UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Universitá Politecnica delle Marche - Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

CONTROLLO DI UN MANIPOLATORE INDUSTRIALE MEDIANTE COMPUTER VISION

Robotic arm control using computer vision

Candidate:
Giorgio Marmolino

Advisor:
Prof. Gianluca Ippoliti

Coadvisor:
Prof. Giuseppe Orlando

Academic Year 2023-2024

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Universitá Politecnica delle Marche - Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

CONTROLLO DI UN MANIPOLATORE INDUSTRIALE MEDIANTE COMPUTER VISION

Robotic arm control using computer vision

Candidate:
Giorgio Marmolino

Advisor:
Prof. Gianluca Ippoliti

Coadvisor:
Prof. Giuseppe Orlando

Academic Year 2023-2024

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

A chi studia tutto il giorno,
per avere tutto un giorno.

Ringraziamenti

Voglio ringraziare il prof. Ippoliti e il prof. Orlando per l'opportunità, la disponibilità e per i preziosi consigli per la realizzazione di questo elaborato e del progetto che descrive.

Ringrazio Marco, per aver cooperato con me nella riuscita di questo lavoro, per aver affrontato con me ogni difficolta' apparentemente insormontabile condividendo giorno per giorno ogni minimo traguardo.

Voglio ringraziare mia madre, mio padre e i miei fratelli per avermi sostenuto ciecamente nel mio percorso universitario anche quando ho deciso di studiare argomenti di complessita' tale da non sapere neanche come si scrivessero, e per essermi stati vicini nei momenti di maggiore difficolta'.

Una menzione particolare va a Gloria Astolfi, per tutti i suggerimenti che mi ha saputo dare nell'ambito universitario, per aver gioito con me di ogni minimo successo, per aver ascoltato ogni mia singola lagna senza insultarmi, ma soprattutto per il continuo supporto psicologico.

Voglio ringraziare la Croce Rossa Italiana, per aver rappresentato un rifugio nel quale ripararmi dalle cattive giornate e dai pessimi umori, per avermi cresciuto ancora un po' responsabilizzandomi, ma soprattutto per avermi fatto vivere esperienze tali da rendermi piu' consapevole del valore della vita.

Ringrazio gli amici, in particolare Riccardo che ha condiviso con me ogni giornata universitaria, lezioni e esami, e tutte le persone che ho incontrato in questi anni, che mi hanno regalato la giusta dose di sarcasmo e ironia per affrontare le giornate folli, rendendole divertenti ed interessanti.

Voglio ringraziare la vita per le difficoltà che ha messo lungo il mio percorso, facendomi cadere e insegnandomi a rialzarmi e ad azzardare anche quando le speranze sembrano essere svanite.

Infine ringrazio me stesso, per non aver mai smesso di sognare.

Ancona, Ottobre 2024

Giorgio Marmolino

Abstract

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs. It involves the development of algorithms and models to automate tasks that the human visual system can do, such as object recognition, image classification, and scene understanding. Object detection is a specific task within computer vision that involves identifying and locating objects within an image or video. It not only classifies objects (e.g., cat, dog, car) but also provides their coordinates in the frame. Common techniques and models used for object detection include traditional methods such as Haar cascades and HOG (Histogram of Orient,ed Gradients) detectors, or Deep Learning-Based Methods: Including Convolutional Neural Networks (CNNs) and more advanced architectures like R-CNN (Region-based CNN), YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) and others. In industrial robotics, computer vision and object detection are pivotal for automating various tasks, enhancing precision, and improving efficiency for applications like quality control and inspection, machine vision in collaborative robots (Cobots) or vision and measurement. This thesis is a report about my project in which I want to control an industrial manipulator using a Variable Structur Control (precisely a Sliding Mode Control) combined with computer vision to allow it to recognise target objects using object detection, find their coordinates in the workspace and touch them with robot's end-effector, using a pre-calculated trajectory based on the intial and final positions, limits of speed and acceleration.

Sommario

La computer vision è un campo dell'intelligenga artificiale (IA) che permette ai computer e ai sistemi di ottenere informazioni significative da immagini digitali, video e altri input visivi. Coinvolge lo sviluppo di algoritmi e modelli per l'automatizzazione di processi che possono essere svolti attraverso il sistema visivo umano come il rilevamento di oggetti, classificazione delle immagini e comprensione delle scene. Il rilevamento di oggetti è un compito particolare che fa uso della computer vision che consiste nell'identificare e localizzare oggetti in immagini o video. Non solo classifica oggetti (gatto, cane, auto, ecc.) ma ne determina anche le coordinate nel frame. Tecniche comuni e modelli usati nel rilevamento di oggetti includono metodi tradizionali quali Haar cascades e HOG (Histogram of Orient,e Gradients) rilevatori o metodi basati sul deep learning quali Convolutional Neural Networks (CNNs) e architetture più avanzate come R-CNN (Region-based CNN), YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) e altri. Nella robotica industriale, la computer vision e il rilevamento degli oggetti sono fondamentali per l'automazione di diversi processi, aumentando la precisione e migliorando l'efficenza per applicazioni riguardo controllo qualità e ispezione, visione per robot collaborativi (Cobots) o visione e misurazione. Questa tesi è un report riguardo questo progetto nel quale si desidera controllare un manipolatore industriale utilizzando un controllo a struttura variabile (VSC) in particolare uno Sliding mode control (SMC) combinato con l'udo della visione artificiale per il riconoscimento di oggetti target, trovarne le coordinate sul piano di lavoro e permettere al manipolatore di toccare l'oggetto con la sua pinsa utilizzando traiettorie pre-calcolate basandosi sulla posizione iniziale e finale, con limiti di velocità ed accelerazione.

Contents

List of Figures

List of Tables

Chapter 1

Introduzione

1.1 Introduzione alla tematica

L'argomento trattato all'interno di questo elaborato riguarda il mondo dei sistemi industriali (catene di produzione, controlli automatici, reti di sensori, ecc.), ognuno dei quali può essere schematizzato come un blocco generico che richiede una sollecitazione di un certo tipo in ingresso per poter produrre, secondo il suo utilizzo, un'uscita deguata. Per fare in modo che il processo lavorativo di questi sistemi si attenga alle specifiche desiderate è sempre necessario che ci sia un controllo su quello che arriva al sistema e quello che esso produce come risultato. Ragion per cui ci si affida a dei sistemi di controllo, i quali scambiano informazioni con i processi fisici (attraverso sensori ed attuatori) o con un operatore umano. Tali sistemi di controllo realizzano, in maniera automatica, gli algoritmi necessari affinché il comportamento del processo fisico sia quello desiderato. Un dispositivo di controllo è un particolare sistema per l'elaborazione dell'informazione, destinato al controllo dei processi fisici, il quale deve potersi interfacciare con l'ambiente esterno. Le sue funzionalità generalmente devono essere quelle di controllo a ciclo chiuso classico del sistema (regolazione o asservimento), calcolo dei valori di riferimento (set-point), gestione di eventuali allarmi ed anomalie e infine realizzazione dell'interfaccia di comunicazione con operatore o altri dispositivi. Al giorno d'oggi esistono molti tipi di sistemi di controllo, più o meno sofisticati, realizzati con diverse tecniche; qui ci occuperemo di una particolare tipologia chiamata VSC.

1.2 Tipologia di controllo sul manipolatore

La scelta del controllo da applicare sull'antropomorfo Barras ERICC è ricaduta su di una tipologia di controllo mediante il VSC, denominata Sliding Mode Control (SMC). Si è scelto di controllare il robot in posizione e corrente. Gia da qui si capisce che il controllo tiene in considerazione le due dinamiche rispettivamente quella dei motori e quella dell'antropomorfo. Si è scelta la corrente come controllo interno perché tale segnale permette di ridurre la sensibilità alle variazioni dei parametri del motore, cosa che non sarebbe stato possibile ottenere controllando il motore in tensione. In questo modo gli attuatori si comportano come generatori controllati di coppia, infatti

si può vedere in seguito come la coppia sia direttamente proporzionale alla corrente.

1.3 Uso della visione

Al fine di permettere al sistema di capire cosa è presente sul piano di lavoro si fa uso della computer vision, in questo modo è possibile non solo rilevare l'oggetto, ma capire se è il target e nel caso ricavarne le cordinate al fine di poterlo raggiungere o effettuare operazioni su quest'ultimo. Per fare ciò è sufficente una semplice usb camera tra quelle presenti in commercio, in modo tale da rivelarsi una scelta corretta in quanto permette di ridurre la sensoristica e aumentare l'informazione. Tra i vari metodi di rilevamento si è scleto l'ACF (Aggregate Channel Features) in quanto di semplice implementazione e con una complessità computazionale relativamente bassa rispeto ad altri sistemi, consentendo in questo modo di causare latenze non considerevoli dal momento in cui viene catturata l'immagine del piano di lavoro a quando il target viene rilevato.

1.4 Toolbox utilizzati

Per la realizzazione di questo progetto è stato necessario installare sul PC esterno dei toolbox in particolare al fine di permettere di effettuare il rilevamento dell'oggetto e il calcolo delle sue coordinate. I toolbox necessari sono:

- Computer vision toolbox;
- Computer vision toolbox model for YOLO v4 Object Detection;
- Image processing toolbox;
- Instrument control toolbox;
- Matlab Coder;
- Control system toolbox;
- Deep Learning Toolbox;
- Symbolic Math Toolbox;

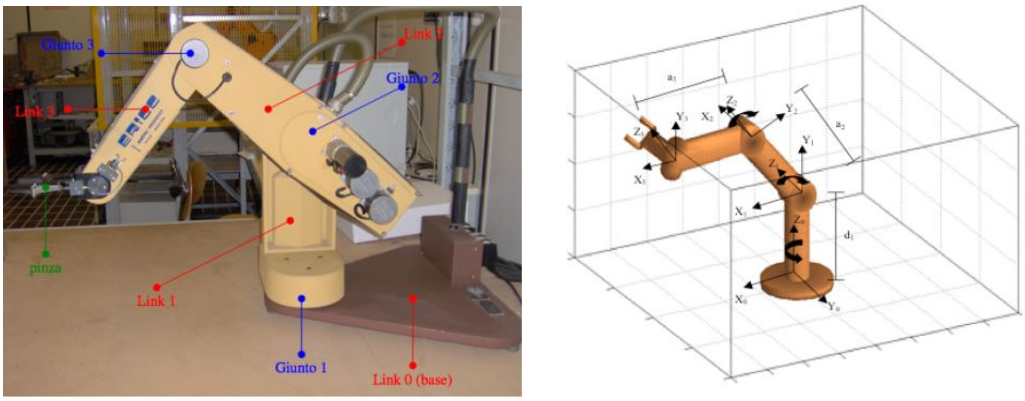
Sull'eleboratore di laboratorio collegato al manipolatore Barras ERICC non è necessario installare ulteriori toolbox a quelli già presenti.

Chapter 2

Il manipolatore

2.1 Il manipolatore

Il robot utilizzato è un manipolatore antropomorfo Barras ERICC a 5 gradi di libertà: tre per il posizionamento dell'end-effector e due per il suo orientamento. Per ricavare i parametri relativi al modello in spazio di stato che descrivono il robot (e che ne consentiranno il controllo) si scelgono i sistemi di riferimento solidali ad ogni link in accordo con la convenzione di Denavit-Hartenberg, dove però si è omesso che l'orientamento dell'end-effector non si mantiene costante a causa della struttura del manipolatore (e quindi facendo ulteriormente variare la matrice d'inerzia del terzo link e di conseguenza di tutto il manipolatore), considerando quindi che l'end-effector sia fisso e in linea con l'avambraccio.



Rappresentazione del manipolatore secondo la convenzione di Denavit-Hartenberg

Nello studio in questione i parametri utilizzati per il modello in spazio di stato e per le applicazioni in simulink sono stati estrapolati dalla tesi di Colombo Luigi "Controllo di coppia di un robot industriale", di seguito riportati.

Parametri relativi al link 1:

Table 2.1: Parametri realtivi al link 1

$\alpha_1[°]$	$a_1[m]$	$\theta_1[°]$	$d_1[m]$	$\sigma_1[m]$	$m_1[kg]$
90	0	0	0.34	0	10.1939

$r_{1x}[m]$	$r_{1y}[m]$	$r_{1z}[m]$
0.028	-0.1405	-0.0995

$I_1[kg \cdot m^2]$	x	y	z
x	0.1533	0	0
y	-0.0144	0.0748	0
z	0.0078	0.0144	0.1757

$J_{m1}[kg \cdot m^2]$	G_1	$B_1[N \cdot m \cdot s]$	$T_{e1}^{+}[N \cdot m \cdot s]$	$T_{e1}^{-}[N \cdot m \cdot s]$
33·E-7	410	1.48·E-3	0.395	-0.435

Parametri relativi al link 2:

Table 2.2: Parametri realtivi al link 2

$\alpha_2[^\circ]$	$a_2[m]$	$\theta_2[^\circ]$	$d_2[m]$	$\sigma_2[m]$	$m_2[kg]$
0	0.28	0	0	0	10.5102

$r_{2x}[m]$	$r_{2y}[m]$	$r_{2z}[m]$
-0.0142	0	0.003

$I_2[kg \cdot m^2]$	x	y	z
x	0.0457	0	0
y	0.0029	0.3371	0
z	0.0117	0.0011	0.3159

$J_{m2}[kg \cdot m^2]$	G_2	$B_2[N \cdot m \cdot s]$	$T_{e2}^{+}[N \cdot m \cdot s]$	$T_{e2}^{-}[N \cdot m \cdot s]$
33·E-7	1233.33	0.817·E-3	0.126	-0.071

Parametri relativi al link 3:

Table 2.3: Parametri realtivi al link 3

$\alpha_3[°]$	$a_3[m]$	$\theta_3[°]$	$d_3[m]$	$\sigma_3[m]$	$m_3[kg]$
0	0.3175	0	0	0	2.2449

$r_{3x}[m]$	$r_{3y}[m]$	$r_{3z}[m]$
-0.266	-0.0052	-0.0013

$I_3[kg \cdot m^2]$	x	y	z
x	0.0044	0	0
y	-0.0012	0.0061	0
z	0.0002	0	0.05851

$J_{m3}[kg \cdot m^2]$	G_3	$B_3[N \cdot m \cdot s]$	$T_{e3}^{+}[N \cdot m \cdot s]$	$T_{e3}^{-}[N \cdot m \cdot s]$
33·E-7	52.0833	1.38·E-3	0.132	-0.105

2.2 I Motori

Per generare il movimento dei giunti si utilizzano motori in corrente continua; questo tipo di attuatori sono dotati di un magnete permanente a due poli che funge da statore e un avvolgimento conduttore che funge da rotore, dove attraverso un dispositivo a spazzole, dato da una combinazione strisciante di segmenti di rame e grafite, si genera la commutazione elettrica necessaria a provocare la rotazione del flusso magnetico del rotore. La potenza nominale dei motori è di 27W con una tensione di azionamento di tra ±18V. Le equazioni caratteristiche del motore sono:

$$\begin{cases} V_a = R_a I_a + L_a \dot{I}_a + E_m \\ E_m = K_e \Omega \\ C_m = K_m I_A \end{cases} .$$

Con:

- V_a tensione di armatura;
- I_a corrente di armatura;
- R_a, L_a rispettivamente resistenza e induttanza degli avvolgimenti di armatura;
- E_m forza controelettromotrice, legata dalla costante K_e alla velocità angolare Ω del'asse del motore;
- C_m coppia prodotta, K_m costante di coppia;

2.3 I Trasduttori

Per conoscere la posizione e la velocità, su ogni giunto del manipolatore sono presenti potenziometri (direttamente installati sugli assi dei giunti) e dinamo tachimetriche (installate sull'asse di ogni singolo motore). Il potenziometro è di tipo a strato a plastiche conduttrici con errore di linearità dello 0.05% e costante di trasduzione di 44.4 mV/° . Le dinamo tachimetriche sono dispositivi con eccitazione a magneti permanenti; la relazione che li caratterizza è:

$$V_u = K_t \Omega$$

con K_t costante tachimetrica, Ω velocità angolare e V_u tensione in uscita. Nelle dinamo installate la costante tachimetrica è di 1V/1000RPM.

2.4 Console di controllo e interfaccia

Il sistema di controllo del manipolatore Barras ERICC (Figura 2.1) è costituito da:

- Scheda madre dotata di microprocessore;
- 5 moduli di potenza (power boards) per il controllo dei motori;
- CPU;
- Scheda di interfaccia PC-robot.

