POLITECNICO DI TORINO

Corso di Laurea Magistrale in

Ingegneria Meccanica

Tesi di Laurea Magistrale

Analisi del braccio robotico collaborativo UR3 ed esempi di programmazione

Relatore/i

Prof. Stefano Paolo Pastorelli

Prof. Stefano Mauro

Candidato

Fiorenzo Apicella

Matr: S241958

A.A.2017/2018

Come robot avrei potuto vivere per sempre, ma dico
a tutti voi oggi, che preferisco morire come uomo,
che vivere per tutta l'eternità come macchina.

Dedicata alla nonna Lucrezia e al collega Antonio

Sommario

1. INTRODUZIONE

1.1 Robotica collaborativa

I robot collaborativi rappresentano una grande opportunità di avanzamento tecnologico in molti settori in cui la robotica è ancora lontana. Questo sta diventando sempre più uno dei temi principali nell'ambito dell'*Industria 4.0* come tecnologie abilitanti di sistemi adattativi, di flessibilità della produzione, di riconfigurabilità, di efficienza. In maniera più rilevante questi tipi di robot stanno apportando un graduale ma radicale cambiamento riguardo le condizioni di lavoro combinando l'aspetto legato alla produttività a quello della salute di lavoratori e utenti. Col termine "Robotica collaborativa" si indica la condivisione del lavoro. Nel termine c'è lo spirito di una delle forme di robotica industriale di maggiore impatto nel corso attuale del manifatturiero e dell'automazione. Il significato va oltre l'usuale concetto di automazione dei compiti umani che ha segnato la nascita dei robot, si parla spesso, infatti, di *co-worker o cobot* (macchine dotate di limitatori di forza). Non esiste una definizione unica di robotica collaborativa: le molte sfumature comprendono la coesistenza, la cooperazione, l'assistenza, la compresenza dell'uomo. In breve, è collaborativa qualsiasi forma di interazione tra uomo e sistema robotizzato funzionale all'esercizio di un compito produttivo che non potrebbe essere eseguito altrimenti o in modo altrettanto efficace o remunerativo.

La "*collaboratività"* si manifesta con l'accesso al sistema robotizzato e allo spazio di lavoro disponibile in qualsiasi ordine, per compiere azioni funzionalmente legate, simultanee, contestuali. Sarebbe più adeguato parlare di applicazioni collaborative piuttosto che di robot; la collaborazione infatti deriva dall'uso di una macchina oltre che a conoscere e sfruttare le sue proprietà.

L'utilità di un'applicazione collaborativa è perciò un criterio fondamentale per definire e progettare un sistema adatto alla presenza umana. L'interesse del mondo industriale per la robotica collaborativa deriva da questo elemento: gli operatori intuiscono la flessibilità d'uso delle soluzioni manuali-automatiche, in un contesto di rinnovata centralità dell'elemento umano nella produzione.

Evidentemente perciò lo sviluppo e la scelta della robotica collaborativa è dettata da motivazioni economiche, di salute occupazionale e di utilizzo efficiente dello spazio di fabbrica [1].

Sebbene sia stato illustrato in breve il concetto di cobot, il fine ultimo della progettazione di queste macchine ancora non è stato ben chiarito. Principalmente vengono pensati per sostituire i robot industriali nelle loro operazioni più semplici. Nella tabella riportata di seguito è possibile individuare quelle che sono le principali differenze tra un classico robot industriale e un robot di tipo collaborativo secondo come indicato in alcuni documenti di "Robotiq".

Tabella 1: differenze tra robot industriali e cobot [2]

PROPRIETA'	ROBOT INDUSTRIALI	COBOT
Muovere pezzi intorno	✓✓	✓✓
Seguire un percorso/traiettoria	✓✓	✓✓
Lavorare autonomamente per un prolungato periodo di tempo	✓✓	✓✓
Migliorare la produttività e la qualità del prodotto	✓✓	✓✓
Ridurre infortuni muscolari dei lavoratori	✓✓	✓✓
Richiesta di sensori e/o sistemi visivi per la sicurezza	✓✓	
Richiesta di un'estesa conoscenza della robotica	✓✓	
Occupare molta parte del piano di lavoro	✓✓	
Sono costosi	✓✓	
Sono semplici per programmatori non esperti		✓✓
Sono semplici da inserire nell'aria di lavoro esistente		✓✓
Sono facili da configurare per nuove task		✓✓
Semplici passaggi da una task all'altra		✓✓
Sono semplici da istallare		✓✓

Questi tipi di robot dunque, tendono a ricoprire i ruoli delle classiche macchine di industria pesante, apportando importanti benefici per gli operatori umani.

Anche se l'utilizzo può risultare ancora abbastanza incerto, sicuramente la maggior parte delle attività che possono svolgere devono possedere due requisiti importanti [2]:

- **Alta prevedibilità,** dove i movimenti tendono ad essere quasi sempre gli stessi con poche piccole variazioni, quali:
  - quando bisogna prelevare oggetti sempre nella stessa posizione oppure organizzati tutti allo stesso modo;
  - quando bisogna spostare oggetti con simili dimensioni, peso e forma;
  - quando bisogna spostare oggetti simili ma di forme irregolari.

- **Ripetibilità:**
  - operazioni coerenti tra oggetti dello stesso lotto o tra più lotti;
  - in operazioni a lungo termine;
  - domanda con alta frequenza.

1.2 Descrizione del sistema di controllo e del robot UR3

Il prodotto trattato ed analizzato nel presente elaborato è uno dei robot progettato da “Universal Robots”, ovvero il braccio meccanico UR3. Questo robot costituisce un esempio significativo di robotica collaborativa in tutti i suoi aspetti e caratteristiche. Come già accennato nell’introduzione l’aggettivo “collaborativo” può assumere molteplici accezioni. Questi sistemi meccanici sono in grado di lavorare fianco a fianco con i dipendenti grazie ad un controllo di forza integrato che arresta automaticamente i robot quando incontrano un ostacolo sul loro percorso. Possono inoltre essere programmati in modo da poter lavorare in modalità ridotta quando una persona entra nella zona di lavoro. La collaborazione però, non è solo riferita alla sicurezza ma assume un senso molto più ampio: riguarda allo stesso tempo anche la facilità d’uso e la semplicità con cui si gestisce e riprogramma le attività della macchina [3]

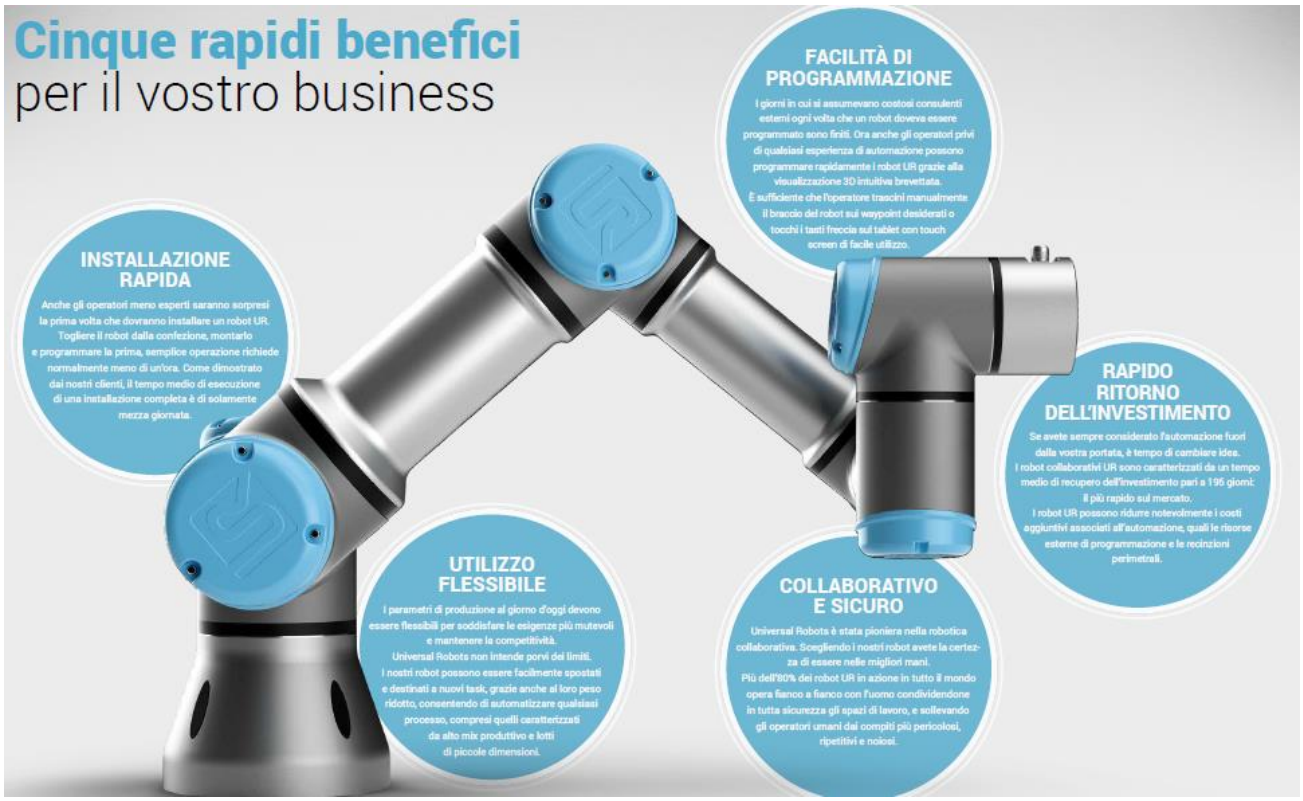


Figura 1: UR3 e possibili vantaggi della robotica collaborativa [3]

Come si osserva in figura 1, tra alcuni aspetti positivi di questa macchina compare una certa flessibilità dal punto di vista applicativo. Sono diverse, infatti, le operazioni che questo robot può svolgere. Alcune di queste sono suggerite dalla casa costruttrice e illustrate nelle figure successive.

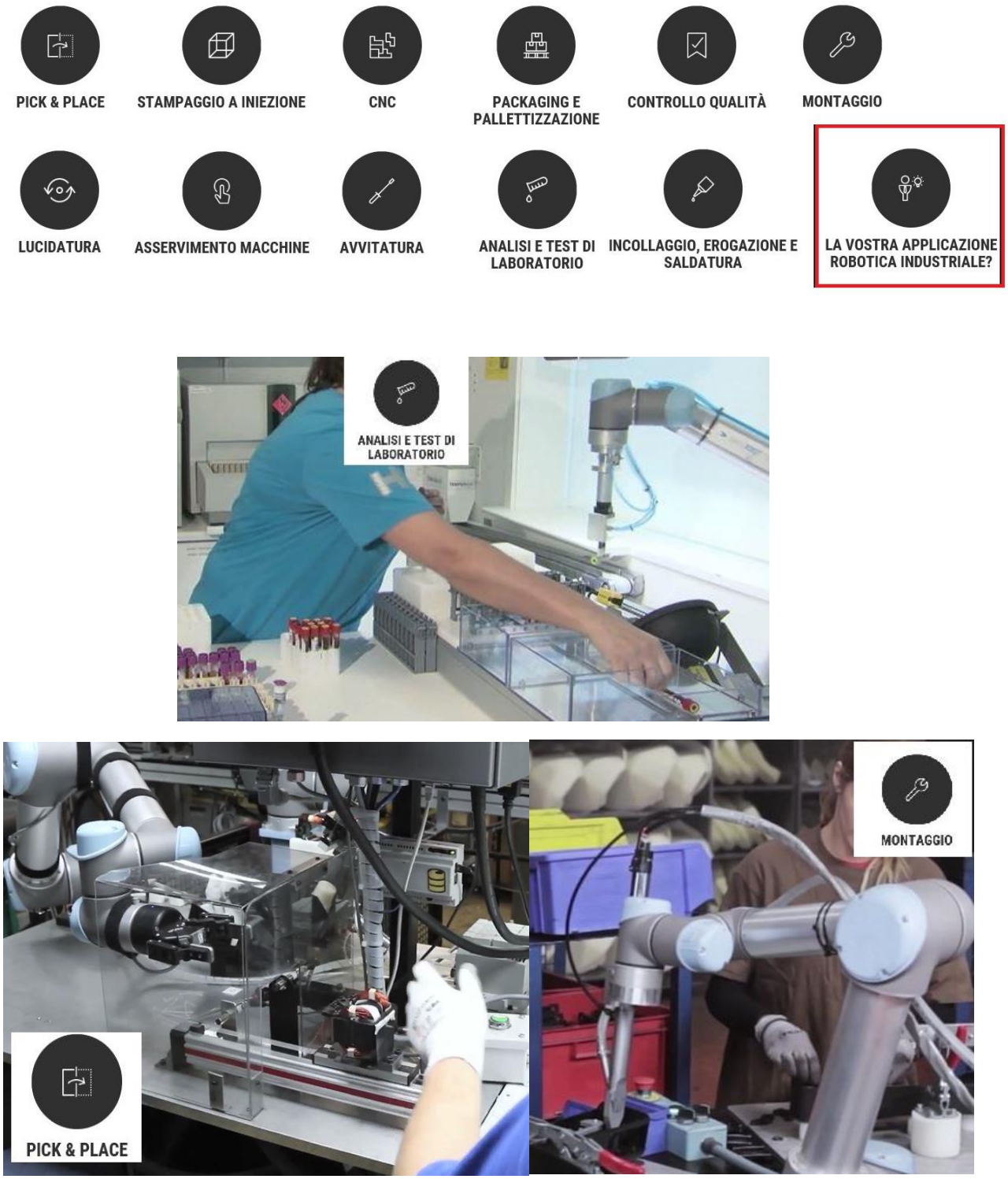


Figura 2: alcune delle applicazioni collaborative [4]

Prima di descrivere nel dettaglio l'ambiente di programmazione del robot, si ritiene opportuno illustrare e descrivere brevemente la struttura fisica dell'intero sistema, compreso di case di controllo e di tutta la rete di cavi di connessione. I componenti principali schematizzati in figura 3 sono i seguenti:

- 1: ***teach pendant*** dalla quale è possibile programmare i movimenti del robot;
- **2**: ***controller*** in cui sono presenti la scheda ESD e tutti le porte per il collegamento dei cavi;
- **3**: ***UR3*** robot fisico;
- **4**: ***cavetti di connessione*** (non è riportato nello schema ma è presente anche un cavo uscente dal controller che viene utilizzato per l'acquisizione dati).

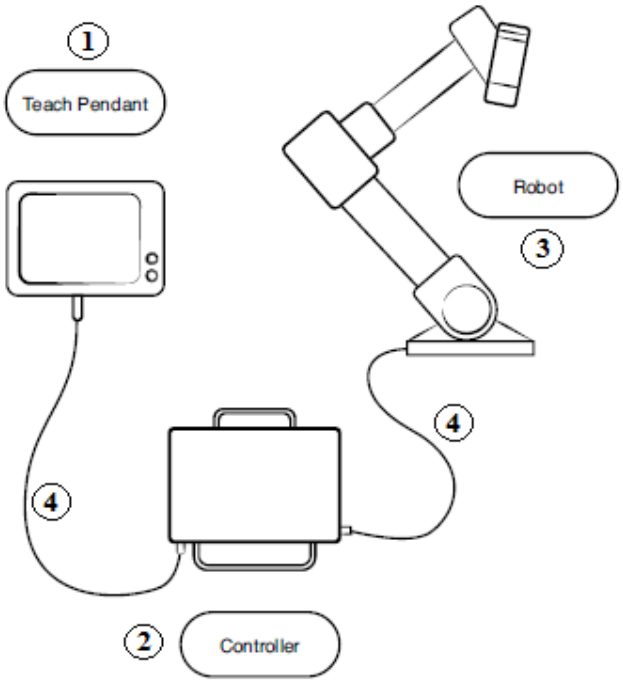


Figura 3: schematizzazione della struttura del sistema [5]

Per poter utilizzare il robot per prima cosa bisogna capire come montare i componenti. Bisogna collegare il teach pendant e il robot fisicamente al controller mediante i seguenti attacchi situati alla base del controller:

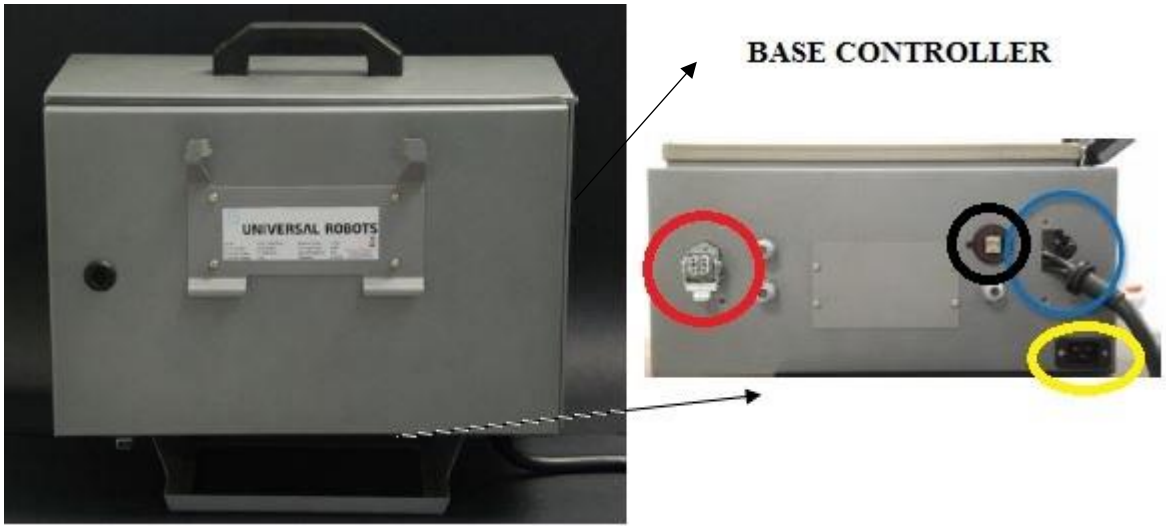
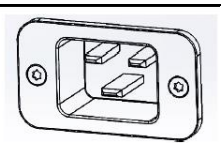
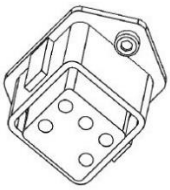
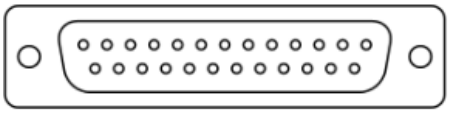


Figura 4: controller e diverse porte di collegamento [6]

Tabella 2: diverse porte di collegamento presenti sulla base del controller

	Nel cerchio giallo l'attacco **IEC C20** per la connessione del sistema alla rete elettrica.
	Nel cerchio rosso il connettore per il collegamento del robot al sistema di controllo. Attenzione non scollegare mai durante il funzionamento del robot.
	Nel cerchio nero il collegamento per connessione **ETHERNET** utilizzata per i moduli di espansione I/O, per l'accesso e controllo a distanza (utilizzato nell'analisi per l'acquisizione dati)
	Nel cerchio blu la porta per il cavo **RS323** per la connessione al controller del teach pendant.

Una volta effettuato il collegamento tra i vari componenti è possibile incominciare ad operare attivando il teach pendant. Questo componente si presenta come in figura 5:

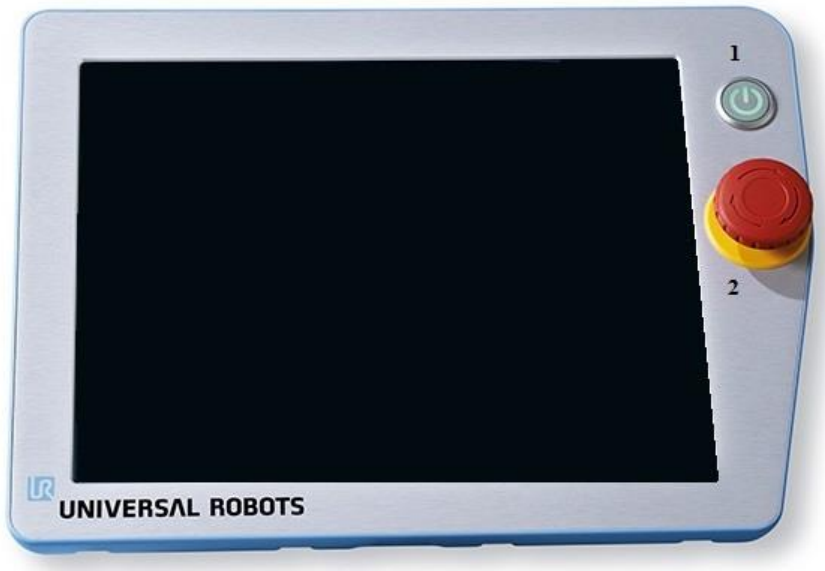


Figura 5: teach pendant

Con il pulsante numero uno si accende il sistema, con il pulsante rosso numero due invece, si arresta la macchina in caso di emergenza. Il pulsante rosso rimane schiacciato finché non lo si risolleva manualmente. È presente un ulteriore pulsante sul retro che, se tenuto premuto, permette di far muovere il robot pilotandolo tramite l'operatore. Inoltre, questo dispone anche di un ingresso USB, per poter salvare i programmi scritti su una memoria esterna e trasferirli su qualsiasi altro dispositivo (anche in formato .txt).

Per concludere si può descrivere il robot che è oggetto di studio di questa tesi. L'UR3 è un robot a sei assi in cui i primi tre costituiscono i gradi di libertà del braccio mentre gli ultimi tre del polso non sferico. Il braccio di "Universal Robot" è costituito da tubi (con materiali come alluminio, plastica PP, acciaio) e giunti. I giunti sono indicati nella figura seguente con i loro nomi abituali. La **Base** è il supporto su cui è montato il robot, mentre all'altra estremità (**Polso 3**) è fissato l'utensile del robot (il punto centrale della flangia di collegamento sarà spesso indicato nel testo come **tool** o **TCP)** [7]. Coordinando il movimento di ciascuno dei giunti, il robot può muovere liberamente l'utensile, con l'eccezione dell'area direttamente sopra e direttamente sotto alla base. In figura sono riportate anche le coppie di serraggio da applicare, nella fase di montaggio, tra un componente e l'altro [6] .

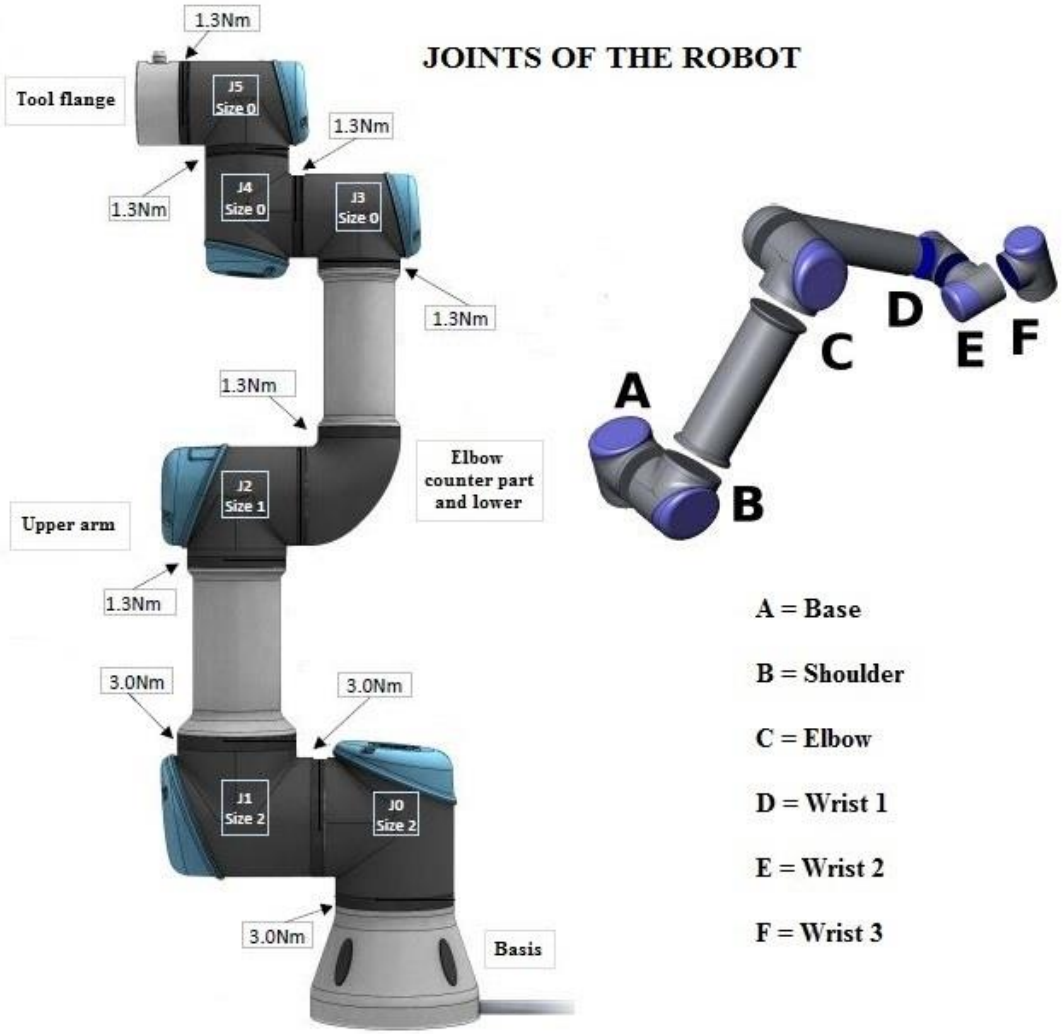


Figura 6: UR3, motori e coppie di serraggio tra i diversi link, nome dei giunti

Tabella 3: coppie massime dei motori

SIZE MOTOR	MAX TORQUE [Nm]
Size 0	12
Size 1	28
Size 2	56

UR3

Prestazioni

Ripetibilità	±0.1 mm / ±0.0039 in (4 mils)
Gamma di temperatura ambiente	0-50°*
Consumo di corrente	Min 90W, Tipico 125W, Max 250W
Funzionamento in collaborazione	15 funzioni di sicurezza avanzate regolabili. Funzione di sicurezza approvata TüV NORD Collaudata in conformità con: EN ISO 13849:2008 PL d

Specifiche

Carico utile	3 kg / 6.6 lbs
Portata:	500 mm / 19.7 in
Gradi di libertà	6 giunti rotanti
Programmazione	Interfaccia utente grafica Polyscope su schermo touch da 12 pollici con supporto

Movimento

Movimento assiale del braccio robotico	Raggio d'azione	Velocità massima
Base	± 360°	± 180°/Sec.
Spalla	± 360°	± 180°/Sec.
Gomito	± 360°	± 180°/Sec.
Polso 1	± 360°	± 360°/Sec.
Polso 2	± 360°	± 360°/Sec.
Polso 3	Infinita	± 360°/Sec.
Attrezzo tipico		1 m/Sec. / 39.4 in/Sec.

