagini, ed è proprio qui che entra in atto il labeling dell'immagine. Questo consiste nell'andare a creare dei riquadri nelle immagini attorno al target del riconoscimento tramite un tool apposito, in questo caso si è utilizzato l' *Image Labeler* messo a disposizione tra i toolbox della Mathworks.

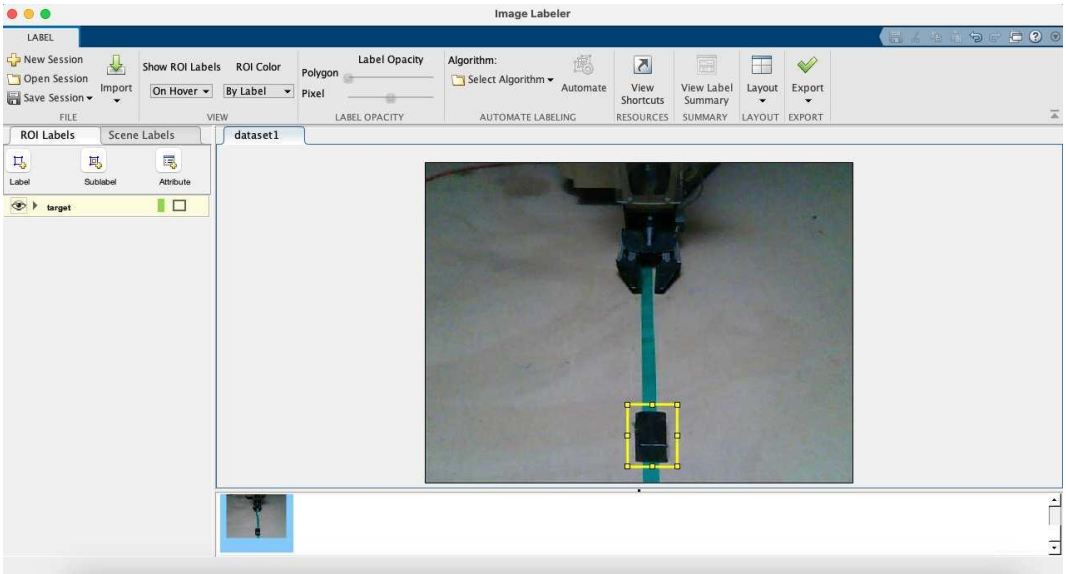


Figure 6.2: Ambiente Image Labeler, in giallo la label che identifica il pezzo

Effettuando questa operazione di labeling, quello che effettivamente si sta andando a fare è arricchire i file delle immagini con l'informazione delle posizioni (espresse in

pixel) dei vertici del rettangolo che individua il target, così facendo l'oggetto in fase di training verrà identificato dalle features estratte da quella porzione di immagine. Per terminare la procedura di labeling, bisogna infine esportare il dataset etichettato come file *.mat*, e sarà proprio il documento che dovrà essere caricato con l'apposita istruzione in fase di training.

6.3 Training del detector

Per allenare il modello sono state utilizzate le seguenti linee di codice:

```
clc; clear all; close all;

%% caricamento dataset e immagini positive

path_img_positive = "C:\Users\Giorgio Marmolino\Desktop\UNIVPM\Lezioni
\Tirocinio\RobotArmControlling\RoboticArmControlling
\Creazione_dataset\New_foto_dataset\immagini_positive";

path_groundTruth ="C:\Users\GiorgioMarmolino\Desktop\UNIVPM\Lezioni
\Tirocinio\RobotArmControlling\RoboticArmControlling
\Creazione_dataset\New_gTruth_toTes.mat";

load(path_groundTruth);
```

Nel codice sovrastante si vanno a salvare i path assoluti verso il dataset precedentemente costruito e verso le immagini positive che serviranno in un secondo momento. Successivamente il dataset viene caricato nel worskpace.

```
%% training rete neurale

[imList,boxLabels] = objectDetectorTrainingData(gTruth, ...
    SamplingFactor=4, ...
    WriteLocation=path_img_positive);

imsWithBoxLabels = combine(imList,boxLabels);
detector = trainACFObjectDetector(imsWithBoxLabels,NumStages=5);
save("New_detector_created.mat", "detector");
```

6.4 Estrapolazione posizioni cartesiane

Dopo aver effettuato il training del detector quest'ultimo è pronto per poter riconoscere e permettere di localizzare nello spazio l'oggetto. Per poter effettuare queste operazioni viene impiegato il seguente codice:

```
lim_basso = 85;
ext_y = 310;

cam = webcam(2);

pause(20)

%% FASE 1

numBoxes = 0;

while(1) %ripete il ciclo fino a quando non rileva qualcosa

        disp("Cerca di rilevare oggetto");
        img = snapshot(cam);
        img = imcrop(img, [200 lim_basso 260 ext_y]);
        imshow(img)
        [bbox,score] = detect(detector,img);

        bbox = bbox(score>30,:);
        score = score(score>30);
        [selectedBbox,selectedScore] = selectStrongestBbox(bbox,score, ...
            OverlapThreshold=0.1, ...
            NumStrongest=1);

        numBoxes = size(selectedBbox,1);
        if(numBoxes ~= 0) %%controlla che sia stato rilevato qualcosa
            break;
        end
        pause(3)
    end
```

In questa fase viene fatto quanto segue:

- Si definiscono i limiti in pixel che serviranno poi a fare il *crop* dell'immagine, utile a semplificare il riconoscimento,
- Si acquisisce l'immagine, viene ritagliata e successivamente data in input al detector,
- Vengono filtrati tra tutti i *bounding box* che circondano gli oggetti rilevati, quelli con uno score superiore ad una certa soglia, questo perchè seppure si metta un pezzo alla volta potrebbe capitare che per delle macchie o

imperfezioni sulla superficie si vadano a rilevare dei falsi oggetti, così facendo si è invece sicuri di avere tra le mani un solo *bounding box* e che faccia riferimento all'oggetto vero e proprio.

Si completa poi la procedura ottenendo le coordinate del centro del *bounding box* (dunque le coordinate del centro dell'oggetto), per quest'operazione viene impiegato il seguente codice:

```
centro = [(bbox(1)+bbox(3)/2) (bbox(2)+bbox(4)/2)];
coord_y = (centro(2)*0.435)/ext_y + 0.155;
```

6.5 Cinematica inversa

Quello che si vuole tirare fuori dalla fase di visione, sono i $set - point$ che si devono dare al sistema di controllo, dunque le posizioni angolari che devono essere raggiunte dai giunti. Il punto è che come si è potuto osservare nella sezione precedente, le informazioni che vengono estrapolate dall'algoritmo di computer vision altro non sono che delle coordinate cartesiane che indicano la posizione dell'oggetto nello spazio. Dunque si deve aggiungere uno step finale nel quale si va a tradurre la posizione cartesiana del target in posizioni angolari. Dunque si effettuano dei conti di cinematica inversa, i calcoli che devono essere fatti variano in funzione della geometria del manipolatore in questione, nel progetto in esame si sono ovviamente utilizzate delle equazioni valide per un manipolatori planare, che sono le seguenti:

$$x = a_2 \cos\theta_2 + a_2 \cos\theta_2 + \theta_3 \tag{6.1}$$

$$y = a_2 \sin\theta_2 + a_2 \sin\theta_2 + \theta_3 \tag{6.2}$$

$$\theta_3 = \pm\frac{x^2 + y^2 - {a_2}^2 - {a_3}^2}{2a_2a_3} \tag{6.3}$$

$$\theta_2 = \arctan 2[-a_3 \sin\theta_3 x + (a_2 + a_3 \cos\theta_3)y, (a_2 + a_3 \cos\theta_3)x + a_3 \sin\theta_3 y] \tag{6.4}$$

Dove la (6.1) e la (6.2) sono le equazioni in cinematica diretta che mettono in relazione le variabili di giunto e le posizioni cartesiane dell'end effector, mentre la (6.3) e la (6.4) sono invece le soluzioni in cinematica inversa, che esprimono le variabili di giunto in funzione delle posizioni cartesiane.
Tali equazioni vengono poi sfruttate nel codice come mostrato di seguito:

```
a1 = 0.34; %lunghezze link manipolatore
a2 = 0.28;
a3 = 0.3175;
```

```
syms o1 o2

eq1 = a2*cos(o1)+a3*cos(o1+o2-90*pi/180) == coord_y;
eq2 = a2*sin(o1)+a3*sin(o1+o2-90*pi/180) + a3 == 0.04 ;

sol = solve([eq1, eq2], [o1, o2]);

o1 = sol.o1 * 180/pi;
o2 = sol.o2 * 180/pi;

o1 = double(o1(1));
o2 = double(o2(1));
```

Dunque non si ha bisogno di implementare direttamente le equazioni della cinamatica inversa ed è possibile scriverle in forma "implicita", quindi come se si stesse ragionando in cinematica diretta e sfruttare poi il comando *solve* per lasciar fare al MATLAB la procedura di estrapolazione delle posizioni angolari. Le variabili che fanno riferimento a quest'ultime, devono essere opportunamente dichiarate come variabile simboliche.