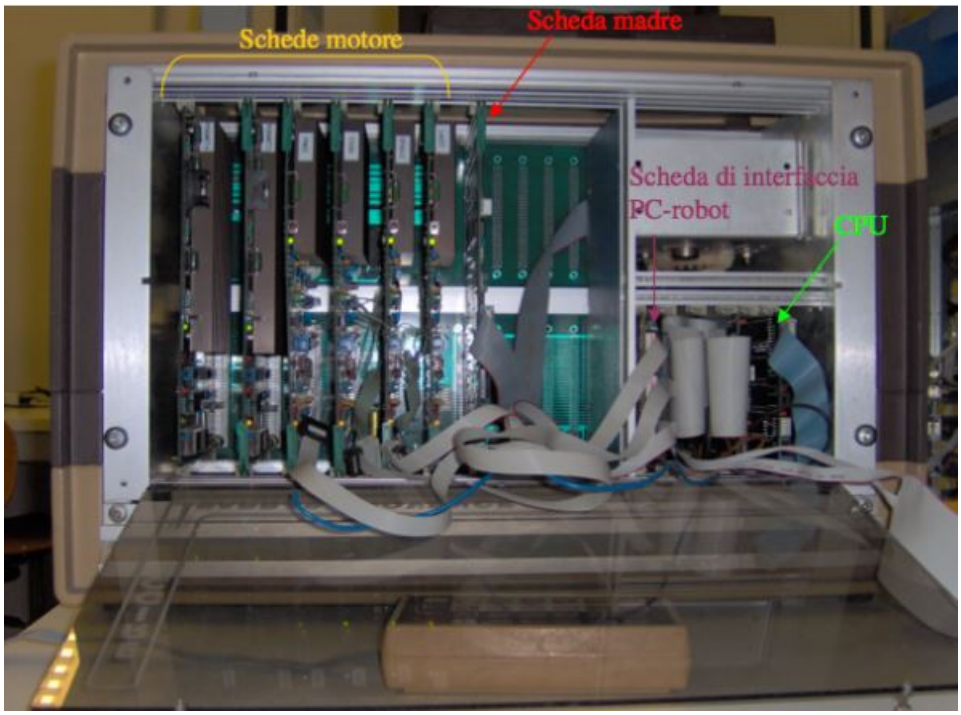


Figure 2.1: Console di controllo

Quest'ultima è fondamentale in quanto nel funzionamento proposto si esclude volutamente il controllore originale della console e si implementa l'asservimento via software attraverso il PC. In questo modo data una traiettoria che si vuole eseguire, attraverso i trasduttori precedentemente citati, il software di controllo genera un segnale che viene trasmesso dalla scheda di acquisizione in ingresso allo stadio di amplificazione dei moduli di potenza dei motori. Nella Figura 2.2 è riportato lo schema di collegamento della console.

2.5 Interfaccia di collegamento, controllo e trasmissione dati

L'interfaccia di collegamento (Figura 2.3) ha il compito di abilitare le funzioni software (disabilitando il controllo originale) e permettere l'acquisizione dei dati. La scheda

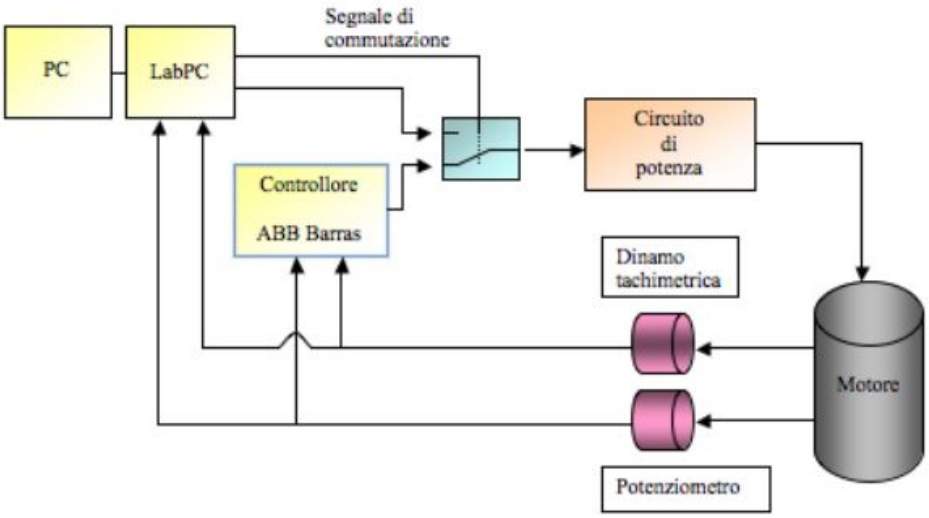
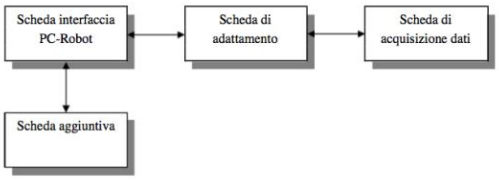
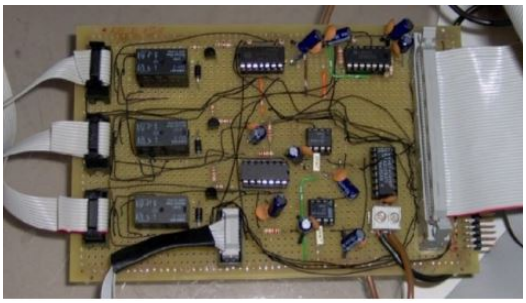


Figure 2.2: Schema di collegamento della console

per l'acquisizione dei dati è una PCI-6024E prodotta dalla National Instruments, installabile direttamente su motherboard di un PC mediante slot PCI. La scheda viene riconosciuta e pilotata mediante il software Matlab/Simulink in tempo reale; inoltre questa attraverso delle uscite digitali collegate a relè ha il compito di escludere la logica di controllo originale e realizzare anche lo "stop" di emergenza attraverso il pulsante a fungo.



(a) Schema di collegamento delle schede



(b) Scheda interfaccia PC-Robot

Figure 2.3: Interfaccia di collegamento

La scheda di interfaccia pc-robot, di cui si evidenzia un lato in cui sono presenti i relè per l'esclusione del controllo originale (e quindi l'abilitazione di quello software) e l'altro lato della scheda con i circuiti analogici che realizzano il sistema di multiplexing (che consentirebbe anche il controllo del primo giunto). Consente lo scambio dei dati con la scheda di acquisizione mediante una scheda di adattamento (Figura 2.4), dove il compito di quest'ultima è quello di adattare le uscite della prima scheda (connessa con cavo flat da 50 linee) a quello della seconda (connessa con cavo flat da 68 linee), attraverso un sistema di morsettiere collegate mediante fili.

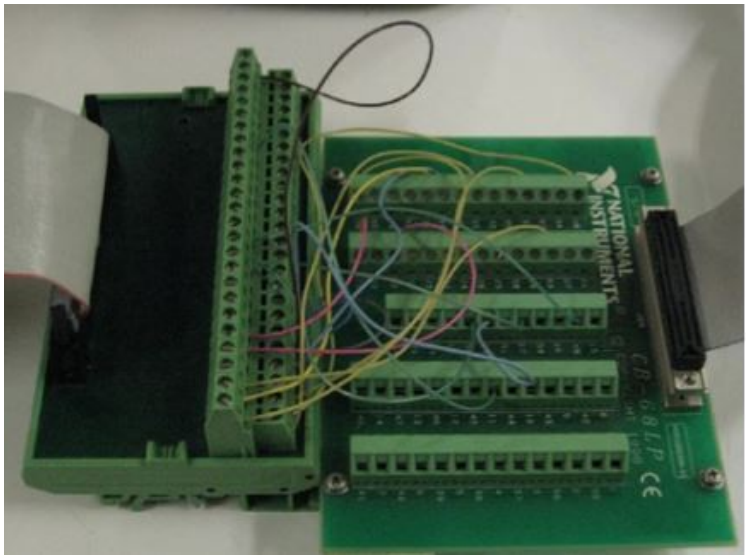


Figure 2.4: Scheda di adattamento

Infine, la scheda aggiuntiva in Figura 2.5, collegata direttamente sia alla scheda di interfaccia che alla DAQ, permette l'acquisizione delle correnti di armatura (in maniera indiretta) e reinstradare nella console i segnali delle velocità dei giunti. La lettura della corrente avviene leggendo la caduta di tensione su una resistenza inserita in serie al motore da cui si è poi calcolata la corrente mediante la legge di ohm. Il dimensionamento del resistore è stato pensato per favorire la precisione della lettura della posizione per un miglior controllo a discapito delle prestazioni dei motori di ERICC dal momento che il compito è quello di rendere il controllo il più preciso possibile e di poter dissipare potenza in ogni condizione di utilizzo prendendo come riferimento la condizione operativa più sfavorevole, ovvero quella in cui l'assorbimento di corrente è massimo.

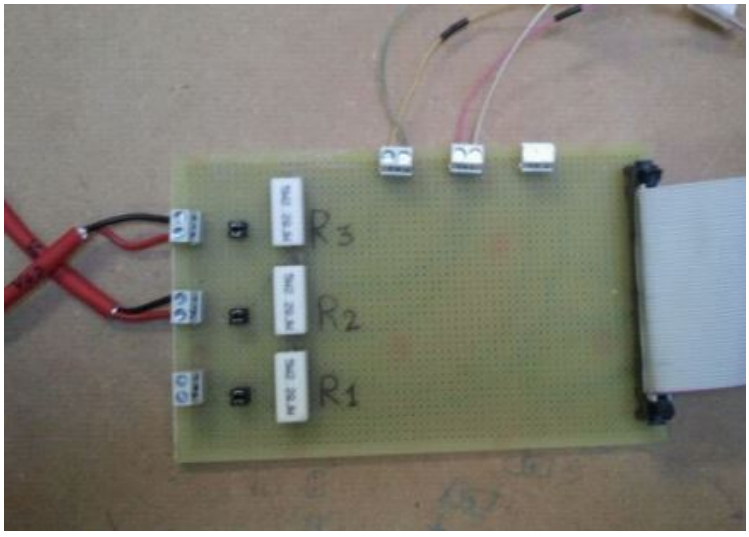


Figure 2.5: Scheda aggiuntiva

2.6 Modello in spazio di stato del manipolatore

Il braccio antropomorfo in questione dal punto di vista meccanico può essere visto come una catena di corpi rigidi (link) interconnessi da giunti, dove un estremità della catena è fissata ad una base di supporto mentre l'opposta è dotata di una pinza (end-effector) per la manipolazione di oggetti. Dovendo effettuare un'analisi cinematica del robot, si possono utilizzare due tipi di approcci, uno diretto secondo il quale, note le posizioni assunte dai giunti è possibile ricavare la posizione dell'end-effector, e uno inverso, per cui nota la posizione e l'orientamento dell'end-effector, insieme ai parametri geometrici si possono definire gli angoli che devono assumere i giunti al fine di consentire quella posizione. Nel caso in analisi, volendo posizionare il robot in una maniera prestabilita, si utilizzerà il secondo approccio. Al fine di ottenere una rappresentazione spaziale del braccio si utilizza l'equazione di Denavit-Hartenberg, scritta solitamente come sequenza di trasformazioni geometriche tra i sistemi di riferimento associati a ciascun giunto del robot, definendo come una specifica configurazione delle articolazioni del robot influenzi la posizione e l'orientamento dell'end-effector.

$$A_i = R_{(z,\theta_i)} T_{(z,d_i)} T_{(x,a_i)} R_{(x,\alpha_i)} \tag{2.1}$$

Dove:

- A_i è la matrice di trasformazione omogenea associata al giunto i.
- $R_{(z,\theta_i)}$ è una rotazione attorno all'asse Z di un angolo θ_i
- $T_{(z,d_i)}$ è una traslazione lungo l'asse Z di una distanza d_i
- $T_{(x,a_i)}$ è una traslazione lungo l'asse X di una distanza a_i
- $R_{(x,\alpha_i)}$ è una rotazione attorno all'asse X di un angolo α_i

Queste equazioni devono essere utilizzate in sequenza per ottenere la trasformazione completa dal sistema di riferimento del giunto i-esimo al sistema di riferimento del giunto (i+1)-esimo. Come precedentemente accennato queste equazioni hanno validità per le catene cinematiche aperte, quindi per dispositivi i cui grafi descrittivi non presentano diramazioni e di cui i giunti hanno un solo grado di libertà (come nel caso di giunti rotazionali o traslazionali).

In questo modo, al fine di determinare le matrici di trasformazione, sarà sufficiente numerare i giunti da 1 a n, e i link da 0 a n (ricordando che i link sono in numero n+1), dove il link *0* rappresenta la base del manipolatore e l'n-esimo è solidale con il sistema di riferimento dell'end-effector.

Volendo calcolare la sopracitata matrice di trasformazione, si deve considerare la tipologia di giunto che determinerà l'asse 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 quello di rotazione.

L'asse di corpo è definito come la normale comune agli assi di giunto adiacenti. Così facendo per ogni A_i si fissa il sistema di coordinate solidale al giunto i-esimo come segue:

1. Si sceglie l'origine del sistema di riferimento O_i nell'intersezione tra l'asse di corpo i e l'asse di giunto i + 1;
2. Si sceglie l'asse z_i coincidente con l'asse di giunto i + 1 con verso arbitrario;
3. Si sceglie l'asse x_i coincidente con l'asse di corpo i + 1 con verso dal giunto i al giunto i + 1; nel caso di assi di giunto intersecanti si sceglie il verso del prodotto vettoriale $\vec{z_{i-1}} \times \vec{z_i}$;
4. L'asse y_i è determinato in modo da formare una terna destrorsa.

Una volta stabiliti i sistemi di riferimento solidali ai link, la geometria del manipolatore viene definita dai seguenti parametri:

- Dimensioni del link i:
  - lunghezza o distanza normale a_i: è la distanza tra gli assi z_i;
  - angolo di avvitamento α_i: è l'angolo formato dalle proiezioni ortogonali degli assi z_i;
- Posizione relativa dei due link connessi al giunto i:
  - distanza tra i due link d_i: è la distanza tra l'origine O_{i-1} e l'asse x_i misurata lungo l'asse z_{i-1};
  - angolo tra i due link θ_i: è l'angolo formato dalle proiezioni ortogonali degli assi x_{i-1} e x_i su un piano perpendicolare all'asse z_{i-1}.

In figura 2.6 sono mostrati il modello di coordinate dei link e il modello di Denavit-Hartenberg.

Sulla base di quanto stabilito si può sviluppare una matrice di trasformazione omogenea che metta in relazione il sistema di coordinate i-esimo col sistema di coordinate (i-1)-esimo:

$$A_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

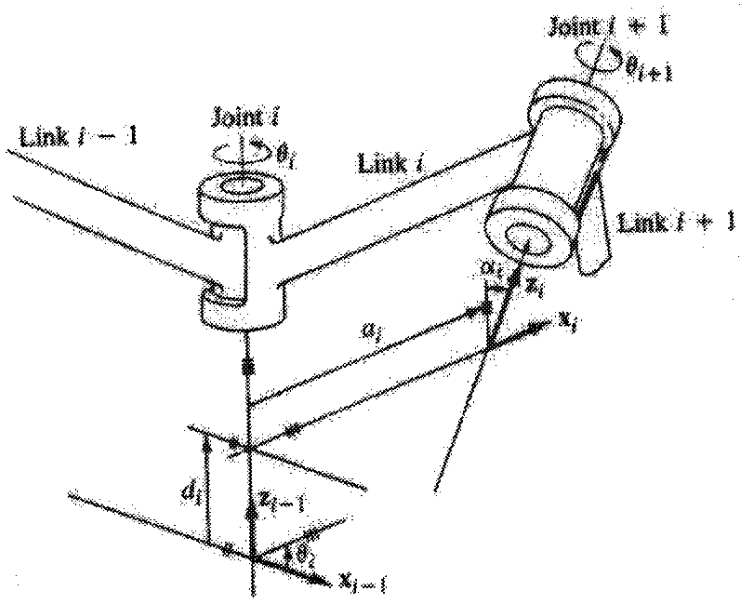


Figure 2.6: Sistema di coordinate dei link

2.7 Equazioni di Lagrange

Le equazioni di Lagrange permettono una descrizione fisica alternativa a quella Newtoniana, facendo uso delle forze generalizzate di Lagrange, che al contrario di come suggerisce il nome non hanno sempre la dimensioni fisiche di una forza ma talvolta possono anche esprimere un momento (che prenderà il nome di momento generalizzato). Queste equazioni possono essere usate per lo studio di sistemi conservativi e non, tra cui il braccio antropomorfo in esame, di cui si vuole conoscere il momento generalizzato τ_i che consenta il movimento dell'i-esimo link del manipolatore. La funzione lagrangiana L, è definita come differenza tra l'energia cinetica T di un copro rigido e la sua energia potenziale V, e attraverso una serie di trasformazioni (omesse) è possibile ricavare le equazioni di Lagrange per sistemi non conservativi:

$$\tau_i = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right) \tag{2.3}$$

In forma matriciale si ottiene il modello matematico del braccio robotico.

$$u(t) = M(q(t))\ddot{q}(t) + V(q(t), \dot{q}(t)) + H(q(t)) \tag{2.4}$$

Dove:

- $q(.) \in \mathbb{R}^n$ è il vettore delle posizioni generalizzate;
- $\dot{q}(.) \in \mathbb{R}^n$ è il vettore delle velocità generalizzate;
- $\ddot{q}(.) \in \mathbb{R}^n$ è il vettore delle accelerazioni generalizzate;

- $M(q(.))$ è la matrice inerziale simmetrica *nxn* relativa all'accelerazione e dipende dalla posizione;
- $V(q(.), \dot{q}(.))$ è una matrice *nxn* che descrive gli effetti centrifughi e di Coriolis;
- $H(q(.))$ è il vettore che tiene conto della forza di gravità;
- $u(.)$ è il vettore delle forze generalizzate applicate ai giunti.