Caratteristiche

Classificazione IP	IP64	
Classe ISO Clean Room	5	
Rumorosità	70dB	
Posizione di installazione robot	Qualsiasi	
Porte I/O	Ingresso digitale	2
	Uscita digitale	2
	Ingresso analogico	2
	Uscita analogica	0
Alimentazione al tool	12 V/24 V 600 mA al tool	

Corpo robot

Ingombro alla base	Ø128mm
Materiali	Alluminio, plastiche PP
Tipo di connettore Tool	M8
Lunghezza cavo di collegamento robot	6 mm / 236 in
Peso meccanica robot con cavo	11 kg /24.3 lbs

QUADRO ELETTRICO

Caratteristiche

Classificazione IP	IP20	
Classe ISO Clean Room	6	
Rumorosità	<65dB(A)	
Porte I/O	Ingresso digitale	16
	Uscita digitale	16
	Ingresso analogico	2
	Uscita analogica	2
Alimentazione I/O:	24V 2A	
Comunicazione	TCP/IP 100Mbit, Modbus TCP, Profinet, EthernetIP	
Fonte di alimentazione	100-240 VAC, 50-60 Hz	
Gamma di temperatura ambiente	0-50°	

Struttura

Dimensioni Quadro Elettrico	475mm x 423mm x 268mm / 18.7 x 16.7 x 10.6 in
Peso	15 kg / 33,1 lbs
Materiali	Lamiera

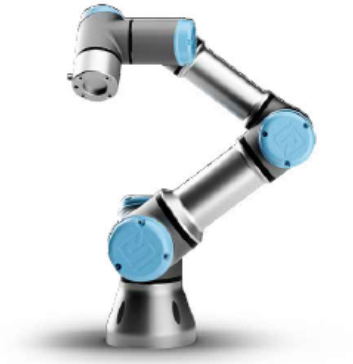
TERMINALE DI PROGRAMMAZIONE

Caratteristiche

Classificazione IP	IP20

Struttura

Materiali	Alluminio, PP
Peso	1,5 kg
Lunghezza cavo	4,5 m / 177 in



*Il robot può operare ad un intervallo di temperatura di 0-50°C. In caso di velocità sostenuta e continua dei giunti, la temperatura ambiente è ridotta.

Figura 7: scheda tecnica del robot [4]

Sull'ultimo giunto è possibile collegare differenti oggetti. Nel caso preso in considerazione sono stati montati i seguenti componenti aggiuntivi: la cella di carico *Robotiq FT300 Force Torque Sensor* e il gripper a due dita *Robotiq 2F-85.*

Seguendo le istruzioni del manuale è stato eseguito il montaggio, meccanico ed elettrico, e la calibrazione.

Si deve porre molta attenzione al collegamento elettrico (identico nei due casi) da effettuare con i diversi cavetti. Se ne possono trovare cinque tutti di diverso colore (bianco, rosso, nero, verde e grigio).

Come indicato nelle due immagini successive, il cavetto nero e rosso devono essere collegati rispettivamente ai morsetti 0V e 24V presenti sulla scheda di controllo, mentre gli altri tre cavetti sono collegati ad un adattatore USB che viene connesso al controller.

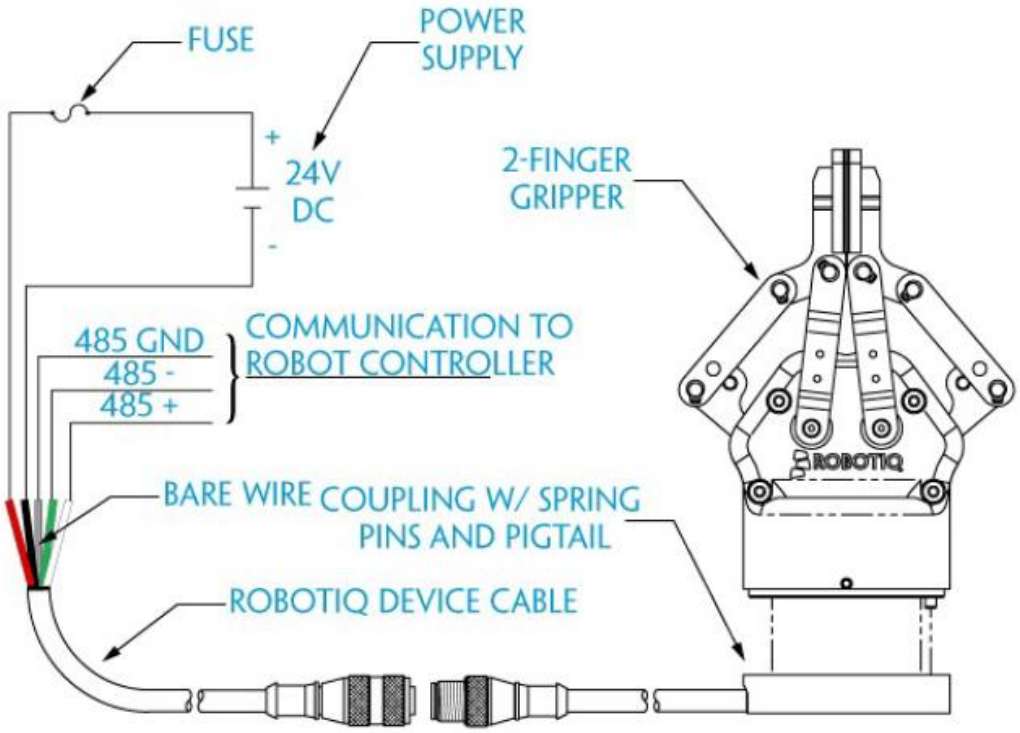


Figura 8: collegamento cavi della pinza (ugual modo quello della cella) [8]

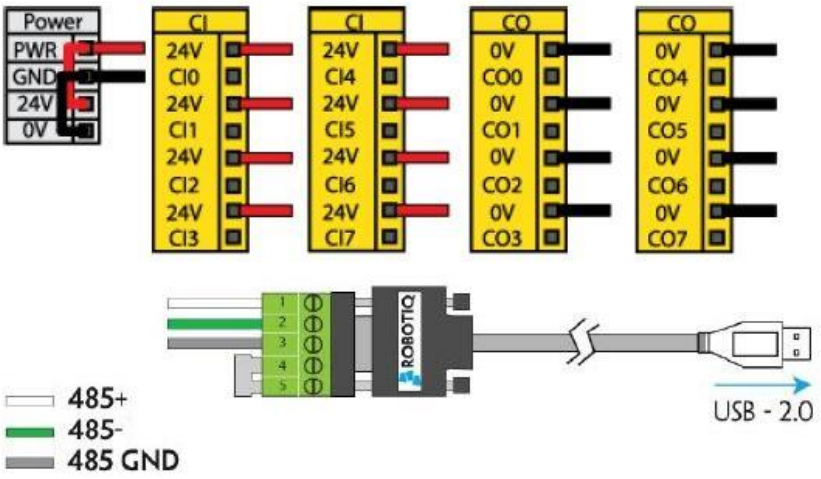


Figura 9: dettaglio collegamento cavi [9]

Il processo di calibrazione invece consiste, per quanto riguarda la cella, nel collocarla in diverse posizioni suggerite dal programma ed effettuare delle misurazioni di forza a tool scarico, mentre per la pinza consiste semplicemente nel tarare la chiusura e apertura delle dita in termini di posizione.

Ovviamente la calibrazione della cella è avvenuta in primo luogo senza gripper montato, successivamente è stata ripetuta ma a pinza collegata.

Si possono dunque descrivere in breve le principali impostazioni dei due componenti.

1.2.1 Robotiq FT 300 Force Torque Sensor

Figura 10: cella di carico [2]

Con questo componente aggiuntivo è possibile misurare le forze e le coppie lungo le tre direzioni nello spazio. Ovviamente con l'aggiunta della cella di carico e del gripper è stato necessario settare nuovamente le impostazioni del tool.

Dalle operazioni di calibratura con cella e gripper montato, scarico e aperto, risultano misurate le grandezze riportate in tabella 4. Risulta infatti esserci una componente di forza significativa lungo la direzione z poiché corrisponde al peso della pinza montata in serie alla cella.

Tabella 4: valori letti dalla cella di carico in seguito a calibrazione

grandezza	valore
Fx [N]	-3.6
Fy [N]	0.9
Fz [N]	21.2

grandezza	valore
Mx [Nm]	-0.07
My [Nm]	0.11
Mz [Nm]	0.05

I valori massimi e le principali specifiche della cella sono riprese dal catalogo e riportate in figura 11 [2]:

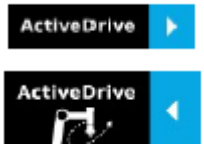
SPECIFICATION		FT 300 (prior to 10/2017)	FT 300
Measuring range	Force	±300 N	
	Torque	± 30 Nm	
Signal noise*	Fx, Fy	1.2 N	0.1 N
	Fz	0.5 N	0.1 N
	Mx, My	0.02 Nm	0.005 Nm
	Mz	0.03 Nm	0.003 Nm
Recommended threshold	Fx, Fy	5 N	1 N
	Fz	2 N	1 N
	Mx, My	0.08 Nm	0.02 Nm
	Mz	0.12 Nm	0.01 Nm
External noise sensitivity**		Immune	
Data output rate		100 Hz	
Temperature compensation***		15 to 35 °C	

Figura 11: principali specifiche della cella [10]

Nel momento in cui viene istallata la cella di carico è possibile vedere in automatico il toolbar che compare in alto sulle schermate del software. In questo toolbar si individuano le principali funzioni che possono essere utilizzate per mezzo della cella.

Vengono riportate di seguito i diversi comandi:

Tabella 5: impostazioni fondamentali della toolbar [2]

	ActiveDrive	barra degli strumenti: nel momento che diventa di colore grigia non è disponibile e bisogna attivarla
	Start/Stop	quando è attivo permette di spostare il robot applicando forze sull'end effector
	Modes	modalità ActiveDrive consentono di limitare il robot a movimenti specifici
	Drift Control	arresta il robot dalla deriva
	Speed	consente di effettuare movimenti precisi a rallentatore. Il sensore di coppia è in grado di rallentare automaticamente in caso di urto con un oggetto
	Square robot	allinea l'orientamento del TCP con la base del robot

Con l'istallazione della cella di carico infine è possibile utilizzare anche un nuovo comando indicato con PATH nella sezione structure. Con quest'ultimo è possibile "registrare" dei movimenti che vengono compiuti dal robot per mezzo di forze applicate al tool (per approfondimenti consultare il manuale [2]).

Figura 12: schermata principale path [10]

1.2.2 Robotiq 2F-85

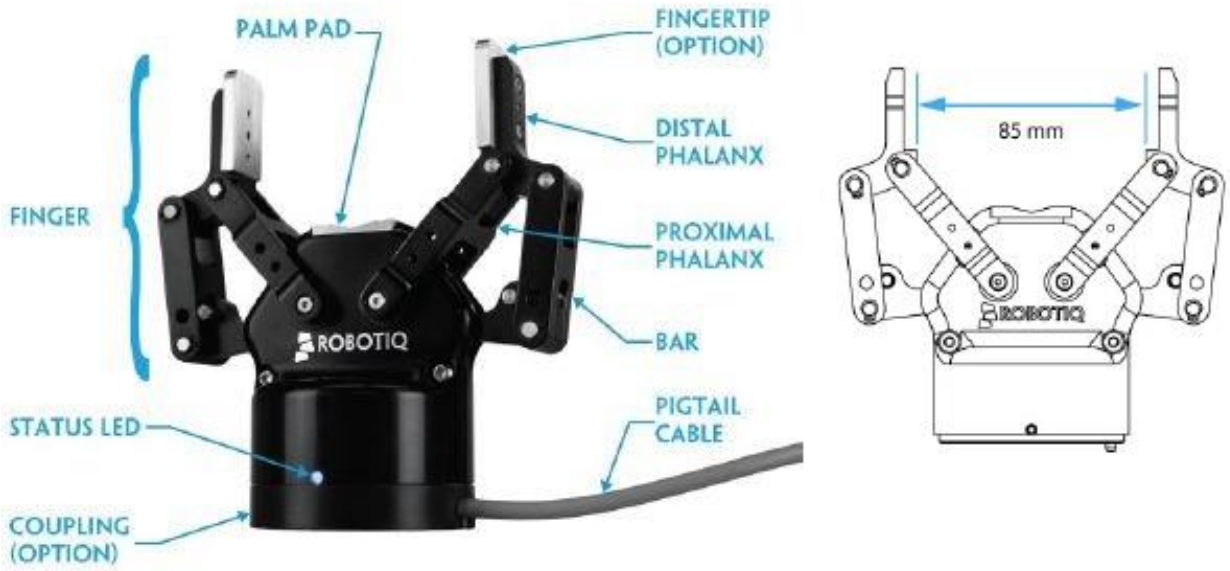


Figura 13: pinza [3]

Per poter programmare il gripper viene utilizzata un'interfaccia molto semplice in cui l'utente può scegliere la posizione di apertura e chiusura delle dita, la velocità con cui si apre e chiude e la forza con cui viene stretto l'oggetto.

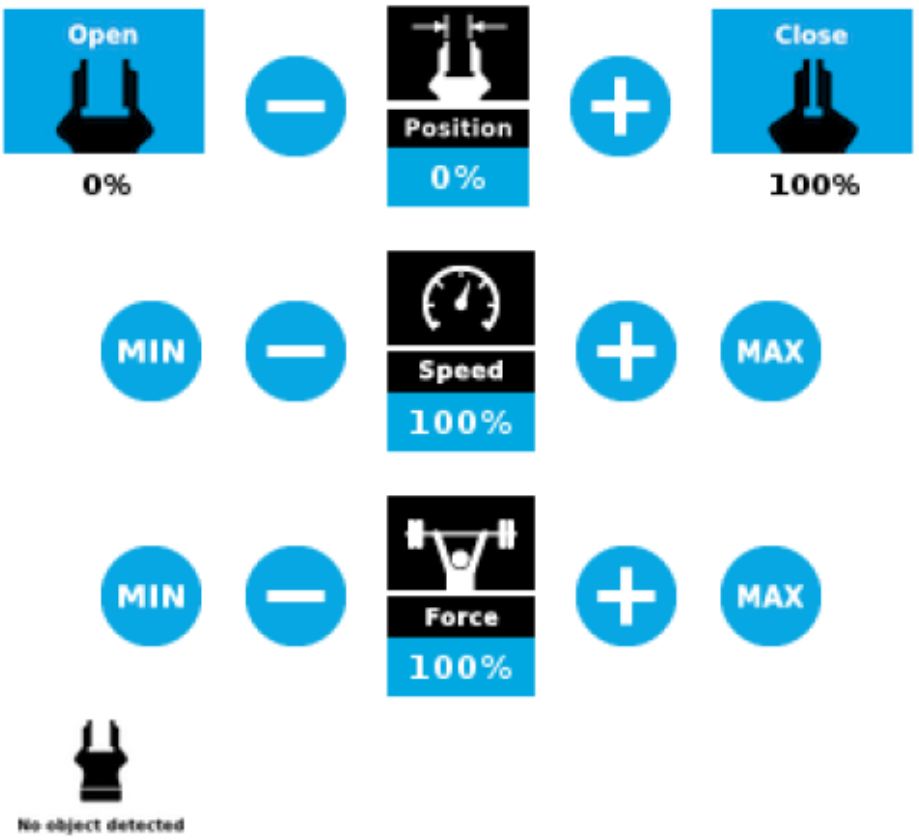


Figura 14: interfaccia utente del gripper [3]

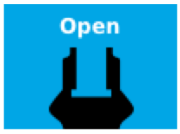
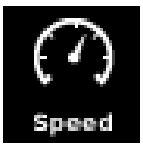
È possibile scegliere in termini percentuali sia l'apertura della pinza sia i valori di velocità e forza. Ovviamente tali parametri saranno compresi tra valori limite riportati nella seguente tabella [3]:

Tabella 6: valori limite per le proprietà della pinza

	MIN	MAX
position [mm]	0	84
speed [mm/s]	20	150
force [N]	20	235

Durante la presa di un oggetto, anche se è possibile scegliere arbitrariamente un valore di chiusura, è sempre consigliabile utilizzare il comando GripperClose(1), ovvero chiudere completamente la pinza durante la presa. Questo permette di afferrare l'oggetto in maniera più sicura. È importante inoltre, la capacità di tale gripper di adattarsi in fase di chiusura al tipo di geometria dell'oggetto che sta prelevando e al punto in cui viene preso. I parametri di forza inoltre potrebbero leggermente variare in funzione del tipo di materiale che viene posto sulle dita (per nozioni più approfondite è possibile consultare il manuale del gripper). Vengono riportati di seguito le principali icone che possono essere visualizzate nella sua interfaccia. Nel momento in cui si decide come far funzionare il gripper è possibile salvare le impostazioni del comando con il pulsante in basso "save action".

Tabella 7: impostazioni fondamentali dell'interfaccia [3]

	Open	apre completamente il gripper
	Close	chiude completamente il gripper
	Requested Position	mostra l'attuale posizione del gripper. Se non è calibrato i valori saranno in percentuale
	Requested Speed	mostra la velocità attuata del gripper
	Requested Force	mostra la forza attuata dal gripper
	No object detected/ object detected	mostra se un oggetto è prelevato o meno

1.2.3 Connessione e funzioni per l'acquisizione dati

Una volta visti gli elementi principali costituenti il sistema, è possibile attuare delle connessioni tra il robot ed un qualsiasi calcolatore. Prima di tutto è opportuno procurarsi un cavetto ethernet da collegare da un'estremità al calcolatore e dall'altra al fondo del controller nell'opportuna porta vista in precedenza.

Successivamente bisogna stabilire un collegamento tra i due componenti mediante dei socket di comunicazione, ovvero delle astrazioni software progettate per poter utilizzare delle API standard e condivise, per la trasmissione e la ricezione di dati attraverso una rete oppure come meccanismo di IPC. Quindi risulta essere il punto in cui il codice applicativo di un processo accede al canale di comunicazione per mezzo di una porta, ottenendo una comunicazione tra processi che lavorano su due macchine fisicamente separate [11]. Dal punto di vista di un programmatore perciò, un socket è un particolare oggetto sul quale leggere e scrivere i dati da trasmettere o ricevere. Per poter ottenere questa comunicazione c'è bisogno di specificare un indirizzo IP statico sul software del robot. Per poterlo inserire bisogna accendere il tablet come visto in precedenza e seguire le istruzioni dell'immagine successiva:

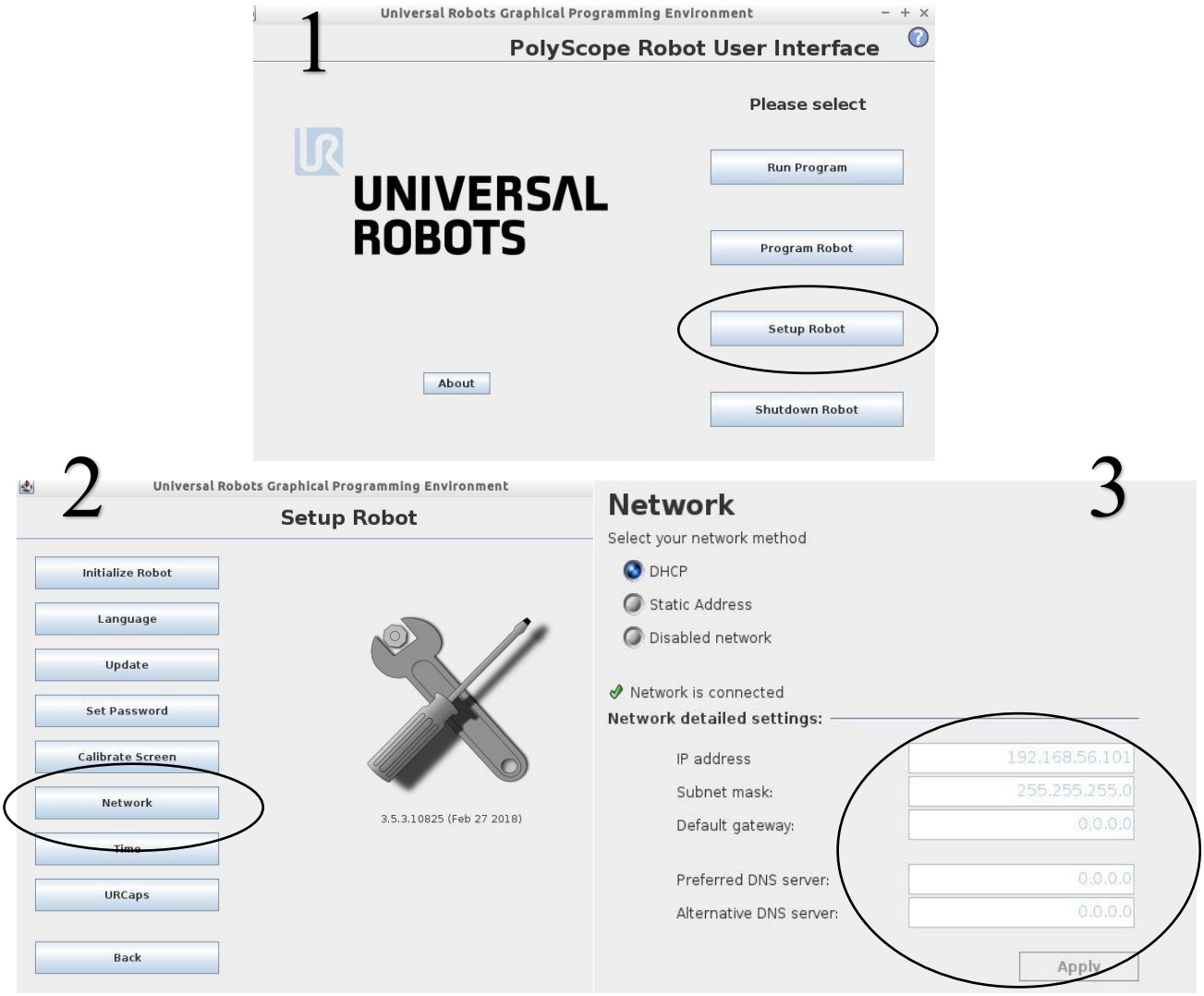


Figura 15: istruzioni per inserire l'indirizzo IP

Una volta inserito questo dato è stato possibile, attraverso la scrittura di un programma (nel caso del presente lavoro è stato utilizzato Matlab), accedere al socket ed alla fase di acquisizione dei dati forniti dal controllore del sistema. Di seguito viene spiegato brevemente questo programma utilizzato per l'acquisizione:

```
%% FILE ACQUISIZIONE DATI

clear all; close all; clc
```

1

```
% se si vuole acquisire da simulatore
% IP='192.168.56.101'; %IP simulator

% se si vuole acquisire dal robot
  IP='192.168.56.103'; %IP Robot
```

2

```
% apertura della GUI per l'acquisizione

fig = uifigure('Position',[100 500 200 150]);
GUIswitch=uiswitch(fig,'toggle',...
    'Items',{'Stop','Start'},...
    'Position',[95 60 20 45],...
    'ValueChangedFcn',@(GUIswitch,event) switchMoved(event)); %si può usare
anche con fig,'toggle'

acquisizione=[];
i=1;
```

3

```
% input per poter avviare l'acquisizione

input('Premere Start su GUI e poi invio per iniziare ad acquisire.');
Status=switchMoved(GUIswitch);
Connection2UR_ON_35(IP);
```

4

```
% apertura di un ciclo while per l'acquisizione dei dati.
% l'acquisizione terminerà solamente quando indicato dall'operatore nella
gui

      while Status == 1
            [vettore_dati] = UR3Stream30003_V6_35();
            acquisizione(i,1:138)=vettore_dati;
            i=i+1;
            pause(0.0000001)
            Status = switchMoved(GUIswitch)
      end

Connection2UR_OFF();
```

5

```
% salvataggio di tutti i dati acquisiti in una structure

prompt = 'inserire nome del file : ';
str = input(prompt,'s');
save(['C:\Users\utente\Desktop\libreria\PROVE\' str],'acquisizione')
```

Queste poche righe di codice sono importanti poiché permettono di collegare il robot al calcolatore e di ottenere i dati. Il programma viene suddiviso in vari blocchi per facilitarne la comprensione:

- **BLOCCO 1,** per prima cosa bisogna indicare il codice IP del robot per poter aprire una connessione socket (o l'IP del simulatore del software fornito da UR);
- **BLOCCO 2,** viene generata una GUI con un pulsante di switch per poter avviare l'acquisizione quando il pulsante è in ON e bloccarla quando il pulsante è in OFF;
- **BLOCCO 3,** viene richiesto di dare un input da tastiera per poter procedere con il programma e quindi entrare nel while che permette l'acquisizione. In questo blocco viene avviata la funzione *"Connection2UR_ON_35.m"* che richiede come input l'IP del robot specificato nel blocco 1.

```
function [] = Connection2UR_ON_35( IP )
%%Chosen the IP by the user the function connect the PC to the UR
with two
%%different sockets, the first for receve datas in realtime and the
second
%%for send datas in realtime
Port_RT1=30003; %real time port 125Hz
global DATA;
DATA=tcpip(IP,Port_RT1,'InputBufferSize',1108); %connection with UR3
as server and matlab as client
fopen(DATA);
end
```

Con questa funzione si crea, tramite la porta 30003 che comunica a 125Hz, il collegamento tra UR3, che funge da server, e il software Matlab, che funge da client;

- **BLOCCO 4,** viene avviato un while che acquisisce i dati finché non viene spento il pulsante generato con la GUI. In questo blocco è presente la funzione "*UR3Stream30003_V6_35.m*" che permette di leggere i dati del robot nel socket in una structure chiamata "DATA".

```
function [vettore_dati] = UR3Stream30003_V6_35()
%% Function for elaborate streaming data from UR3.
global DATA;
s1=fread(DATA,1,'int');%dimension of bytes receved
vettore_dati=fread(DATA,138,'double')';%datas reved
end
```

Viene inserito un "pause" con un tempo piccolo e trascurabile, più piccolo del tempo di acquisizione, per poter considerare durante l'iterazione del while la condizione del tasto nella GUI.