Chapter 7

Generazione della traiettoria con filtro VSC

7.1 Introduzione alle treiettorie

Con il termine traiettoria si intende definire un percorso nello spazio di lavoro di una parte specifica di un cinematismo di una macchina automatica che si desidera venga compiuto per l'esecuzione di un determinato compito. Come già accennato precedentemente non basta infatti indicare la posizione da raggiungere per un determinato giunto, ma bisogna fornire una traiettoria che tenga conto sia della posizione iniziale del giunto che di quella che si vuole raggiungere. Inoltre la pianificazione di una traiettoria che descrive un percorso che va da un punto A ad un punto B va a fornire informazioni circa il profilo della velocità ed accelerazione associate, dunque non si sta soltanto spiegando ai giunti da dove a dove devono andare, ma bensì anche in che maniera devono compiere gli spostamenti.
Quello che si sta concretamente andando a fare, pianificando una traiettoria nello spazio dei giunti, altro non è che definire una legge di moto che descriva tale spostamento. Nell'intraprendere quest'azione generalmente ci si avvale di funzioni polinomiali, il grado delle quali varia di caso in caso in base alle necessità, tenendo presente che il profilo della traiettoria, la sua velocità e la sua accelerazione devono rispettare dei vincoli in termini di continuità, questo perchè brusche variazioni possono portare a picchi di sforzi di controllo che possono danneggiare gli attuatori o la meccanica del manipolatore stesso.

$$s(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$$

I coefficienti si calcolano sulla base delle condizioni iniziali delle quali si necessita per spostamento, velocità ed accelerazione.
Scegliendo ad esempio un valore di n: $n = 1$ si ottiene:

$$s(t) = a_0 + a_1(t - t_0)$$

e si ha:

$$s(t_0) = q_0 = a_0 \tag{7.1}$$

dunque a_0 coincide con q_0 posizione iniziale desiderata, mentre per a_1 si ha:

$$s(t_1) = a_1 = \frac{q_1 - a_0}{t_1 - t_0} \tag{7.2}$$

dove con q_1 si intende la posizione finale. Con la legge oraria indicata si ottengono i seguenti profili di posizione, velocità ed accelerazione:

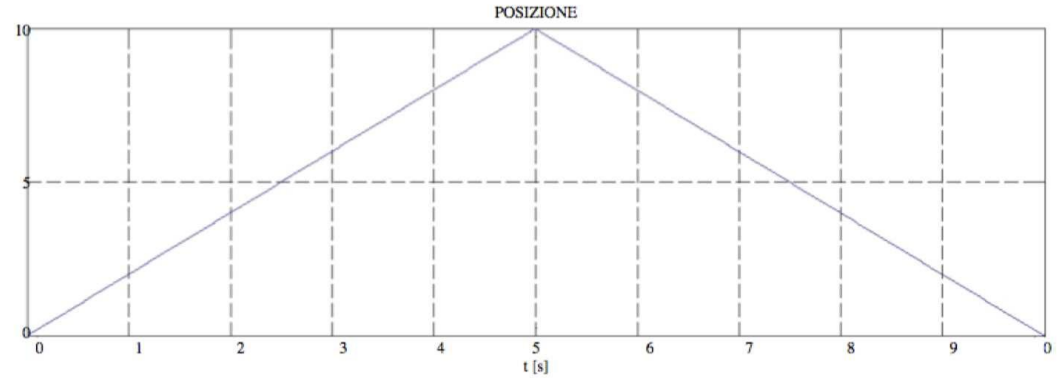


Figure 7.1: Profilo di posizione $s(t)$

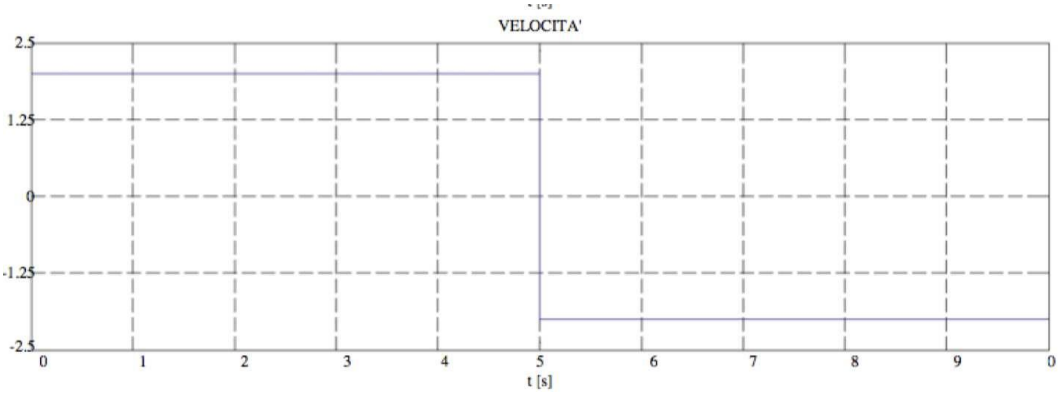


Figure 7.2: Profilo di velocità $\dot{s}(t)$

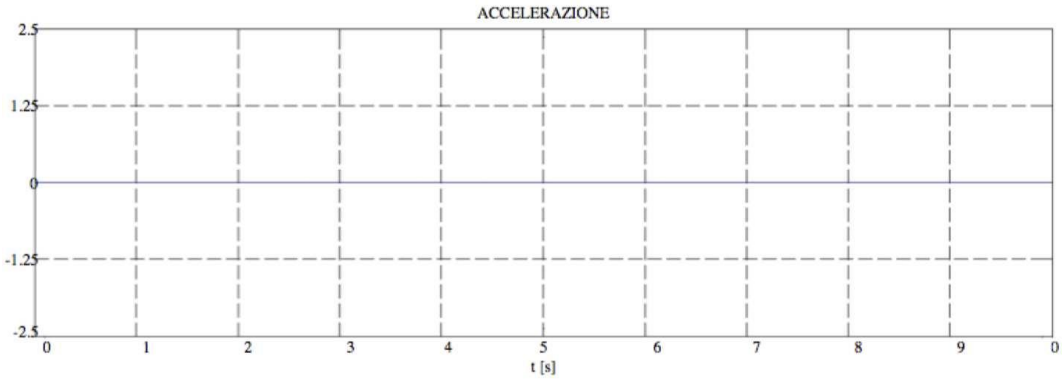


Figure 7.3: Profilo di accelerazione $\ddot{s}(t)$

Per le premesse fatte, è evidente come questa treiettoria (con conseguente velocità ed accelerazione) si porti dietro una difficoltà di applicazione, infatti $s(t)$ presenta una brusca variazione, mentre la $\dot{s}(t)$ ha una discontinuità nell'istante t nel quale la $s(t)$ inizia a decrescere. Per sopperire a questo problema basta prendere polinomi di grado superiore, che consentono inoltre di applicare condizioni iniziali e finali anche a velocità ed accelerazione. Tuttavia utilizzando questo approccio in maniera pura non si riesce ad andare a vincolare i valori massimi di $\dot{s}(t)$ e $\ddot{s}(t)$, rischiando di ottenere dei profili che non possono fisicamente essere seguiti.

7.2 Filtro non lineare per imporre vincoli al moto

Generalmente il problema della generazione della traiettoria con vincoli su velocità ed accelerazione è risolto a priori, calcolando la traiettoria in maniera tale da soddisfare tutte le condizioni per $s(t)$, $\dot{s}(t)$ e $\ddot{s}(t)$, si incappa però in una problematica, ossia se cambiano le condizioni che si vogliono imporre si deve ricalcolare il tutto da zero.

Per la risoluzione di questa problematica si è utilizzato un filtro dinamico non lineare che prende in ingresso profili non direttamente utilizzabili e smussa le traiettorie in maniera tale da rispettare le condizioni che si vogliono imporre. Il sistema filtro si basa su uno schema in retroazione e su tecniche di controllo non lineari a struttura variabile (stesso concetto della legge impiegata per il controllo del manipolatore), con la finalità di inseguire il riferimento al meglio possibile pur rispettando i vincoli che si vogliono imporre alla traiettoria. Nel caso di questo progetto si andranno a passare al filtro dei riferimenti a gradino, che viene poi smussato cercando di creare una traiettoria che lo faccia eseguire in tempo minimo rispettando i limiti imposti, andando a generare un profilo di moto continuo e senza sovraelongazioni, rispettando la condizione di continuità della velocità e generando un profilo di velocità costante a tratti.

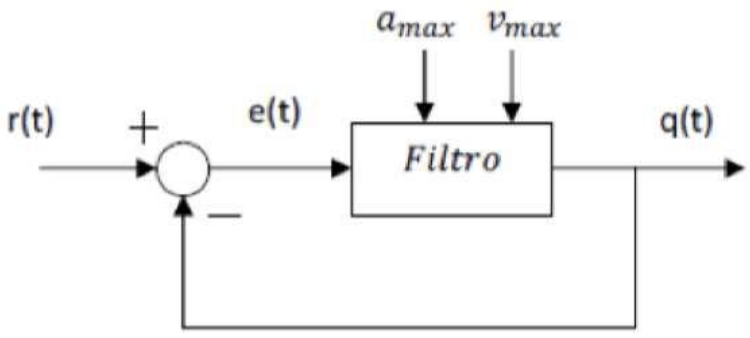


Figure 7.4: Schema a blocchi del sistema filtro

Dove $r(t)$ è un segnale di riferimento che si vuole inseguire, $q(t)$ è la traiettoria generata che insegue al meglio il valore $r(t)$, $e(t) = r(t) - q(t)$ è l'errore di tracking, a_{max} e v_{max} sono rispettivamente i valori massimi di accelerazione e velocità che si stanno imponendo sulla traiettoria $q(t)$. Nello schema in retroazione considerato il blocco controllato genera come variabile di controllo l'accelerazione desiderata, che viene successivamente integrata per ottenere velocità e posizioni desiderate.