Il modello in spazio di stato lo si ottiene portando il modello matematico ottenuto nella forma:

$$\dot{x}(t) = f(x(t), t) + g(x(t), t)v(t) \tag{2.5}$$

Che rappresenta l'espressione generale di un modello in spazio di stato per un sistema dinamico non lineare a tempo continuo; il modello può essere risolto rispetto a $\ddot{q}(t)$:

- $x(t) = \begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix}$
- $v(t) = u(t)$

e ponendo:

$$\tilde{f}(x(t)) = M^{-1}q(t)[-V(q(t), \dot{q}(t))\dot{q}(t) - H(q(t))] \tag{2.6}$$

$$\tilde{g}(x(t)) = M^{-1}q(t) \tag{2.7}$$

$$f(x(t)) = \begin{bmatrix} \dot{q}(t) \\ \tilde{f}(x(t)) \end{bmatrix} \tag{2.8}$$

$$g(x(t)) = \begin{bmatrix} 0_n \\ \tilde{g}(x(t)) \end{bmatrix} \tag{2.9}$$

si ottiene il modello in spazio di stato del manipolatore.

2.8 Le Traiettorie

Per traiettoria si intende il percorso nello spazio di lavoro di una parte di un cinematismo al fine di eseguire un determinato task. Si è specificato che la traiettoria viene seguita nello spazio di lavoro, ma il movimento viene generato nello spazio degli attuatori che può differire dallo spazio di lavoro. La pianificazione di un generico percorso da un punto iniziale A ad un punto finale B contiene implicitamente informazioni sul profilo della velocità e dell'accelerazione, che possono variare a seconda della tecnica utilizzata per la generazione della traiettoria, dove quest'ultima dovrà sottostare a vincoli fisici dovuti alle capacità massimali degli attuatori e alle massime forze cui il manipolatore può essere sottoposto.

2.8.1 Generazione delle traiettorie mediante polinomi

Pianificare la traiettoria nello spazio degli attuatori (e quindi nel caso in esame nello spazio dei giunti) coincide con definire la legge del moto che di conseguenza determina il movimento associato. Oltre ai vincoli precedentemente citati su velocità e accelerazione massime consentite, se ne aggiungono altri sulla continuità della traiettoria e delle sue derivate parziali del primo e secondo ordine (corrispondenti quindi a velocità ed accelerazione). Al fine di rispettare tutti i vincoli sono state utilizzate delle traiettorie basate su funzioni polinomiali:

$$s(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n \tag{2.10}$$

dove il grado del polinomio influisce sul tipo di traiettoria (lineare, parabolica ecc.), mentre i coefficienti $a_0, \ldots, a_n$ sono calcolati in base alle condizioni iniziali su posizioni, velocità ed accelerazioni. Per n=1 si ottiene una traiettoria lineare:

$$s(t) = a_0 + a_1(t - t_0) \tag{2.11}$$

Dove i coefficienti:

$$a_0 = q_0 a_1 = \frac{q_1 - q_0}{t_1 - t_0} \tag{2.12}$$

con q_0 e q_1 rispettivamente posizione iniziale e finale della traiettoria. In questo modo si ottengono i profili di velocità ed accelerazione riportati in Figura 2.7:

dove si può notare che l'andamento della traiettoria presenta brusche variazioni, quello della velocità una discontinuità nel punto di inversione della traiettoria. Questi difetti possono essere corretti aumentando opportunamente il grado della funzione polinomiale che descrive la traiettoria, al fine di ottenerne uno con un profilo più smussato e che non presenti discontinuità. In questo modo è possibile definire il generatore di traiettorie (che fornisce profili come rampe e gradini), dove sarà necessario collegare in cascata un filtro, basato su di uno schema in retroazione (Figura 2.8) e su tecniche di controllo non lineari a struttura variabile (VSC), è progettato in modo che la sua uscita segua al meglio il riferimento esterno, rispettando i vincoli di velocità ed accelerazione massima consentita, attraverso un riferimento più smussato di quello proposto. Quindi la generazione della traiettoria segue questo schema:

Dove $r(t)$ è la traiettoria che si desidera seguire che ha un andamento come quella mostrato nella Figura 2.9:

Mentre $q(t)$ in uscita dallo schema è la traiettoria ottenuta facendo uso del filtro che insegue al meglio la traiettoria precedentemente proposta, rispettando i vincoli su velocità e accelerazioni massime consentite.

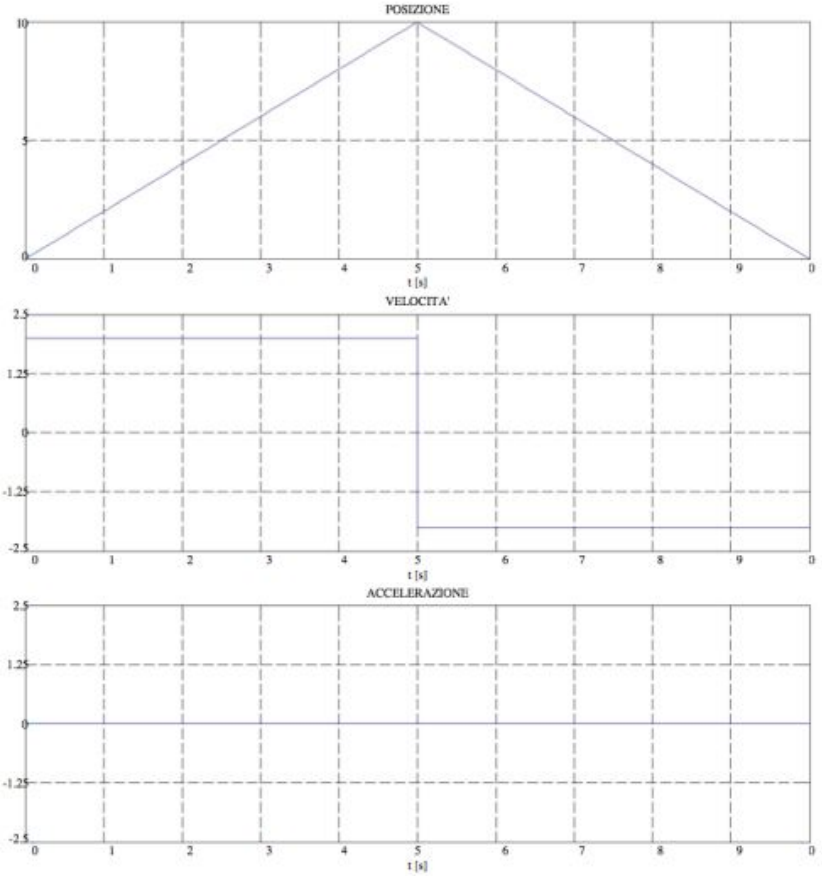


Figure 2.7: Traiettorie mediante polinomi

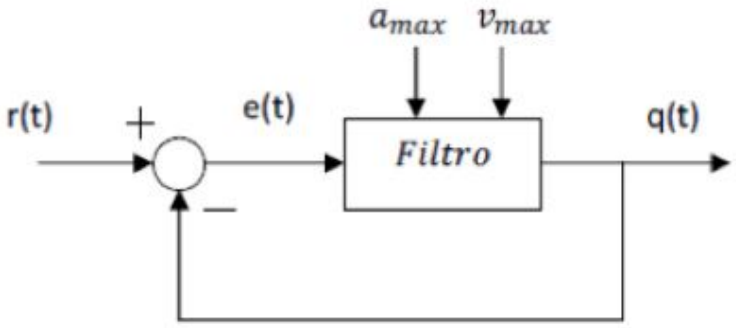


Figure 2.8: Schema in retroazione

2.8.2 Il generatore di traiettorie

Il generatore di traiettorie in simulink è realizzato a tempo discreto (Figura 2.12), dove la legge di controllo negli istanti $t = kT_c$ calcola l'accelerazione u_k che viene poi integrata per ottenere velocità e posizioni desiderate (Figura 2.10). In particolare nella sua creazione gli ingressi di interesse per la generazione della traiettoria sono due:

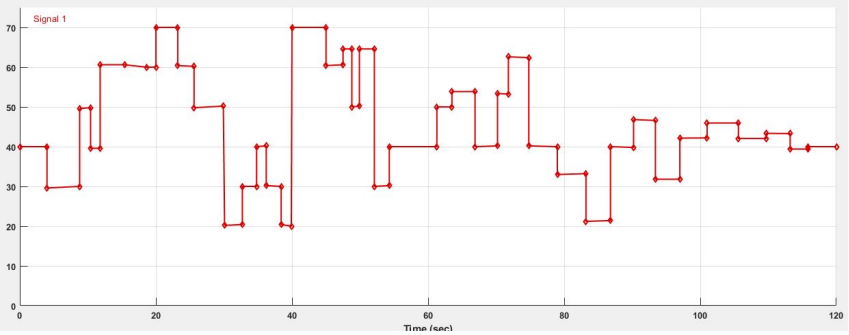


Figure 2.9: Traiettoria non filtrata

il primo r(k) dove entra l'effettivo andamento della traiettoria (indipendentemente che sia un riferimento variabile o costante nel tempo) e il secondo è il q_{20} / q_{30} che rappresentano i valori delle posizioni inziali; questi valori vengono dati al sistema facendo una lettura preliminare dei valori di posizioni al fine di permettere al generatore di restituire una traiettoria da seguire che parta dal valore attuale del riferimento; al contrario, se la traiettoria non iniziasse dallo stesso punto in cui si trova il manipolatore si avrebbe da subito un errore elavato che la dinamica del sistema potrebbe non essere in grado di seguire nel tempo, non raggiungendo mai la posizione desiderata o seguendo male il riferimento variabile nel tempo. Rispetto al generatore originale sono state effettuate delle modifiche, in particolare all'uscita del generatore di traiettorie non si ha più il segnale che viene ritardato di due campioni ma quello attuale (in questo caso l'uscita numero 3).

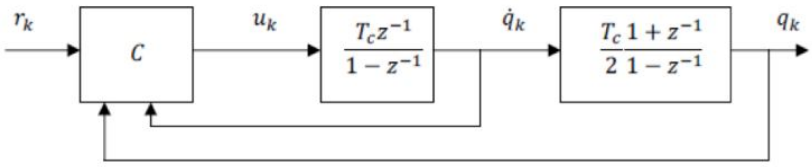


Figure 2.10: Traiettoria a tempo discreto

Da cui si ricavano, facendo uso di un'approssimazione di tipo rettangolare per la velocità e di una di tipo trapezoidale per la posizione:

$$\begin{cases} \dot{q}_k = \dot{q}_{k-1} + T_c u_{k-1} \\ q_k = q_{k-1} + \dfrac{T_c}{2}(\dot{q}_k + \dot{q}_{k-1} \end{cases} . \tag{2.13}$$

Che permette infine di ottenere i profili di posizione, velocità e accelerazione mostrati in Figura 2.11:

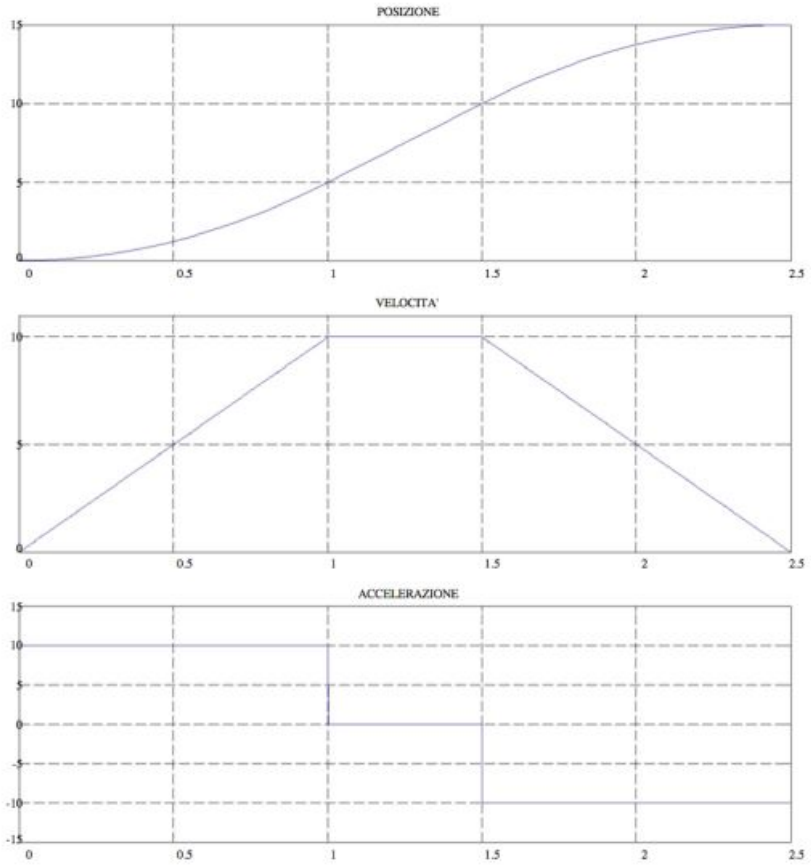


Figure 2.11: Profili filtrati ottenuti

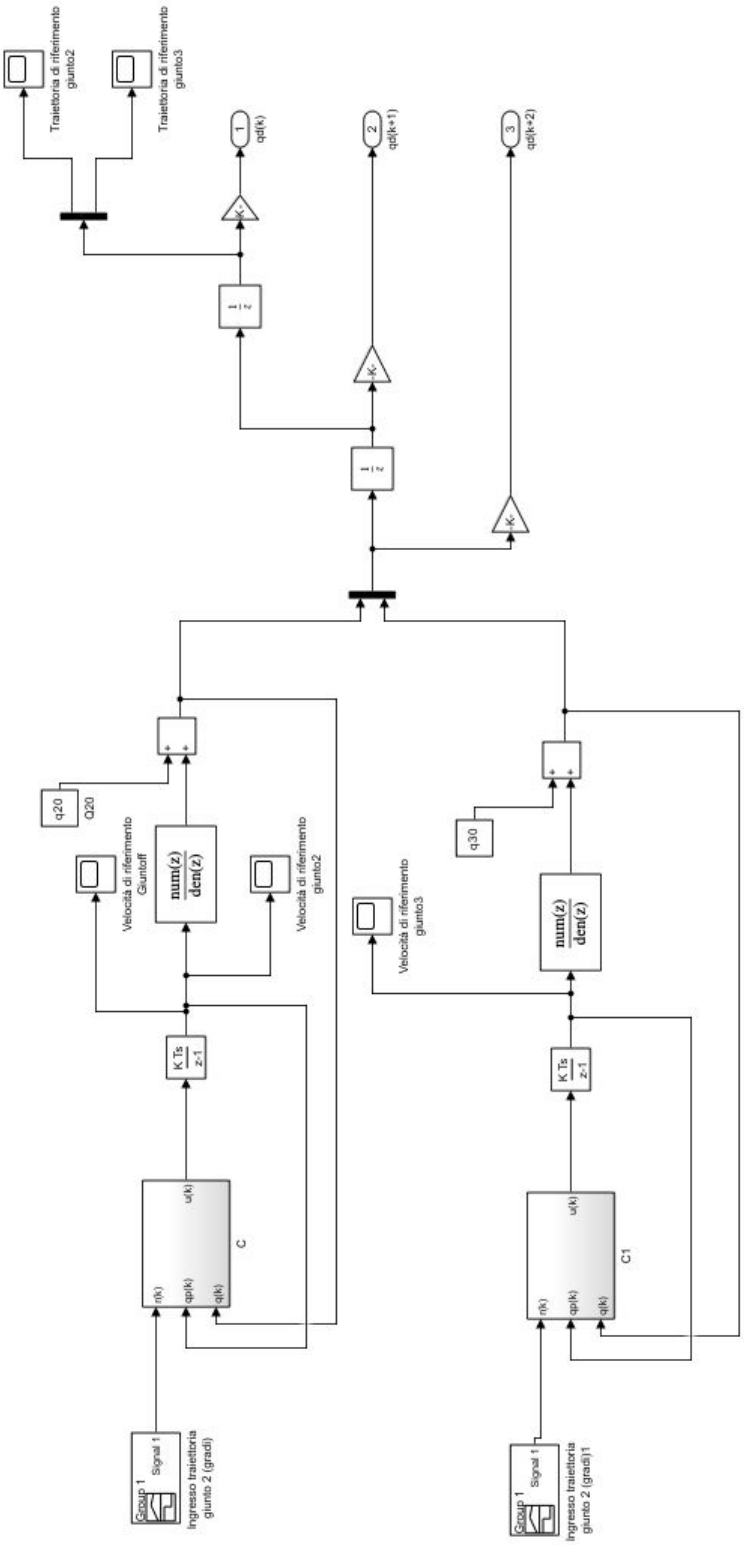


Figure 2.12: Generatore di traiettorie

2.8.3 Utilizzo

All'interno di questo progetto questo sistema verrà utilizzato al fine di interpolare le posizioni che devono assumere i giunti di volta in volta, come il passaggio dalla posizione iniziale a quella di Home, o a seguire il passaggio dalla posizione di home a quella ricavata mediante cinematica inversa per portare il manipolatore al di sopra dell'oggetto rilevato. Analizzando nel tempo il profilo delle posizioni non filtrato si otterrà qualcosa di simile a quello in 2.9, per questo si ha la necessità di filtrare le traiettorie.