Una volta terminata l'iterazione, quindi quando si esce dal ciclo, si interrompe la connessione socket tra i due componenti mediante la funzione "*Connection2UR_OFF.m*"

```
function [] = Connection2UR_OFF()
global DATA;
fclose(DATA);
end
```

- **BLOCCO 5,** nell'ultimo blocco sono presenti semplicemente dei comandi che salvano tutti i dati acquisiti in una grande structure.

Una volta acquisiti i dati sono stati scritti dei semplici file, quali:

- "*file_estrazione.m*", permette di scompattare la grande struttura acquisita in tante variabili fisiche ed analizzare in questo modo i movimenti del robot;
- *"plot_singoli_tutte_le_grandezze.m",* permette di visualizzare gli andamenti di tutte le grandezze acquisite durante il movimento;
- *"plot_grandezze_importanti.m",* permette di visualizzare alcune grandezze importanti facendo delle sovrapposizioni;
- *"sovrapponi_grafici.m",* permette di sovrapporre gli andamenti delle grandezze principali di una stessa prova ma effettuata con parametri cinematici differenti.

Grazie a questa procedura è stato possibile nel corso dell'elaborato acquisire le diverse informazioni dal robot. Dalla scomposizione del file di acquisizione si possono ottenere le seguenti grandezze:

Tabella 8 : dati acquisibili [4]

nome della variabile	descrizione
Time	tempo
q_target	movimento dei giunti di target
qd_target	velocità dei giunti di target
qdd_target	accelerazione dei giunti di target
I_target	correnti dei giunti di target
M_target	coppie di target
q_actual	movimento dei giunti misurato
qd_actual	velocità dei giunti misurata
I_actual	correnti dei giunti misurate
I_control	correnti di controllo dei giunti
Tool_vector_actual	posizione e orientazione del TCP misurate
TCP_speed_actual	velocità lineari e angolari del TCP misurate
TCP_force	forze generalizzate nel TCP [N] e [Nm]
Tool_vector_target	posizione e orientazione del TCP di target
TCP_speed_target	velocità lineare e angolare del TCP di target
Digital_input_bits	stato corrente dei bit digitali di input (posso dare delle grandezze di input di corrente e tensione): 0-7 standard, 8-15 configurabile, 16-17 tool
Motor_temperatures	temperature dei motori

Controller_timer	controllo real-time del tempo di esecuzione del thread (suddivisione di un processo in due o più sottoprocessi che vengono eseguiti insieme da un sistema di elaborazione mono o multiprocessore
Robot_mode	messaggi: stato del robot
Joint_modes	messaggi: stato dei giunti
Safety_mode	messaggi: stato di sicurezza
Tool_accelerometer_values	valori x, y e z dell'accelerometro del tool
Speed scaling	scala di velocità del limite di traiettoria (da 0 a 1)
Linear_momentum_norm	norma della quantità di moto (cartesiana)
V_main	pannello controllo di sicurezza: tensione principale
V_robot	pannello controllo di sicurezza: tensione del robot 48 [V]
I_robot	pannello controllo sicurezza: corrente del robot
V_actual	tensioni (volt) dei giunti misurate
Digital_outputs	Stato corrente dei bit digitali di output (posso ricevere delle grandezze di output di corrente e tensione da eventuali sensori): 0-7: Standard, 8-15: Configurabile, 16-17: Tool
Programm_state	stato del programma (variabile legata al play, stop e pause del pad)
Elbow_position	posizione gomito
Elbow_velocity	velocità gomito

Sfruttando l'acquisizione dati è stato possibile spiegare il funzionamento di alcuni comandi di base illustrati nel paragrafo successivo e convalidare il modello del robot creato e descritto invece nei capitoli successivi.

1.3 Ambiente di programmazione

1.3.1 Funzioni di base del robot

Per poter scrivere un semplice programma che permetta al robot di svolgere determinate operazioni bisogna approfondire la conoscenza dei movimenti di base che la macchina può compiere.

Nel momento in cui viene acceso il software principale, compare la seguente schermata in cui è possibile selezionare diverse operazioni:

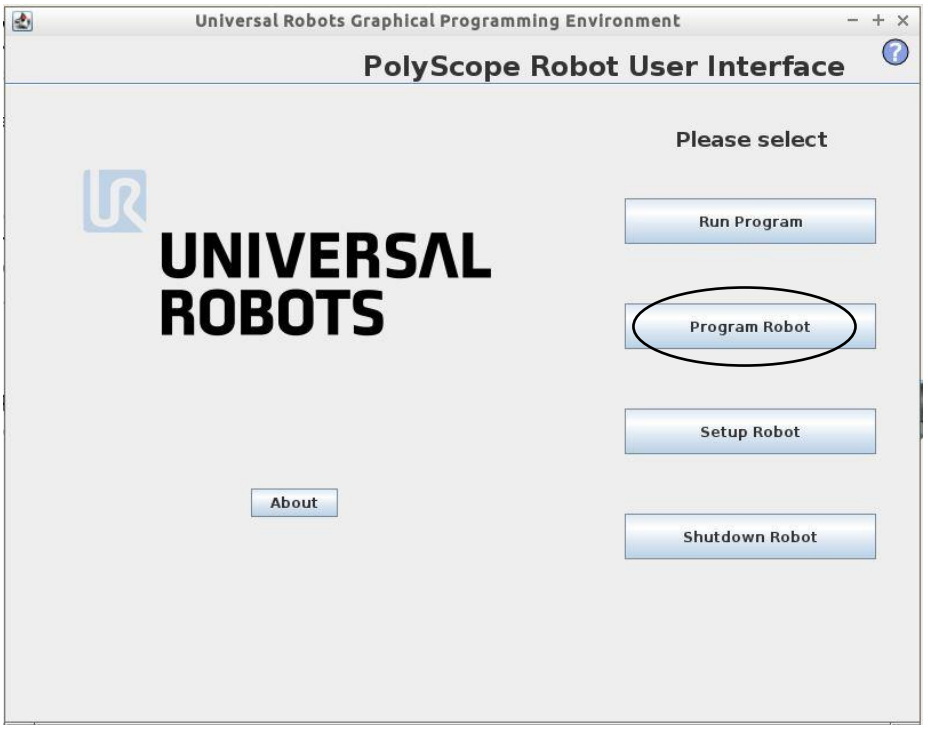


Figura 16: STEP1 schermata iniziale PolyScope

- Selezionando *Run Program* si può far muovere il robot senza alcuna programmazione;
- Selezionando *Program Robot* si può creare un programma per farlo muovere;
- Selezionando *Setup Robot* è possibile entrare nelle impostazioni di setup;
- Selezionando *Shutdown Robot* si spegne la macchina.

Per poter dunque creare un piccolo programma si digiti Program Robot.

Figura 17: STEP 2 realizzare un programma da zero

A questo punto è possibile caricare un altro programma già precedentemente creato e salvato o realizzarne di nuovi utilizzando il template per il *pick and place* o partendo totalmente da zero.

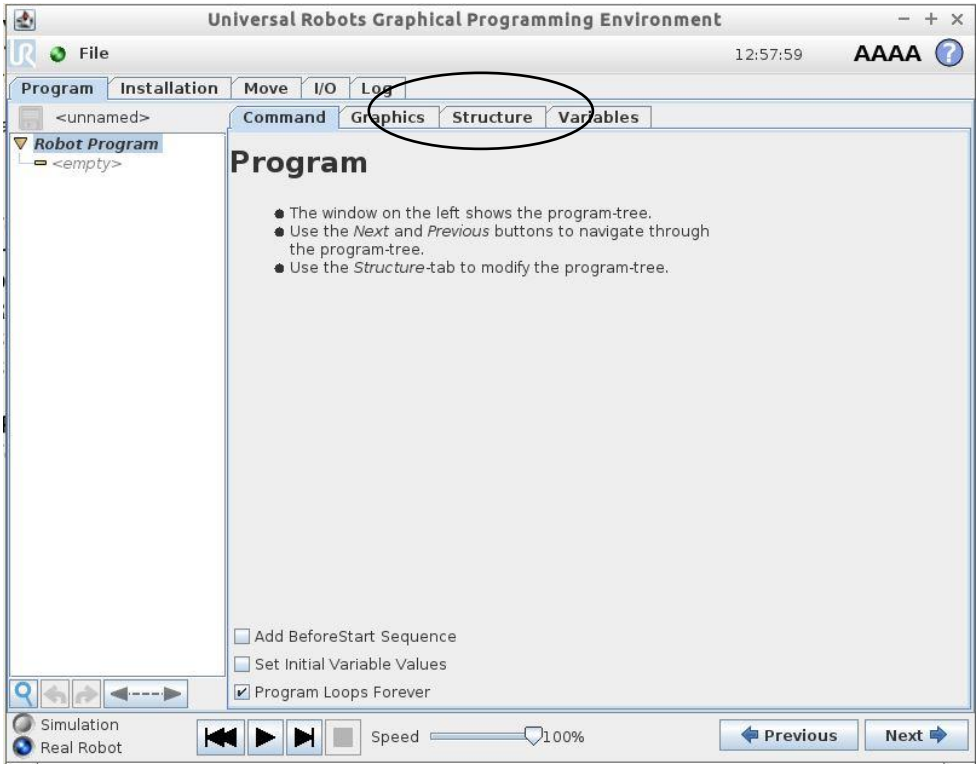


Figura 18: STEP 3 inserire nel programma un comando

A questo punto compare la pagina del programma da compilare. Premendo Structure è possibile visualizzare la pagina dei comandi.

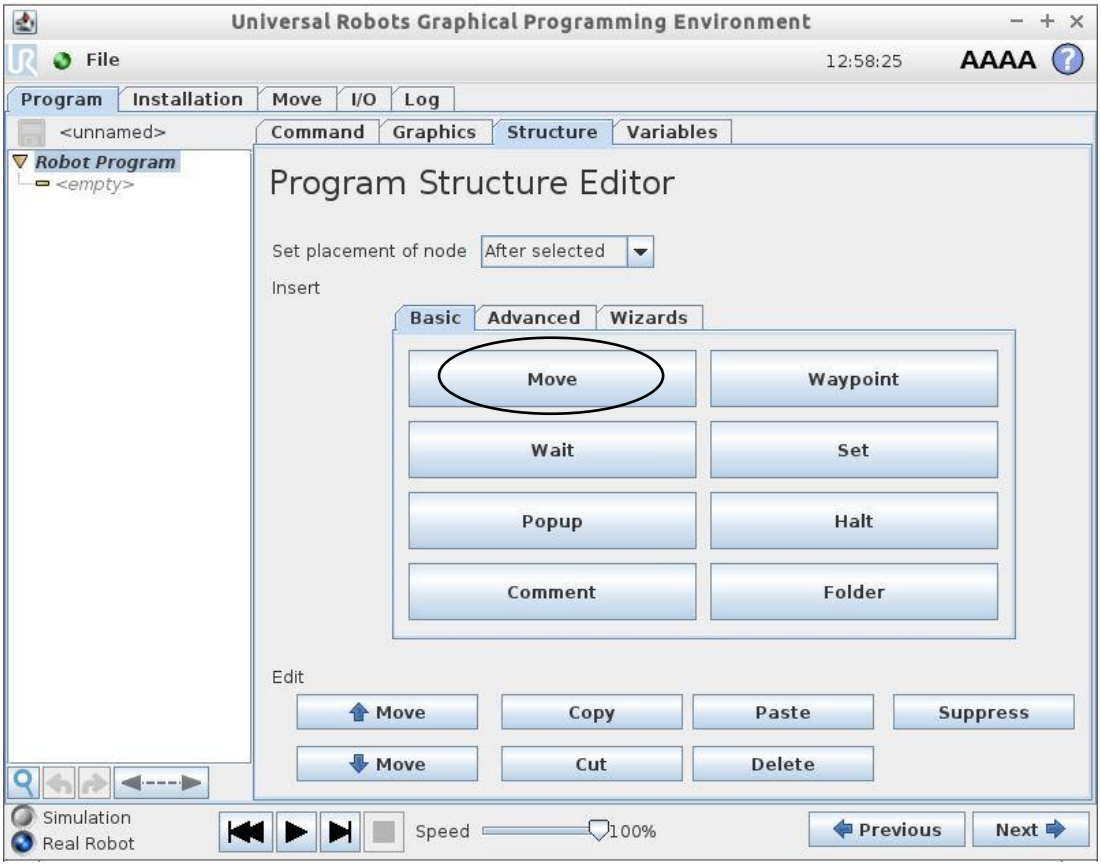


Figura 19: STEP 4 selezionare il comando Move

Si possono trovare diversi comandi da dare al robot. Per adesso si consideri soltanto l'uso di quelli base [4]:

- *MOVE* permette di introdurre nel programma un movimento che può essere MoveJ, MoveL o MoveP;
- *WAYPOINT* permette di introdurre nel programma un punto di riferimento da raggiungere;
- *WAIT* permette di far attendere il robot in una certa posizione per un determinato intervallo di tempo oppure un determinato segnale proveniente dai sensori;
- *SET* permette di poter introdurre determinati output digitali, analogici o variabili;
- *POPUP* ferma il movimento e permette di aprire una finestra di dialogo e fare una scelta. Si può riprendere l'operazione oppure fermarla;
- *HALT* permette di fermare l'esecuzione del programma;
- *COMMENT* permette di fare dei commenti nel programma per qualche movimento o azione;
- *FOLDER* permette di raggruppare in una cartella determinati movimenti.

Si è scelta l'opzione ***Move.***

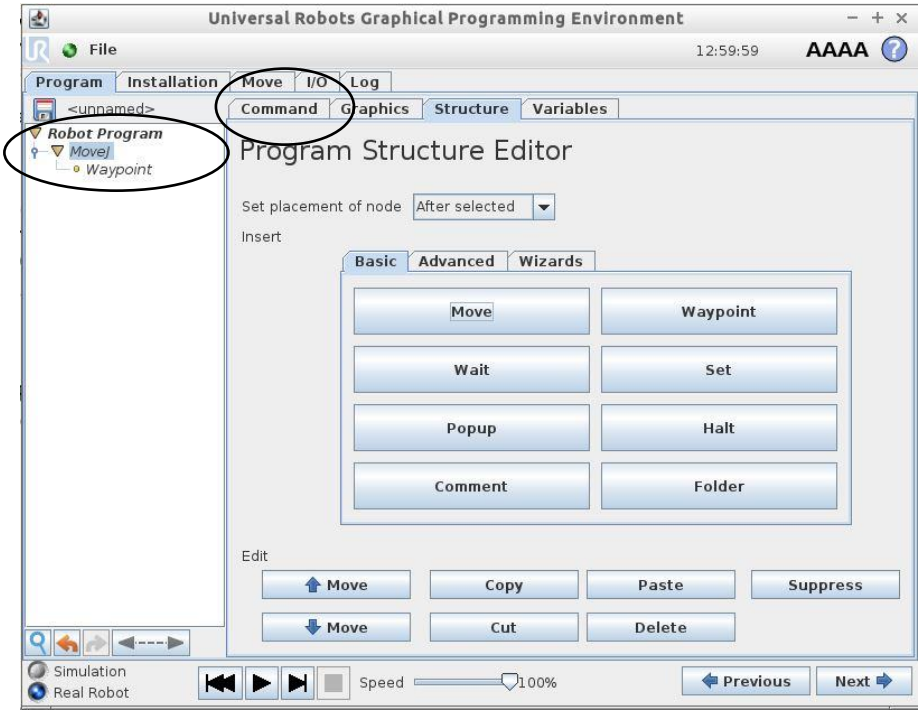


Figura 20: STEP 5 visualizzare la pagina delle impostazioni del comando

A questo punto apparirà nel programma un "Move". Sarà possibile impostare e variare le sue caratteristiche digitando la sezione *command*.

A questo punto sono stati analizzati con una certa cura i vari movimenti sfruttando l'acquisizione dati illustrata precedentemente.

MoveJ

Il primo movimento analizzato è MoveJ. È possibile impostare le sue proprietà dalla seguente pagina:

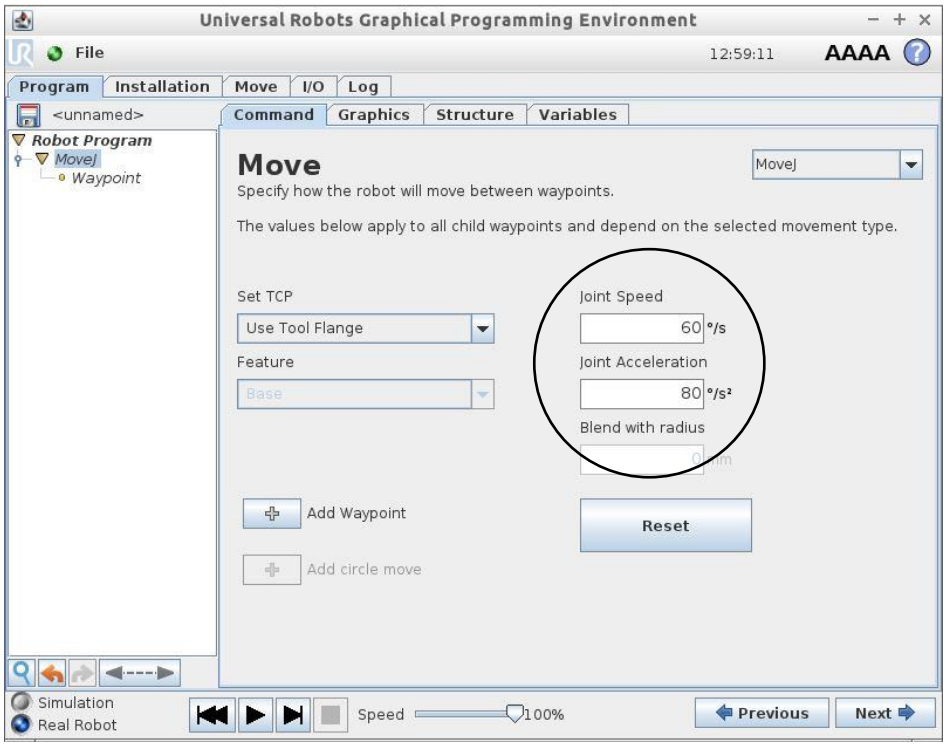


Figura 21: schermata MoveJ

Questo movimento permette di poter raggiungere una determinata posizione di target attuando i giunti attraverso delle leggi trapezoidali di velocità. È possibile imporre una velocità massima da raggiungere e l'inclinazione del tratto obliquo del trapezio tramite l'accelerazione.

Alla base di questo movimento vi sono i seguenti vincoli da rispettare, tipici di un movimento pianificato con legge di velocità di tipo trapezoidale:

- Vincoli di traiettoria:
  - Posizione iniziale e finale;
  - Velocità iniziale e finale pari a 0;
  - Assunzione della simmetria del profilo.
- Vincolo addizionale, velocità di crociera
  - Range di velocità

$$velocity\ range: \frac{|q_f - q_i|}{t_f} < |\dot{q}_c| \leq \frac{2|q_f - q_i|}{t_f} \rightarrow$$

$$t_c = \frac{q_i - q_f + \dot{q}_c t_f}{\dot{q}_c}\ and\ \ddot{q}_c = \frac{\dot{q}_c^2}{q_i - q_f + \dot{q}_c t_f}$$

posizione iniziale	posizione finale	velocità di crociera	accelerazione di crociera	tempo di accelerazione	tempo totale
q_i	q_f	$\dot{q}_c$	q_c	t_c	t_f

La velocità che prende in input tale comando (nella sezione command osservato in figura 21) è quella media di crociera calcolata tra la velocità massima e minima del range per le leggi di tipo trapezoidali. Scelto un tempo di crociera e delle posizioni iniziali e finali, si verifica infatti che il robot compie il movimento in quel tempo prestabilito, comandato con quei specifici valori cinematici.

Sono state svolte alcune prove cambiando i vari parametri per effettuare queste verifiche. Di seguito viene specificato il nome di ogni acquisizione, riportata negli indici in appendice dove viene specificato, in un primo indice a quale programma scritto corrisponde quella determinata acquisizione, e in un secondo a cosa è servita quella specifica acquisizione.

- **Prova 1 (***ACQUISIZIONE_moveJ_prova_1.mat***)**

Tabella 9: proprietà della prima prova MoveJ

q_i del giunto 1	q_f del giunto 1	18°/s < $\dot{q}$ < 36°/s con un valore medio	Un'accelerazione $\ddot{q}$
0°	90°	27°/s	16.2°/s^2

Calcolando il t_f dalla formula dell'accelerazione risulta che il movimento dovrebbe avvenire in 5s.

Dal grafico ottenuto grazie all'acquisizione si può osservare il risultato dei valori attesi.

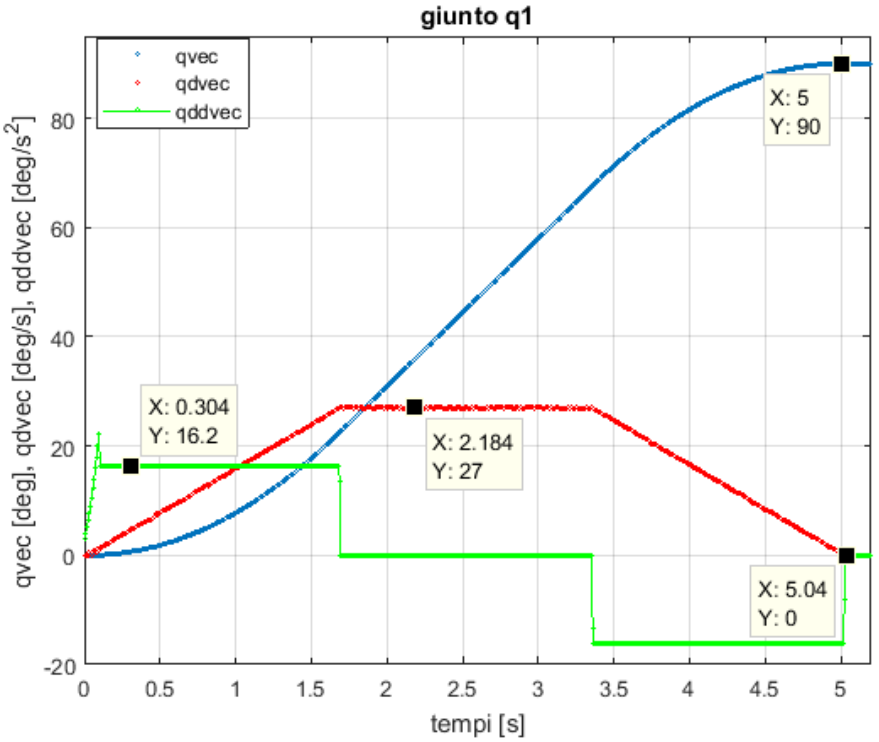


Figura 22: qvec actual, qdvec actual, qddvec target

- **Prova 2** (*ACQUISIZIONE_moveJ_prova_2.mat*)

È possibile inoltre effettuare il movimento modificando le impostazioni presenti nella pagina del waypoint impostato al di sotto del Move corrispondente:

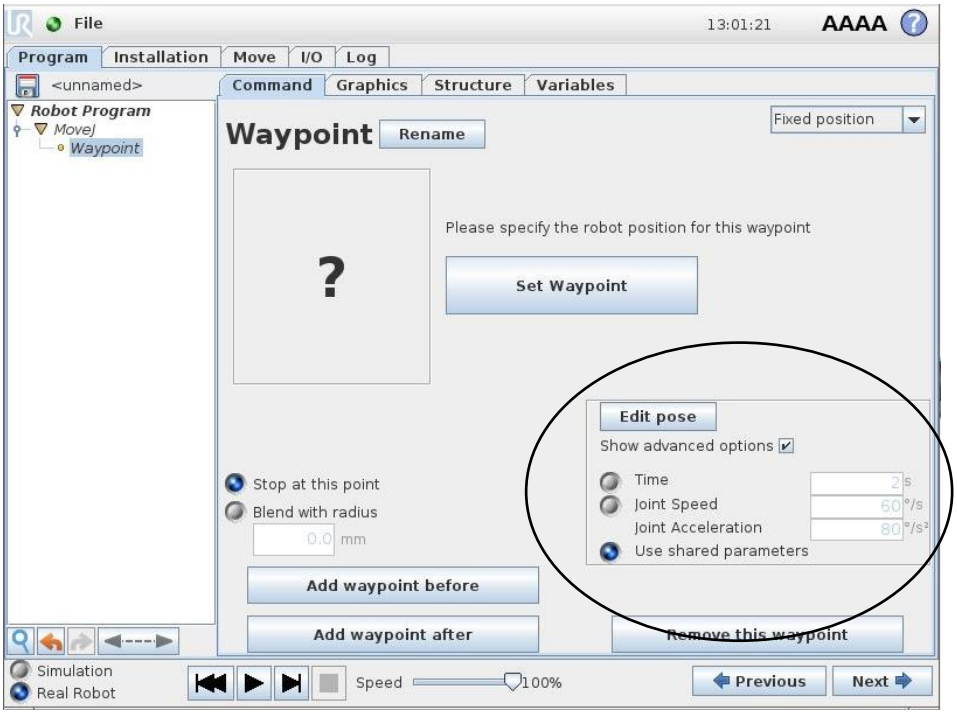


Figura 23: pagina del waypoint

È possibile far muovere il robot non solo inserendo i valori di velocità e accelerazione, ma anche solo impostando un determinato tempo entro la quale si desidera che avvenga il movimento.

Tabella 10: proprietà seconda prova MoveJ

q_i del giunto 1	q_f del giunto 1	tempo t_f [s]
0°	90°	3

Si è verificato che la velocità e l'accelerazione con cui si muove il robot sono calcolate dal software dell'UR3 come visto in precedenza e che come velocità il robot utilizza il valore medio del limite del range.