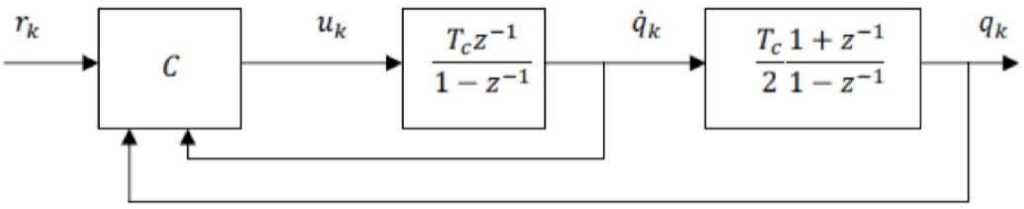


Figure 7.5: u_k accelerazione che viene itegrata ad ottenere $\dot{q}_k$ e q_k

Il generatore di traiettorie è realizzato a tempo discreto, per la legge di controllo viene calcolata l'accelerazione $\ddot{q}_k = u_k$ all'sistante t: $t = kT_c$ con $k = 1, \ldots, n$. Il controllore C in figura 7.6 è definito come segue:

$$C = \begin{cases} Z_k = \frac{1}{T_C a_{\text{mac}}} \left(\frac{e_k}{T_C} + \frac{\dot{e}_k}{2} \right) \\ m = \text{Int} \left[\frac{1+\sqrt{1+8|Z_k|}}{2} \right] \\ \sigma_k = \dot{Z}_k + \frac{Z_k}{m} + \frac{m-1}{2} \text{sgn}\left(Z_k\right) \\ u_k = -a_{\max} \text{sat}\left(\sigma_k\right) \frac{1+\text{sgn}(\dot{q}_k sgn(\sigma_k) + v_{\max} - T_C a_{\max})}{2} \end{cases} \tag{7.3}$$

dove $e_k = q_k - r_k$ è l'errore in posizione, ed analogamente dove $\dot{e}_k = \dot{q}_k - \dot{r}_k$ è l'errore in velocità con $\dot{r}_k$ che è la derivata del riferimento, calcolata come segue:

$$\dot{r}_k = \frac{2}{T_C} \left(r_k - r_{k-1} \right) - \dot{r}_{k-1}$$

si noti che nelle precedenti equazioni introdotte, $Int(.)$ è la funzione *parte intera*, mentre $sgn(.)$ è la funzione *segno* e $sat(.)$ è la funzione *saturazione*, quest'ultime definite come mostrato di seguito:

$$\text{sgn}(x) = \begin{cases} 1, & \text{se } x \geq 1 \\ -1 & \text{se } x < 0 \end{cases}$$

$$\text{sat}(x) = \begin{cases} 1, & \text{se } x > 1 \\ x & \text{se } -1 \leq x1 \leq \\ -1 & \text{se } x < -1 \end{cases}$$

Infine, una volta definita l'accelerazione desiderata u_k si può conseguentemente calcolare velocità e posizione:

$$\dot{q}_k = \dot{q}_{k-1} + T_C u_{k-1}$$

$$q_k = q_{k-1} + \frac{T_C}{2}\left(\dot{q}_k + \dot{q}_{k-1}\right)$$

7.3 Risultati per un riferimento a gradino

Si esaminano ora i profili di posizione, velocità in output dal filtro per un ingresso a gradino avente le seguenti proprietà:

- la traiettoria va dal punto $q_0 = 0°$ al punto $q_1 = 15°$ (ampiezza del gradino pari a 15°)
- velocità massima pari a $\dot{q}_{max} = 10°/s$
- accelerazione massima a $\ddot{q}_{max} = 10°/s^2$

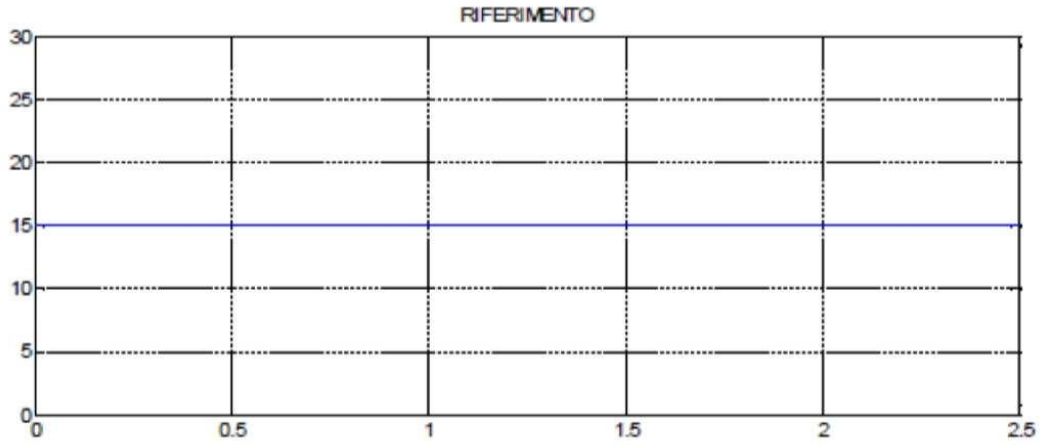


Figure 7.6: Segnale di riferimento $r(t)$

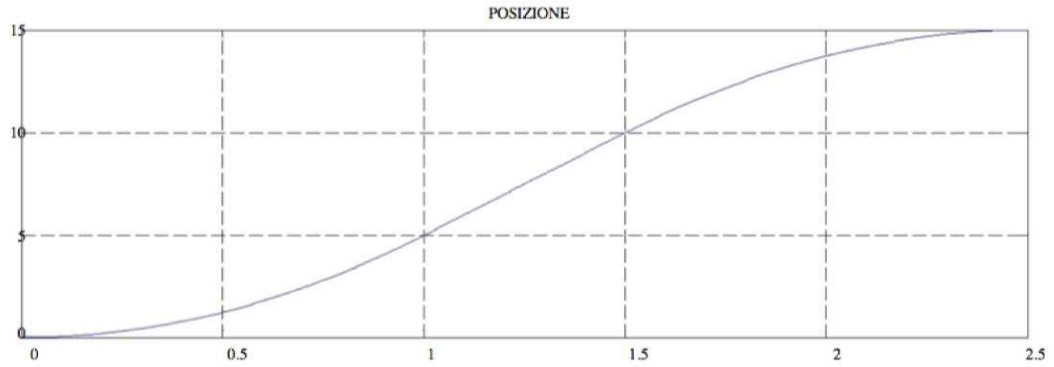


Figure 7.7: Profilo di posizione $q(t)$

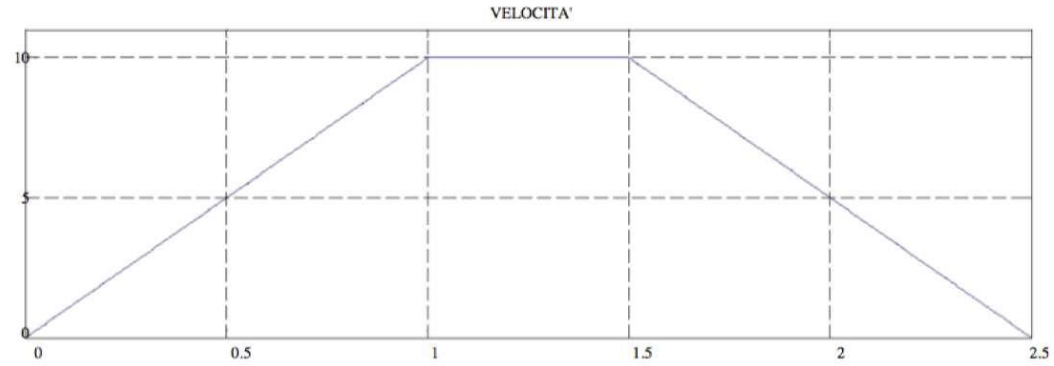


Figure 7.8: Profilo di velocità $\dot{q}(t)$

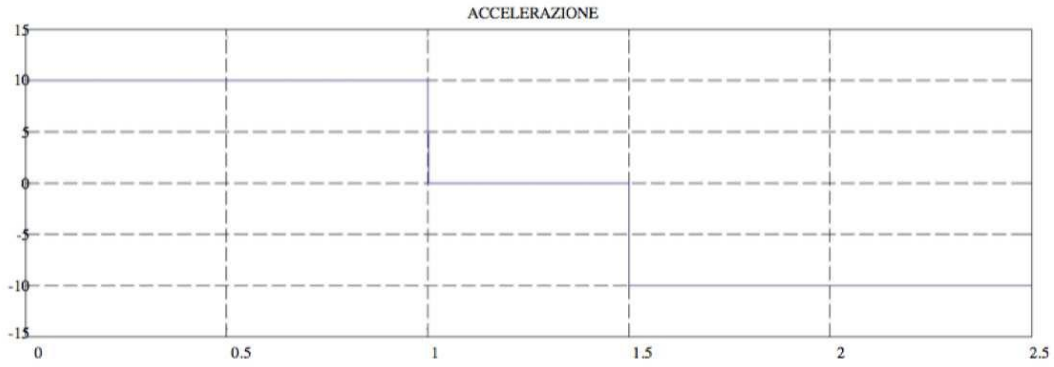


Figure 7.9: Profilo di accelerazione $\ddot{q}(t)$

Come è possibile notare dai diagrammi riportati precedentemente, si va a formare una traiettoria che appartiene alla famiglia delle "traiettorie lineari con raccordi parabolici", caratterizzate dall'assenza di tratti discontinui nel profilo di velocità. Nel caso in esame infatti, si può osservare un andamento di velocità trapezoidale, nei tratti lineari del profilo di posizione la velocità si mantiene costante, per poi crescere o decrescere linearmente. Andando a "progettare" il moto secondo quest'approccio si vanno ad imporre entrambe le proprietà prefissate in precedenza, ovvero:

- inseguimento del riferimento $r(t)$
- minimizzare il tempo della traiettoria pur rispettando le specifiche sui vincoli di velocità ed accelerazione.