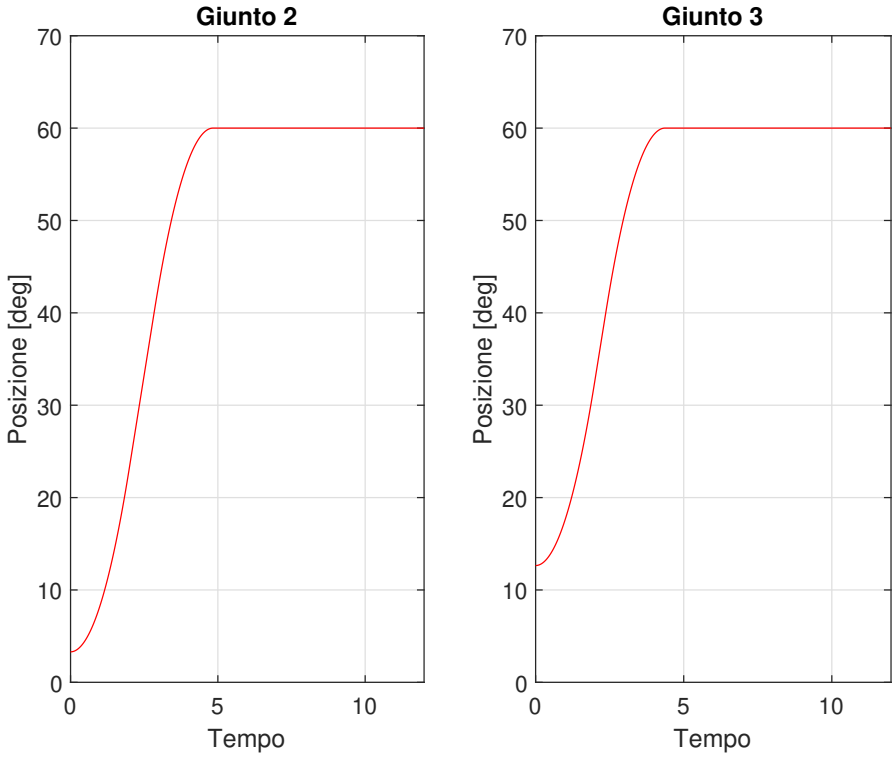


Figure 2.13: Generiche traiettorie di riferimento

2.9 Parametri del manipolatore utilizzati in simulazione

Nel codice seguente si hanno i parametri che vengono utilizzati nelle esecuzioni; i parametri sono relativi al VSC, alle caratteristiche fisiche del manipolatore e a vincoli sulle traiettorie:

```
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];

%% in base a dove deve andare corregge i parametri

%fa la lettura della posizione iniziale
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)

getq20deg = (get_q20*180)/pi;
getq30deg = (get_q30*180)/pi;

go_pos_2 = 65;
go_pos_3 = 65;


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

lam = [lam1 0;0 lam2];
Kradtodeg=180/pi;

disp('Posizione di HOME')
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)

getq20deg = (get_q20*180)/pi;
getq30deg = (get_q30*180)/pi;
```

Come è possibile vedere per i parametri relativi al controlo (ϵ, λ e ρ) viene fatta una distinzione mediante un if in base a se il manipolatore compie un movimento verso l'alto o verso il basso in quanto durante le prove in laboratorio il manipolatore ha dimostrato di risentire anche della direzione di movimento. In un capitolo successivo verrà introdotta la tecnica di tuning dei parametri del VSC.

2.10 Matlab r2023b

Matrix laboratory è il nome completo del programma rilasciato dalla MathWorks e che si è largamente diffuso in molti campi della tecnologia e delle scienze. Matlab è un linguaggio interpretato ed un potente strumento di calcolo numerico, la logica che implementa è basata sulle matrici, come ci si può aspettare dal nome, consentendo il calcolo di espressioni molto complicate in modo rapido. Matlab offre un'ampia gamma di strumenti ed è in grado di costruire interfacce utente, visualizzare funzioni e dati, utilizzare funzioni specifiche o programmarne di nuove. Inoltre sono stati sviluppati software che consentono l'interfacciamento di Matlab con altri software. Le funzioni predisposte nel programma coprono una vastissima gamma di campi, dall'elaborazione del segnale alla progettazione di un controllore, dalla realizzazione grafica di modelli alla simulazione di questi ultimi e comprende anche funzioni utili nel campo della chimica, delle telecomunicazioni e dell'ingegneria genetica. Queste funzioni sono spesso associate a dei veri e propri strumenti di progettazione chiamati tools, che rendono più comodo e agevole il lavoro. I tools sono M-file, estensione dei file di Matlab, che consentono l'espansione del programma base; nell'ambito della controllistica sono molto utilizzati il SISOtool, utile per la progettazione del controllore di un sistema, l'ltiview, che consente la visualizzazione dei vari tipi di grafici utilizzati dai controllisti per il dimensionamento, il Simulink, importante strumento per la simulazione del comportamento di un sistema e per la modellizzazione di processi fisici di cui si conosce il modello matematico.

2.11 Simulink

Questo tool è importante nella fase di progettazione di un controllore, in quanto è possibile modellizzare il processo fisico da controllare, simularne l'andamento e la risposta e progettare un controllore adatto. Al termine della fase di progettazione è possibile simulare il comportamento dell'intero sistema in controreazione e verificare il soddisfacimento delle specifiche di progetto. Il Simulink è un linguaggio di programmazione ad alto livello, con una notevole libreria di oggetti in grado di implementare virtualmente sistemi lineari e non lineari, modellandoli con sorgenti, funzioni di trasferimento e operatori di somma e moltiplicazione. Il sistema analizzato può essere ricreato usando la rappresentazione a blocchi, tipica nell'ambito controllistico. Inoltre Simulink consente la scomposizione di un sistema in sottomodelli, in modo da renderne più semplice la programmazione. La rappresentazione con blocchetti interconnessi in modo adeguato include l'utilizzo di blocchi appositi con i quali è possibile richiamare funzioni Matlab, in questo modo si possono sostituire complicate modellizzazioni con codice matlab equivalente. Un importante esempio nella modellizzazione tramite codice è l'S-Function, una funzione predefinita di Matlab con uno scheletro standard nel quale si devono inserire i dati del sistema modellato, quali ingressi, uscite, equazioni differenziali, stati e stati iniziali. Il modello del

manipolatore ERICC è non lineare ed è estremamente complesso ricostruirlo con i vari blocchetti Simulink, in questo caso si può ricorrere all'utilizzo dell'S-function che rende più semplice la costruzione di tale modello.

2.12 Real-Time Workshop

Questo modulo estende le funzionalità del Simulink aggiungendo funzioni supplementari, ad esempio la generazione di S-Function, senza doverla importare dall'esterno, oppure la possibilità di inserire Embedded function, funzioni implementate direttamente nel sistema, sono meno efficienti e meno complete, ma possono essere trasportate in una dll che invece non supporta gli altri tipi di funzione. Inoltre questo modulo consente di passare da un ambiente simulato alla realizzazione di un controllo tramite Hardware in the loop, che utilizza il PC di progettazione come dispositivo di controllo.

2.13 Embedded function

Le embedded functions mettono a disposizione dell'utente di Simulink un potente mezzo per ampliare notevolmente il sistema. In effetti si potrebbe dire che ogni singolo blocco di Simulink in effetti altro non é che una Embedded Function. Le Embedded Function utilizzano una sintassi particolare che permette loro di integrarsi perfettamente nei metodi di simulazione di Simulink. Una Embedded Function non è altro che un blocco con un certo numero di entrate (anche nulle) e un certo numero di uscite (anche nulle). Al suo interno possono trovarsi degli stati continui o discreti che possono poi essere risolti e integrati dal Solver di Simulink. Durante la simulazione Simulink chiama ripetutamente le procedure all'interno della Embedded Function secondo le necessità:

1. inizializzazione, chiamata all'inizio della simulazione:
   - inizializzazione della struttura di simulazione (entrate, uscite, stati ecc.);
   - inizializzazione dei tempi (campionamento ecc.);
   - inizializzazione delle matrici e delle variabili temporanee;
   - scegliere il range entro cui il segnale può variare;
2. cliccare sul pulsante play per l'avvio dell'ottimizzazione;
3. calcolo del prossimo tempo di elaborazione, per i sistemi a passo variabile di integrazione;
4. calcolo delle uscite, al "major time step";
5. update degli stati discreti, al "major time step"; integrazione, nel caso di sistemi con stati continui;

Il sistema lavora sempre in questo modo, indipendentemente dal fatto che le Embedded Function siano state implementate come M-Files o come C-MEX. Nella realizzazione tramite C-MEX tutte queste attività vengono realizzate tramite funzioni distinte, mentre nel caso di M-Files, la differenziazione viene fatta tramite tags.

Fase di simulazione	C-MEX Function	M-File S-Function
Inizializzazione	mdlInitializeSize	flag = 0
Prossimo passo	mdlGetTimeOfNextVarHit	flag = 4
Output	mdlOutputt	flag = 3
Update	mdlUpdate	flag = 2
derivate	mdlDerivate	flag = 1
finesimulazione	mdlTerminate	flag = 9

Table 2.4: Fasi di simulazione

Una Embedded Function di tipo M-File viene creata utilizzando unicamente funzioni da Matlab, ottenendo n file di tipo ASCII con tutte le funzioni necessarie. La funzione ha sempre la chiamata seguente:

$$function[sys, x0, str, ts] = ERICC(t, x, u, flag, par1, par2, ...)$$

dove par1, par2, ... sono parametri opzionali che si possono passare alla funzione. Nella funzione principale viene fatta la distinzione tramite il parametro flag sulle attività da svolgere, tramite un controllo di tipo switch. I parametri in entrata sono:

- "t" è l'attuale tempo di simulazione;
- "x" sono i valori attuali degli stati;
- "u" è il valore dell'entrata;
- "flag" determina l'attività attuale da svolgere nella funzione.

Questi 4 parametri sono passati in entrata da parte di simulink ogni volta che la funzione deve essere effettuata. I valori in uscita dipendono dalla funzione svolta; in particolare nel nostro modello Simulink, la Embedded Function è stata utilizzata per simulare il comportamento del Manipolatore ERICC in corrispondenza a determinati segnali di controllo. L'Embedded Function relativa al robot prevede l'assegnazione delle variabili di stato iniziale del manipolatore dall'esterno; quindi oltre ai parametri t,u,x(),flag dovremo inviare anche le variabili 'q_{20}, q_{30}, $\dot{q_{20}}$, $\dot{q_{30}}$'. La chiamata della funzione 'main' dell'Embedded Function diventa quindi:

$$function[sys, x0, str, ts] = ERICC(t, x, u, flag, q_{20}, q_{30}, \dot{q_{20}}, \dot{q_{30}})$$

. Tali variabili dovranno poi essere inviate anche alla funzione di inizializzazione dell'S-Function che verrà eseguita come prima procedura in corrispondenza al valore di flag = 0. La chiamata della funzione di inizializzazione è la seguente:

$$[sys, x0, str, ts] = mdlInitializeSizes(q_{20}, q_{30}, \dot{q_{20}}, \dot{q_{30}})$$

Se il flag=1, invece, viene richiamata la sottofunzione 'mdlDerivatives()', nella quale viene definito l'intero sistema di equazioni differenziali che costituiscono il modello non lineare del sistema. A tale scopo, è necessario inviare a tale funzione tutti i parametri da utilizzare nelle equazioni; le variabili passate dall'esterno e le tre variabili standard 't,x,u'. Nella nostra funzione, i parametri vengono impostati come costanti direttamente all'interno della funzione, per evitare massicci invii di costanti dall'esterno che aumenterebbero la probabilità di errore. Le equazioni implementate in questa funzione dell'Embedded Function sono quelle descrivono la dinamica del robot (studiata nei capitoli precedenti). Il modello deve essere espresso in una notazione matriciale per consentire a matlab la corretta elaborazione dei dati. Si definisco, a questo scopo, le matrici B, C, g, Fv ed utilizzando l'algebra delle matrici si risolve il sistema trovando la struttura finale dx, che costituisce l'intero sistema differenziale che modella il robot, quindi la variabile di default sys viene posta uguale a tale struttura. Proseguendo in ordine, si trovano le funzioni di definizione delle uscite (flag=3) e di terminazione del codice (flag=2,4,9). Queste funzioni non presentano particolarità; il loro scopo è semplicemente quello di definire la variabile di default della uscite sys e di terminare l'esecuzione del codice.

2.14 Hardware in the loop

Per testare il controllore VSC, si è realizzata una simulazione detta *hardware in the loop* in tempo reale, ovvero si è connesso una struttura fisica (il manipolatore) con una struttura software (il controllore) implementata in ambiente Simulink. L'interfacciamento tra ambiente Simulink e processo reale avviene mediante i toolbox Real-Time Windows Target e Real-Time Workshop, toolbox che permettono la compilazione di un modello di Simulink *.mdl in un codice sorgente eseguibile in 'external mode' sotto simulink: le maschere dei blocchi Simulink diventano di conseguenza una interfaccia grafica tra utente e processo reale per l'immissione e la modifica di parametri in tempo reale. E' stato inoltre introdotto uno switch manuale di sicurezza che permette, in caso di problematiche di varia natura, di interrompere il controllo da parte della scheda di acquisizione e restituirlo all'unità centrale. Le connessioni tra il modello in Simulink e la scheda di acquisizione dati avviene mediante l'utilizzo dei blocchi messi a disposizione nel Real-Time Windows Target che rappresentano gli ingressi e le uscite digitali e analogiche della PCI-6024E. Questo concetto viene applicato facendo uso del file simulink relativo a Figura 3.5.

Chapter 3

Progettazione

3.1 Collegamento PC-PC via ethernet

Non potendo effettuare direttamente i calcoli sul PC collegato all'unita' di controllo del manipolatore e' necessario effettuare un collegamento ad un PC esterno che consenta di svolgere queste operazioni che richiedono toolbox recenti e una maggiore capacità computazionale.

3.1.1 Collegamento fisico

Per fare cio' si e' deciso di effettuare il collegamento mediante un cavo ehternet come segue:

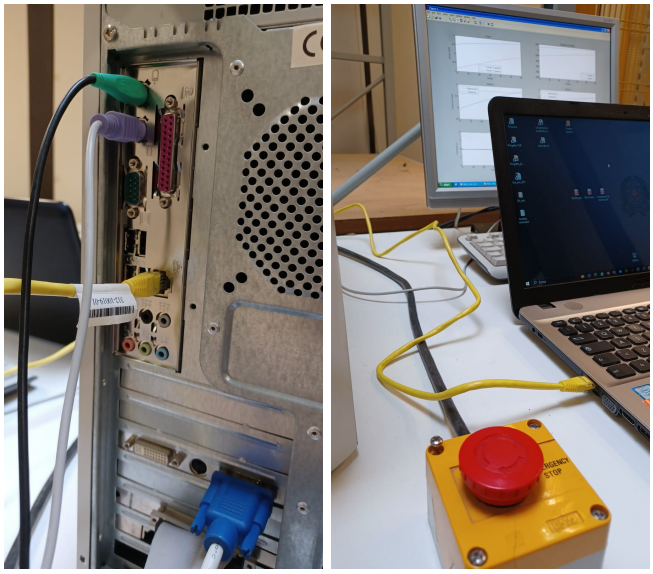


Figure 3.1: Collegamento cavo ethernet

Il cavo deve essere connesso alla porta più alta del PC del laboratorio (collegata alla scheda di rete).