Dalle formule precedenti utilizzando t=3s i valori calcolati sono:

- $30°/s < \dot{q}_c < 60°/s$ con un valore medio di **45°/s**
- Un'accelerazione $\ddot{q}_c$ pari a **45°/s²**

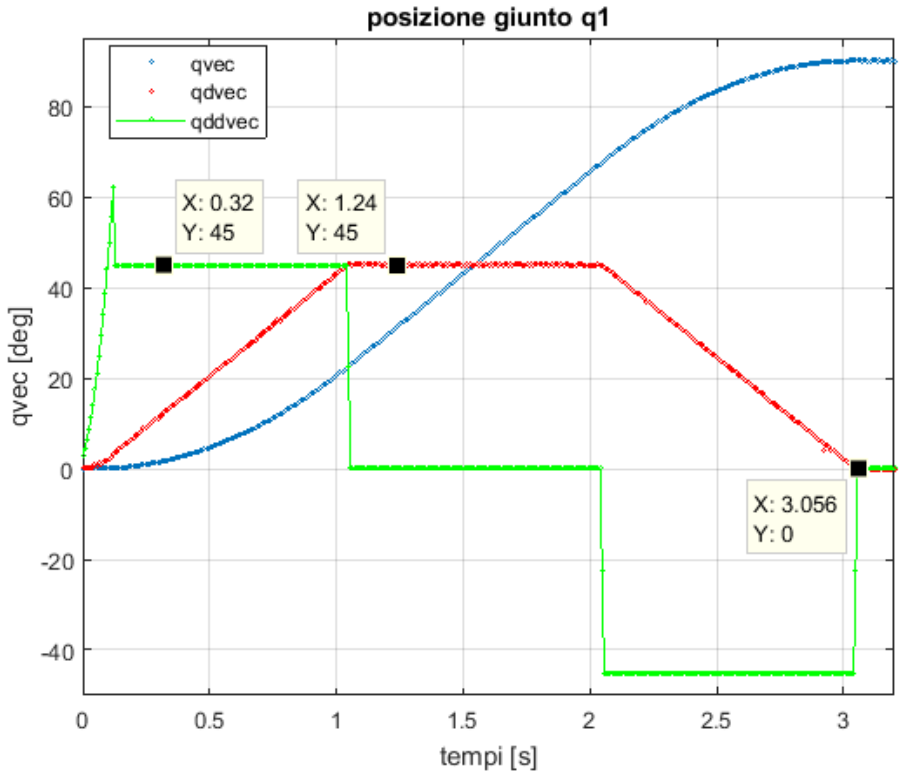


Figura 24: qvec actual, qdvec actual, qddvec target

- **Prova 3.1** (*ACQUISIZIONE_moveJ_prova_3.1.mat*)

Se si inserisce un MoveJ nel programma, e in questo sono compresi più di un waypoint, è possibile:

1 far compiere tutti i movimenti con gli stessi parametri di velocità e accelerazione;
2 far compiere i movimenti con parametri diversi.

Per poter raggiungere tutti i waypoint con le stesse caratteristiche cinematiche basta impostare il comando "USE SHARED PARAMETERS". In questo modo i movimenti vengono effettuati con i valori che si inseriscono nel Movej (come nella prima prova).

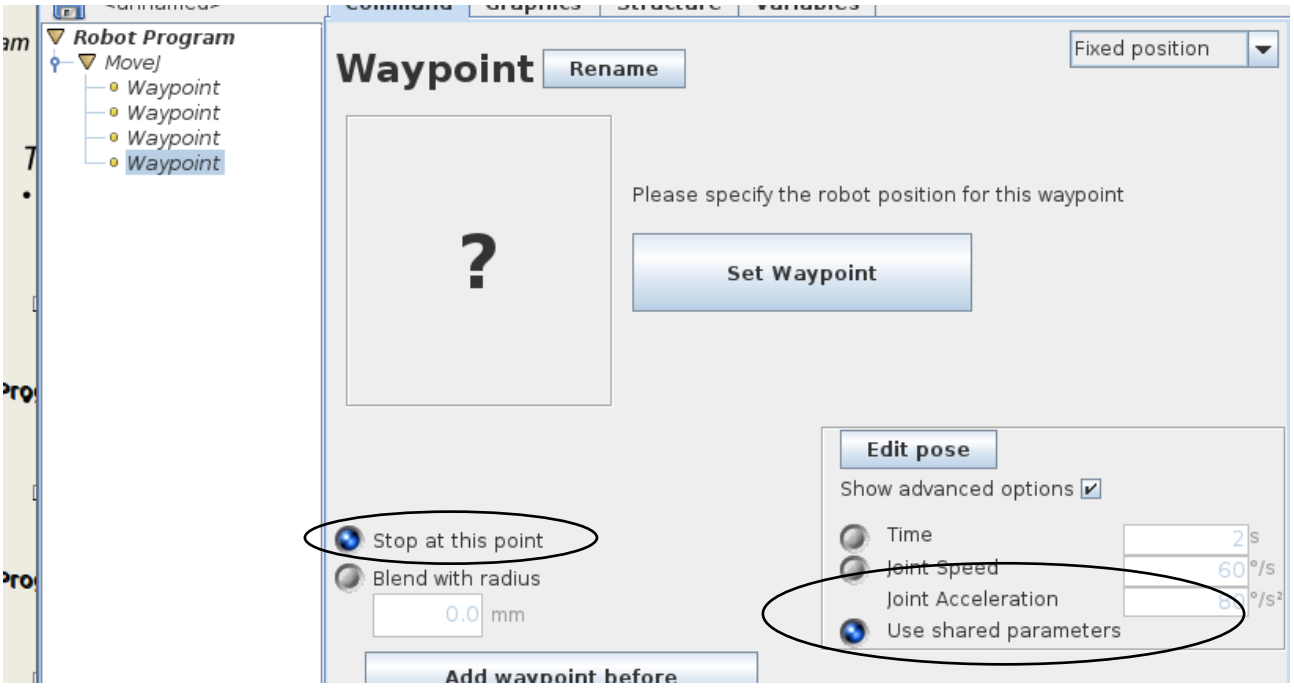


Figura 25: pagina del waypoint, parametri da impostare

Questa prima prova è stata effettuata attuando il primo giunto inizialmente da 0° a 90° e poi da 90° a 180° con valori di velocità e accelerazione uguali ai precedenti:

- $18°/s < \dot{q}_c < 36°/s$ con un valore medio di **27°/s**;
- Un'accelerazione $\ddot{q}_c$ pari a **16.2°/s²**;
- t_f quindi di 5s per ogni movimento.

Dalle acquisizioni emerge che se non viene impostato un certo valore di "BLEND WITH RADIUS" la transizione tra un movimento e l'altro risulta istantanea, ovvero il robot non transita da una posizione all'altra in modo graduale ma istantaneamente si ferma e poi riparte. In questo modo, come è possibile osservare dalla figura 28, si possono individuare due trapezi di velocità.

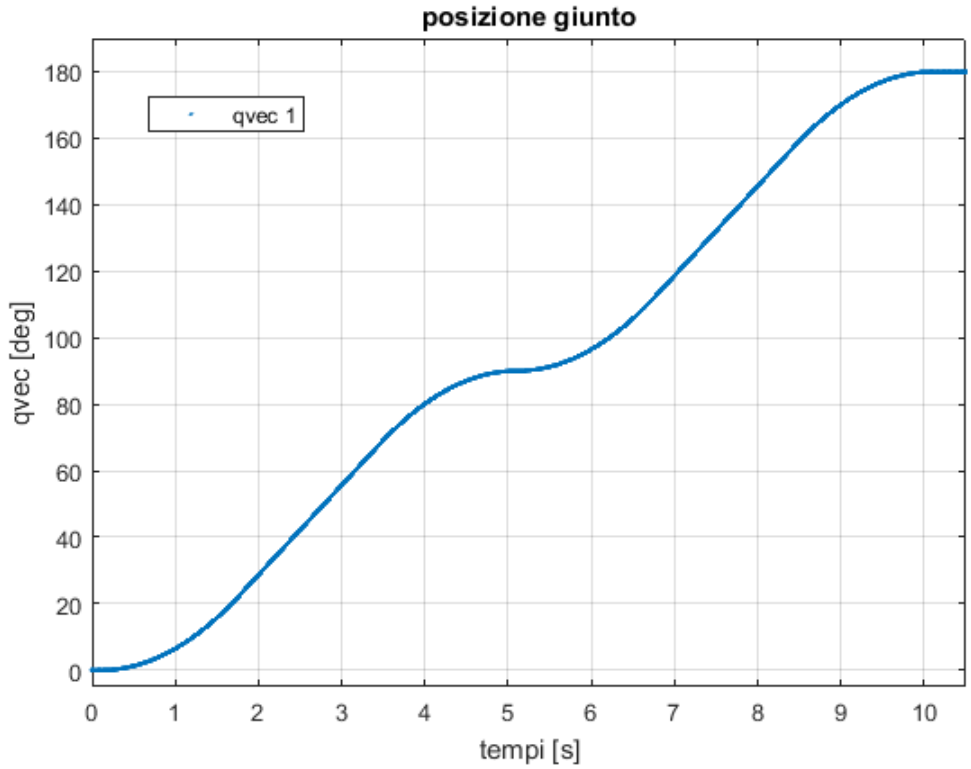


Figura 26: qvec actual giunto 1

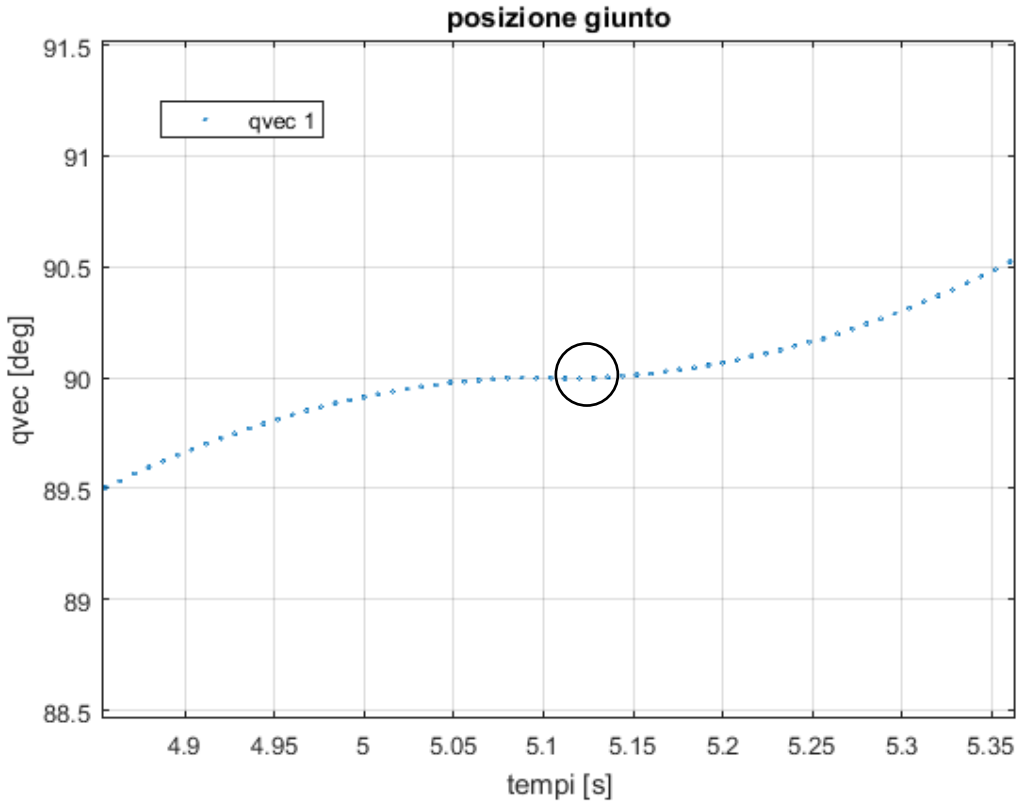


Figura 27: ingrandimento sul qvec, punti nella zona di transizione senza impostare alcun raggio di raccordo

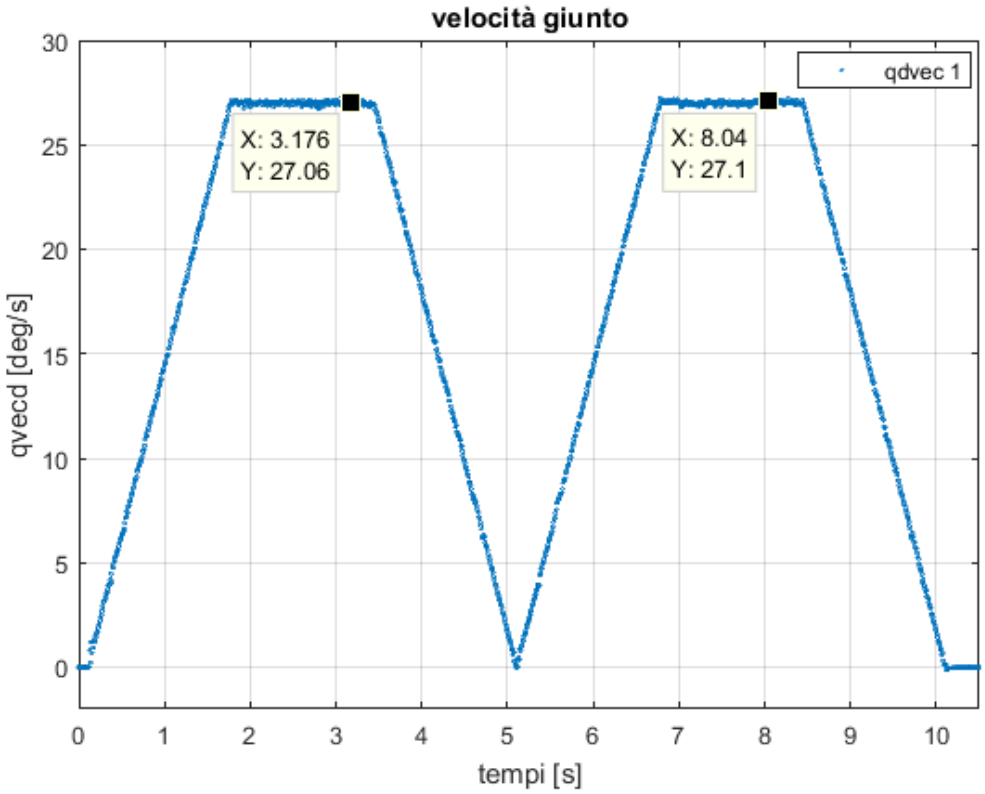


Figura 28: qdvec actual giunto 1

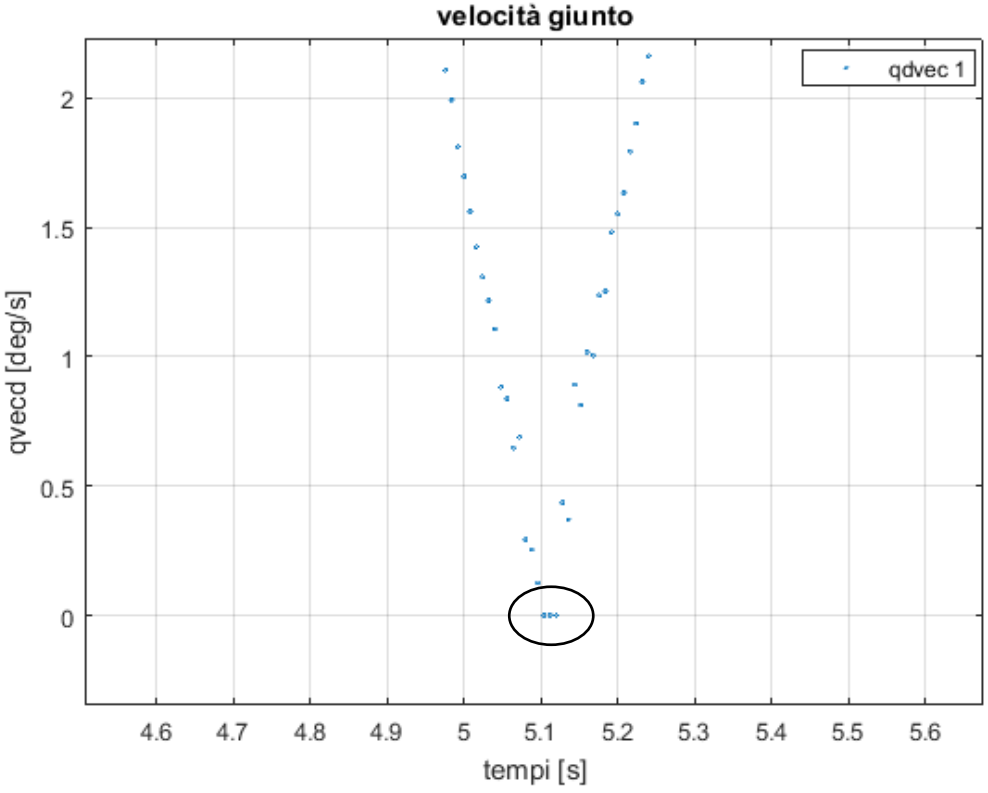


Figura 29: punti zona di transizione senza raggio di raccordo

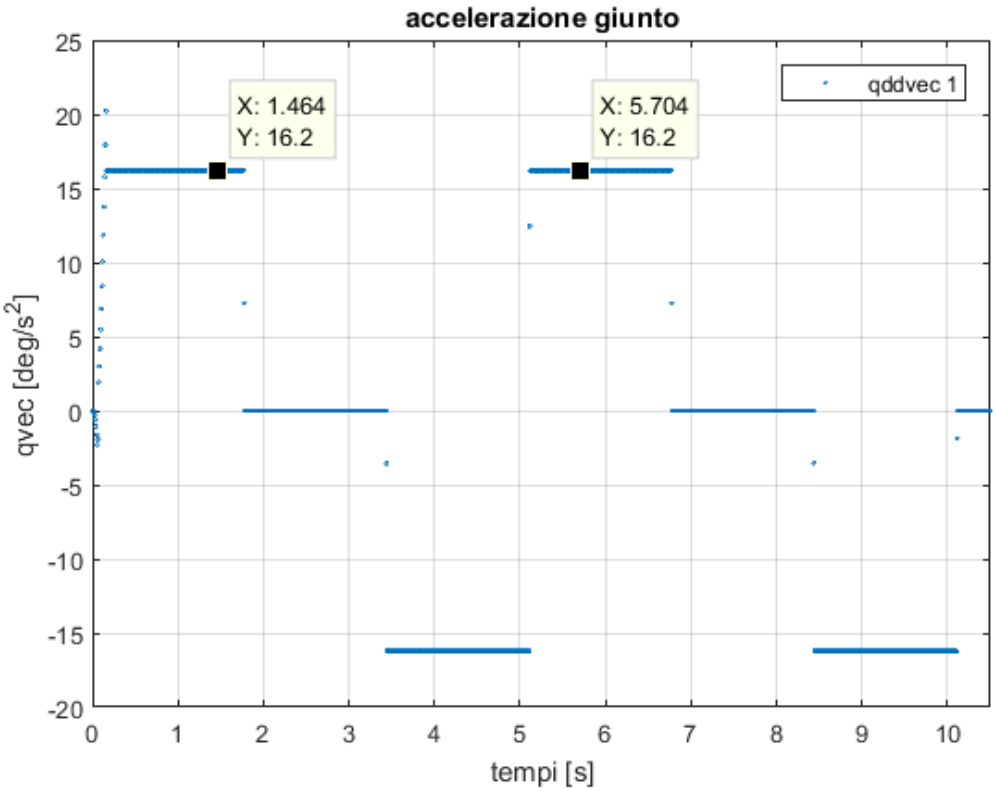


Figura 30: qddvec target giunto 1

- **Prova 3.2** (ACQUISIZIONE_moveJ_prova_3.2.mat)

È stata effettuata un ulteriore prova sfruttando le impostazioni avanzate e andando prima da 0° a 90° e poi da 90° a 180° con valori di velocità e accelerazione questa volta diversi in particolare:

NEL PRIMO TRATTO

- 18°/s < $\dot{q}_c$ < 36°/s con un valore medio di **27°/s**
- Un'accelerazione $\ddot{q}_c$ pari a **16.2°/s²**
- t_f quindi di 5s (non inserito come input ma calcolato offline)

NEL SECONDO TRATTO

- 30°/s < $\dot{q}_c$ < 60°/s con un valore medio di **45°/s**
- Un'accelerazione $\ddot{q}_c$ pari a **45°/s²**
- t_s quindi di 3s (non inserito come input ma calcolato offline)

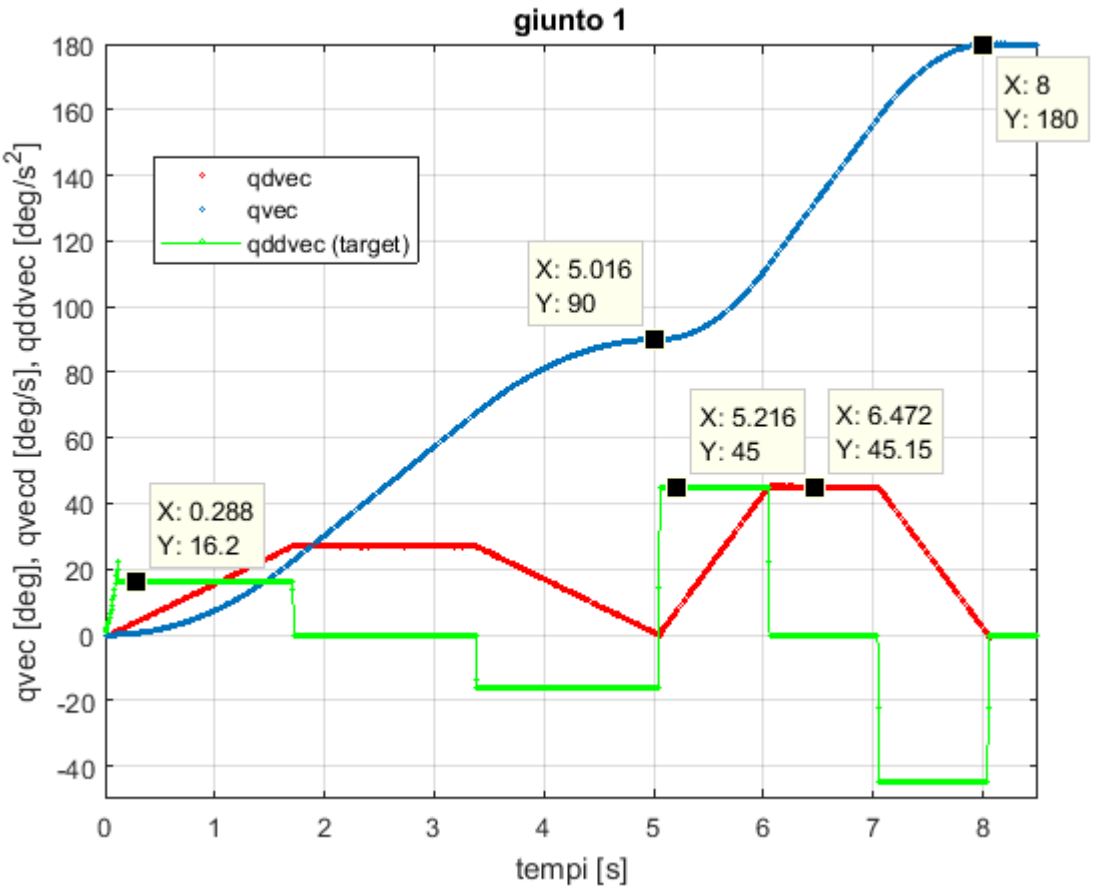


Figura 31: prova con diversi tempi

- **Prova 4** (*ACQUISIZIONE_moveJ_prova_4.mat*)

Questa prova effettuata con il Movej consiste nel capire, nel momento in cui sono attuati più giunti per raggiungere un waypoint, i parametri di velocità e accelerazione impostati a chi si riferiscono.
Si è deciso di far muovere insieme al giunto 1 anche i giunti 3 e 4, facendogli però ricoprire distanze più brevi. È stato impostato:

- il giunto uno si sposta da 0° a 90°;
- il giunto tre si sposta da 0° a 60°;
- il giunto quattro si sposta da 0° a 30°;
- la velocità impostata come shared parameters è di 27°/s;
- l'accelerazione impostata come shared parameters è di 16.2°/s^2.

Si osserva che a muoversi con quei parametri di velocità ed accelerazione è solo il giunto che deve spostarsi di più, ovvero in questo caso il giunto 1. Gli altri due tendono a muoversi sempre nello stesso tempo (5s) con velocità e accelerazioni che il robot calcola come nel primo punto.