E soprattutto si riesce ad avere una flessibilità rispetto ai vincoli su $\dot{q}(t)$ e $\ddot{q}(t)$, se questi cambiano non si ha la necessità di ricalcolare manualmente il tutto, ma sarà il filtro che rielaborerà la traiettoria secondo le nuove specifiche.

Chapter 8

Simulazione

8.1 Programma Simulink in simulazione

Nell'immagine seguente è possibile osservare il software creato in Simulink per simulare il sistema di controllo, sfruttando il modello matematico del robot e non il manipolatore vero e proprio, questo per essere certi che la legge di controllo dia i risultati previsti e non vada a creare dinamiche che rischino di danneggiare la strumentazione.

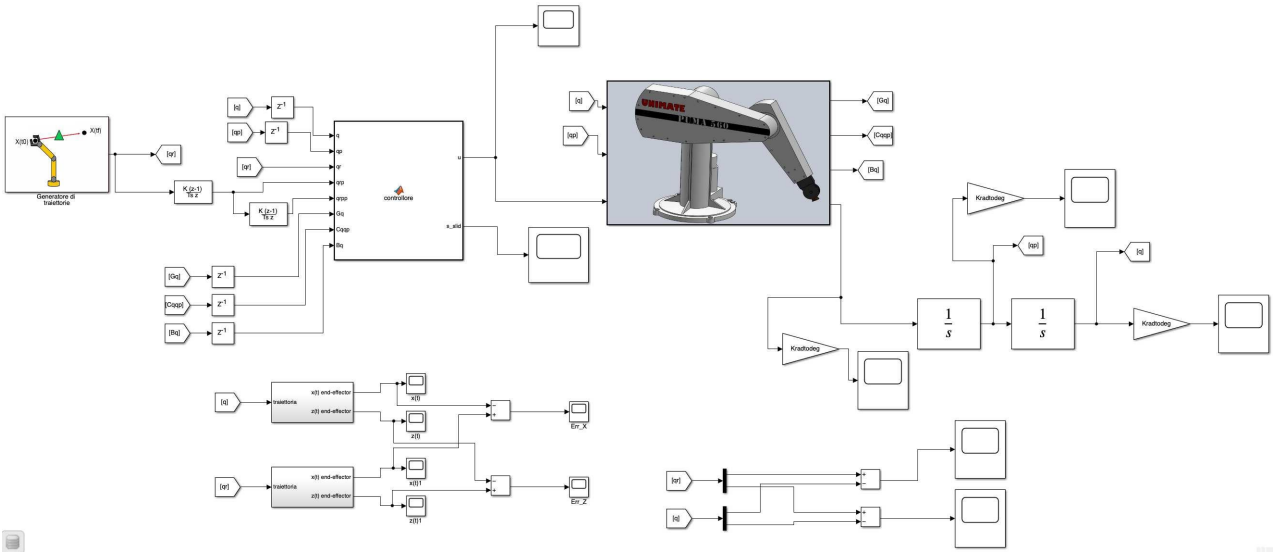


Figure 8.1: Software di controllo applicato in simulazione.

Il tutto si articola in cinque sezioni principali:

- *Generazione delle traiettorie:* Blocco che date una serie di posizioni che i giunti devono raggiungere, va a generare una traiettoria sufficientemente smussata in maniera tale che i motori riescano effettivamente a seguirla,
- *Sezione di controllo:* questa parte presenta il cuore di questo sistema, ossia il VSC, ed un apposito convertitore A/D che si occupa di ricevere la retroazione di posizione e di velocità a tempo continuo, per poi discretizzarle secondo il Tc del nostro sistema ed infine fornirle come input al controllore.

- *Modello matematico del robot:* questo blocco è quello che va effettivamente a simulare il comportamento dinamico del manipolatore, tenendo conto dei suoi parametri costruttivi (ad esempio lunghezze, masse e momenti d'inerzia dei link) permette dunque di ricostruire come il sistema in questione risponde agli stimoli. Questo blocco è realizzato tramite una Matlab Function che prende come input il forzamento in coppia che andiamo a dare ad ognuno dei due giunti e ci restituisce in output i valori di posizione e velocità di ciascuno.

- *Cinematica diretta:* l'insieme di blocchi sottostante al controllore fa in modo tramite opportuni passaggi di cinematica diretta di andare a calcolare le coordinate cartesiane che nel tempo individuano la posizione dell'end-effector (dunque in funzione delle posizioni assunte dai giunti).

- *Calcolo dell'errore:* l'insieme di blocchi posizionato in basso a destra, ha la funzione di andare a disaccoppiare le componenti di q_r e q per andare a calcolare l'errore che si sta compiendo rispetto al riferimento.

8.2 Parametri in simulazione

Come è emerso durante la spiegazione della legge di controllo, in quest'ultima vi sono dei parametri che giocano un ruolo determinante per il corretto funzionamento del manipolatore, che riassumendo sono:

- ρ_2, ρ_3

- λ_2, λ_3

- ϵ_2, ϵ_3

I valori di default utilizzati nel codice per i parametri sono i seguenti:

```
eps1=0.1;
eps2=0.1;
ro1=1;
ro2=1;
lam1=1;
lam2=1;
lam=[lam1 0;0 lam2];
eps=[eps1 0;0 eps2];
```

Dati gli ottimi risultati raggiunti sin da subito in fase di simulazione, non si è presentata la necessità di variare alcun parametro dal suo valore di default, cosa che invece è stata necessaria passando a controllare il manipolatore reale.

8.3 Matlab function per la simulazione del manipolatore

Nel programma in simulazione la risposta del robot agli sfrozi di controllo viene simulata tramite una MATLAB function che dato lo sforzo di controllo e lo stato attuale del manipolatore (posizioni e velcoità angolari dei giunti) va a calcolare:

- $B(q)$,
- $G(q)$,
- $C(q, \dot{q})$,

che vengono poi impiegati per determinare la $\ddot{q}(t)$. Per calcolare le quantità precedentemente citate vengono sfruttate le relazioni matematiche introdotte a pagina 21 quando si parla della dinamica del manipolatore. È però importante notare che sebbene in fase di simulazione queste quantità vengano calcolate nella MATLAB function che simula in manipolatore, nell'applicazione reale del sistema di controllo vengono invece calcolate all'interno della MATLAB function del controllore, come è possibile notare nell'apposito capitolo che mostra il codice di quest'ultimo. Questo perchè nel caso reale sarà il robot vero e proprio a restituire in retroazione le informazioni circa la q e la $\dot{q}$, e la determinazione delle matrici prima citate è necessaria per il calcolo dello sforzo di controllo, mentre in simulazione tali calcoli vengono eseguiti per riprodurre la risposta del manipolatore e vengono poi passate alla funzione che rappresenta il controllore. Si riporta di seguito il codice del quale si è parlato pocanzi:

```
function [Gq,Cqqp,Bq,qpp] = manipolatore(q,qp,u,a2, l2, l3, ml2,
ml3, Il2, Il3, mm3, g,kr2, kr3, Im3, Im2, Fv)
q2=q(1);
q3=q(2);
q2p=qp(1);
q3p=qp(2);


b11q3=Il2+ml2*l2*l2+kr2*kr2*Im2+Il3+ml3(a2*a2+l3*l3+2*a2*l3*cos(q3))+
+Im3+mm3*a2*;
b12q3=Il3+ml3*(l3*l3+a2*l3*cos(q3))+kr3*Im3;
b22=Il3+ml3*l3*l3+kr3*kr3*Im3;

hq3=-ml3*a2*l3*sin(q3);

G1q2q3=(ml2*l2+ml3*a2)*g*cos(q2)+ml3*l3*g*cos(q2+q3);
G2q2q3=ml3*l3*g*cos(q2+q3);
```

```
Bq=[b11q3 b12q3;b12q3 b22];
IBq=inv(Bq);
Gq=[G1q2q3;G2q2q3];
Cqqp=hq3*[q3p (q2p+q3p);-q2p 0];

qpp=IBq*(-Cqqp*qp-Fv*qp-Gq+u);
```

Chapter 9

Implementazione reale

9.1 Real-Time Workshop

Il Real-Time Workshop di MATLAB, ora conosciuto come Simulink Coder, è uno strumento che consente la generazione automatica di codice estrapoalto da modelli Simulink. Questo strumento è particolarmente utile per lo sviluppo di sistemi in tempo reale, come quelli utilizzati in applicazioni embedded, sistemi di controllo e prototipazione rapida. Nel caso di questo progetto è stato infatti sufficiente modellare il sistema di controllo in Simulink in maniera piuttosto intuitiva tramite uno "schema a blocchi", che viene poi tradotto in codice C++ (che implicitamente convertito in linguaggio macchina è in grado di agire sul manipolatore).