3.1.2 Setting del collegamento ethernet

Si effettua il collegamento su un PC con sistema operativo Windows 10; dalla pagina delle impostazioni, stato, centro e connessioni di rete, impostazioni di condivisione avanzate, per tutti i campi (privato - guest - rete) si abilita la condivisione di file e stampanti disabilitando l'uso della password per l'accesso alla connessione.

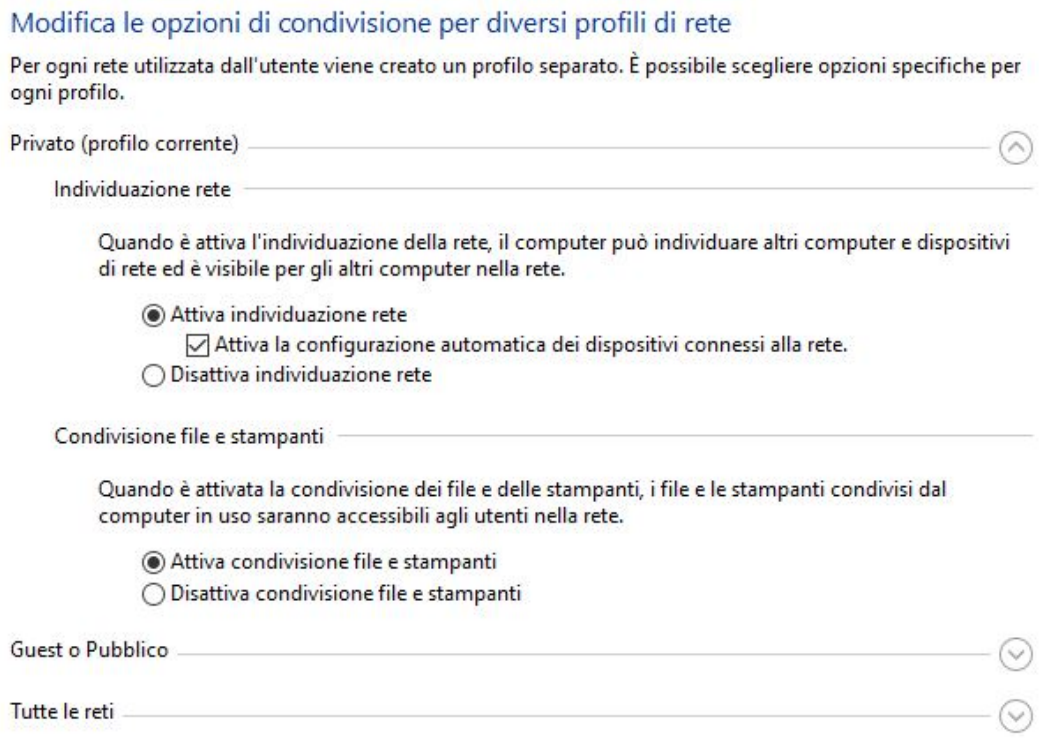


Figure 3.2: Opzioni da modificare per la condivisione

A seguire dalla pagina stato, modifica opzioni scheda, ethernet, proprietà, tcp/ip ipv4, proprietà, si compilano i campi come segue:

Table 3.1: Campi IPv4 computer esterno

IP	192.168.1.1
Subnet mask	255.255.255.0
DNS	8.8.8.8
DNS alternativo	8.8.4.4

3.1.3 Abilitazione supporto per condivisione file SMB1

E' necessario riabilitare il supporto in quanto e' stato disabilitato di recente a seguito di un aggiornamento di windows 10 in quanto al centro di attacchi ransomware.

Seguendo questi semplici passi:

1. pulsante Windows, cerca "pannello di controllo" e aprilo;
2. visualizzazione per categoria;
3. "Programmi" e poi in "Programmi e funzionalità" clicca "Attiva o disattiva funzionalità di Windows";
4. Nell'elenco trovare la voce "Supporto per condivisione file SMB 1.0/CIFS e spuntare alle sole sottovoci "Client SMB 1.0/CIFS" e "Server SMB 1.0/CIFS";

A fine installazione effettuare riavvio. A pc riavviato, in Servizi impostare in automatico l'avvio del servizio "Pubblicazione risorse per individuazione". Successivamente aprire una finestra di Esplora file, Rete e dopo una manciata di secondi verificare la visibilita' della risorsa di rete che si sta cercando.

Impostare sulla rete il nome del WORKGROUP uguale su entrambe le reti (quella del PC con windows 10 e quella del PC collegato al manipolatore quando si effettua la procedura di setting).

Per quanto riguarda il PC collegato al manipolatore, al fine di effettuare il collegamento andare su rete e creare la rete domestica assegnando il nome del workgroup in maniera opportuna (presente procedura guidata). I campi sono da riempire con i seguenti valori:

Table 3.2: Campi IPv4 computer manipolatore

IP	192.168.1.2
Subnet mask	255.255.255.0
DNS	8.8.8.8
DNS alternativo	8.8.4.4

Infine sul PC collegato al manipoatore creare una cartella e attraverso la voce condivisione delle proprietà abilitare la condivisione dei file alla seconda voce della scheda. Al bisogno dal PC esterno si accederà alla cartella indicando il path:

\\192.168.1.2\cartellaCondivisa\file

3.1.4 Collegamento Remote Desktop Protocol

Al fine di permettere il controllo da un unico dispositivo si è realizzato un controllo remoto del PC collegato al manipolatore sfruttando la precedente connessione che oltre a consentire il passaggio dei dati permette anche la gestione dell'ambiente matlab. Per farlo sarà sufficente accedere al PC del manipolatore e abilitare la connessione remota. Per fare ciò si accede come amministratori Risorse del computer, proprietà, Remote, Abilitare desktop remoto, "Consenti agli utenti di connettersi in

remoto a questo computer" e confermare gli utenti abilitati. In questo modo dal PC esterno accedendo a Connessione Desktop Remoto e inserendo le credenziali del pc del manipolatore sarà possibile controllare il computer direttamente dal PC esterno centralizzando il controllo di tutto il sistema.

3.1.5 Gestione dei path all'interno dei programmi MATLAB

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.

3.2 Il controllo

3.2.1 Variable structure control (VSC)

Il VSC (Variable structure control), è una forma di controllo non lineare a struttura variabile che lavora in retroazione sullo stato. Il controllo in sliding mode nasce come metodo di "controllo universale" essendo robusto e versatile. La sintesi di un sistema di controllo che applica direttamente un'azione di tipo discontinuo apre nuovi orizzonti per il controllo di attuatori di tipo ON-OFF che tipicamente sono controllati in PWM. Tra le caratteristiche principali si ha:

- *Robustezza:* le leggi di controllo vengono commutate in base allo stato del sistema, il controllo si adatta automaticamente a variazioni, disturbi e incertezze del sistema;
- *Rapida risposta dinamica:* poiché le leggi di controllo vengono commutate in risposta a cambiamenti nello stato del sistema, il controllo può adattarsi istantaneamente alle nuove condizioni operative, permettendono al sistema di manenere elevate prestazioni reagendo rapidamente a perturbazioni o cambiamenti nelle richieste di controllo;
- *Semplicità di implementazione:* possiamo analizzare la semplicità di implementazione suddividendola in 4 macrocategorie:
  - *Modularità*: ogni modulo può essere progettato per controllare un aspetto specifico del sistema, come il tracciamento della traiettoria, la stabilizzazione del sistema o la reiezione dei disturbi;
  - *Leggi di commutazione*: possono essere implementate utilizzando semplici condizioni logiche o funzioni matematiche, rendendo il processo di commutazione intuitivo;
  - *Flessibilità*: è possibile progettare le leggi di controllo per adattarsi alle caratteristiche del sistema e alle richieste di controllo;
  - *Utilizzo di componenti standard:* il VSC può essere implementato utilizzando componenti hardware e software standard, quali: microcontrollori, sensori ed attuatori affiancati a MATLAB/Simulink per quanto riguarda il software;
- *Applicabilità a sistemi non lineari:* a seconda della situazione è possibile applicare una legge di controllo differente, permettendo di gestire sistemi non lineari che possono comportarsi in modi diversi a seconda delle diverse regioni dello spazio di stato, commutando tra diverse leggi di controllo.

Nella Figura 3.3 è mostrato il funzionamento del VSC.

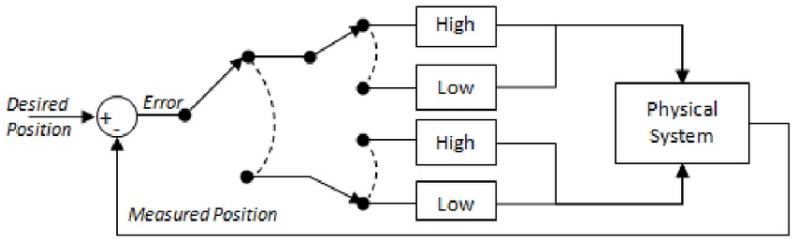


Figure 3.3: Variable structure control

3.2.2 Sliding Mode Control (SMC)

L'SMC (Sliding Mode Control), o controllo di scorrimento, è una forma specifica di VSC. L'obiettivo di questa tecnica è far convergere il sistema verso una superficie di scorrimento predefinita nello spazio di stato mantenendolo a regime (Figura 3.4). L'idea alla base di questa tecnica di controllo è portare il sistema alla superficie di sliding, che rappresenta il riferimento del sistema di controllo. Per raggiungere la zona di lavoro il sistema viene forzato attraverso segnali discontinui ad alta frequenza. Nella fase transitoria le traiettorie oscillano intorno alla superficie stessa, definito come "chattering", e l'ampiezza delle oscillazioni è tanto più piccola quanto maggiore è la frequenza del segnale di controllo. L'aspetto chiave su cui si basa l'SMC è la superficie di scorrimento che deve garantire il raggiungimento ed il mantenimento della stabilità del sistema.

La superficie di scorrimento è definita come una superficie nello spazio di stato che rappresenta una condizione desiderata o una traiettoria di controllo per il sistema; il sistema è progettato per convergere e rimanere sulla superficie di scorrimento per garantire le prestazioni desiderate del sistema. La legge di controllo deve generare ingressi di controllo che spingono il sistema verso la superficie di scorrimento e lo mantengono su di essa. Anche la superficie di scorrimento deve essere scelta in modo tale che sia raggiungibile dal sistema e ne garantisca la stabilità a regime; il sistema è progettato per rimanere sulla superficie di scorrimento, è quindi in grado di mantenere le prestazioni desiderate anche in presenza di variazioni del sistema e disturbi esterni. La progettazione dipende dalle specifiche esigenze di controllo del sistema. Può essere definito in base agli obiettivi di controllo, come il raggiungimento di una determinata traiettoria o il mantenimento di una certa regione nello spazio di stato.

Per quanto riguarda la configurazione e la scelta dei parametri, quali ad esempio il guadagno di controllo ed i termini di commutazione, possono essere regolati per ottenere una convergenza ottimale del sistema. Questo processo di tuning dei parametri può essere guidato da tecniche di ottimizzazione, ad esempio un algoritmo di discesa del gradiente, o esperienza pratica. I microcontrollori eseguono il software in modo sequenziale a intervalli discreti di tempo, determinati dalla frequenza di campionamento del sistema. Questo significa che il controllore esegue un'iterazione del codice di controllo a intervalli regolari, prendendo misure dai sensori, calcolando

l'azione di controllo e inviando comandi ai motori o agli attuatori. Per questa ragione, si è optato per l'implementazione a tempo discreto di questa versione di VSC. L'ultimo passo prima di mettere in funzione il sistema è quello di eseguire delle simulazioni. È importante testare la legge di controllo attraverso le simulazioni per valutare le prestazioni ed identificare eventuali problemi o potenziali miglioramenti. Questo processo aiuta a garantire che il sistema converga in modo efficace e stabile durante il funzionamento reale.

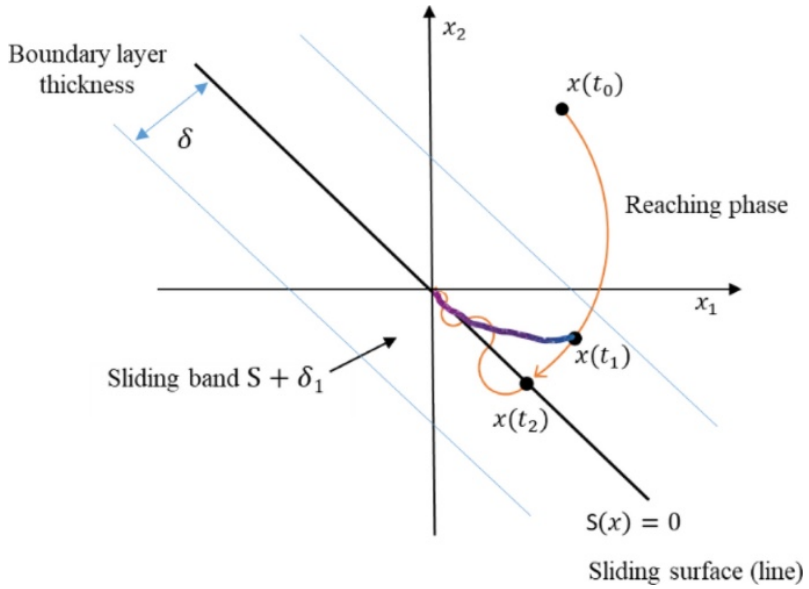


Figure 3.4: Sliding mode control

La sintesi della legge di controllo avviene in due fasi distinte:

- Prima si sceglie la regione di spazio di dimensione (n-m), sulla quale si desidera che avvenga lo sliding mode. Si definisce, quindi la Superficie di Sliding:
  - $s(x) = 0$, con $s(x)$: $\mathcal{R} \to \mathcal{R}$, in cui n è la dimensione dello stato e m la dimensione dell'ingresso. La scelta di tale regione è legata al tipo di specifiche che si vogliono imporre al sistema.

- Successivamente, si scelgono le m componenti del controllo, cioè le funzioni $u(x,t)$ per i = 1...m, in modo che la condizione sia verificata da un certo istante t, in poi.
  La funzione di controllo è del tipo:

$$u_i(x,t) = \begin{cases} u_i^+(x,t) & \text{se} \quad s_i(x) > 0 \\ u_i^-(x,t) & \text{se} \quad s_i(x) < 0 \end{cases} \tag{3.1}$$

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

3.3 Incertezza parametrica

Introducendo il controllo utilizzato è necessario trattare la questione relativa all'incertezza parametrica, in quanto come precedentemente accennato i valori del manipolatore sono stati ottenuti in simulazione. L'incertezza parametrica nella teoria del controllo riguarda la mancanza di conoscenza precisa dei parametri del sistema che si intende controllare. Questo può derivare da diverse cause, quali modellazione approssimativa, dove i modelli matematici utilizzati non riescono a catturare tutte le dinamiche del sistema reale, oppure da variazioni ambientali e operative che influenzano i parametri nel tempo (es. la variazione di resistenza di un resistore nel tempo o con la temperatura). Queste incertezze rendono difficile prevedere esattamente il comportamento del sistema e progettare un controllo ottimale, poiché il controllore deve essere robusto abbastanza da gestire queste variazioni e mantenere il sistema stabile e performante nonostante le incertezze.

3.4 Costruzione simulink del VSC

Nella Figura 3.5 è possibile notare l'implementazione del sistema di controllo che sfrutta il VSC.