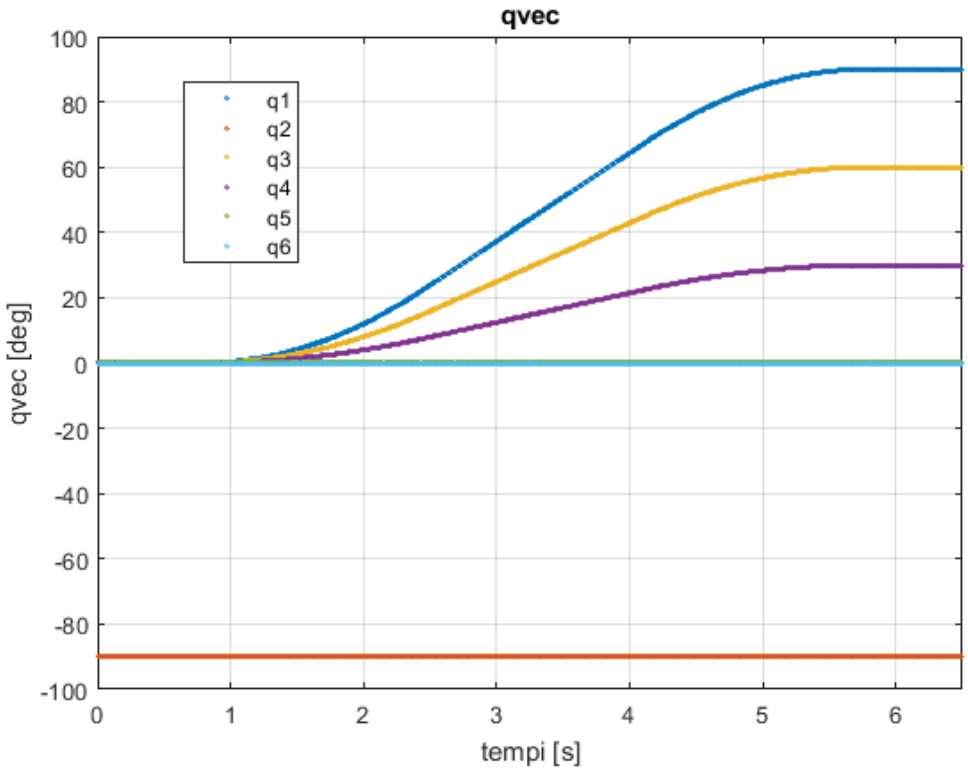


Figura 32: qvec actual

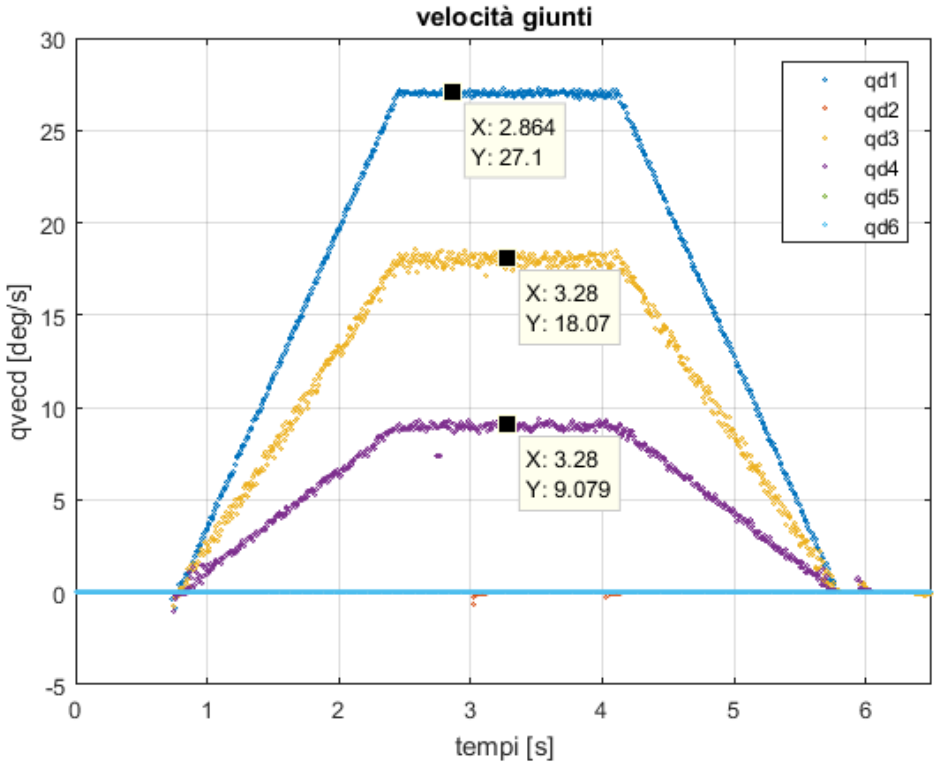


Figura 33: qdvec actual

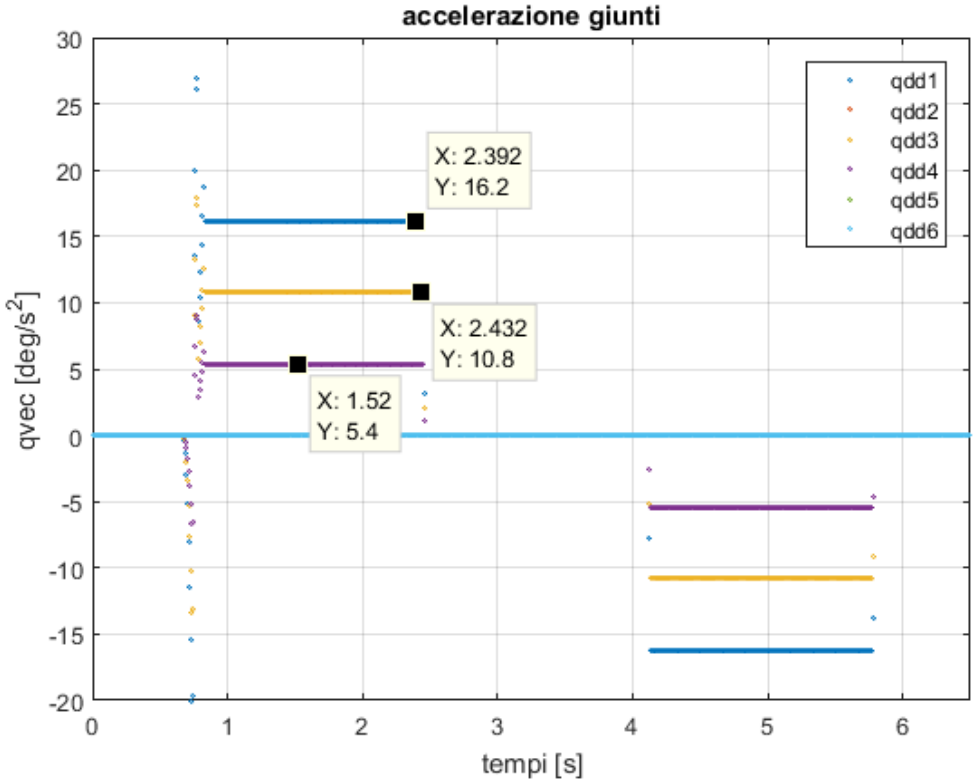


Figura 34: qddvec target

- **Prova 5** (*ACQUISIZIONE_moveJ_prova_5.mat*)

Si può inoltre impostare un movimento con dei waypoint relativi. Si possono considerare due specifiche configurazioni e con questo tipo di comando si vanno a misurare la differenza di posizione e di orientazione del TCP tra le due. Tali grandezze poi verranno utilizzate per poter giungere, a partire da un generico waypoint, una configurazione successiva in cui il TCP dista rispetto al punto precedente degli stessi valori impostati con le due configurazioni iniziali. Ovviamente se tale processo viene ripetuto più volte si rischia di uscire fuori traiettoria.

MoveL

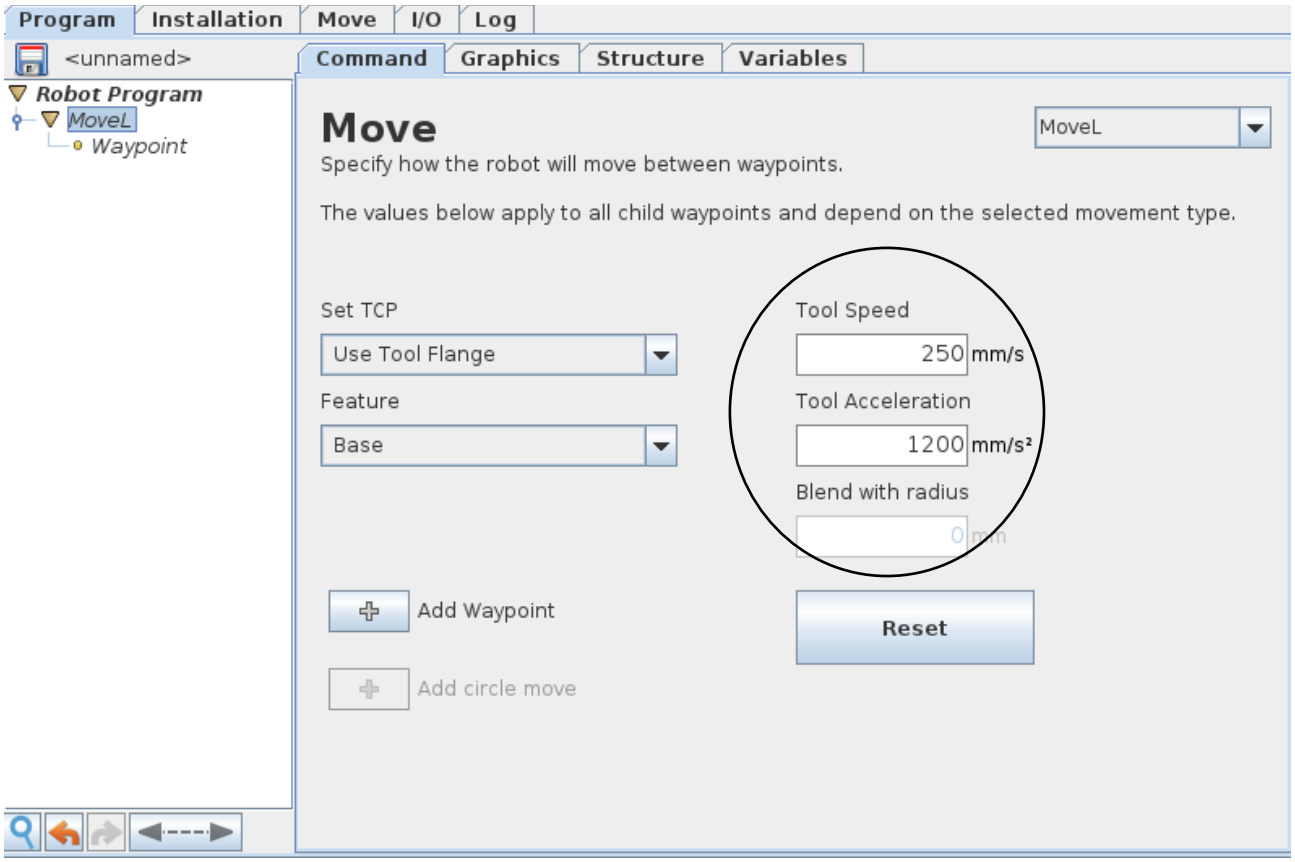


Figura 35: schermata MoveL

Con il MoveL è possibile impostare un movimento nello spazio operativo. I parametri di input richiesti sono la velocità e l'accelerazione del tool. Il tipo di movimento che viene attuato dal robot è uno spostamento lineare nello spazio operativo ovviamente non sempre possibile. Può accadere infatti che durante questo spostamento il robot passi per delle configurazioni vicine a quelle di singolarità e si blocca.

Se si hanno due movimenti successivi in MoveL il robot nel punto di transizione tende a fermarsi e ad assumere una velocità nulla. Si può impostare un raggio di raccordo cosi come nel MoveJ per poter garantire una transizione meno brusca ma comunque a velocità non costante.

- **Prova 1** (*ACQUISIZIONE_moveL_prova_1.mat*)

Sono stati effettuati due movimenti in MoveL senza impostare un raggio di transizione.

Si osserva soprattutto nel grafico delle velocità (fig. 38 e 39) come nel momento della transizione si abbia una certa discontinuità con il valore di velocità che tende a zero e poi risale al valore impostato da input.

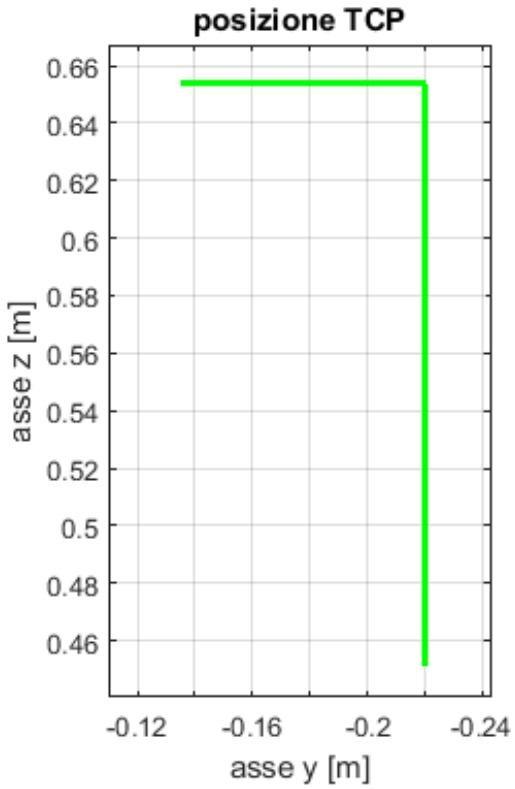


Figura 36: percorso TCP

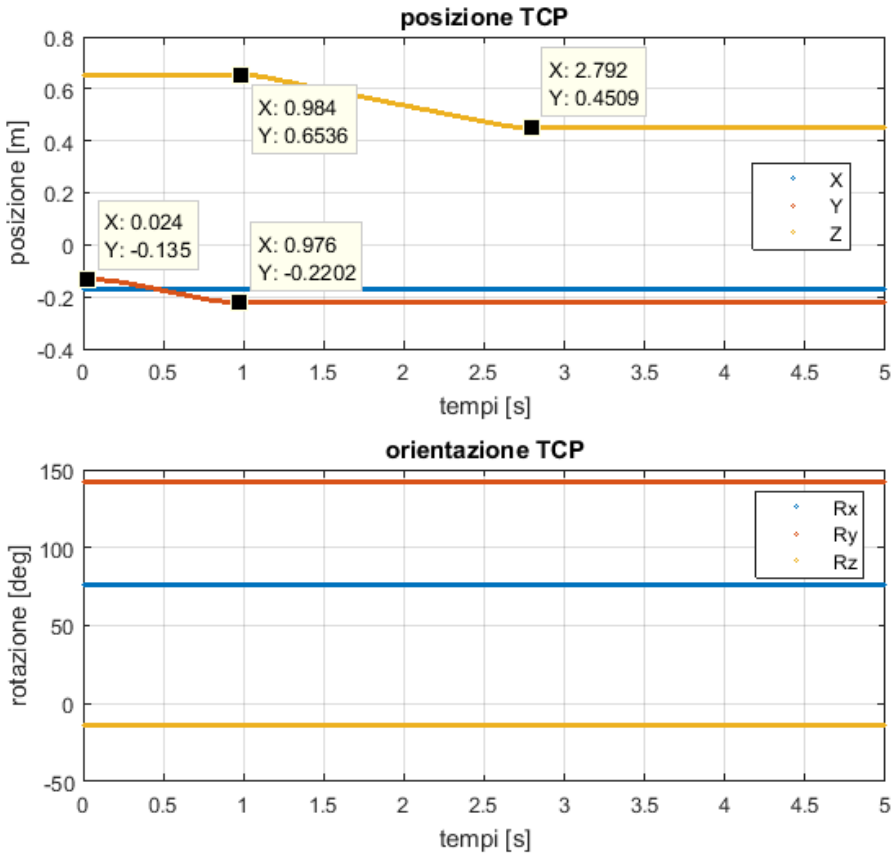


Figura 37: posizione e orientazione del TCP

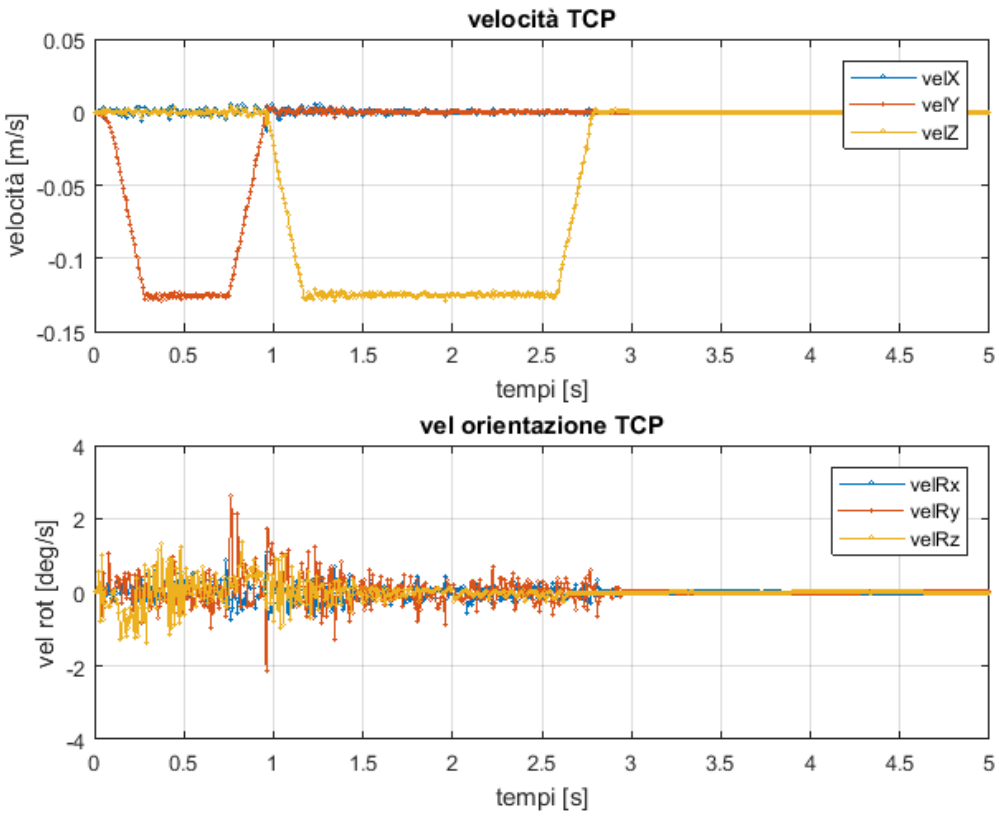


Figura 38: velocità lineare e angolare del TCP

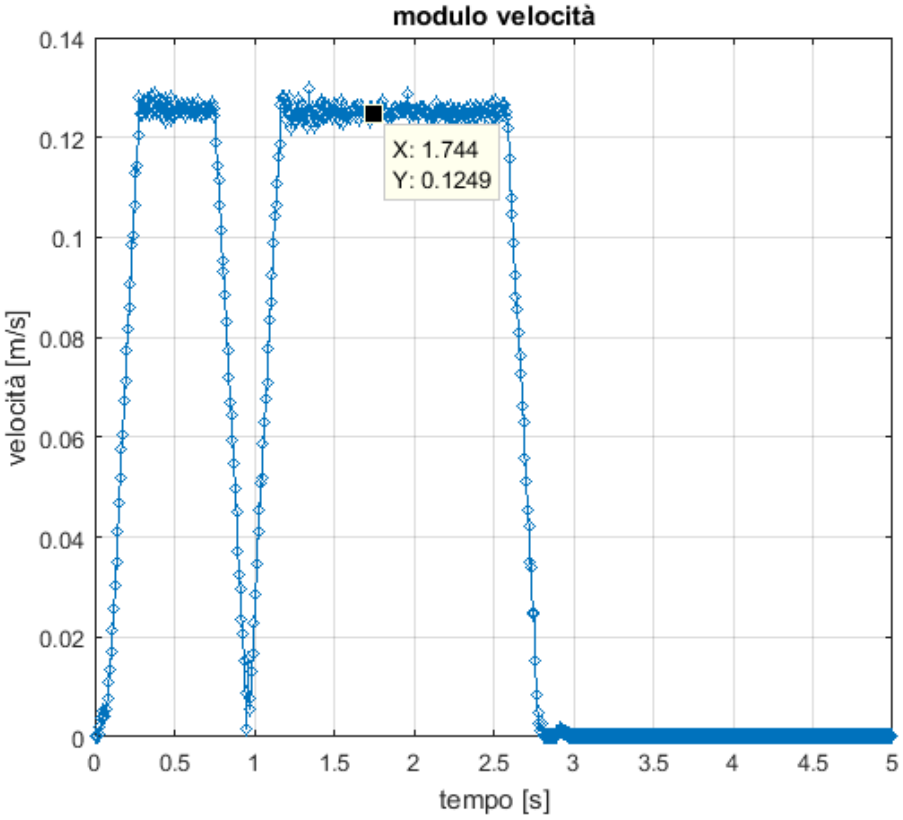


Figura 39: modulo velocità lineare del TCP

- **Prova 2** (*ACQUISIZIONE_moveL_prova_2.mat*)

È stato effettuato lo stesso movimento della prova 1, questa volta impostando un raggio di transizione pari a 25mm. Si osserva in questo caso che la transizione è molto più dolce e, dal grafico in fig. 43, la velocità diminuisce ma di poco e non tende a zero. Questo implica dal punto di vista dinamico una minore sollecitazione inerziale.

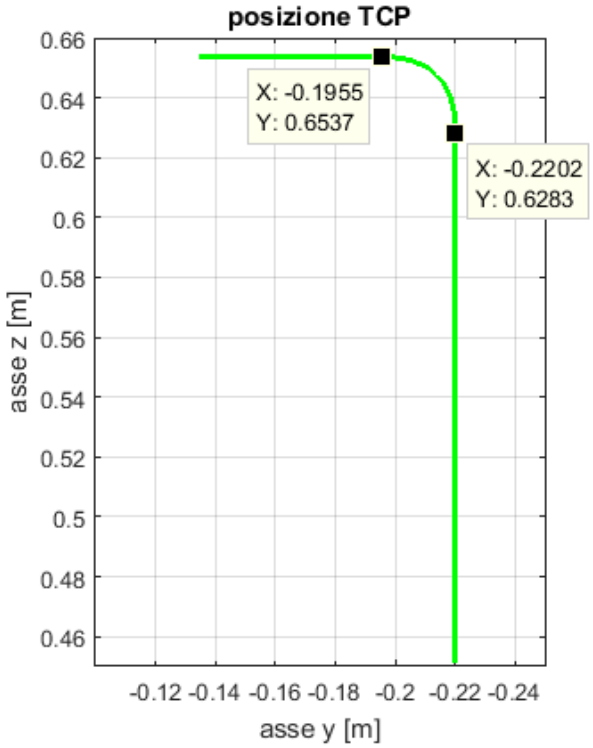


Figura 40: percorso TCP con raggio di raccordo durante la transizione

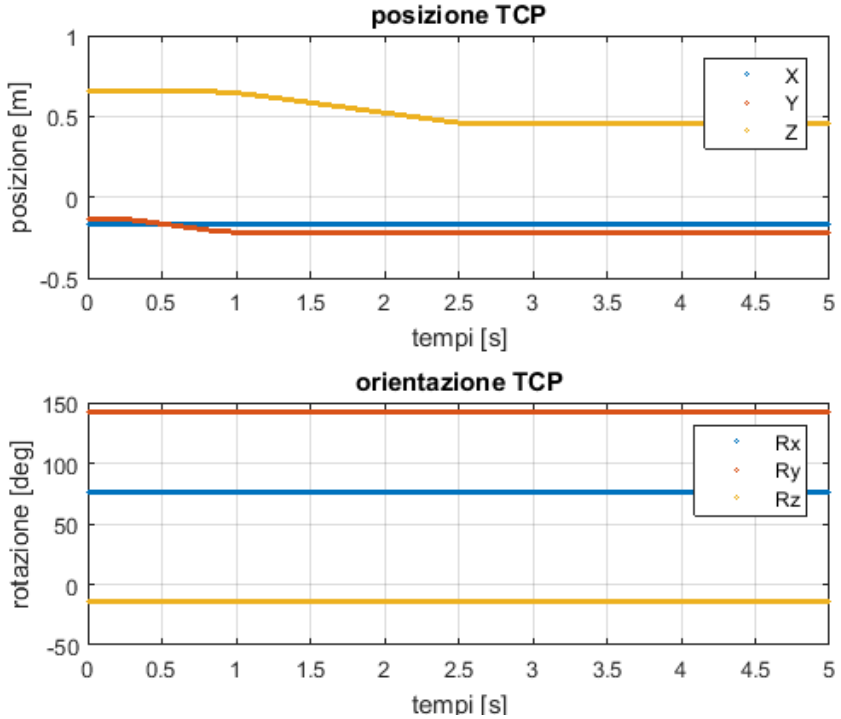


Figura 41: posizione e orientazione del TCP

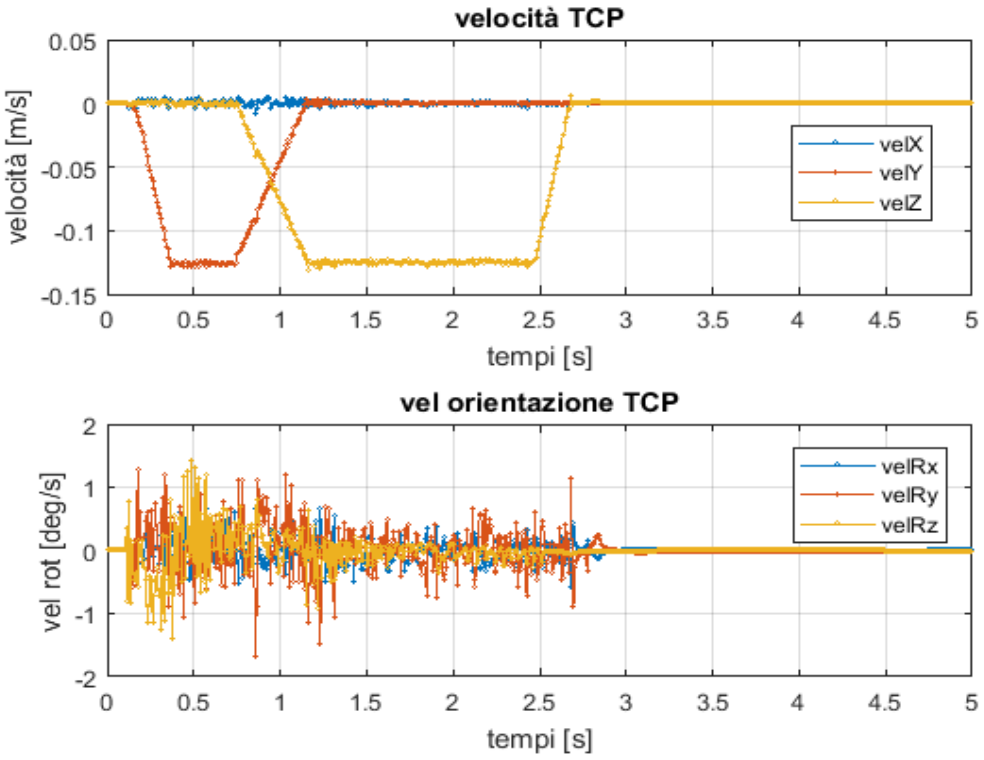


Figura 42: velocità lineare e angolare del TCP

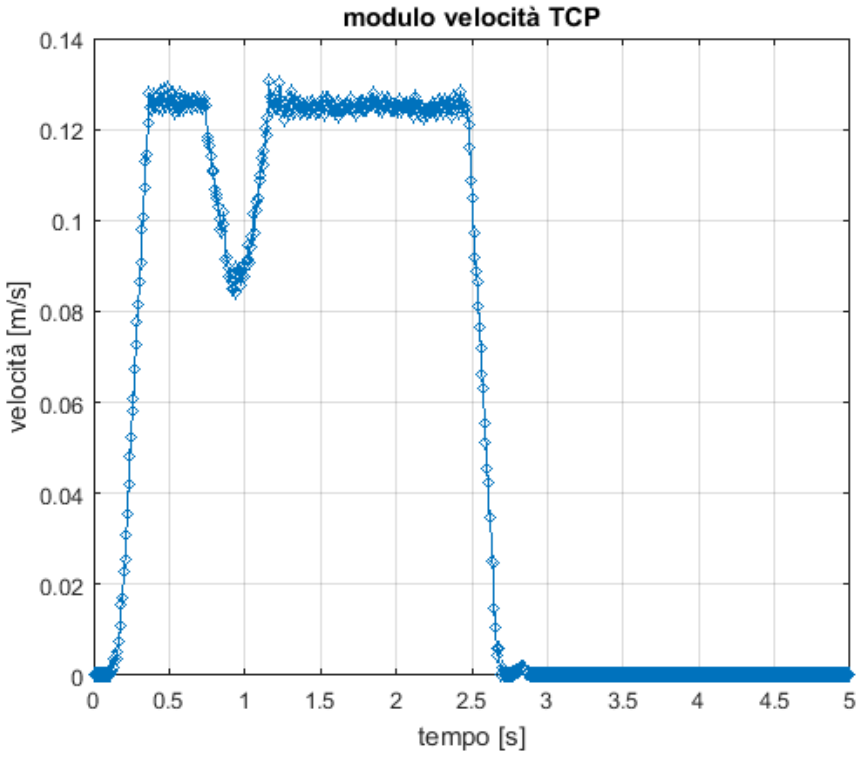


Figura 43: modulo velocità lineare del TCP

MoveP

Con questo comando è possibile ottenere un movimento lineare nello spazio operativo impostando un certo raggio di transizione ma mantenendo costante la velocità del TCP durante tutto il movimento.