Gli aspetti principali del real-time workshop sono i seguenti:

- *Generazione di codice:* Simulink Coder permette di generare codice C o C++ direttamente dai modelli Simulink. Questo codice può essere utilizzato per l'implementazione su hardware embedded, microcontrollori o processori DSP;
- *Ottimizzazione del codice:* il codice generato è altamente ottimizzato per le prestazioni in tempo reale, riducendo il carico di lavoro manuale e minimizzando gli errori;
- *Esecuzione su target specifici:* il codice generato può essere eseguito su sistemi operativi in tempo reale (RTOS), target dedicati o schede di prototipazione come Arduino, Raspberry Pi, e altri;
- *Integrazione con altri strumenti:* Simulink Coder si integra con strumenti di terze parti per il debugging, la profilazione e l'analisi delle prestazioni;
- *Code reuse:* Il codice generato è modulare, facilitando il riutilizzo in altri progetti o sistemi;
- *Compatibilità tra compilatori:* il codice può essere compilato usando diversi compilatori supportati, come GCC, MSVC, e altri;
- *Codice per più piattaforme:*supporta la generazione di codice per piattaforme diverse, incluse x86, ARM, e PowerPC.

Sfruttare questo strumento in fase di sviluppo conferisce numerosi vantaggi; tra i quali in primis si ha una riduzione dei tempi di lavoro grazie all'automatizzazione della stesura del codice, sempre per questa ragione si va a minimizzare il rischio di errori che possono invece verificarsi con la scrittura manuale. Si conferisce una notevole flessibilità al programma eliminando la necessità di mettere mano al codice per effettuare delle modifiche, e la modularità del codice ne consente l'integrazione con altri sistemi e l'adattamento a diverse esigenze.

9.2 Inizializzazione in posizione di home

Come primo step per far lavorare il sistema, grazie al programma *manipulator_parameters.m* viene inizializzato l'ambiente, sia in termini concreti che in termini virtuali, si effettua quanto segue. Dapprima vengono caricati tutti i parametri necessari per il corretto funzionamento del sistema di controllo:

```
Ts=0.001;
Tc=0.001;

amax = 10;
vmax = 20;


a2=0.28;
l2=-0.0145;
l3=0.0178;
ml2=10.5102;
ml3=2.2449;
Il2=0.3159;
Il3=0.05851;
kr2=1233;
kr3=552;
Im2=33*10^-7;
Im3=33*10^-7;
mm3=5;
kv2=0.817*10^-3;
kv3=1.38*10^-3;

Fv = [kv2 0;0 kv3];
g = 9.81;

Kp1=1;
Kp2=1;
```

```
Kd1=1;
Kd2=1;

qr2pp=0;
qr3pp=0;
qrpp=[qr2pp;qr3pp];
```

Successivamente, viene effettuata la lettura delle posizioni iniziali dei due giunti e, dopo aver impostato la posizione di home come meta da raggiungere e viene effettuato il controllo atto a verificare il verso dello spostamento e variare, in funzione di questo, i parametri per il VSC.

```
if(go_pos_2 > getq20deg)
    eps1 = 0.4; %originale 0.3
    eps2 = 0.4; %originale 0.3
    ro1 = 7; %originale 5
    ro2 = 6.8;%originale 4.5
    lam1 = 200; %originale 200
    lam2 = 200; %originale 200
else
    eps1 = 0.5; %originale 0.3
    eps2 = 0.8; %originale 0.3
    ro1 = 12.5; %originale 5
    ro2 = 7;%originale 4.5
    lam1 = 500; %originale 200
    lam2 = 200; %originale 200
end-
```

Infine, viene lanciato il modello simulink, con l'implementazione reale del sistema di controllo.

```
modello = 'ERICC_MP_traiettoria';
load_system(modello);
set_param(modello,'SimulationMode','external');
set_param(modello,'SimulationCommand','connect');
set_param(modello,'SimulationCommand','start');

pause(15);

get_pos = 'get_initial_pos';
load_system(get_pos);
set_param(get_pos,'SimulationMode','external');
```

```
set_param(get_pos,'SimulationCommand','connect');
set_param(get_pos,'SimulationCommand','start');

pause(3)
```

9.3 Trasmissione dei risultati della visione

Come già spiegato nell'apposito, il set-point per il raggiungimento del pezzo viene fornito dall'algoritmo di visione (da notare che l'integrazione tra la visione ed il controllo non è stata implementata in fase di simulazione perchè non necessario), per poi essere processato dal filtro inglobato nel generatore di traiettorie. Dopo che dall'algoritmo di visione vengono calcolati gli angoli necesari per i giunti, quest'ultimi devono essere trasmessi dal PC esterno al PC di laboratorio, per far ciò ci si avvale della connessione ethernet precedentemente introdotta e si vanno a salvare le posizioni elaborate nella cartella in rete:

```
go_pos_2 = o1(1); %posizioni che ha ricavato
go_pos_3 = o2(1); %posizioni che ha ricavato
go_pos = [go_pos_2 go_pos_3];
signal = 0;
save('\\192.168.1.2\PM\found_position.mat', "go_pos", "signal");
```

successivamente all'interno del programma principale del PC dell'ERICC tali informazioni vengono recuperate come mostrato di seguito:

```
load('C:\Documents and Settings\lumafra\Desktop\PD\found_position');

correzione = 0;
go_pos_2 = go_pos(1) - correzione
go_pos_3 = go_pos(2)

pause(3)
```

E viene poi controllato che il robot debba effettivamente eseguire uno spostamento, controllando che i valori delle variabili *go_pos*2 e *go_pos*3 differiscano da quelli precedentemente assegnati, ossia gli angoli dei giunti per avere il manipolatore in posizione di home.

```
if(go_pos_2 == 65 )
    error('posizione non cambiata')
end
```

Infine, viene caricato il modello e lanciata l'esecuzione come già effettuato per portare l'ERICC in posizione di home.

9.4 Parametri reali e differenziazione

Come già illusrtrato, in fase di simulazione non si è presentato il bisogno di modificare i valori dei parametri, il sistema di controllo riusciva ad ottenere ottimi risultati dalle simulazioni, cosa che non è accaduta invece attuando il controllo del manipolatore reale. Sin da subito infatti non solo il controllo non era ottimale, ma con i valori mostrati in simulazione non si riusciva per nulla a far compiere movimenti al robot, per questo motivo dopo svariate prove modificando i valori, sulla base del significato dei singoli parametri, si sono raggiunte ottime prestazioni del sistema. Un'altra accortezza che si è dovuta adottare è stata la differenziazione del set di parametri tra movimenti in salita del braccio ed in discesa, questo perchè si è notato che mantenendo gli stessi parametri le prestazioni ottenute erano diverse tra traiettorie che prevedevano l'innalzarsi del braccio rispetto a quando quest'ultimo doveva abbassarsi. Dunque questa suddivisione è stata implementata a livello di codice come segue:

```
if(go_pos_2 > getq20deg)
    eps1 = 0.4;
    eps2 = 0.4;
    ro1 = 7;
    ro2 = 6.8;
    lam1 = 200;
    lam2 = 200;
else
    eps1 = 0.5;
    eps2 = 0.8;
    ro1 = 12.5;
    ro2 = 7;
    lam1 = 500;
    lam2 = 200;
end
```

Cioè si va a controllare la posizione angolare che deve raggiungere il giunto di spalla, se quest'ultima è maggiore della posizione angolare allora si adotteranno i parametri "da salita", altrimenti verranno caricati quelli "da discesa".

9.5 Programma simulink reale

Viene mostrato di seguito il programma simulink responsabile di effettuare il controllo al manipolatore reale:

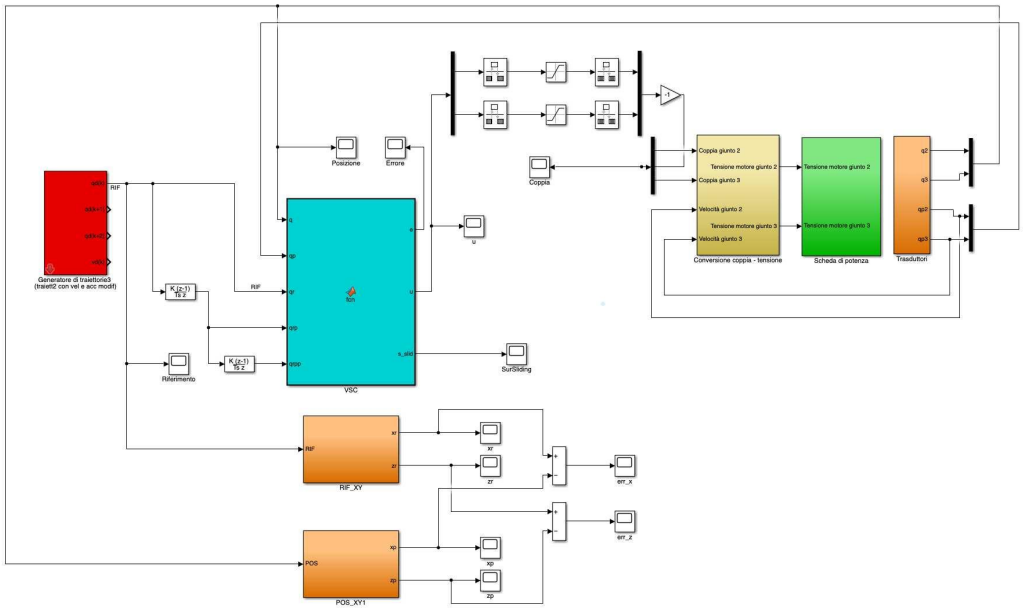


Figure 9.1: Programma simulink per il controllo del manipolatore reale

si possono notare delle sostanziali differenze con il software per la simulazione. Sicuramente la prima cosa che salta all'occhio è la mancanza della MATLAB function che simula il manipolatore, quest'ultima per ovvi motivi è stata sostituita con gli appositi blocchi caratteristici del real-time workshop che consentono di avere input e output verso una scheda di acquisizione dati. Si effettuano operazioni di I/O verso la scheda grazie a quanto raffigurato di seguito:

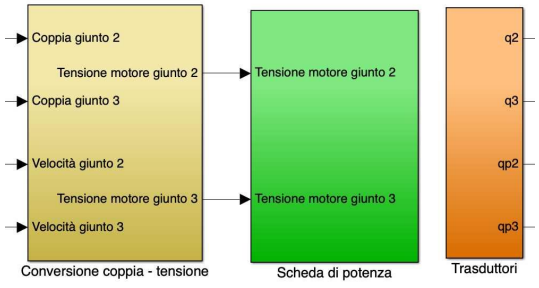


Figure 9.2: Blocchi per scambiare informazioni con la scheda di acquisizione dati

Il blocco *conversione coppia − tensione* si occupa per l'appunto di convertire il segnale che proviene dal controllore, dunque lo sforzo di controllo, in un valore in basse tensione che dovrà essere inoltrato alla DAQ, tale valore verrà poi letto e amplificato dalle schede di potenza, l'interno del blocco appare come segue:

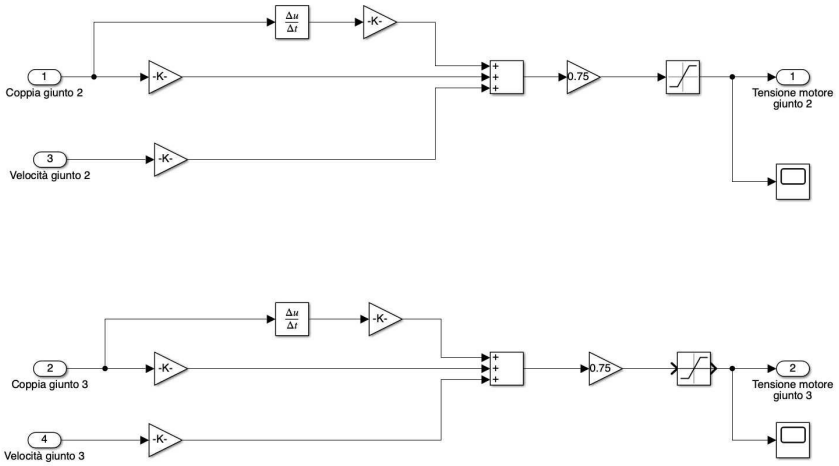


Figure 9.3: Blocco per la conversione della coppia in tensione

mentre il blocco *scheda di potenza* inoltra il segnale in tensione, calcolato precedentemente, alla DAQ sfruttando i blocchi del real-time workshop prima menzionati:

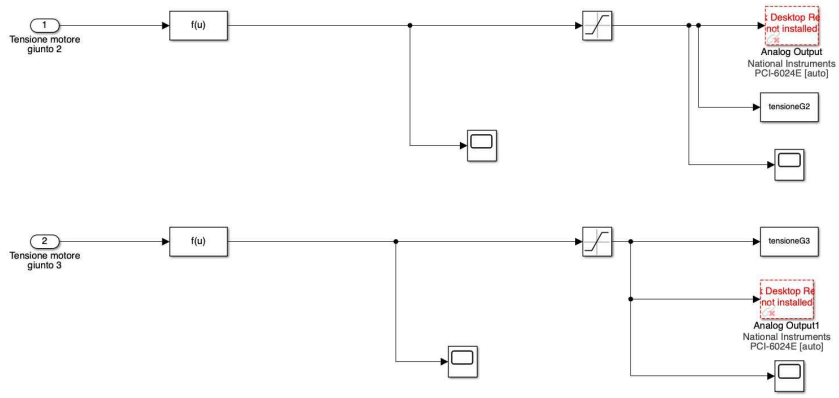


Figure 9.4: Blocco per l'output verso le schede di potenza

invece nel blocco *trasduttori* vengono recuperati grazie agli opportuni strumenti di input, i valori di posizioni e velocità angolari dai giunti che devono essere mandati in retroazione al controllore.

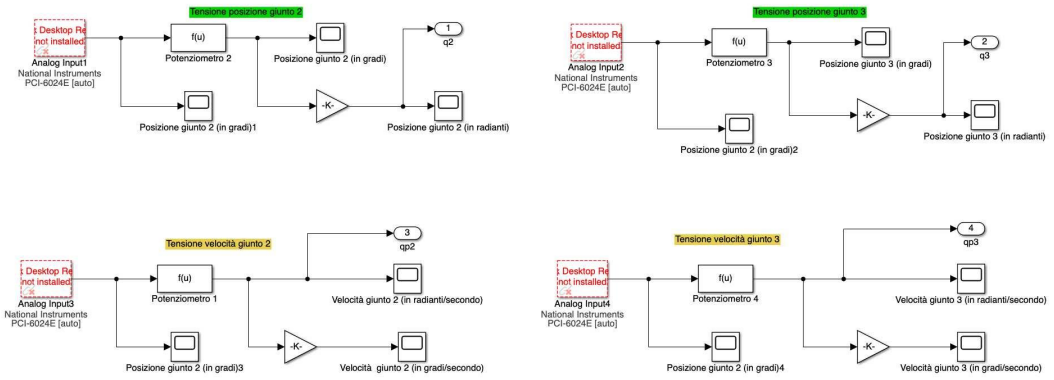


Figure 9.5: Blocco per l'input dai trasduttori montati sul manipolatore

Un'altra differenza rispetto al software in simulazione è la presenza dei seguenti elementi:

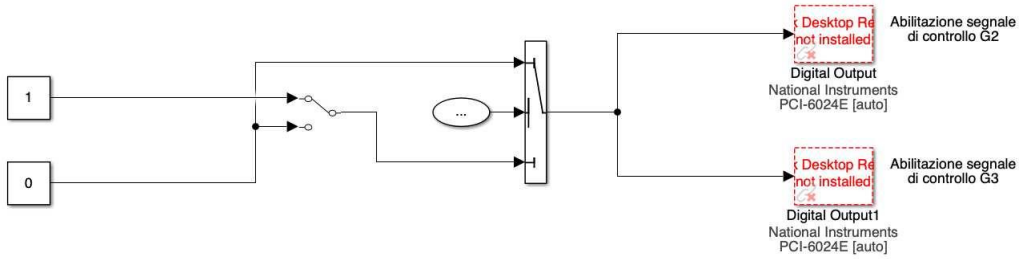


Figure 9.6: Blocco per l'esclusione del controllo nativo

tali blocchi sono necessari per andare ad agire elettronicamente sul relè che permette di escludere il controllore nativo ed abilitare invece il controllo da PC, senza di questo infatti tutto il software in MATLAB non avrebbe effetto ed il manipolatore rimarrebbe immobile.

Chapter 10

Risultati sperimentali

10.1 Risultati in simulazione

Si è testato il sistema di controllo in simulazione sull'inseguimento di due traiettorie riportate qua di seguito:

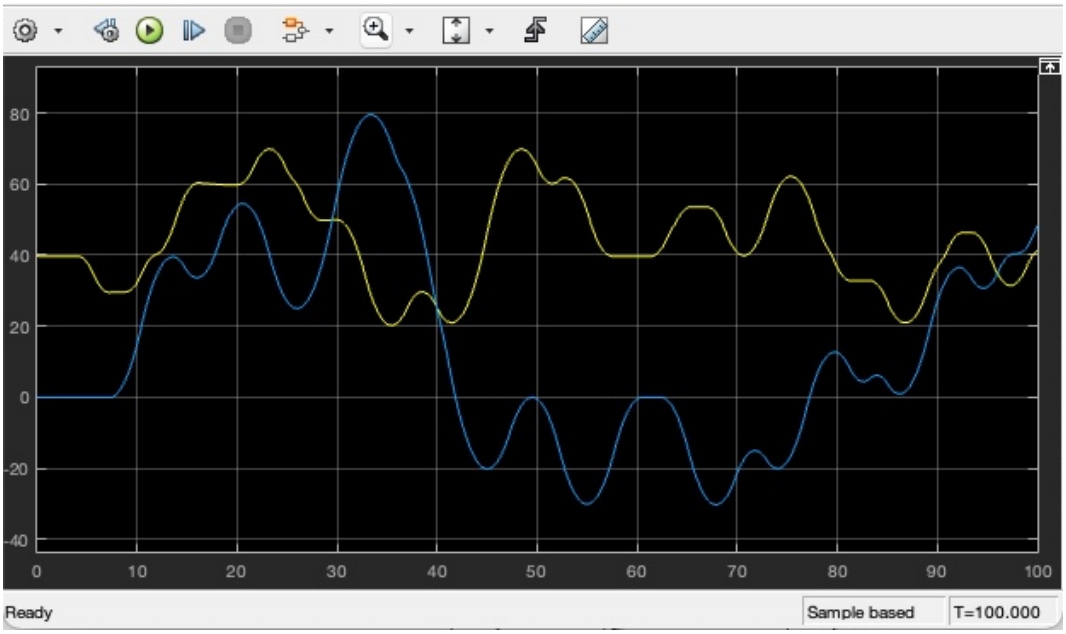


Figure 10.1: Andamento delle posizioni del giunto 2 (giallo) e del giunto 3 (blu)

nell'eseguire quanto mostrato, si sono visualizzati gli andamenti di alcune grandezze coinvolte nel sistema, mostrate a seguire.