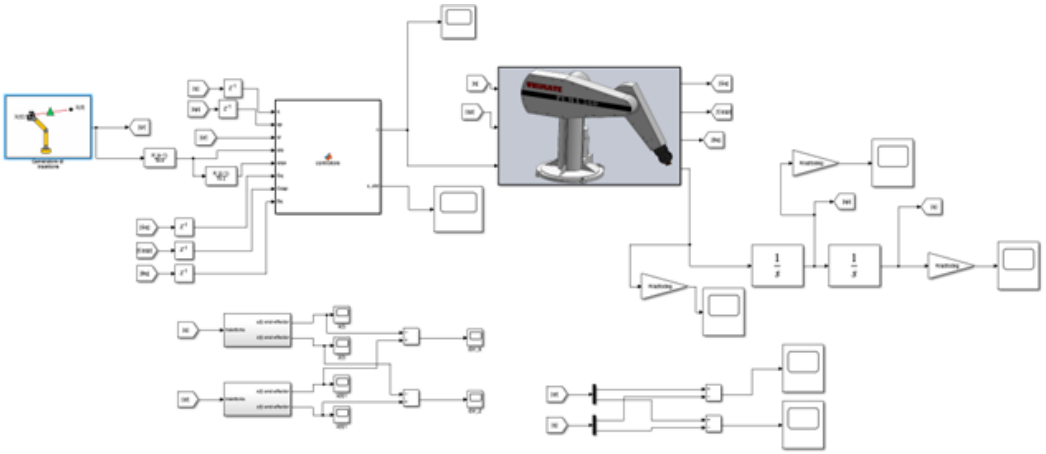


Figure 3.5: Schema a blocchi in simulazione del secondo VSC

Il tutto si articola in quattro sezioni principali:

- *Generazione delle traiettorie*: blocco che va a fornire i valori di riferimento delle posizioni angolari per i due giunti;
- *Sezione di controllo*: il VSC realizzato a tempo discreto, in questa versione implementato direttamente come un'unica Matlab Function che restituisce in output l'andamento della superfice di sliding e lo sforzo di controllo applicato al manipolatore nel tempo;

- *Modello matematico del robot*: anche qui questo blocco ci permette di simulare il comportamento del manipolatore, tenendo conto della dinamica propria di quest'ultimo, realizzato mediante una Matlab Function;
- *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 funzioni delle posizioni assunte dai giunti).

In questo modello la retroazione non è realizzata mediante i classici collegamenti, ma sfruttando i blocchi GoTo e From, ciò non comporta alcun tipo di miglioramento in termini di efficienza, ma conferisce sicuramente una migliore leggibilità al tutto.

Ulteriore nota per il fatto che vengono passati al controllore anche i profili di velocita e accelerazione desiderati, opportunamente estrapolati dai riferimenti di posizione mediante un derivatore (discreto). Descrivendo più nel dettaglio il funzionamento del VSC possiamo osservare le seguenti fasi:

1. Calcolo dell'errore in posizione ed in velocità, per poi calcolare il valore attuale della superfice di sliding:

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

2. Calcolo del vettore delle coppie applicati attualmente ai giunti:

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

3. Determinazione degli sforzi di controllo per i due giunti in funzione del valore della superficie di sliding, e calcolo della coppia definitiva:

```
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];
```

Osservando cosa succede nella parte aggiunta riguardo la cinematica diretta (Figura 3.6) si nota che questo blocco prende in input la traiettoria percorsa dai giunti e restituisce in output i valori in coordinate cartesiane che l'end-effector percorre nel tempo. Per far ciò viene utilizzata una Matlab Function che sfrutta la componente di traslazione nella matrice di trasformazione del robot (Figura **??**).

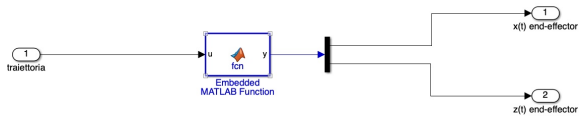


Figure 3.6: Cinematica diretta

Questo blocco, dunque, prende in input la traiettoria percorsa dai giunti e restituisce in output i valori in coordinate cartesiane che l'end-effector percorre nel tempo. Per far ciò viene utilizzata una Matlab Function che sfrutta la componente di traslazione nella matrice di trasformazione del robot.

```
function y = fcn(u)
sys = [0;0];
a2 = 0.28;
a3 = 0.3175;

sys(1,1) = a2*cos(u(1)) + a3*cos(u(1)+u(2)-90*pi/180);
sys(2,1) = a2*sin(u(1)) + a3*sin(u(1)+u(2)-90*pi/180) + 0.34;
y = sys;
```

Dove a2 ed a3 sono rispettivamente le lunghezze del secondo e terzo link. Come si può notare dall'immagine iniziale questo procedimento viene effettuato anche dando in ingresso la traiettoria di riferimento delle posizioni di giunto, così facendo si può determinare l'errore commesso nello spostamento sia sull'asse X che sull'asse Z, ossia i due assi che determinano il piano nel quale si muove l'end-effector.

3.4.1 Calibrazione dei parametri per l'ottimizzazione del controllo

Al fine di ottimizzare i parametri si è scelto un approccio pratico, valutando di volta in volta la la correttezza dei parametri osservando:

- **andamento del manipolatore**: vibrazioni, movimento continuo in una direzione fuorviante dalla traiettoria di riferimento in direzioni esterne a questa;
- **grafici**;

Importante sottolineare l'importanza dei grafici in quanto descrivono in maniera completa l'andamento del manipolatore e del controllo durante tutto il tempo di esecuzione. In particolare si è scelto di evidenziare al fine di valutare il controllo:

- errore sul movimento dei giunti in gradi;
- errore sul movimento nello spazio cartesiano;
- confronto tra traiettorie di riferimento e posizione effettiva dei giunti;
- confronto tra traiettoria di riferimento e posizione effettiva del manipolatore nello spazio carteisano;
- movimento effettivo sul piano XZ del manipolatore;
- tensione fornita ai motori;
- coppia dei motori;

Attraverso questi grafici si possono ricavare informazioni che permettono di effettuare considerazioni riguardo la necessità di correggere i parametri e al massimo rivalutare il modello matematico del manipolatore al fine di regolare i parametri ρ, λ e ϵ. In particolare sono stati scelti valori di λ intorno a 200 in maniera predefinita, per poi effettuare un tuning preciso riguardo i valori di ϵ e ρ che servono a garantire un corretto inseguimento della traiettoria. Per fare ciò si parte da valori elevati di ρ e ristretti di ϵ, che possono causare vibrazioni dei giunti del manipolatore, e si prosegue diminuendo continuamente i valori di ρ e aumentando quelli di ϵ gradualmente in maniera alternativa, fino a raggiungere delle prestazioni ritenute accettabili. Basandosi sui grafici si osservano i grafici relativi alla coppia, e alla traiettoria confrontata con il riferimento: nel caso di un movimento caratterizzato da intense oscillazioni attorno al riferimento costante, si possono osservare queste oscillazioni anche all'intenro dei grafici delle coppie, mentre nei grafici relativi alla traiettoria si hanno delle oscillazioni della posizione del giunto intorno al riferimento. Nel caso di un corretto movimento del manipolatore, attraverso il grafico dell'errore si dovrebbe osservare la presenza di un errore iniziale che tende ad annullarsi durante l'esecuzione del movimento, i grafici relativi al confronto tra la traiettoria di riferimento ed il movimento effettivo dovrebbero mostrare delle linee che tendono a sovrapporsi e

i grafici relativi alle coppie dovrebbero essere caratterizzati da uno sforzo iniziale per consentire il movimento del manipolatore e successivamente delle oscillazioni minime per consentire il mantenimento della posizione raggiunta rappresentata dal riferimento. Inoltre in tutti i casi è importante osservare i grafici relativi alle tensioni in quanto nel caso il manipolatore dovesse effettuare sforzi particolari nei suoi movimenti, al fine di garantire il movimento si avrebbero dei picchi di tensione che, per quanto limitati dal modello in simulink, possono risultare pericolosi per i motori.

Questo processo può dimostrarsi lungo e complicato in quanto i parametri del VSC che con il modello in simulazione si dimostrano ottimi in realtà possono non tenere conto dell'incertezza parametrica come introdotto precedentemente). Questo concetto sarà trattato in maniera più approfondita nella sezione relativa alle prove reali con il manipolatore dove saranno elencati diversi fattori che possono influire sulla qualità del controllo; se ne introducono aluni quali temperatura, direzione di movimento, posizione finale e tipo di riferimento (traiettoria o punto fisso da raggiungere), ecc. .

3.5 La visione

3.5.1 La computer vision

La computer vision è una disciplina dell'ingegneria che si occupa di come i computer possano acquisire, elaborare e comprendere immagini digitali. L'obiettivo principale della computer vision è sviluppare metodi che permettano ai computer di interpretare e comprendere il contenuto visivo in modo simile a come lo farebbe un essere umano.

L'acquisizione di immagini è il primo passo e comporta l'uso di dispositivi di imaging per catturare rappresentazioni digitali del mondo reale. Queste immagini possono essere statiche, come le fotografie. Successivamente, le immagini acquisite vengono pre-processate per migliorare la qualità e facilitare l'analisi successiva. Questo può includere la correzione delle distorsioni, il filtraggio del rumore, e la regolazione del contrasto e della luminosità.

L'estrazione delle caratteristiche è un passo cruciale che coinvolge l'identificazione di elementi distintivi all'interno delle immagini. Questi elementi possono essere semplici, come punti e linee, o più complessi, come pattern e forme. La segmentazione dell'immagine consiste nel dividere un'immagine in regioni significative che possono rappresentare oggetti o parti di oggetti. Questo processo è fondamentale per isolare gli elementi di interesse nell'immagine per ulteriori analisi.

La classificazione e il riconoscimento sono le fasi finali della pipeline di computer vision. Questi processi utilizzano algoritmi di apprendimento automatico per identificare e classificare gli oggetti presenti nelle immagini. Ciò può comportare il riconoscimento di volti, testi, gesti, movimenti, e molto altro.

3.5.2 Uso della usb-cam, impostazione e posizionamento

Per il posizionamento della camera usb si è scelto di ricercare la posizione più in alto possibile al fine di ottenere una vista del piano di lavoro completa e che nel momento in cui si volesse ricavare la posizione di un oggetto sul piano risentisse il meno possibile dell'inclinazione della camera.

In questo modo all'interno della command window di matlab, attraverso i comandi:

- **webcamlist**, si ottiene una lista di tutte le cam usb, tra queste compare la webcam del pc (in caso di portatile) e la usb cam collegata al pc;
- **cam = webcam(i)**, si seleziona la i-esima cam mostrata dalla webcamlist;
- **preview(cam)**, si apre una preview live della usb cam selezionata, utile per il suo posizionamento;

In particolare la cam andrà posizionata in modo che il bordo alto e basso che viene inquadrato dalla cam coincidano con i nastri orizzontali che sono stati posizionati sul tavolo (che hanno lo scopo di delimitare l'area in cui possono essere posizionati gli oggetti che devono essere rilevati).

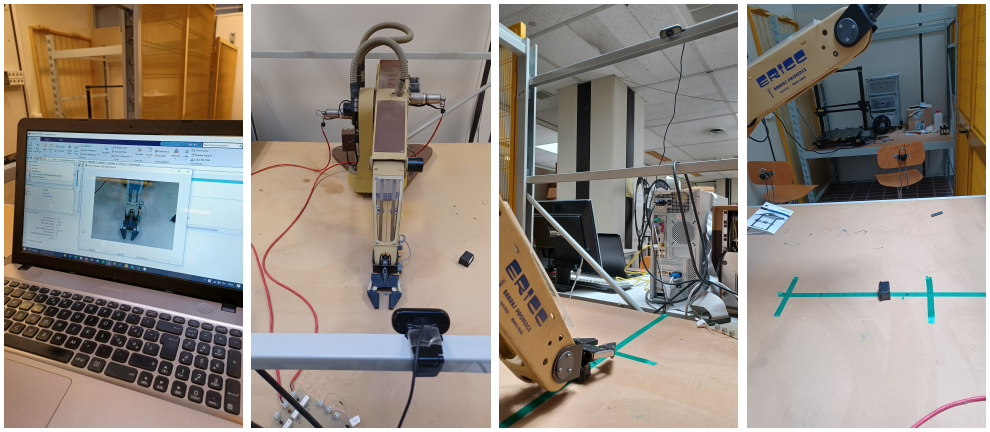


Figure 3.7: Posizionamento della camera usb nella cella di lavoro del manipolatore

3.5.3 Detector utilizzato

L'ACF (Aggregate Channel Features) è un metodo di rilevamento degli oggetti utilizzato principalmente per la rilevazione di pedoni, ma è applicabile anche ad altri tipi di oggetti. Il metodo ACF è stato sviluppato per essere rapido ed efficiente, combinando diverse caratteristiche visive per migliorare la precisione del rilevamento. L'ACF utilizza più canali di immagine per catturare diverse caratteristiche dell'immagine. Questi canali includono l'intensità (in scala di grigi), gradienti orizzontali e verticali, e canali di colore (RGB o altri spazi colore). I canali vengono aggregati e ridotti di dimensione tramite tecniche di pooling, che riducono la risoluzione dell'immagine preservando le informazioni essenziali. Dal set di canali ridotti, vengono estratte caratteristiche visive utilizzando filtri specifici. Queste caratteristiche vengono utilizzate per costruire un vettore di caratteristiche per ogni finestra di immagine. Una finestra mobile scorre sull'immagine a diverse scale. Per ogni posizione e scala della finestra, vengono estratte le caratteristiche e passate attraverso un classificatore per determinare la presenza o meno di un oggetto. L'ACF utilizza solitamente un classificatore basato su un albero decisionale o su una foresta casuale (random forest). Questo classificatore viene addestrato su un set di dati di immagini con annotazioni degli oggetti di interesse. Dopo il rilevamento degli oggetti, viene applicata la soppressione non massima (Non-Maximum Suppression) per rimuovere le rilevazioni duplicate e mantenere solo le più accurate. Si hanno vantaggi di efficienza, in quanto l'ACF è progettato per essere veloce, rendendolo adatto per applicazioni in tempo reale, preciso in quanto utilizzando più canali e tecniche di aggregazione questo riesce a migliorare la precisione del rilevamento rispetto ai metodi basati su singoli canali; inoltre vanta semplicità di implementazione in quanto rispetto ai modelli di deep learning, l'ACF è relativamente semplice da implementare e richiede meno risorse computazionali. Risulta limitato però se utilizzato con oggetti complessi per cui si possono avere forme variabili o oggetti inseriti in scenari complessi e dipendenza da

caratteristiche predefinite per cui potrebbe non adattarsi bene a tutte le situazioni senza un addestramento specifico.

3.5.4 Creazione del dataset

Al fine di creare il dataset per l'addestramento del'ACF sono state scattate un minimo di 150 foto positive (ovvero contenenti l'oggetto che si vuole rilevare). Secondo il funzionamento dell'ACF per il suo addestramento non è necessario fare uso di foto negative (ovvero che non contengono l'oggetto da rilevare). In questo modo si procede scattando le foto con la cam usb posizionata:

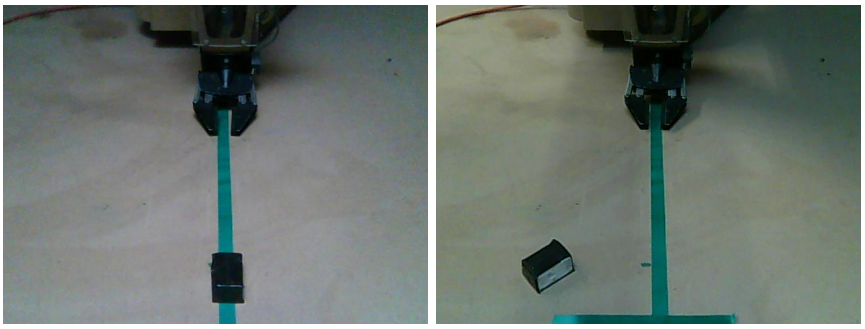


Figure 3.8: Generiche immagini scattate per la creazione del dataset

Quando si genera il dataset è importante cambiare la posizone dell'oggetto in particolare cambiando la sua disposizione nel piano di lavoro (nonostante in questo caso il suo uso prevede il posizionamento sull'asse di movimento del manipolatore) e la sua angolazione rispetto alla cam. Inoltre è utile effettuare foto variando la luminosità del piano inquadrato (semplicemente accendendo e spegnendo le luci).

3.5.5 Labeling

Questa fase consiste nell'etichettare le immagini con informazioni che il modello utilizzerà per imparare a riconoscere e classificare gli oggetti di interesse. Questa procedura avviene inquadrando con un rettangolo (detto bounding-box), l'oggetto di interesse e assegnando a questo un nome che ne definirà l'etichetta.

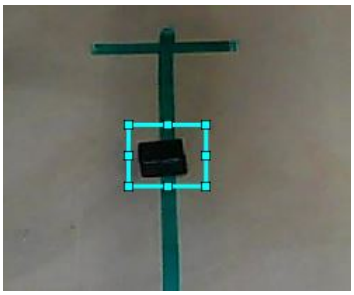


Figure 3.9: Generica immagine etichettata

3.5.6 Training dell'ACF

Scelto il tipo di detector e creato il dataset si procede con la fase di addestramento (o training).