- **Prova 1** (*ACQUISIZIONE_moveP_prova_1.mat*)

Si decide di effettuare la stessa traiettoria e utilizzare gli stessi parametri della prova due del MoveL.

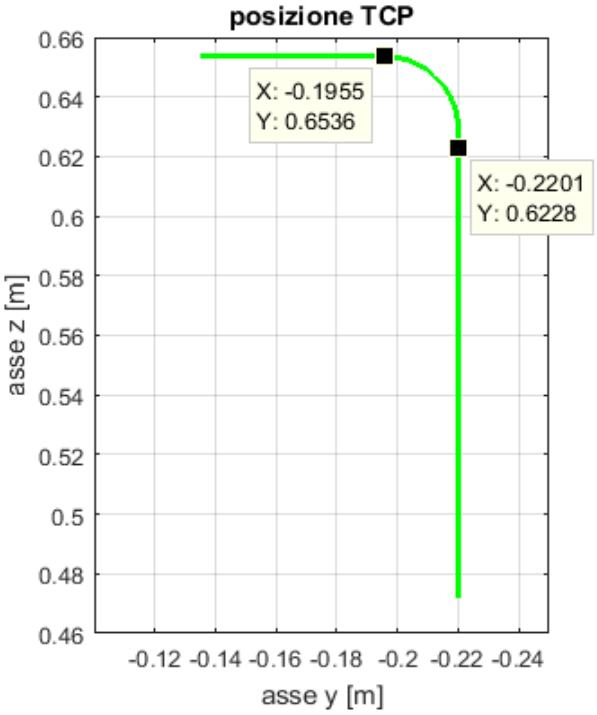


Figura 44: percorso del TCP

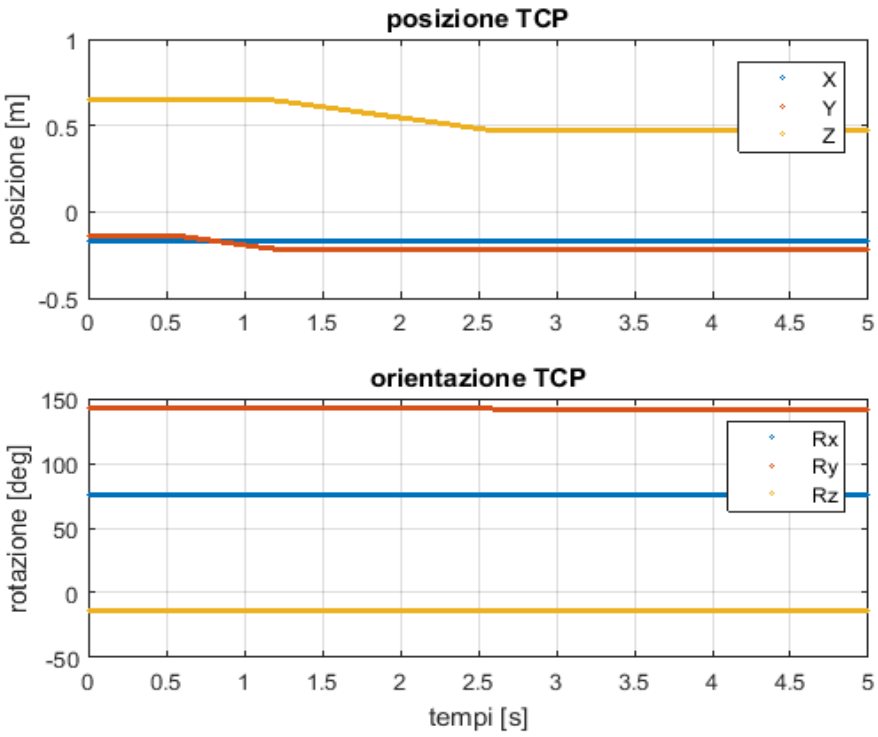


Figura 45: posizione e orientazione del TCP

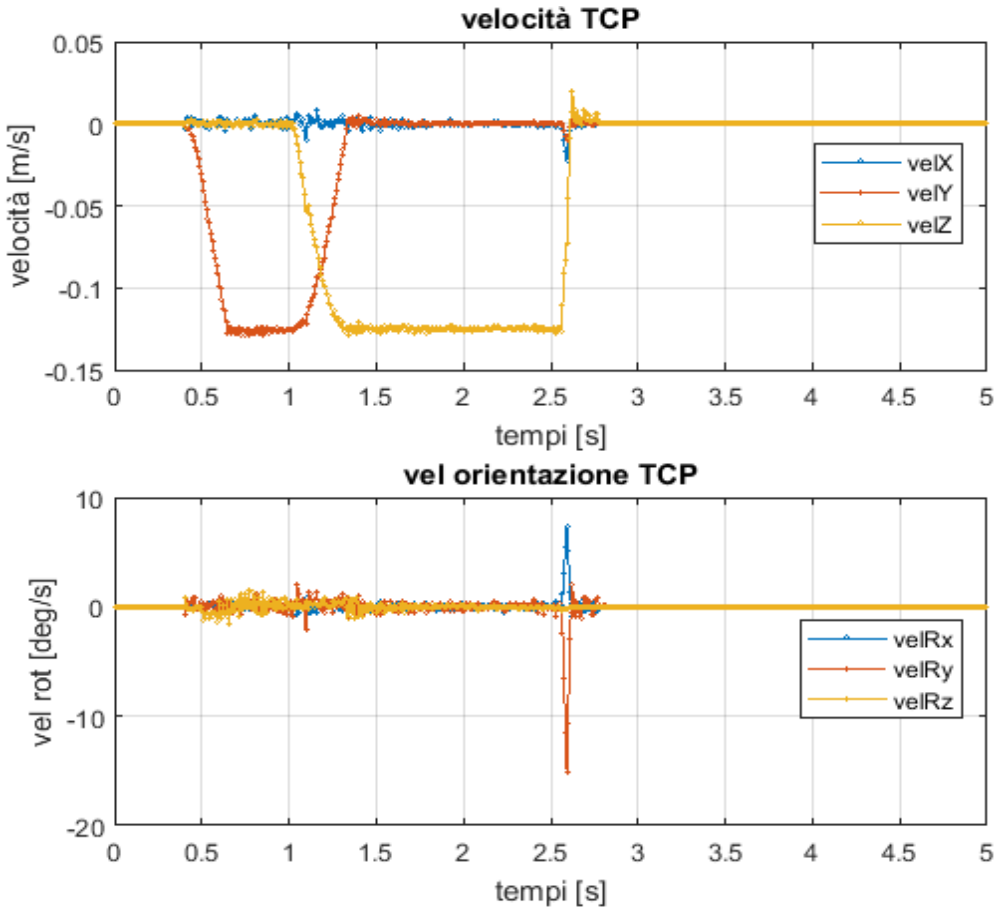


Figura 46: velocità lineare e angolare del TCP

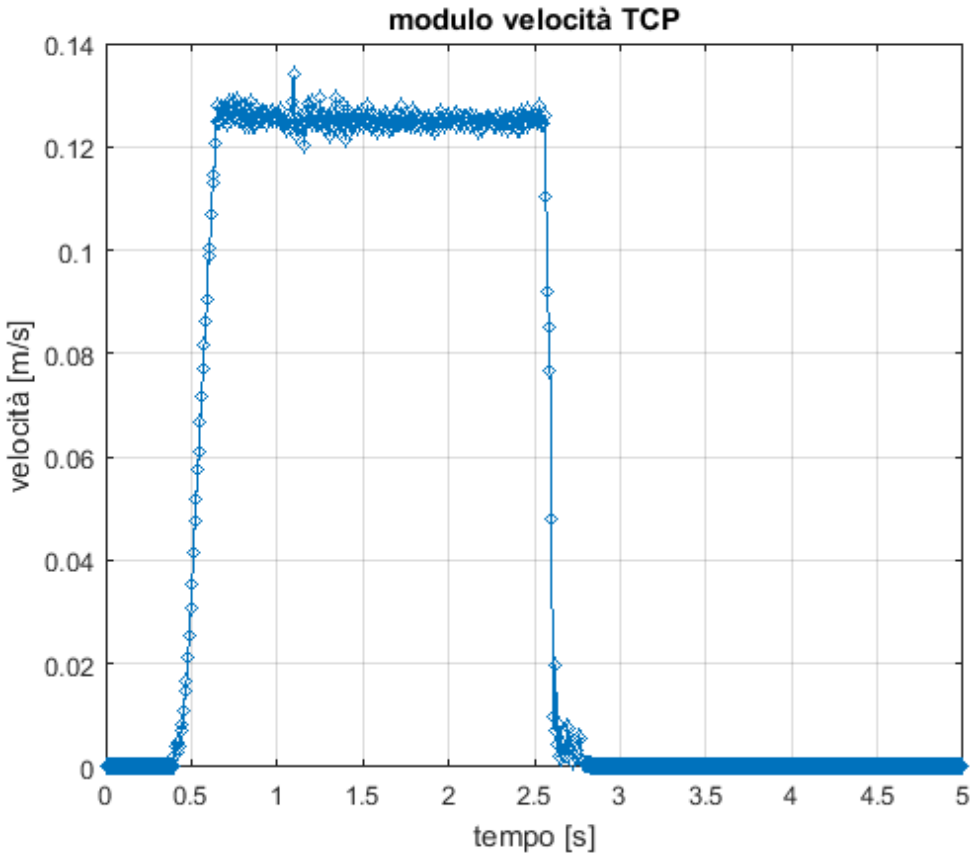


Figura 47: modulo velocità lineare

Si osserva che la velocità lineare del TCP rimane costante anche nella zona di transizione tra un movimento e l'altro. Ovviamente a seconda del raggio di transizione scelto si può avere un percorso più o meno spigoloso. Per valori piccoli il percorso presenterà molte irregolarità, mentre per valori più grandi sarà più uniforme.

Durante questo movimento però l'unità di controllo del robot non può nè attendere un'azione dell'operatore nè un'operazione di I/O; ciò potrebbe causare l'arresto del robot e innescare un blocco di protezione [4].

Nel MoveP è integrata anche un'altra funzione ovvero la "*CircleMove",* che permette di compiere movimenti circolari. Il robot inizia il movimento dalla posizione in cui si trova prima di compiere questo comando, si sposta attraverso un **punto di transizione** definito sull'arco circolare e un **punto finale** che completa il movimento circolare. L'orientazione del tool può essere mantenuta:

- *Fixed,* dove solo il punto iniziale viene utilizzato per definire l'orientamento;
- *Unconstrained* dove il punto di partenza diventa il punto finale nella definizione dell'orientazione.

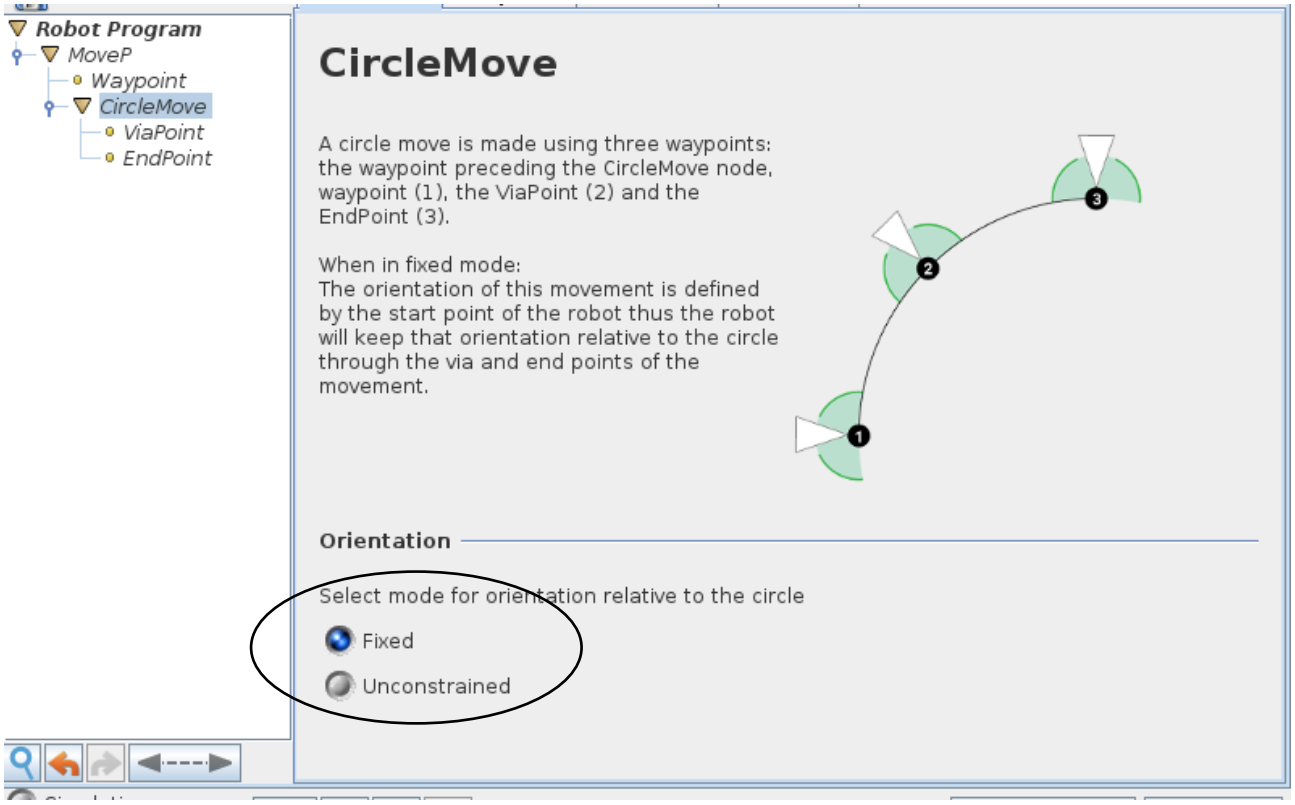


Figura 48: circle move

L'aspetto riguardante questa funzione e l'analisi dei raccordi sono ripresi durante l'illustrazione di alcuni movimenti programmati dal robot, nell'ultimo capitolo.

1.3.2 *Funzioni avanzate del robot*

Sul software di "Universal Robot" sono state implementate anche delle funzioni avanzate che permettono al robot di compiere particolari operazioni, usando sempre la semplice interfaccia grafica e senza utilizzare un linguaggio di più basso livello. La schermata che appare è la seguente:

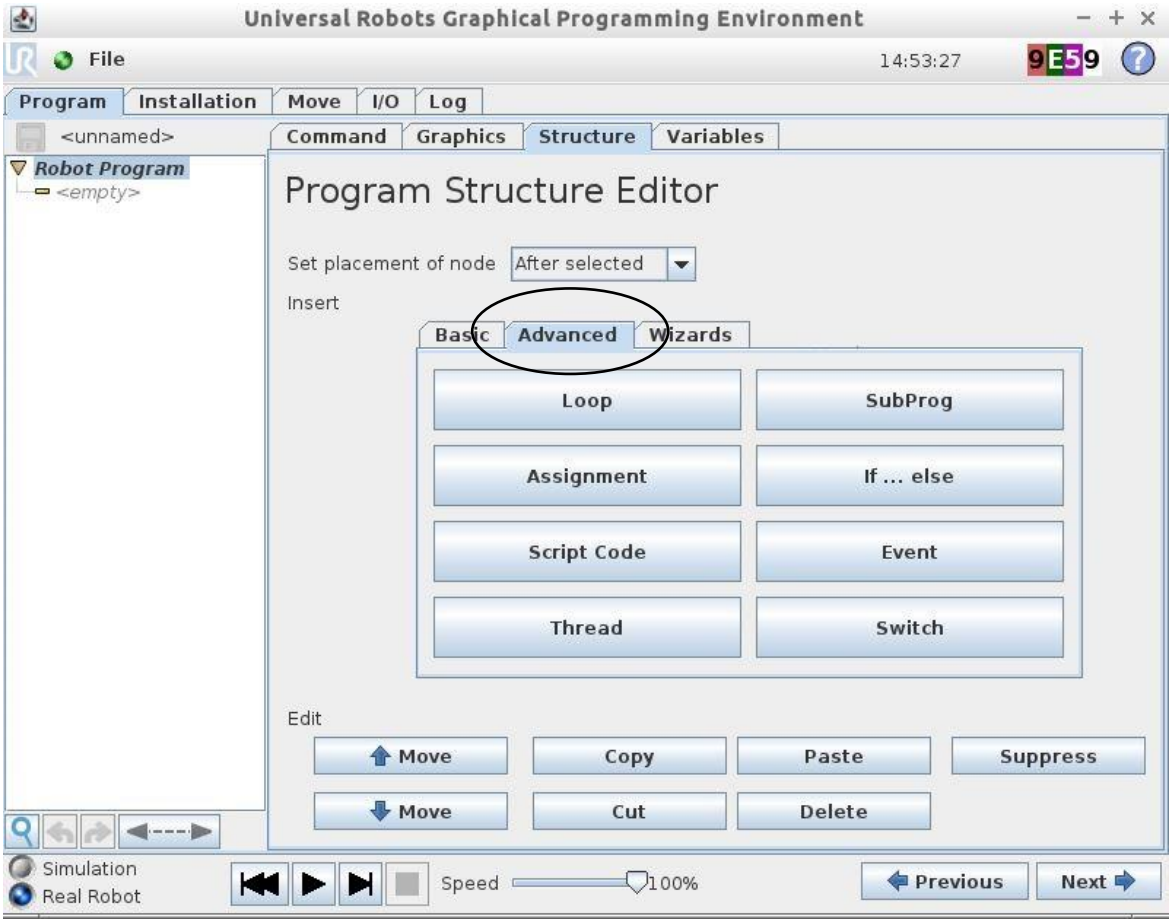


Figura 49: comandi nella sezione advanced

Si possono trovare i seguenti comandi [4]:

- *LOOP* permette di impostare un loop o infinito o di un certo numero di cicli o fino ad un particolare evento o segnale esterno;
- *SUBPROG* permette di richiamare un programma precedentemente scritto e salvato;
- *ASSIGNMENT* permette di inserire una variabile numerica o una stringa di testo;
- *IF…ELSE* permette di inserire delle condizioni e a seconda di quale si verifica vengono eseguiti determinate azioni (è stato scritto il "*PROGRAMMA_if_nopinza.urp*" per capirne il funzionamento);
- *SCRIPT CODE* permette di scrivere le funzioni del robot in forma di codice quindi con un linguaggio di più basso livello;

- *EVENT* permette di monitorare un input e successivamente eseguire un'azione o implementare una variabile;
- *THREAD* permette di eseguire delle azioni in parallelo al programma (usato soprattutto per la gestione dei segnali);
- *SWITCH* permette in base ai segnali che riceve di eseguire dei comandi piuttosto che altri (simile all'if, è stato scritto il "*PROGRAMMA_switch_nopinza.urp*" per capirne il funzionamento).

Nei capitoli successivi saranno illustrati dei programmi che presentano questi comandi avanzati.

Per concludere è necessario descrivere anche i comandi particolari che si trovano nella finestra "Wizards".

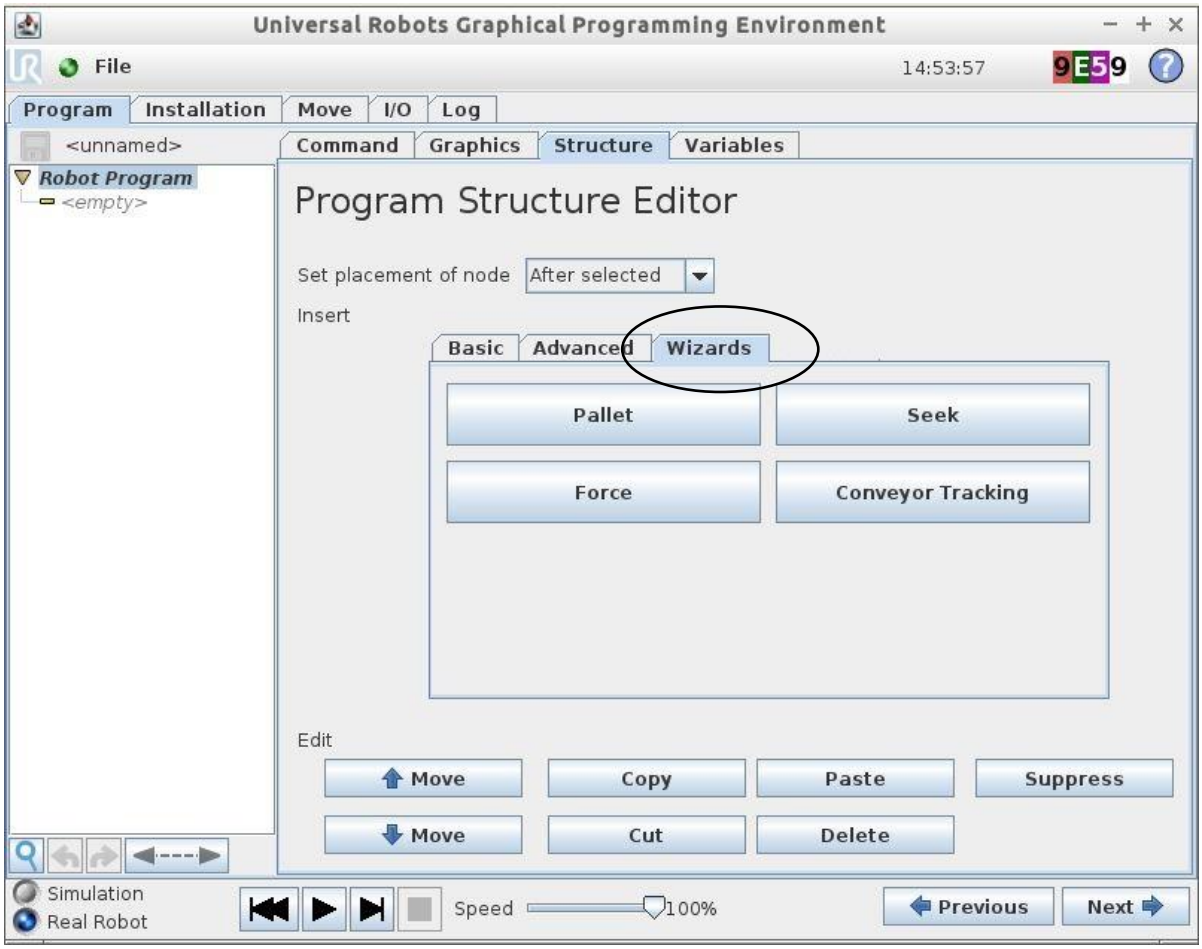


Figura 50: comandi nella sezione wizard

- *CONVEYOR TRACKING* permette di gestire i movimenti della macchina in presenza di un nastro trasportatore;
- *FORCE* permette di comandare i movimenti della macchina impostando l'entità e la direzione di una forza applicata al TCP.

I due comandi successivi possono essere illustrati attraverso dei programmi che sono stati scritti per verificarne il giusto funzionamento:

- *PALLET* permette di programmare il movimento di palletizzazione. Di seguito è illustrato il programma "*PROGRAMMA_pallet_nopinza.urp*" scritto per simulare la presa di una serie di componenti tramite azione di pick and place e la collocazione di questi su di un pallet di una certa grandezza.