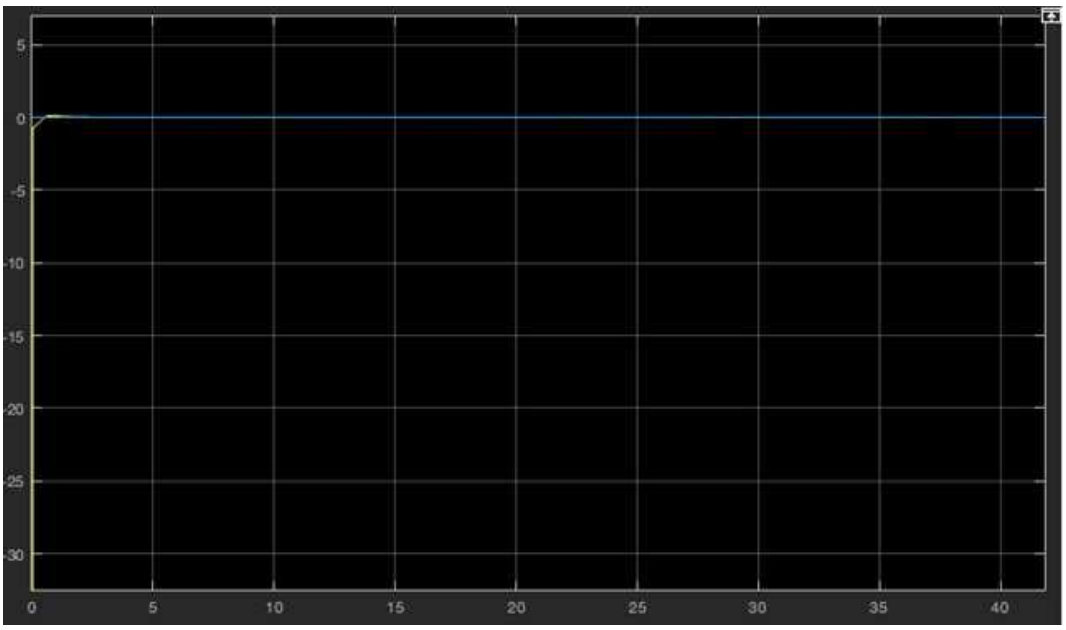


Figure 10.2: Andamento delle posizioni del giunto 2 (giallo) e del giunto 3 (blu)

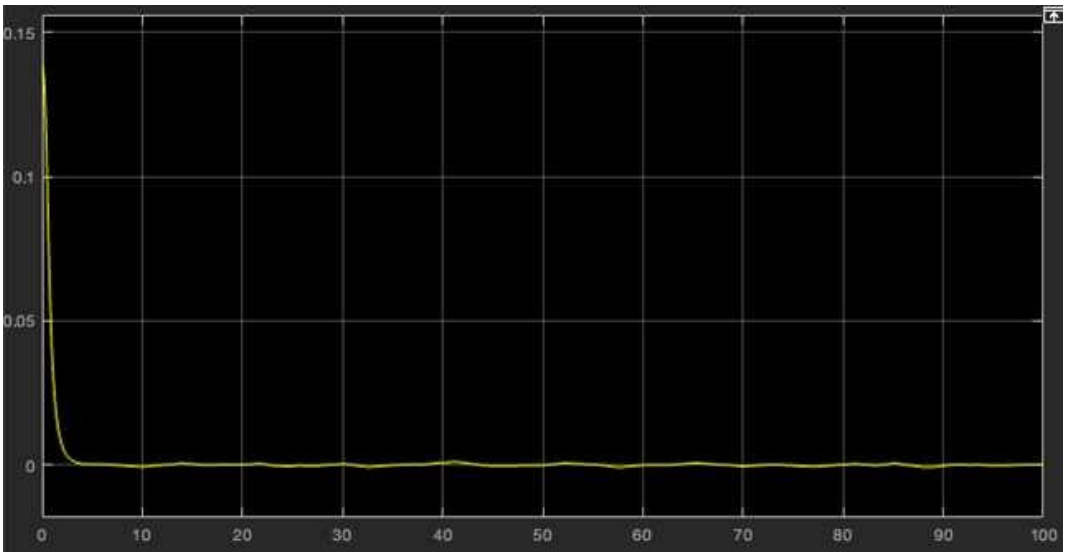


Figure 10.3: Errore della traiettoria dell'end-effector sull'asse X

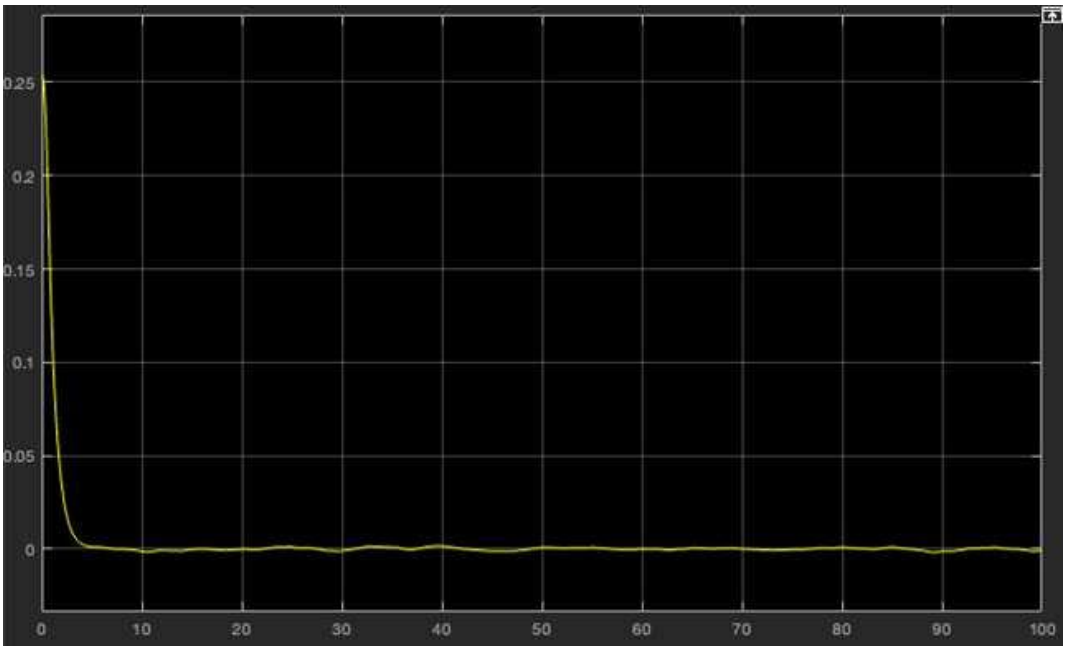


Figure 10.4: Errore della traiettoria dell'end-effector sull'asse Z

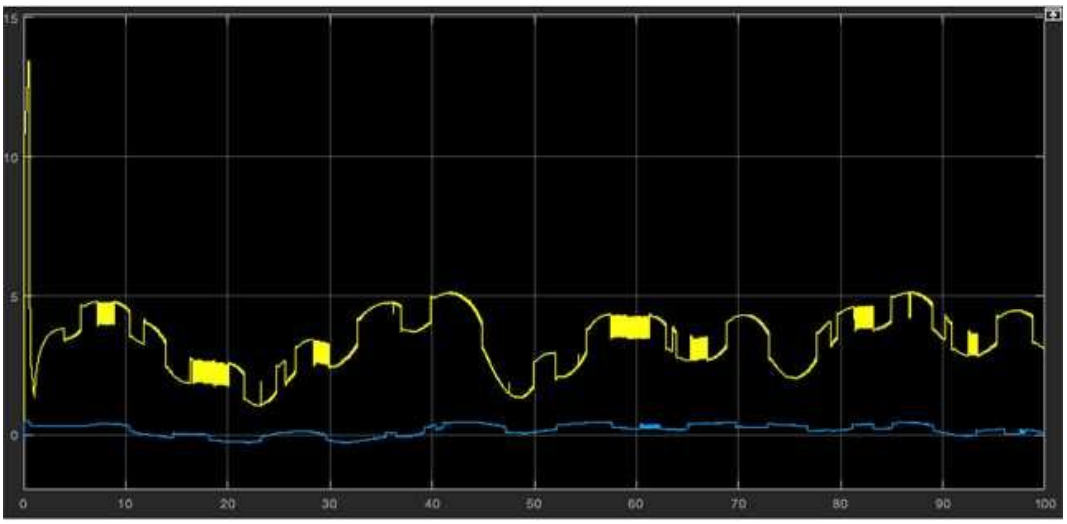


Figure 10.5: Andamento coppia del giunto 2 (giallo), e del giunto 3 (blu)

Come si era già detto (e come si può anche osservare) i risultati della simulazione sono stati ottimi pur non modificando i parametri, le superfici di sliding convergono perfettamente a 0 (nonostante un picco iniziale), stesso discorso può essere fatto per l'errore di posizionamento sui due assi dell'end-effector, che segue bene la traiettoria di riferimento. L'andamento della coppia è piuttosto reglare, l'unica irregolarità che salta all'occhio è un picco di coppia iniziale, va però puntualizzato che questo non è presente durante le prove reali, infatti questo picco in simulazione è dovuto a come vengono impostate le condizioni iniziali