Generalmente, si utilizza un classificatore basato su un albero decisionale o su una foresta di alberi decisionali, come gli algoritmi di boosting. Durante l'addestramento, il classificatore apprende a separare le feature degli oggetti target da quelle dello sfondo. Questo avviene iterativamente, dove il modello viene aggiornato per ridurre gli errori di classificazione e migliorare la precisione. Durante il training, si applicano anche tecniche di validazione per garantire che il modello non stia sovradattandosi al dataset di training. Questo implica l'uso di set di validazione separati per monitorare le prestazioni del modello e fare aggiustamenti necessari. Infine, il modello addestrato viene testato su un set di immagini non visto prima, per valutare la sua capacità di generalizzare a nuovi dati. La misura della performance, viene utilizzata per misurare l'efficacia del detector nel rilevare correttamente gli oggetti "target".

Al fine è stato realizzato un programma matlab in grado di produrre un detector e salvarlo, in modo tale da poter essere poi utilizzato nel programma che permette di effettuare l'esecuzione del task predefinito. Il programma in questione è diviso in più fasi:

1. indicare path delle foto positive non etichettate e loading del ground truth (ovvero la verità di base) contenente le coordinate dei buonding box di ogni immagine;
2. addestramento e salvataggio;
3. validazione;

Nella prima fase si procede indicando il path delle immagini positive e caricando il ground truth; questa è una fase preliminare al traning effettivo. Si combinano le immagini con le coordinate dei bounding box al fine preparare una variabile da usare come dataset di addestramento del detector. Nella seconda fase si procede con il training effettivo usando la funzione *trainACFObjectDetector* che prende come argomenti il dataset di addestramento e il numero di stage di addestramento; è importante valutare bene questo valore, in quanto se inferiore al numero di stage necessari potrebbe portare ad errori e non rilevare oggetti presenti nel piano di lavoro. Al contrario effettuare un numero di stage di training troppo elevato porterebbe ad un sovradattamento del detector al dataset di addestramento, per cui non riuscirebbe a riconoscere immagini troppo diverse da quelle usate originariamente per l'addestramento, e inoltre aumentare il numero di stage per il training comporterebbe anche un certo tempo di addestramento del detector in quanto questa fase è computazionalmente complessa. Considerando il task in questione si ritiene sufficente un numero di stage di training pari a 5, ottenendo un valido compromesso tra la capacità

di rilevare oggetti e il tempo impiegato per l'addestramento. Infine si procede alla validazione, per cui con la usb cam posizionata e opportunemente collegata si scatta una foto al piano di lavoro in presenza di uno o più oggetti; al fine di validare il detector la foto può essere scattata cambiando luci, ombre e posizione dell'oggetto nel piano di lavoro.

Al fine di velocizzare il processo di detecting, l'immagine può essere opportunemente ritagliata, mantenendo incluso l'oggetto, considerando l'area di lavoro individuata dai nastri verdi. Arrivati a questo punto con il comando detect, fornendo come argomenti il detector e l'immagine, si ottiene in output una serie di bounding boxes accompagnati da uno score, che indica il livello di confidenza del rilevamento. Questo dato permette di determinare l'affidabilità del rilevamento, in particolare in questo caso viene sfruttata per selezionare solo gli oggetti con uno score minimo (es. 30) al fine di escludere semplici ombre o nel caso sia presente, l'end-effector del manipolatore. All'atto nella stampa della figura con il bounding box e l'etichetta target può essere utile aggiungere informazioni quali:

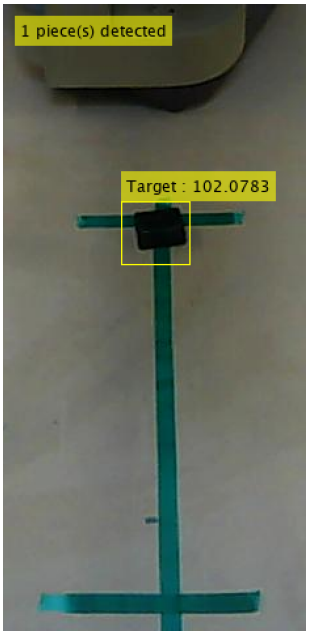


Figure 3.10: Risultato rilevamento di singolo oggetto

- numero di oggetti rilevati;
- score dell'oggetto rilevato;
- coordinate dell'oggetto rilevato;
- posizione dei giunti del manipolatore al fine di raggiungere l'oggetto rilevato;

Non-Maximum Suppression (NMS)

Durante il processo di rilevamento di oggetti, un modello può rilevare lo stesso oggetto più volte con diversi bounding box, quindi al fine di considerare solo un certo numero di oggetti con uno score superiore alla soglia, si utilizza la funzione selecStrongestBox, permettendo di considerare un numero massimo di oggetti (valore che nel caso del task sarà impostato a 1) e una *overlap threshold* = 0.1. Questa procedura data dall'NMS, NMS utilizza l'overlap threshold per eliminare i bounding box ridondanti mantenendo solo i più confidenti.

Overlap threshold

La overlap threshold (soglia di sovrapposizione) si riferisce a un valore di soglia critica utilizzato per determinare se due regioni, solitamente rappresentate da bounding box, si sovrappongono sufficientemente per essere considerate la stessa entità o oggetto.

Questo concetto è cruciale in vari algoritmi e tecniche, come il Non-Maximum Suppression (NMS) e la valutazione delle prestazioni di modelli di object detection.

La misura comunemente utilizzata per quantificare questa sovrapposizione è "l'intersezione su unione" (Intersection over Union, IoU), definita come il rapporto tra l'area di intersezione dei due bounding box e l'area della loro unione:

$$IoU = \frac{AreadiIntersezione}{AreadiUnione} \tag{3.2}$$

L'area di intersezione è individuata dall'area comune tra i due bounding box, mentre l'area di unione è l'area complessiva coperta dai due bounding box.

Applicazione dell'Overlap Threshold

Quando si applica un overlap threshold, si stabilisce un valore limite per la IoU (in genere compreso tra 0.1 e 0.5). Due bounding box vengono considerati come rappresentanti lo stesso oggetto se la loro IoU supera la soglia. In altre parole:

$$\begin{cases} IoU \geq OverlapThreshold & \text{stesso oggetto} \\ IoU < OverlapThreshold & \text{oggetti differenti} \end{cases}$$

3.6 Calcolo delle coordinate dell'oggetto rilevato

Una volta rilevato l'oggetto si necessita di ricavare le posizioni che devono assumere i giunti al fine di permettere al manipolatore di toccare l'oggetto. Per fare questo una volta rilevato l'oggetto si procede ricavando le coordinate del bounding box relative al valore del centro. Considerando che generalmente il bounding box viene definito come:

$$[xBase \quad yBase \quad xEstensione \quad yEstensione]$$

Per ricavare il centro del bounding box si calcolerà come:

$$centro = [\frac{bbox(1) + bbox(3)}{2} \quad \frac{bbox(2) + bbox(4)}{2}] \tag{3.3}$$

Dove per il task prefissato si è interessati solo al valore della y (ricordando che nel caso di un immagine il valore della y indica il numero di pixel a partire dall'alto). Da qui si effettua una proporzione, i cui valori sono indicativi e dipendono dalla precisione con cui è stata posizionata inizialmente la cam usb:

$$coord_y = \frac{centro(2) * 0.435}{yEstensione} + 0.155 \tag{3.4}$$

Dove 0.155m è la distanza misurata tra la base del manipolatore e l'inizio della prima striscia orizzonatale, $yEstensione$ il numero di pixel in verticale della foto ritagliata e 0.435m è la distanza tra le due strisce. Per ottenere la distanza dell'oggetto dal manipolatore agli 0.155m misurati si somma la distanza rapportata al numero di pixel. Così facendo si ottiene la distanza effettiva tra il manipolatore e l'oggetto rilevato, per cui attraverso la cinematica inversa si possono ricavare le posizioni che i giunti devono assumere per permettere all'end-effector di toccare l'oggetto rilevato:

$$\begin{cases} a_2 cos(\theta_1) + a_3 cos(\theta_1 + \theta_2 - 90 * \pi/180) = C_y \\ a_2 sin(\theta_1) + a_3 sin(\theta_1 + \theta_2 - 90 * \pi/180) + a_3 = H_{min} \end{cases} \tag{3.5}$$

Dove a_1, a_2 e a_3 sono le lunghezze dei link del manipolatore; H_{min} indica il valore di altezza dell'oggetto che deve essere rilevato (in questo caso circa 2cm). Effettuando il calcolo tramite la 3.5 si possono riscontrare due casi: ottenere valori di θ_1 e θ_2 reali o complessi; ottenere valori reali significa che esiste una coppia di valori (θ_1, θ_2) per cui il manipolatore riesce a raggiungere la posizione individuata; al contrario una coppia di valori complessi indica che la posizione è al di fuori della portata del manipolatore (in questo caso il manipolatore rimane fermo nella posizione di home).

3.7 Realizzazione in matlab

Per realizzare la parte di visione lo script matlab è stato diviso in più sezioni.

Nella prima parte del programma oltre ai generici comandi di pulizia di workspace e command window, chiusura delle figure precedentemente aperte si assegna la usb cam ad una variabile con vicino dei valori da utilizzare nella funzione imcrop al fine di ritagliare l'immagine catturata dalla usb cam prima di passarla al detector (al fine di veloccizzare la funzione di rilevamento); inoltre si ha una successione di comandi di delete() per l'eliminazione dei valori di coordinate caricati in memoria al pc collegato al manipolatore nella precedente esecuzione, load per caricare nel workspace il detector realizzato nella fase di training, e infine si imposta la variabile signal a 0; questa variabile serve per realizzare una comunicazione tra i due pc.

Nella seconda parte, denominata prima fase, dopo un'attesa di 20 secondi (tempo stimato per avviare il programma sul pc del laboratorio e permettere al manipolatore di raggiungere la posizione di Home al di fuori della portata della usb cam) si entra in un ciclo while nel quale viene scattata una foto del piano di lavoro, viene ritagliata e passata al detector; questo ciclo continua fino a quando il numero di oggetti rilevati che hanno conseguito un certo score è non nullo.

```
while(1)

    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)
        break;
    end
    pause(3)
end
```

A seguito del rilevamento dell'oggetto (e quindi dopo aver ottenuto le coordinate in pixel del bounding box) si fa uso della cinematica inversa per calcolare le coordinate che devono assumere i giunti al fine di raggiungere l'oggetto rilevato:

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

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));
```

Ricavate le coordinate viene mostrata l'immagine con il target rilevato, lo score associato e le coordinate dei giunti.

```
%% stampa immagine ottenuta: img2 immagine con puntino,
imgcounted immagine con riquadro

coordinates = "angle 1 = " + o1 + " angle 2 = " + o2;

img2 = insertShape(img, "circle", [centro(1) centro(2) 2],
"LineWidth", 10, "ShapeColour", green);
img2 = insertText(img2, [10 10], coordinates);
imwrite(img2, "Piece_detected.jpg");

img = insertObjectAnnotation(img,"rectangle", ...
selectedBbox,"Target : " + selectedScore);

imgCounted = insertText(img,[10 10],coordinates);

figure
subplot(1,2,1)
imshow(img2)
subplot(1,2,2)
imshow(imgCounted)
```

Ottenute le coordinate dell'oggetto non rimane che caricarle in memoria al pc collegato al manipolatore, che sfruttando il simulink con il VSC, andrà ad interpolare la posizione attuale del manipolatore con quella che deve raggiungere definendo un riferimento che i giunti devono seguire:

```
%% trasmette le posizioni che ha ricavato

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; %resetta il segnale
save('\\192.168.1.2\found_position.mat', "go_pos", "signal");
```

Nel programma utilizzato è presente un'ulteriore sezione che non viene riportata in quanto non presenta niente di utile all'esecuzione se non una serie di plot di grafici relativi a dati raccolti dall'esecuzione.

Dalla parte del pc collegato al manipolatore si caricano i parametri e si effettua la simulazione per poi salvare i dati ottenuti dalla simulazione in memoria in modo che il pc esterno li possa leggere e usarli per stampare i grafici utili alla valutazione della bontà dell'esecuzione:

```
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)

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

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 = 0.25; %originale 0.3
```

```
    eps2 = 0.6; %originale 0.3
    ro1 = 15; %originale 5
    ro2 = 7;%originale 4.5
    lam1 = 500; %originale 200
    lam2 = 200; %originale 200
end

lam = [lam1 0;0 lam2];
Kradtodeg=180/pi;

pause(3)

%% Simulazione
disp('INIZIO SIMULAZIONE')
modello = 'ERICC_MP_traiettoria';
% modello = 'ERICC_MP_rif_cost';

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

pause(15);
%impostare tempo simulazione da qui e di conseguenza
fa la pausa t+5secondi