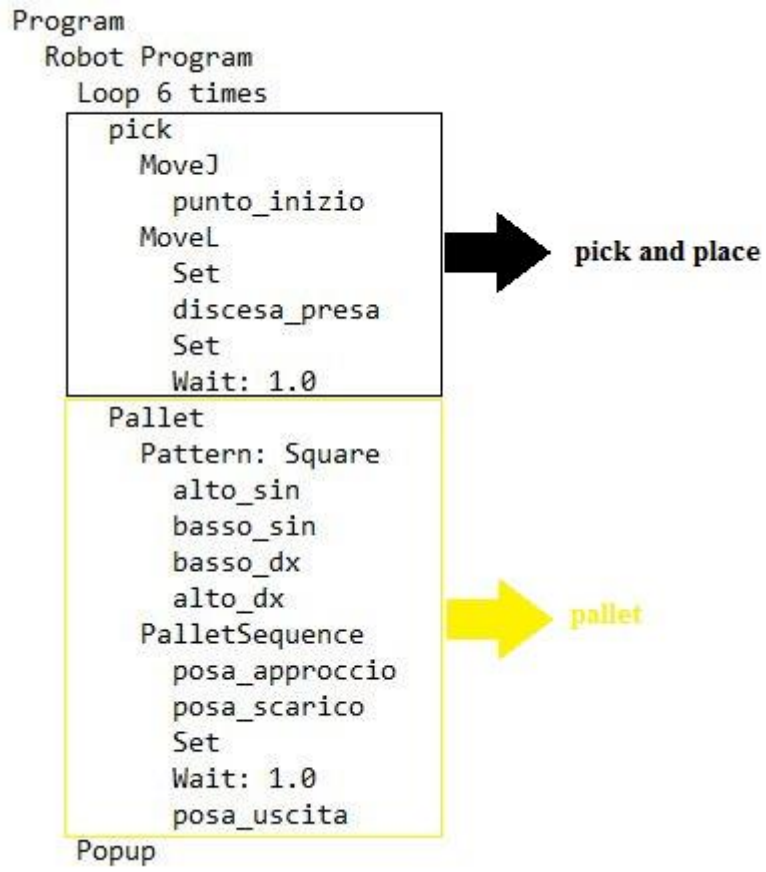


Figura 51: programma di esempio per palletizzazione

Il comando pallet è costituito da:

- Pattern, in cui si deve indicare la forma del pallet e la posizione nello spazio attraverso dei waypoint. Ovviamente la sequenza di punti non deve essere scelta a caso ma secondo la logica di deposito pezzo sul pallet individuato. Le geometrie possono essere dei punti, delle rette, dei piani o dei volumi;

- PalletSequence, in cui si definisce la sequenza di pose da eseguire. Vi è in genera una posa di approccio ad una certa altezza di sicurezza, una posa di avvicinamento/scarico con la quale si definisce la posizione di deposito pezzo e infine, una posa di uscita dove il robot raggiunge un ulteriore posizione ad un certo piano di sicurezza allontanandosi dai pallet in maniera cautelativa.

- *SEEK* permette di afferrare (destacking) e depositare (stacking) oggetti collocati uno sopra l'altro. Una volta selezionato questo comando si può scegliere quali delle due azioni compiere. Nel programma che segue (*PROGRAMMA_stack_nopinza.urp*) si è pensato di realizzare in serie prima un'operazione di distacco e poi una di deposito.

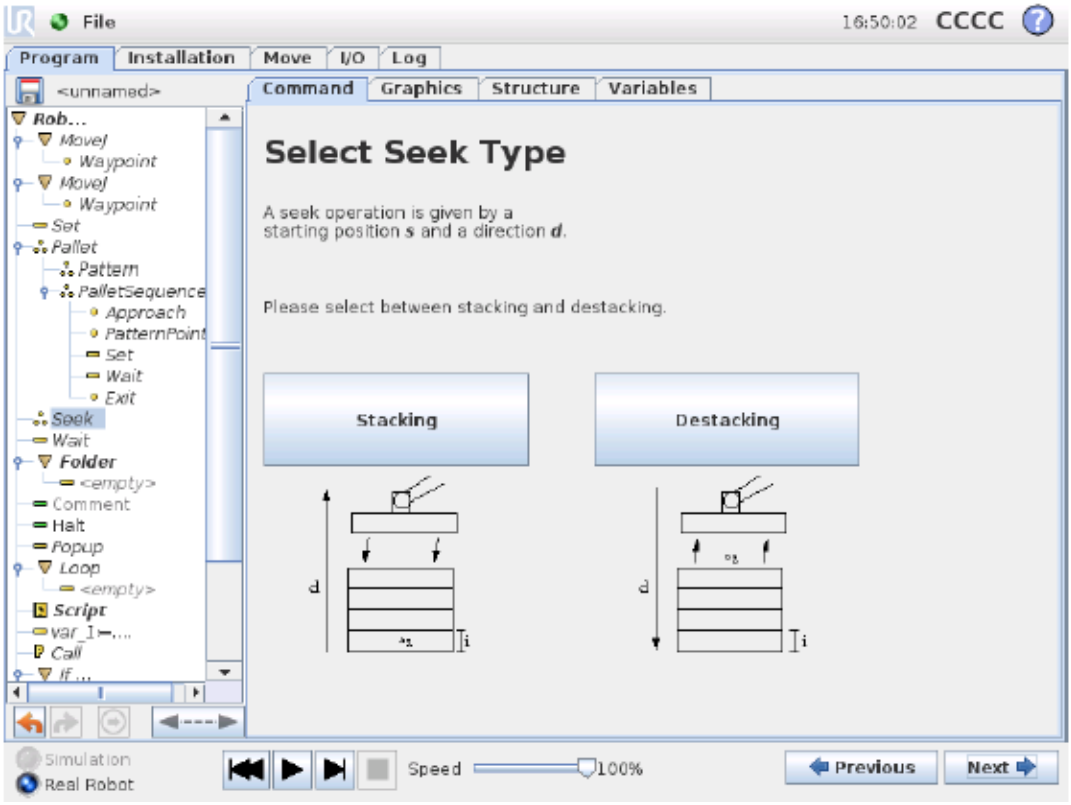


Figura 52: pagina principale del comando seek

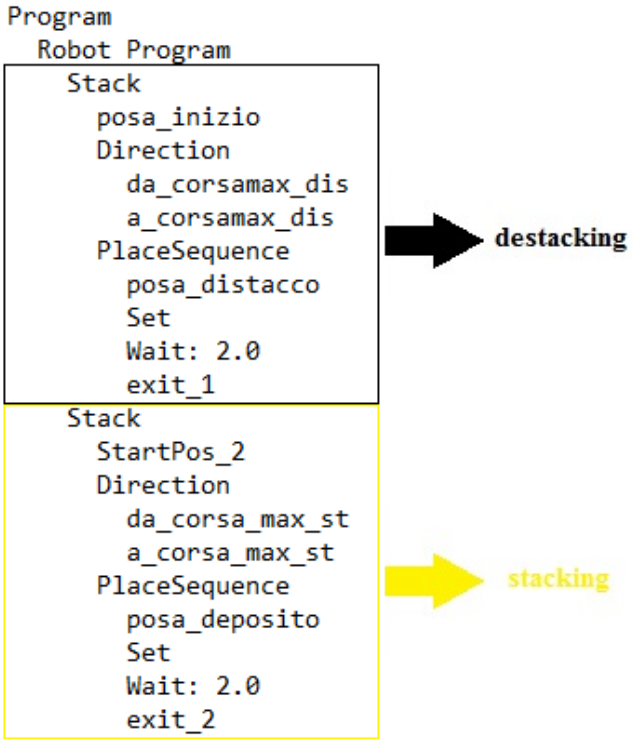


Figura 53: programma di esempio del seek

Il comando seek è costituito da:

- Startposition, in cui il robot torna ogni volta che termina il comando;

- Direction, costituito da due waypoint che definiscono la direzione e il massimo della corsa che deve compiere la macchina;

- PlaceSequence, costituito da una serie di waypoint, uno che individua la posizione di posa o prelievo del pezzo e un ultimo che permette l'allontanamento verso un piano di sicurezza senza urtare gli oggetti precedentemente mossi.

Nella pagina principale del comando vengono richieste due informazioni: una riguarda un segnale con la quale il robot capisce quando passare all'azione successiva, mentre l'altra è legata alla grandezza (in termine di altezza) degli oggetti. Quest'ultima informazione è fondamentale poiché il software in automatico riesce a calcolare, in base alla distanza specificata in Direction, il numero di oggetti da manipolare.
Infine, ovviamente, anche in questo caso è possibile indicare una velocità e un'accelerazione dei movimenti.

1.3.3 URSim 3.5.3 (simulatore del PolyScope)

Tutti questi movimenti illustrati fino a questo punto possono essere simulati e acquisiti, prima di utilizzare il robot vero e il suo sistema di comando, attraverso il software fornito da "Universal Robot" URSim (in questo lavoro è stata utilizzata la versione 3.5.3). In questo ambiente di lavoro è possibile trovare tutti i comandi nominali che servono per muovere il robot, visti fin ora e in cui si ha la possibilità di verificare e programmare la macchina anche quando il componente fisico non è presente. In questo elaborato tutte le acquisizioni sono state ottenute da movimenti effettuati direttamente dall'UR3, ma alcuni programmi, prima di essere scritti nel suo ambiente di lavoro, sono stati pensati ed implementati al simulatore.

Questo software però è stato creato per un sistema operativo Linux. Per l'esecuzione del simulatore in un altro sistema operativo è necessaria una macchina virtuale. Viene consigliato da UR di installare VMWare Player oppure, com'è stato fatto per questo lavoro, VirtualBox. Queste macchine virtuali sono state create appositamente per gli utenti di UR. Contengono dei simulatori dei software per l'UR3, UR5 o per l'UR10. Non è però possibile lanciare più di un simulatore per volta [4].

Di seguito è illustrata la schermata principale del simulatore una volta lanciato URSim in VirtualBox. È possibile osservare i tre software di simulazione e le relative cartelle con all'interno i programmi scritti e salvati in questo ambiente.

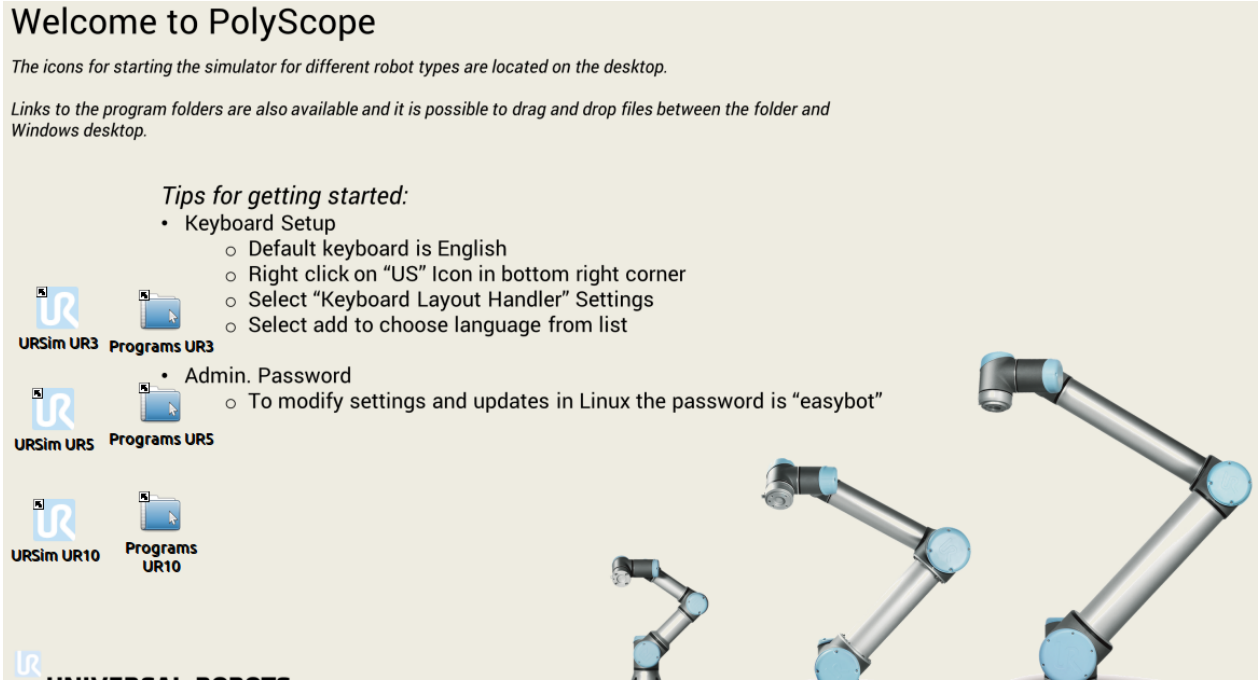


Figura 54: schermata principale URSim su VirtualBox

Per poter usare il simulatore bisogna prima istallare la macchina virtuale sul proprio pc. Prima di tutto bisogna essere sicuri di possedere almeno 8Gb liberi sul proprio disco rigido, scaricare dal sito dell'UR l'archivio che contiene la macchina virtuale e estrarre i file al suo interno in una qualsiasi cartella. Successivamente si eseguono le seguenti istruzioni [4]:

1 avviare VirtualBox e premere nuovo;
2 definire il nome (arbitrario) e il tipo (Linux, versione Ubuntu);
3 selezionare la dimensione della memoria 768Mb e premere avanti;
4 selezionare la voce "utilizza un file del disco rigido" e definire il percorso in cui è stato scompattato il file compresso, poi premere crea;
5 premere start per avviare la macchina virtuale;
6 se si verifica l'errore "Hardware acceleration is not avaiable" bisogna entrare nelle impostazioni del BIOS e abilitare l'accesso alle macchine virtuali, successivamente riavviare Windows, VirtualBox e la macchina virtuale;
7 la macchina virtuale è attiva.

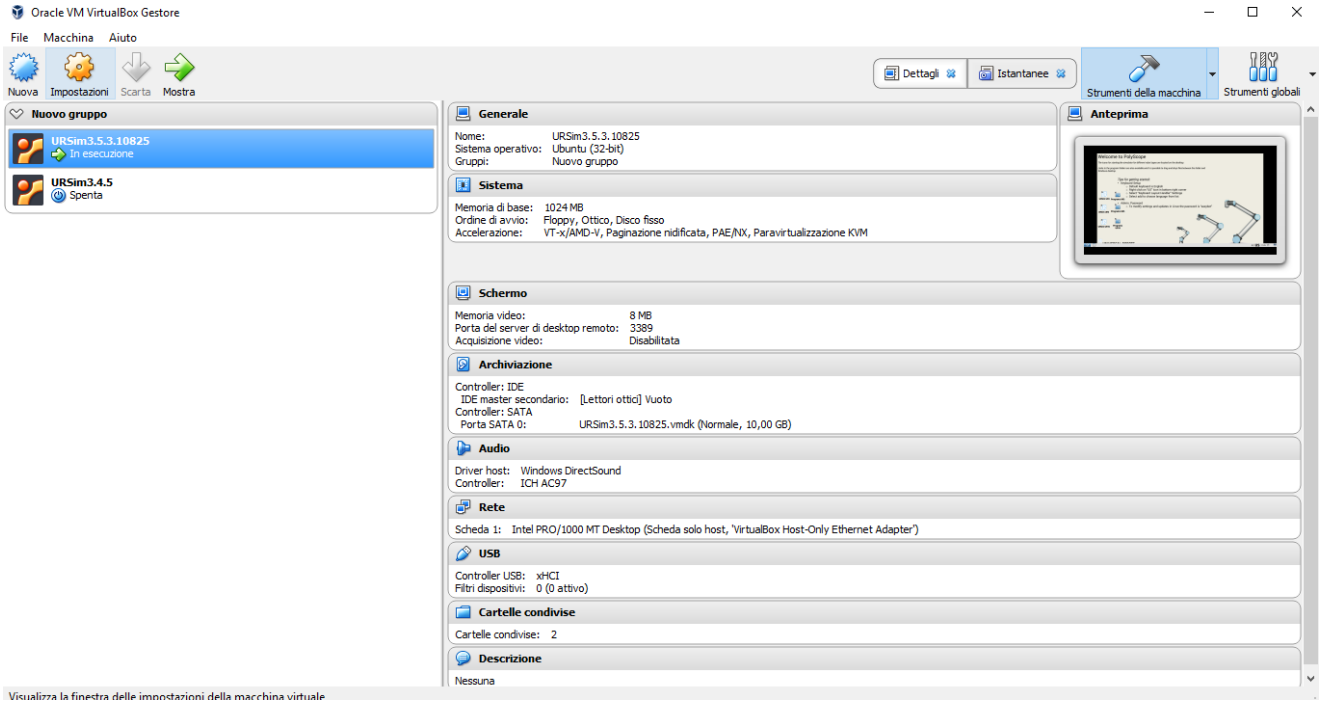


Figura 55: schermata principale VirtualBox

In alto a sinistra è possibile lanciare il simulatore del robot e successivamente visualizzare la schermata in figura 54.

2. MODELLO MULTIBODY DEL ROBOT

Una volta capito come programmare il robot nel suo ambiente di programmazione, è possibile realizzare un suo modello multi-body per simularne i movimenti ed effettuare diversi calcoli cinematici e dinamici. Tale modello, nel caso del presente elaborato, viene realizzato mediante il software Matlab sfruttando una sua estensione, ovvero il toolbox versione 9.1 di Peter Corke.

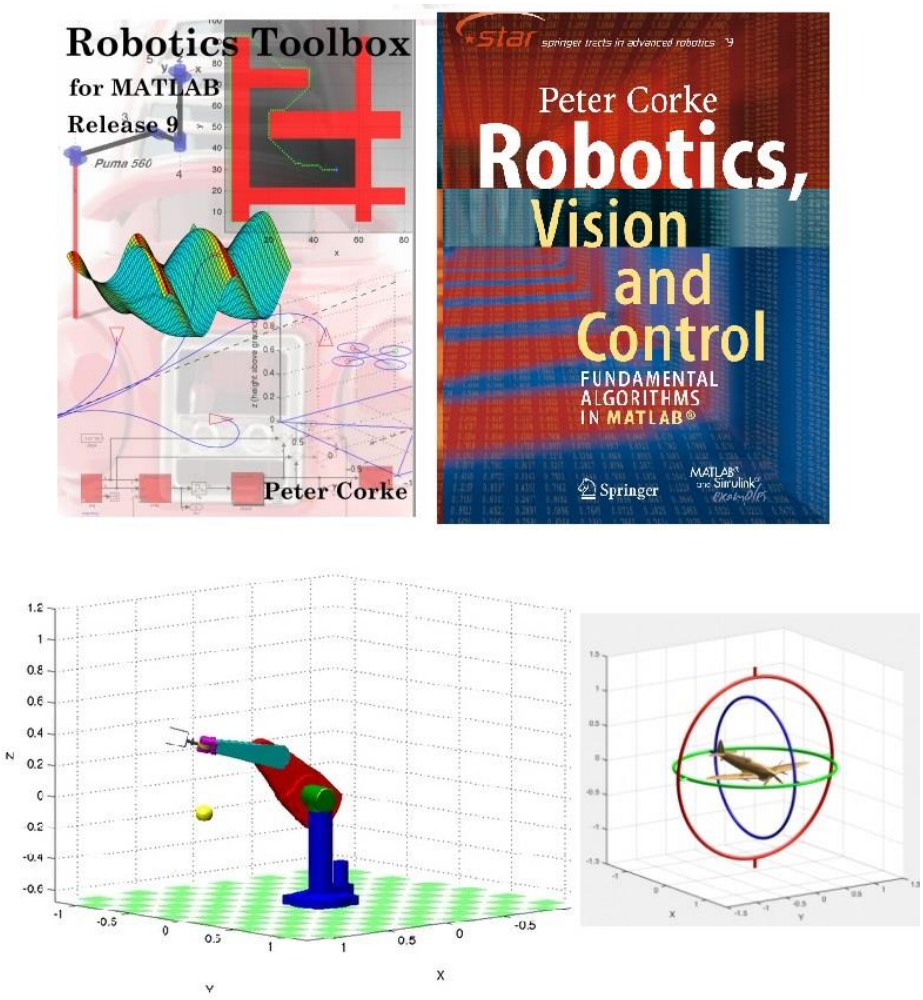


Figura 56: facciata manuale del Toolbox 9.1, facciata del manuale degli algoritmi ed esempi di riproduzione 3D

Questa estensione contiene molte funzioni che permettono di simulare diversi tipi di braccia robotiche e studiarne aspetti come la cinematica, la dinamica o anche la generazione di traiettorie. Inoltre, contiene file che permettono di rappresentare in 2D e in 3D l'orientazione e la posa dell'oggetto analizzato [11]. L'utilizzo di questo toolbox comporta alcuni vantaggi tra cui:

- il codice è già abbastanza completo e fornisce un buon punto di confronto per future implementazioni degli stessi algoritmi;
- le funzioni sono generalmente scritte in maniera abbastanza semplice e comprensibile, talvolta però a discapito dell'efficienza computazionale. Se si desidera migliorare tale aspetto si può tranquillamente modificare o riscrivere le funzioni;

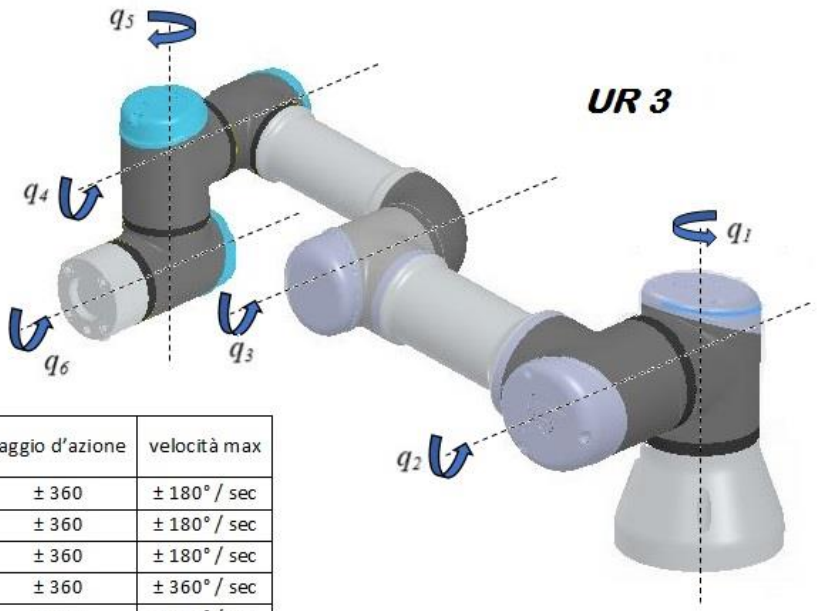
- poiché il codice sorgente è disponibile, è possibile utilizzarlo per la comprensione delle analisi e soprattutto per l'insegnamento.

Si procede dunque con l'illustrare come è stato costruito il modello e quelle che sono state le funzioni e le equazioni utilizzate per lo studio del robot.

2.1 Convenzioni di Denavit-Hartenberg

Per poter affrontare lo studio del robot, è necessario comprendere come ogni singolo giunto deve essere controllato cinematicamente per poter ottenere una data posizione del tool. I giunti (joint), infatti, sono gli elementi di collegamento dei singoli corpi (comunemente noti come link) costituenti il robot, azionati di solito da determinati motori. Si procede introducendo dei sistemi di riferimento relativi ad ogni link, per poter cosi individuare la posa del sistema di riferimento del tool rispetto alla base del robot, sfruttando le relazioni tra i singoli riferimenti intermedi della catena. Ci sono due logiche secondo le quali possono essere introdotte le terne e sono una alla base della convenzione standard di Denavit-Hartenberg e una alla base della convenzione modificata. In questo elaborato sono stati sviluppati entrambi i modelli.

Vengono riportati di seguito i versi di rotazione positivi dei joint essenziali per la scelta dei sistemi di riferimento di Denavit-Hartenberg (abbreviato D-H).



Movimento assiale del robot	raggio d'azione	velocità max
base (q1)	± 360	± 180° / sec
spalla (q2)	± 360	± 180° / sec
gomito (q3)	± 360	± 180° / sec
polso 1 (q4)	± 360	± 360° / sec
polso 2 (q5)	± 360	± 360° / sec
polso 3 (q6)	infinita	± 360° / sec

Figura 57: UR3 e verso positivo di rotazione dei giunti

2.1.1 Convenzione di Denavit – Hartenberg Standard

Al fine di calcolare l'equazione della cinematica diretta per un manipolatore a catena aperta, bisogna delineare un metodo generale e sistematico per definire posizione e orientamento relativi di due link consecutivi. Il problema si riconduce all'individuazione delle terne solidali a ciascun link e alla determinazione della trasformazione di coordinate che legano le due terne.

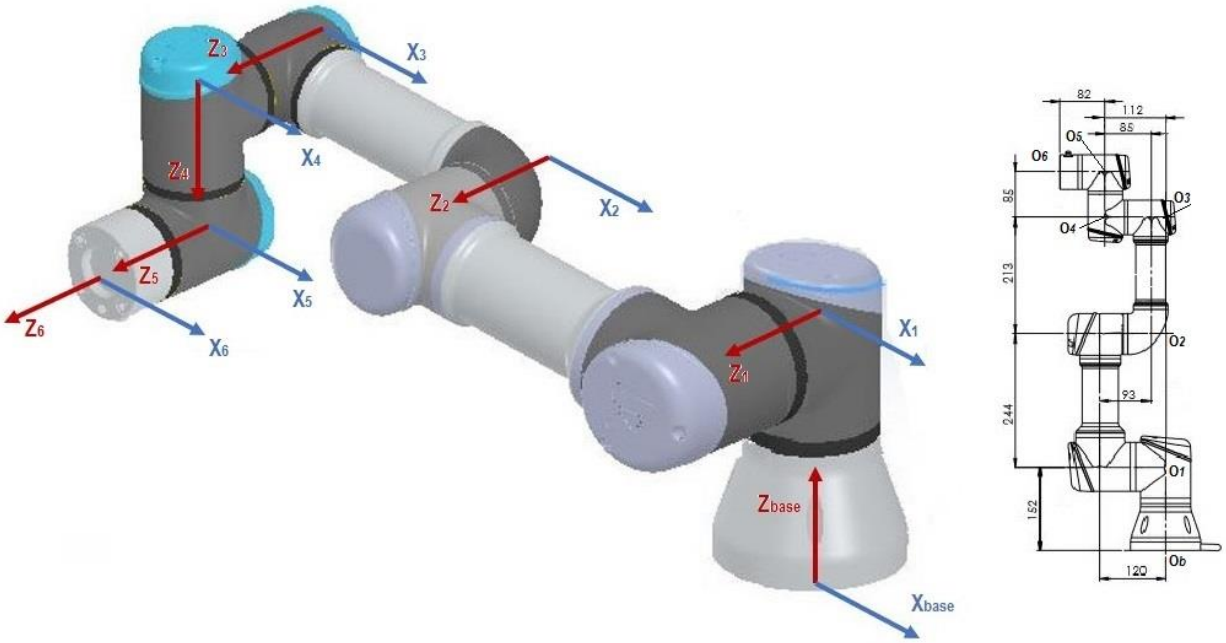


Figura 58 :Scelta delle terne di D-H e geometria per poter individuare i parametri di D-H nella configurazione di tutti zeri

Per prima cosa vengono scelte le terne di riferimento (mostrate in fig. 58) secondo la seguente procedura [5]:

1 si sceglie l'asse z_i giacente lungo l'asse del giunto i+1;
2 si individua O_i all'intersezione dell'asse z_i con la normale comune agli assi z_{i-1} e z_i (se i due assi sono paralleli l'origine si può scegliere in qualsiasi punto della retta che funge da distanza);
3 si sceglie l'asse x_i diretto lungo la normale comune agli assi z_{i-1} e z_i con verso positivo dal giunto i al giunto i+1 (nel caso di intersezione degli assi z, la normale comune è sia uscente sia entrante per cui l'asse x può essere scelta arbitrariamente);
4 si sceglie l'asse y_i (non riportato in figura) in modo da completare una terna levogira.