10.2 Risultati nelle prove reali

Effettuando svarite prove del sistema di controllo implementato sul manipolatore reale, si riporta l'esecuzione che segue per riassumere le prestazioni raggiunte dall'ERICC:

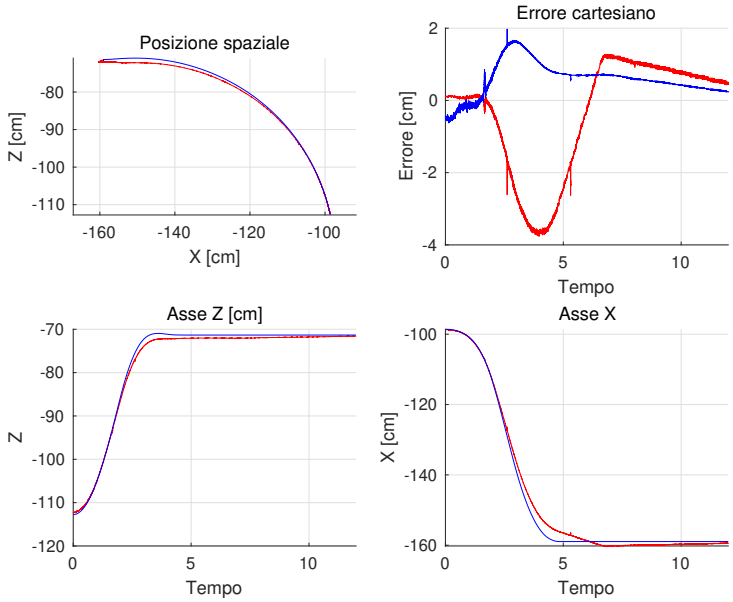


Figure 10.6: Per l'errore osserviamo le due componenti per i giunti, mentre per gli altri grafici si distinguono riferimento e traiettoria reale

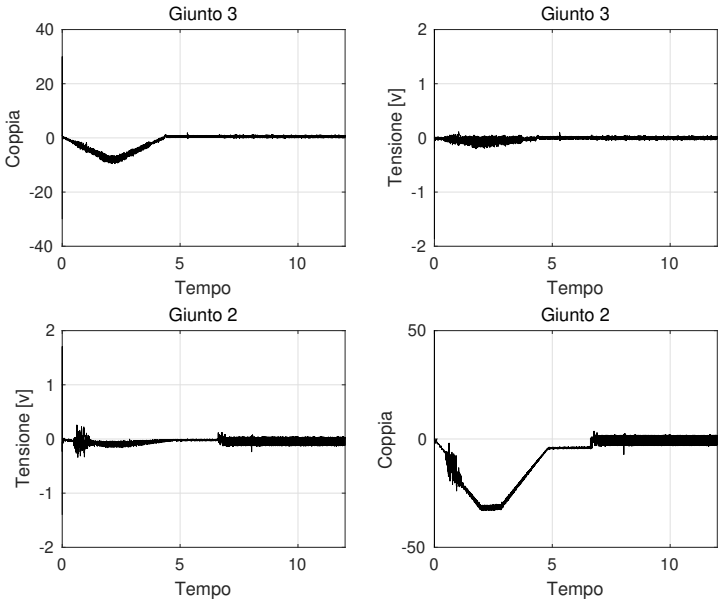


Figure 10.7: Andamenti di coppie e tensioni per entrambi i giunti

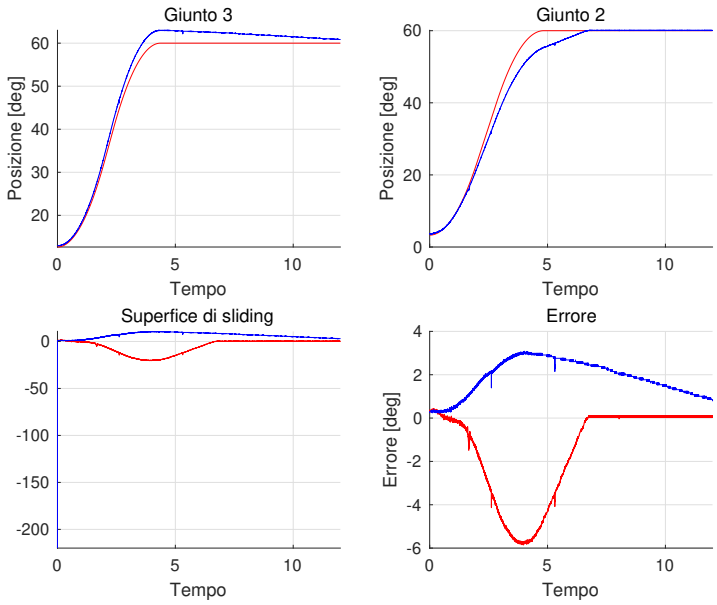


Figure 10.8: Anche qui per l'errore e superifici di sliding osserviamo le due componenti identificative dei giunti, gli altri grafici raffigurano il riferimento e la traiettoria reale

come si può notare i risultati ottenuti sono ampiamente accettabili, con errori delle posizioni angolari dei giunti che si mantengono entro i 6° di range che garantiscono un errore in posizionameto cartesiano entro i 4 *cm* dalla posizione esatta dell'end-effector, e va specificato che entrambi gli errori menzionati hanno un andamento decrescente che tende ad annullarsi, come si può osservare dai grafici sovrastanti. Si può inoltre apprezzare come le superfici di sliding $s_1(x), s_2(x)$ tendano al valore nullo, in accordo con quanto detto pocanzi riguardo gli errori. Infine, si evidenzia come gli andamenti di coppia e tensione siano piuttosto uniformi, non presentando particolari piccoli o brusche variazioni che possano mettere in pericolo gli attuatori.

Chapter 11

Conclusioni e sviluppi futuri

11.1 Conclusioni

Durante il tempo nel quale si è portato avanti questo progetto si sono riscontrate diverse difficoltà.
Dapprima si è dovuto mettere le mani su software piuttosto datati, cercando di mettere d'accordo vecchie versioni di programmi e sistemi operativi con le necessità (attuali nei tempi) di questo progetto. Altre sfide che si sono affrontate sono ad esempio l'adattamento dei parametri di controllo al caso specifico che hanno richiesto numerose prove, e l'interfacciamento di piu script MATLAB che girano su calcolatori diversi e devono attendersi a vicenda dato che scambiano tra loro informazioni necessarie al corretto funzionamento del sistema.
Ad ogni modo il progetto è stato terminato con successo entro i tempi stimati, gli scopi posti in principio sono stati correttamente raggiunti, si è riusciti ad ottenere ottimi risultati sia in simulazione che nell'applicazione reale del sistema di controllo. Si è riusciti anche a costruire correttamente il sistema di visione e ad integrarlo in maniera adeguata con il sistema di controllo del manipolatore. In conclusione, si afferma anche che il lavorare su sistemi non di ultima generazione ha permesso una comprensione approfondita dei fondamenti, permettendo di toccare con mano elementi illustrati a lezione ed inoltre la mancanza di interfacce avanzate, e la consapevolezza dei limiti delle tecnologie precedenti, ha consentito una forte crescita delle skills di problem solving con risorse limitate.

11.2 Sviluppi futuri

Nel corso della costruzione di questo progetto si sono individuati alcuni punti che potrebbero essere ulteriormente sviluppati per incrementare le capacità e la precisione del sistema, tra questi si evidenziano:

- *Implementazione del primo giunto:* si vuole implementare il controllo anche sul primo giunto del robot, questo introdurrebbe un'importante cambiamento nel sistema, in quanto lo spazio di lavoro planare del manipolatore verrebbe esteso ad una sfera di 180° intorno allo stesso;

- *Miglioramento del controllo:* si potrebbero migliorare i parametri per ottenere una fedeltà più ferrea rispetto al riferimento, inoltre modificando i parametri anche per traiettorie differenti da quelle a gradino si avrebbe una mobilità più fluida del manipolatore e la capacità di far compiere a quest'ultimo movimenti più complessi;
- *Miglioramento della visione:* addestrando nuovamente il detector con un dataset più vario si renderebbe il sistema abile a riconoscere molteplici oggetti, con magari una differenzazione del comportamento del robot in funzione della natura del pezzo riconosciuto, costruendo nel modo giusto il set di dati si renderebbe anche più preciso il riconoscimento.
- *Rete neurale per la stima dell'errore:* si potrebbe inoltre introdurre nel software una rete neurale per la stima dell'errore di tracking, così facendo si può rendere il controllo ancora più preciso riuscendo a "prevenire" determinate situazioni.

Bibliography

[1] L. Colombo. *Controllo Robusto di un Manipolatore Industriale con Retroazione di Forza.*

[2] M. Lippera. *Controllo Robusto di un Manipolatore Industriale con Retroazione di Forza.*

[3] L. Villani G. Oriolo B. Siciliano, L. Sciavicco. *"Robotics Modelling, Planning and Control"*. Springer, 2009.

[4] V. Fossi. *Progetto di un Sistema a Struttura Variabile, Basato su Reti Neurali, per il Controllo Robusto di un Manipolatore Industriale.*

[5] S. Antonelli. *Controllo Dinamico a Struttura Variabile di un Manipolatore Antropomorfo.*