%% passo tutti i dati relativi all'esecuzione

save('C:\Documents and Settings\lumafra\Desktop\PD\datiEsecuzione.mat');
```

Chapter 4

Simulazione e prove reali

4.1 Introduzione al capitolo

La verifica delle decisioni effettuate nella fase progettuale è stata effettuata in 2 fasi, una di simulazione per la verifica della funzionalità del controllo e una successiva di prove reali includendo il sistema di visione artificiale.

4.2 Simulazione del VSC

Nei grafici riportati in seguito sono rappresentati gli andamenti delle posizioni e delle superifici di sliding dei giunti, l'errore della traiettoria dell'end-effector sull'asse X e Z e infine l'andamento delle coppie dei due giunti. La simulazione è della durata di 100s. L'andamento delle superfici di sliding è pressochè ottimale in quanto dell'ordine del 10^{-4}; il confronto delle traiettorie con i rispettivi riferimenti restituisce linee che si sovrappongono, suggerendo quindi che l'errore in gradi dei giunti è tendenzialmente nullo; infine il grafico relativo allo sforzo di controllo è in linea con le aspettative, per cui si effettua azione di controllo (in coppia) in corrispondenza dei punti in cui si ha variazione di riferimento; il giunto di spalla subendo anche il carico del giunto di gomito nello sforzo di controllo presenta dei valori maggiori. Non avendo a disposizione un modello che simula il movimento in 3D non si è ritenuto necessario la stampa di grafici riguardo la posizione spaziale, movimenti lungo gli assi e tensioni dei motori dei singoli giunti.

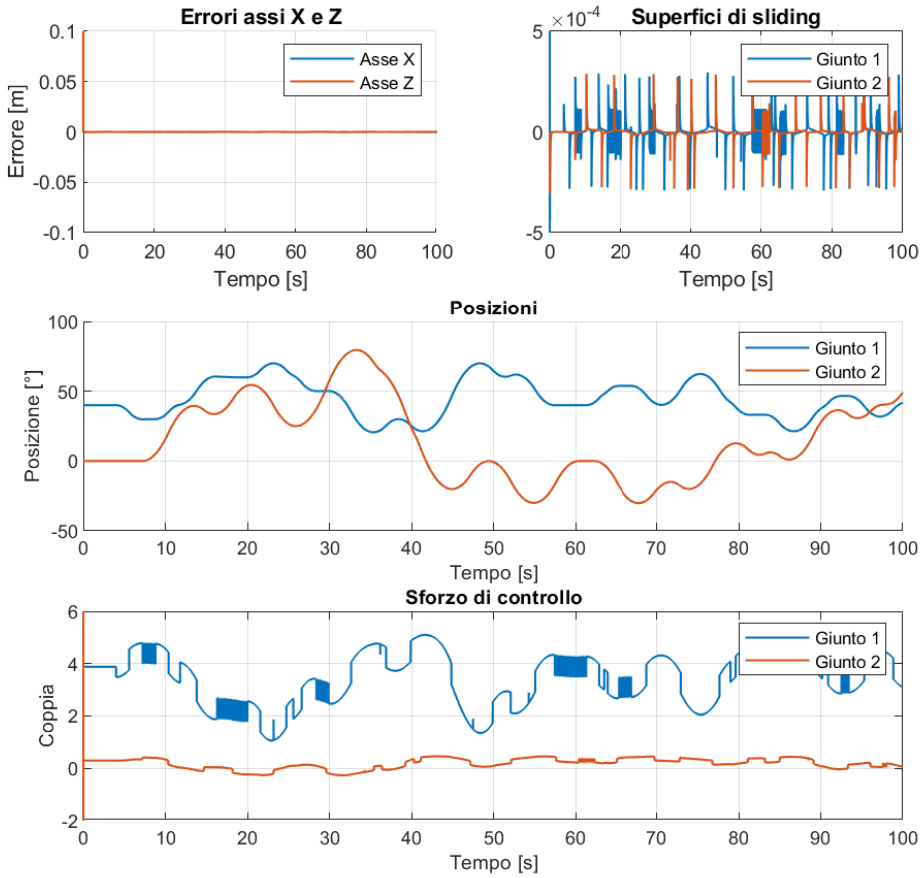
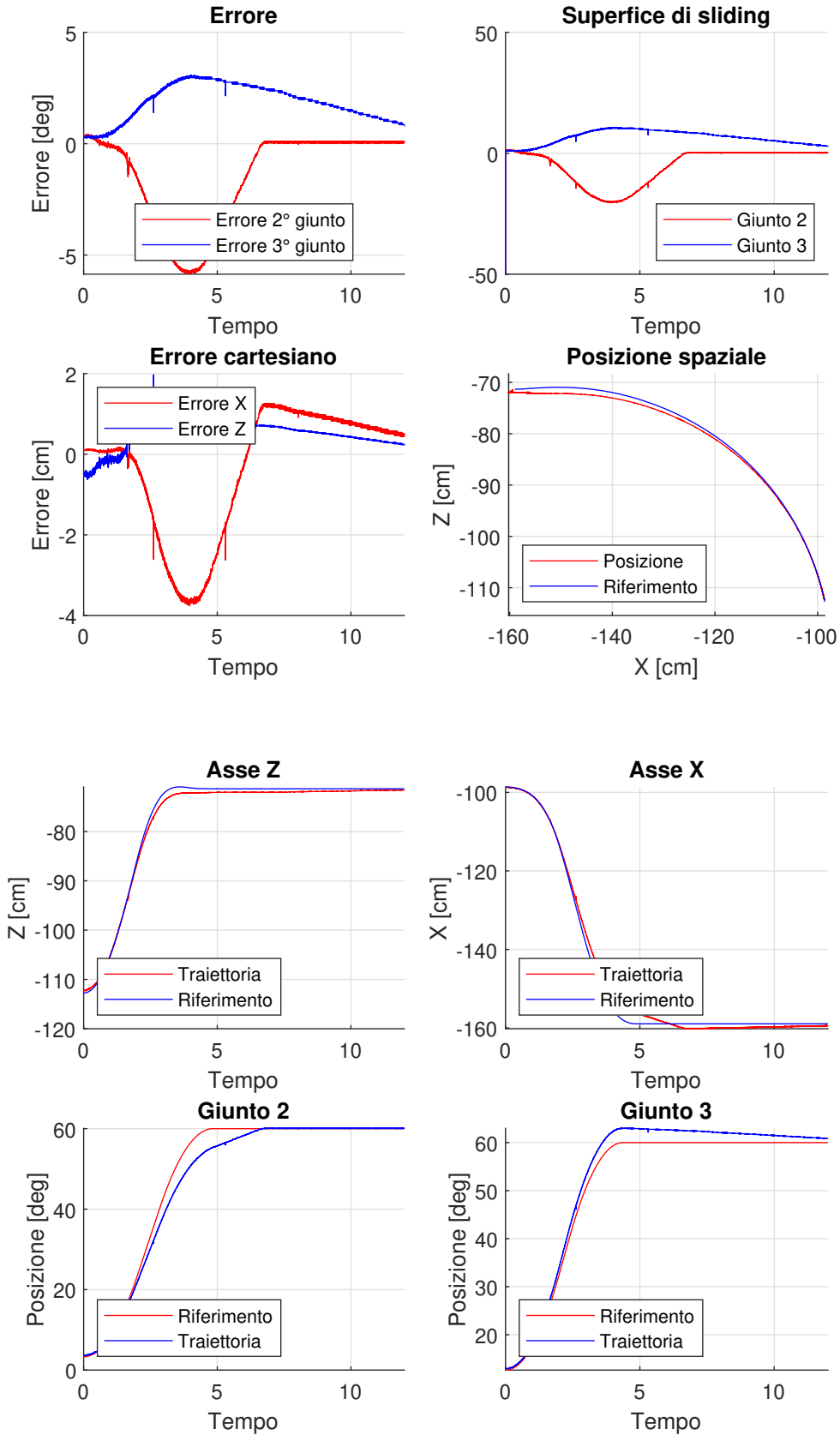


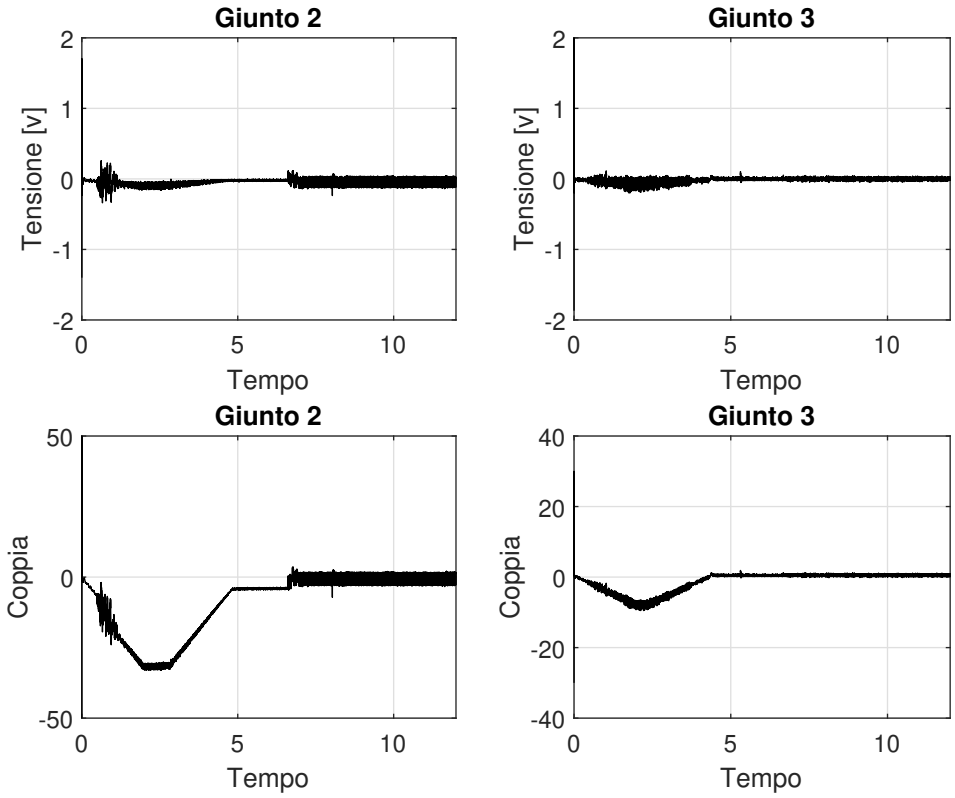
Figure 4.1: Grafici relativi ad errori, superfici si sliding e sforzo di controllo

4.3 Prove reali: controllo e uso della visione artificiale

4.3.1 Risultati del controllo VSC

I grafici ottenuti sono relativi all'esecuzione effettuata con i parametri individuati per un movimento verso l'alto del manipolatore. L'errore massimo commesso per ogni giunto è contenuto in un massimo di 5° che si traduce in un errore massimo di 3 cm sul piano cartesiano. Attraverso i grafici relativi alle tensioni e alle coppie si evince che la scelta dei parametri è ottimale in quanto non sono presenti grandi variazioni che all'atto pratico si traducono in vibrazione dei link del manipolatore; questo è confermato dal fatto che nei grafici delle traiettorie seguite dai giunti una volta raggiunto il valore costante del riferimento non si hanno oscillazioni intorno a questo bensì i giunti mantengono correttamente la posizione. Nei grafici relativi gli errori e le superfici di sliding si vede che dall'inizio del movimento i giunti del manipolatore impiegano circa 5 secondi per raggiungere il valore di riferimento, circa un secondo dopo che il riferimento assume un valore costante indicato dalla posizione che si desidera raggiungere col manipolatore. Va sottolineato che i grafici possono variare di simulazione in simulazione in quanto il manipolatore dimostra di risentire di disturbi ambientali quali temperatura, direzione di movimento, posizione finale e tipo di riferimento (traiettoria o punto fisso da raggiungere); ognuno di questi fattori va ad influire sul tuning dei parametri per cui è importante individuare correttamente il task da far eseguire al sistema prima di effettuare il tuning dei parametri. Un'ulteriore differenza che si nota nel passaggio dalla simulazione alle prove reali è data dell'incertezza parametrica dei valori del manipolatore che come spiegato inizialmente questi sono ottenuti attraverso una stima.


Errore
Errore [deg]
Errore 2° giunto
Errore 3° giunto
Tempo
Superfice di sliding
Giunto 2
Giunto 3
Tempo
Errore cartesiano
Errore [cm]
Errore X
Errore Z
Tempo
Posizione spaziale
Z [cm]
Posizione
Riferimento
X [cm]
Asse Z
Z [cm]
Traiettoria
Riferimento
Tempo
Asse X
X [cm]
Traiettoria
Riferimento
Tempo
Giunto 2
Posizione [deg]
Riferimento
Traiettoria
Tempo
Giunto 3
Posizione [deg]
Riferimento
Traiettoria
Tempo


Giunto 2
Tensione [v]
Tempo
Giunto 3
Tensione [v]
Tempo
Giunto 2
Coppia
Tempo
Giunto 3
Coppia
Tempo

4.3.2 Risultati della visione

Il sistema progettato durante le prove impiega pochi secondi per acquisire l'immagine, processarla e individuare l'oggetto restituendo l'immagine con indicate coordinate e bounding box, accompagnato dalla dicitura *target* e lo score relativo al rilevamento. I tempi ridotti sono dovuti anche al fatto che l'immagine subisce una semplice operazione di preprocessing, per cui una volta acquisita prima di procedere alla fase di detection questa viene ritagliata riducendone significativamente le dimensioni, in particolare questa viene ritagliata verticalmente al fine di individuare nell'immagine solo i limiti individuati dai nastri orizzontali, e orizzontalmente da destra e sinistra vengono eliminate le parti del piano di lavoro non raggiungibili dal manipolatore (e quindi i punti in cui si è certi che per l'applicazione richiesta l'oggetto non sarà posizionato); complessivamente l'immagine viene ridotta quasi a un terzo dell'originale. L'oggetto rilevato può essere indicato con un cerchio centrato sui valori delle cordinate centrali del bbox o con un riquadro che ne indica lo score.

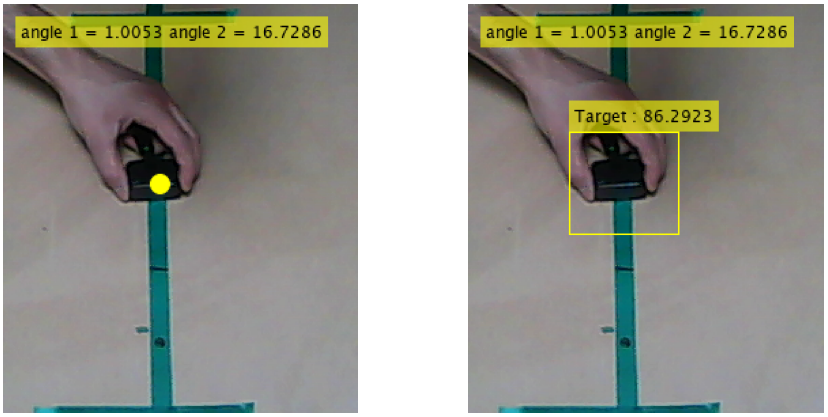


Figure 4.2: Immagine con target e relative coordinate

4.3.3 Valutazione della posizione raggiunta

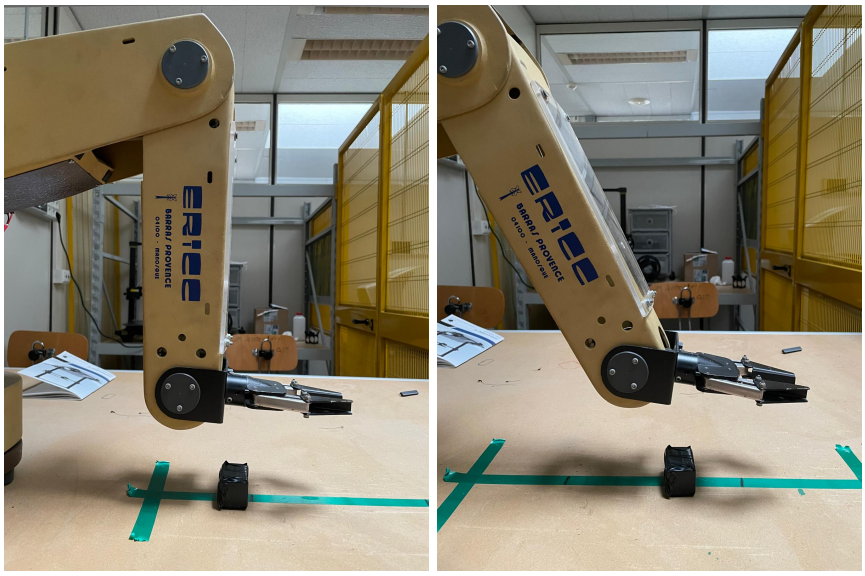


Figure 4.3: Posizioni raggiunte dal manipolatore (vista frontale)

In figura 4.3 si vede come il manipolatore si posiziona con precisione al di sopra dell'oggetto rilevato; il manipolatore si trova leggermente più in alto dell'oggetto in quanto in fase di sperimentazione è stata indicata un altezza maggiore di quella dell'oggetto al fine di prevenire eventuali collisioni dovuti a possibili errori di progettazione. Indicando nei parametri dell'equazione per la cinematica inversa il corretto valore dell'altezza dell'oggetto il manipolatore tenderà a raggiungere l'oggetto rilevato a meno dell'errore ottenuto durante la fase di valutazione del controllo VSC.

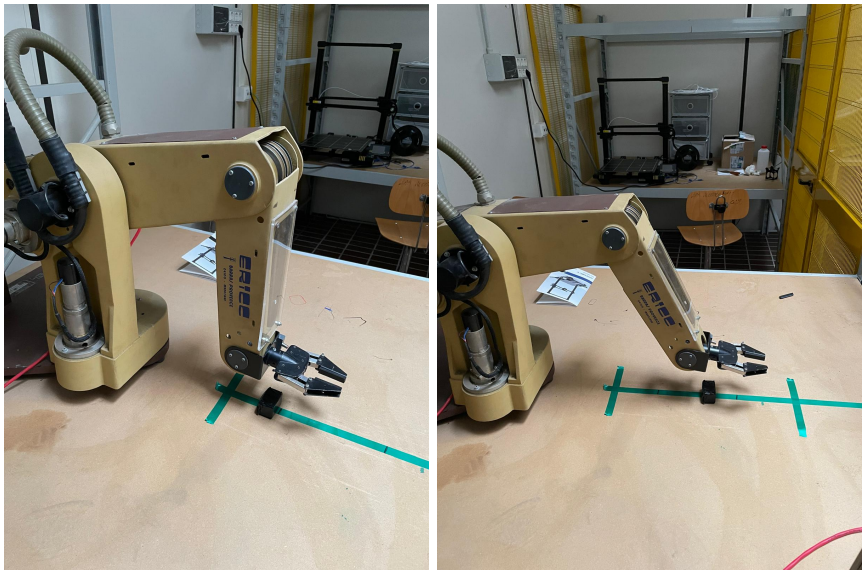


Figure 4.4: Posizioni raggiunte dal manipolatore (vista dall'alto)

4.4 Sviluppi futuri

- Implementazione del controllo del primo giunto per ampliare le funzionalità già esistenti;
- Valutare ulteriori rilevatori o ottimizzazione della funzionalità di quello presente:
  - renderlo più veloce inserendo una fase di preprocessing dell'immagine acquisita;
  - valutare un diverso detector da utilizzare
- Ottimizzazione dei parametri del VSC al fine di migliorare il controllo;
- Uso del VSC affiancato da rete neurale;
- Migliorare il metodo di comunicazione tra i due PC per lo scambio di dati e segnali;
- Implementare un modello 3D per la raffigurazione della simulazione;

Bibliography

[1] L. Colombo "Controllo di coppia di un robot industriale";

[2] M. Lippera “Controllo Robusto di un Manipolatore Industriale con Retroazione di Forza”;

[3] V. Fossi “Progetto di un Sistema a Struttura Variabile, Basato su Reti Neurali, per il Controllo Robusto di un Manipolatore Industriale”;

[4] S. Antonelli “Controllo Dinamico a Struttura Variabile di un Manipolatore Antropomorfo”;

[5] S. Del Giudice “Controllo in Real Time di Robot Industriali";

[6] Collegamento PC/PC attivazione supporto SMB1: `https://answers.microsoft.com/it-it/windows/forum/all/non-riesco-pi%C3%B9-a-vedere-i-pc-collegati-alla/8144161c-e71d-46ff-bfff-24c414d42e03`