Una volta definite le terne, si procede con l'individuazione dei parametri di D-H nel seguente ordine (con O_i' verrà indicata l'intersezione della normale comune x_i con z_{i-1}):

- $\boldsymbol{a_i}$, distanza di O_i da O_i';
- $\boldsymbol{d_i}$, coordinata su z_{i-1} di O_i';
- $\boldsymbol{\alpha_i}$, angolo intorno a x_i tra l'asse z_{i-1} e l'asse z_i valutando positivo in senso antiorario;
- $\boldsymbol{\vartheta_i}$, angolo intorno all'asse z_{i-1} tra l'asse x_{i-1} e l'asse x_i valutando positivo in senso antiorario.

Tabella 11: parametri di D-H convenzione standard

Parametri di Denavit-Hartenberg standard				
link *i*	ϑ_i [°]	a_i [m]	d_i [m]	α_i [°]
1	**q1**	**0**	**0.1519**	**90°**
2	**q2**	**-0.24365**	**0**	**0**
3	**q3**	**-0.21325**	**0**	**0**
4	**q4**	**0**	**0.11235**	**90°**
5	**q5**	**0**	**0.08535**	**-90°**
6	**q6**	**0**	**0.08190**	**0**

È possibile adesso studiare il robot in qualsiasi configurazione, poiché tali parametri sono funzione solo della geometria e quindi costanti, eccetto i diversi ϑ che rappresentano i singoli gradi di libertà del robot. Con l'ausilio del software Matlab e del toolbox istallato è stato possibile ricavare una riproduzione 3D schematizzata del robot nella configurazione di tutti zero sfruttando i parametri di D-H calcolati:

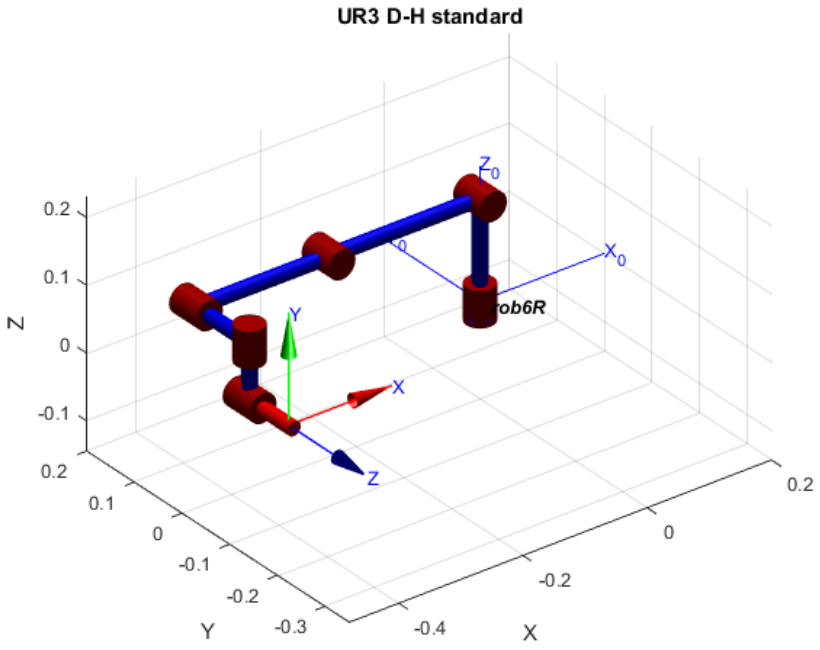


Figura 59: schematizzazione del robot in ambiente Matlab convenzione standard

Lo scheletro del robot riportato in figura 59 (e successivamente in figura 61) viene ottenuto per mezzo dell'utilizzo di due funzioni scritte. I due file sono i seguenti:

- "*build_robot_UR3*", che permette di creare il robot generando i diversi link, e di associare ad ognuno di essi sia i rispettivi parametri di Denavit-Hartenberg sia le diverse proprietà dinamiche come massa, inerzia e posizione del baricentro;
- "*forward_kinematics*", che permette di calcolare tutte le matrici di trasformazione dei singoli link rispetto al sistema di base e di plottare il robot in versione schematica come nelle figure 59 e 61. Inoltre, in questo file vengono calcolate anche delle matrici CAD che servono per trovare la relazione tra i sistemi di riferimento del robot in ambiente Solidworks e quelli di D-H. Queste matrici serviranno per poter visualizzare il robot anche in un altro ambiente di visualizzazione, ovvero in Simulink.

2.1.2 Convenzione di Denavit – Hartenberg Modificata

Lo stesso procedimento viene riproposto questa volta applicando la convenzione di Denavit – Hartenberg modificata. Questa metodologia è leggermente diversa in quanto, durante la scelta delle terne di riferimento, si considerano gli assi i-esimi sul giunto prossimale del link i-esimo, e non su quello distale come succedeva nella convenzione standard. Infatti, la differenza tra le due convenzioni consiste in una diversa posizione delle terne e diverso ordine dei parametri.

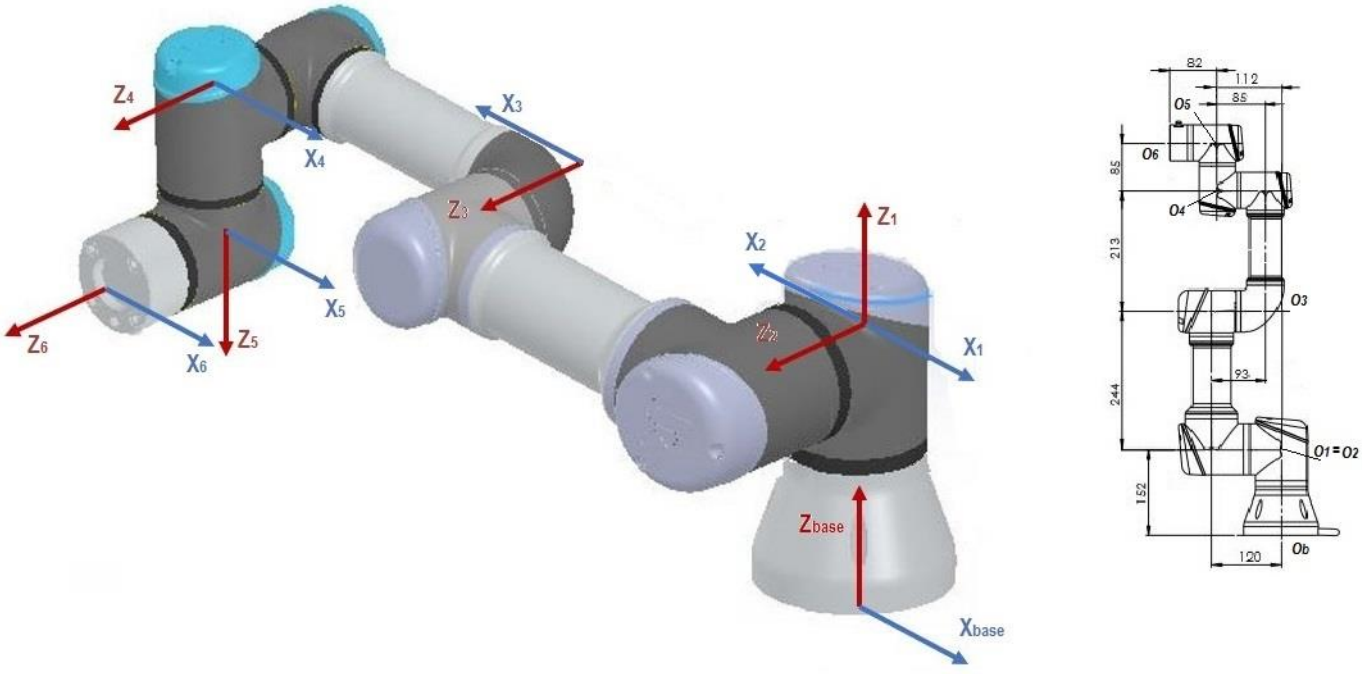


Figura 60: scelta delle terne di D-H e geometria per poter individuare i parametri di D-H nella configurazione di tutti zeri

Scelta delle terne di riferimento di Denavit – Hartenberg convenzione modificata (fig. 60) [5]:

1 si sceglie l'asse z_i giacente lungo l'asse del giunto i;
2 si individua O_i all'intersezione dell'asse z_i con la normale comune agli assi z_{i+1} e z_i (se i due assi sono paralleli l'origine si può scegliere in qualsiasi punto della retta che funge da distanza);
3 si sceglie l'asse x diretto lungo la normale comune agli assi z_{i+1} e z_i con verso positivo dal giunto i al giunto i+1;
4 si sceglie l'asse y (non riportato in figura) in modo da completare una terna levogira.

Una volta definite le terne di riferimento solidali ai link, è possibile specificare completamente la posizione e l'orientamento della terna i-esima rispetto la terna i-1 con i seguenti parametri di D-H (dove O' è sempre definito come l'intersezione della normale comune x_i con l'asse z_{i-1}):

- α_{i-1}, angolo di twist intorno a x_{i-1} tra l'asse z_{i-1} e l'asse z_i valutandolo positivo secondo il verso di x_{i-1};
- a_{i-1}, distanza tra O_i da O_i' (distanza tra i due assi z divenuti paralleli per mezzo dell'angolo di twist);
- d_i, coordinata su z_i di O_i' (distanza tra O_i e O' lungo l'asse z_i);
- ϑ_i, angolo intorno all'asse z_i tra l'asse x_{i-1} e l'asse x_i valutandolo positivo secondo il verso di z_i.

Tabella 12: parametri di D-H convenzione modificata

Parametri di Denavit-Hartenberg				
giunto *i*	α_{i-1} [°]	a_{i-1} [m]	d_i [m]	ϑ_i [°]
1	**0**	**0**	**0,1519**	**q1**
2	**90°**	**0**	**0**	**q2-pi**
3	**0**	**0,24365**	**0**	**q3**
4	**0**	**0,21325**	**0.11235**	**q4-pi**
5	**+90°**	**0**	**0.08535**	**q5**
6	**-90°**	**0**	**0,08190**	**q6**

A questo punto, sempre con l'ausilio del software Matlab, è stato possibile ricavare il plot del robot nella configurazione di tutti zero sfruttando i parametri di D-H calcolati:

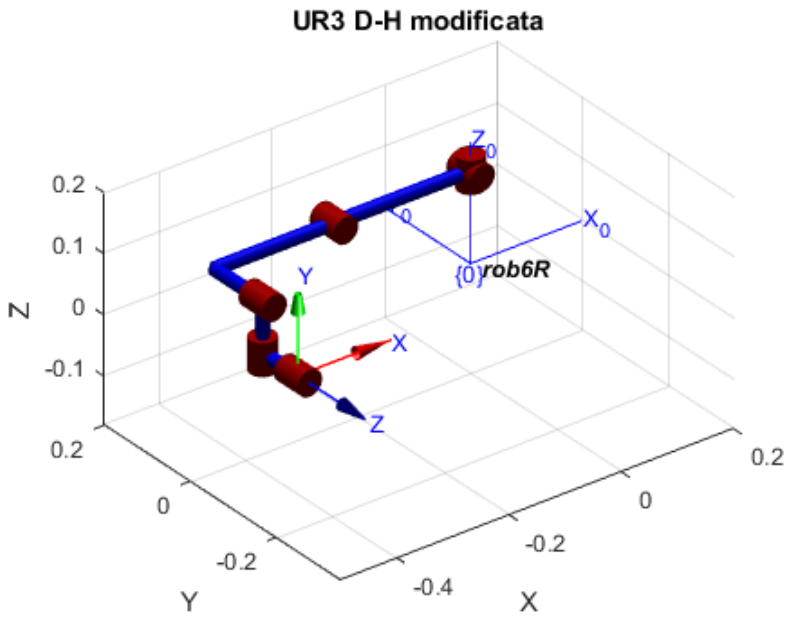


Figura 61: schematizzazione del robot in ambiente Matlab convenzione modificata

Questa convenzione è stata quella scelta per le successive analisi cinematiche e dinamiche.

2.2 Cinematica diretta

Un manipolatore è costituito da un insieme di corpi rigidi connessi in cascata tramite coppie cinematiche o giunti. I giunti possono essere fondamentalmente di due tipi: giunti di rotazione o rotoidali (in questo caso lo sono tutti e sei) e giunti di traslazione o prismatici. Tutta la struttura nel suo insieme costituisce una catena cinematica. Come già visto, un estremo della catena è vincolato ad una base, mentre all'altro estremo è collegato un organo che di solito consente la manipolazione di oggetti nello spazio, in questo caso un gripper a due dita. Da un punto di vista topologico, la catena cinematica viene detta aperta quando vi è una sola sequenza di bracci a connettere i due estremi della catena. In maniera alternativa, un manipolatore contiene una catena cinematica chiusa quando una sequenza di bracci forma un anello [12].

La struttura meccanica di un manipolatore è caratterizzata da un numero di gradi di libertà che ne determinano la posa. Ogni grado di libertà viene tipicamente, come visto, associato a un'articolazione di giunto e costituisce una variabile di giunto. L'obiettivo della cinematica diretta è quello di determinare la posa dell'organo terminale del manipolatore, in funzione dei valori assunti dalle variabili di giunto (ovvero dai gradi di libertà e cioè da come ogni singolo giunto viene attuato) [12].

In catene cinematiche più semplici può essere applicato un approccio di tipo geometrico e trovare quindi in maniera più intuitiva e rapida il legame tra il tool e il sistema di riferimento di base. Come è facile intuire, l'efficacia di un approccio di questo tipo quando la struttura del manipolatore è complessa e il numero dei giunti diventa elevato, non è molto elevata per cui è preferibile l'adozione di una soluzione meno diretta, ma basata su una procedura sistematica e generale [12].

Si consideri il caso studiato in questo elaborato, ovvero una catena aperta costituita da n+1 bracci (7 bracci) connessi tramite n giunti (6 giunti) ove il braccio 0 è convenzionalmente fissato a terra (base). Ogni giunto fornisce un singolo grado di libertà alla struttura meccanica. La costruzione di una procedura operativa per il computo della cinematica diretta scaturisce naturalmente dalla struttura a catena cinematica aperta del manipolatore. Infatti, dal momento che ciascun giunto connette due e solo due bracci consecutivi, è ragionevole considerare dapprima isolatamente il problema della descrizione dei legami cinematici tra bracci successivi e successivamente risolvere in maniera ricorsiva il problema della descrizione complessiva della cinematica del manipolatore. A tale scopo è opportuno definire, come già visto nel paragrafo precedente, una terna di coordinate solidali a ciascun braccio, dalla base all'ennesimo. Pertanto, la trasformazione di coordinate complessiva che esprime posizione e orientamento della terna ennesima rispetto alla base è data:

$$ {}^{0}\boldsymbol{A}_n(\boldsymbol{q}) = {}^{0}\boldsymbol{A}_1(q_1) * {}^{1}\boldsymbol{A}_2(q_2) \dots * {}^{n-1}\boldsymbol{A}_n(q_n) = \begin{bmatrix} {}^{0}\boldsymbol{A}_6 & {}^{0}\boldsymbol{p}_6 \\ \boldsymbol{0} & 1 \end{bmatrix} $$

Quello che alla fine va calcolato dunque, è la matrice ${}^{0}\boldsymbol{A}_6$ che rappresenta l'orientazione del tool rispetto alla terna di riferimento di base e il vettore ${}^{0}\boldsymbol{p}_6$ che rappresenta la posizione nello spazio del tool rispetto allo stesso riferimento di base.

Poiché nello spazio giunti sono noti non solo i valori di posizione ma anche di velocità e accelerazione, è possibile inoltre poter calcolare il vettore di velocità generalizzata del tool (ovvero i valori della velocità lineare e angolare del tool) attraverso una semplice relazione che coinvolge la matrice Jacobiana del tool. Questa matrice rappresenta una trasformazione lineare che mappa il vettore velocità di n dimensioni dei giunti con il vettore velocità generalizzata del tool. Il legame può essere osservato dalla seguente relazione:

$$\underline{V}_{TCP} = J_{TCP} * \underline{\dot{q}}$$

Dove:

- $\underline{V}_{TCP}$ = rappresenta il vettore delle velocità generalizzato di componenti $[\,V_x\ V_y\ V_z\ \omega_x\ \omega_y\ \omega_z\,]^T$;
- $\underline{\dot{q}}$ = rappresenta il vettore di velocità dei giunti $[\,\dot{q}_1\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4\ \dot{q}_5\ \dot{q}_6\,]^T$;
- J_{TCP} = matrice Jacobiana.

Nel modello realizzato viene utilizzata la funzione "*forward_kinematics_inverse_dynamics.m*" nella quale viene risolta sia la cinematica diretta sia la dinamica inversa. Per quanto riguarda la cinematica diretta, nota la storia temporale dei giunti, viene calcolata mediante la "*fkine.m*" la posa del tool per ogni set di posizione dei giunti stessi. In sostanza viene fatto il calcolo in un ciclo for e il numero di cicli è pari al numero di istanti temporali considerati. Inoltre, sfruttando la funzione "Jacob0" è possibile per ogni iterazione calcolare la matrice jacobiana del tool rispetto alla base e, come visto dalle formule precedenti, poter calcolare i componenti del vettore di velocità generalizzato riferito al tool stesso. Viene riportata di seguito la parte di codice che permette la risoluzione della cinematica diretta:

```
for i = 1:length(time)
% forward kinematics
    qvec = deg2rad([q1vet(i) q2vet(i) q3vet(i) q4vet(i) q5vet(i)
q6vet(i)]);
% gfd giunti noti dall'acquisizione [rad]
    [ATCP0, A_all]=rob6R.fkine(qvec);
% soluzione cinematica diretta
    pTCP0vet(:,i)= ATCP0(1:3,4);
% posizione TCP wrt sr.0

% velocità TCP
    JTCP_0=rob6R.jacob0(qvec);
% jacobiana TCP wrt sr.0 (coordiante velocita'  TCP in sr.0)
    qdvec=[q1pvet(i) q2pvet(i) q3pvet(i) q4pvet(i) q5pvet(i)
q6pvet(i)]'*pi/180;    % velocita' gdl giunti nota dall'acquisizione
    vTCP(:,i)=JTCP_0*qdvec;
% velocita'  generalizzata TCP wrt sr.0
    magvTCP(1,i)=norm(vTCP(1:3,i),2);
% modulo velocita' lineare TCP wrt sr.0

    ...
end
```

2.3 Dinamica inversa: equazioni e logica del modello

La deduzione del modello dinamico di un manipolatore gioca un ruolo di estrema importanza in relazione ai problemi di simulazione del moto, di analisi di strutture di manipolazione e di determinazione di algoritmi di controllo. La simulazione del moto di un manipolatore consente di provare strategie di controllo e tecniche di pianificazione di traiettoria, senza la necessità di riferirsi a una struttura di manipolazione fisicamente disponibile. L'analisi del modello dinamico può essere di aiuto nel progetto meccanico di prototipi. La conoscenza di forze e coppie richieste per l'esecuzione di movimenti tipici fornisce informazioni utili al progetto di giunti e trasmissioni e alla scelta di attuatori [12].
I problemi di tipo dinamico in generale prevedono due differenti metodologie di risoluzione:

- un approccio di sistema (o di Lagrange) che si basa su considerazioni esclusivamente energetiche e dove le equazioni del moto possono essere derivate con un approccio indipendente dal sistema di coordinate di riferimento. Questo modello è concettualmente semplice e sistematico e conduce alla derivazione del modello dinamico in forma chiusa;
- un approccio di Newton-Eulero che si basa su diversi bilanci di forze e coppie sui diversi sottosistemi del robot mediante l'ausilio del diagramma a corpo libero. Questo consente di dedurre un modello di tipo ricorsivo che risulta efficiente da un punto di vista computazionale in quanto sfrutta la natura seriale a catena aperta tipica dei manipolatori.

Quest'ultima analisi risulta molto efficiente in questo caso poiché strettamente collegata alla cinematica studiata precedentemente. La procedura di calcolo che si è seguita per poter ottenere le azioni motrici di ogni singolo giunto è la seguente (riportata nello schema in figura 59):

1 **Cinematica diretta,** si parte dall'equazione del moto del primo giunto e, noti i valori di velocità generalizzata e angolare nel link di base, si ricava la cinematica del link 1. La procedura di calcolo si ripete fino al link ennesimo presumendo di conoscere tutte le leggi del moto dei singoli giunti;

2 **Dinamica inversa,** terminata l'analisi al punto 1 si procede a ritroso. Infatti, con le velocità relative al link ennesimo e con le forze esterne (in questo caso considerate agenti solo sull'ultimo link) si riesce a ricavare le reazioni scambiare tra il link n e il link n-1. Così facendo è possibile risalire alle reazioni che vengono generate e che si scaricano sulla base del robot;

3 **Azioni motrici,** i valori delle azioni motrici possono essere ricavate dalle reazioni calcolate volta per volta al punto 2, proiettandole su alcuni versori corrispondenti alla direzione dell'asse

in cui si ha il grado di libertà nel giunto. Queste forze motrici non saranno altro che le forze e le coppie che i motori devono esercitare per produrre quel determinato moto con quella particolare cinematica analizzata precedentemente.

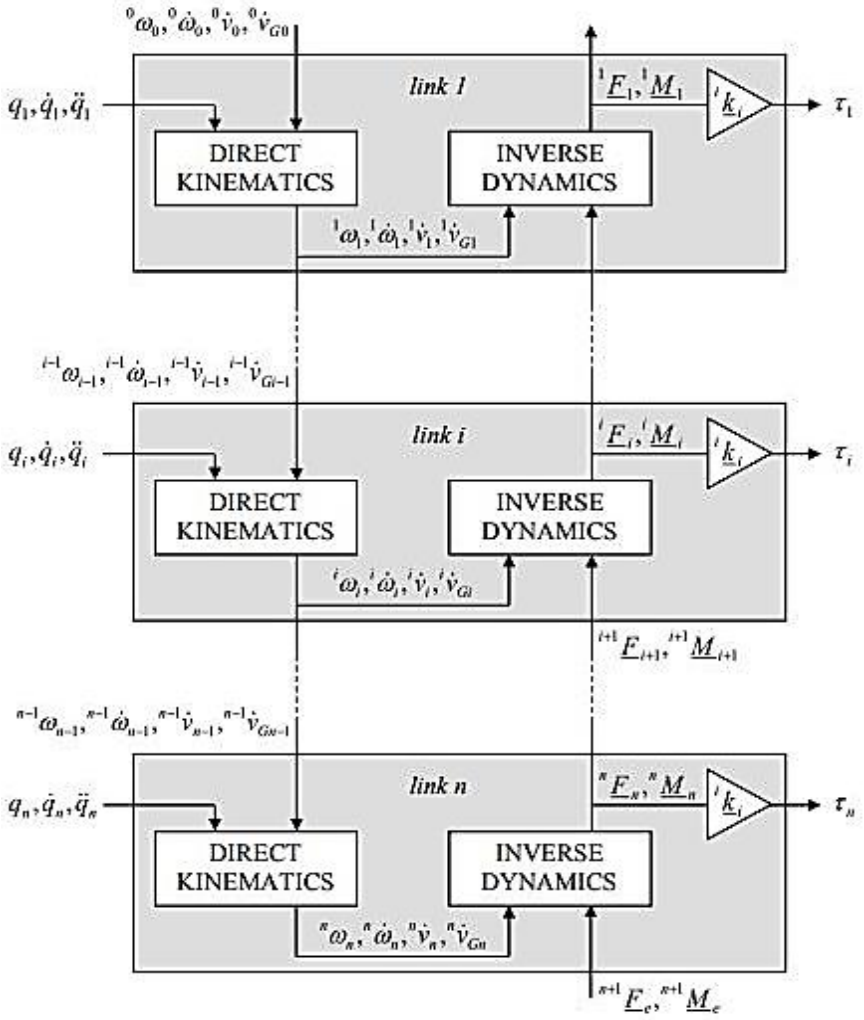


Figura 62: schema risolutivo della dinamica inversa

Durante le varie prove è stato possibile acquisire anche dati relativi alle coppie di target che devono essere applicate per far muovere i singoli link, si decide di utilizzare il modello dinamico per cercare di ricavare questi andamenti risolvendo prima la cinematica diretta e poi, con i valori cinematici in output, risolvere la dinamica inversa. Per poter però realizzare tutto questo è stato necessario individuare:

- i valori delle masse dei link;
- i valori delle posizioni dei baricentri rispetto alle singole terne di D-H modificata;
- i valori dei tensori centrali d'inerzia rispetto alle singole terne di D-H modificata.

Le prime informazioni ricavate dai file di "Universal Robot" sono i valori delle masse e i centri di massa dei link rispetto alle terne di D-H standard.

Tabella 13: massa e coordinate baricentri rispetto le terne di D-H std [4]

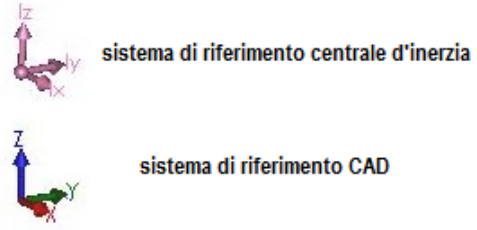
Denavit-Hartenberg parameters for UR3 (standard)						
	Base	Upper Arm	Lower arm	Wrist 2	Wrist 3	Tool Mounting Bracket
mass [Kg]	2	3.42	1.26	0.8	0.8	0.35
center_of_mass [m]	[0,-0.02,0]	[0.13,0,0.1157]	[0.05,0,0.0238]	[0,0,0.01]	[0,0,0.01]	[0,0,-0.02]

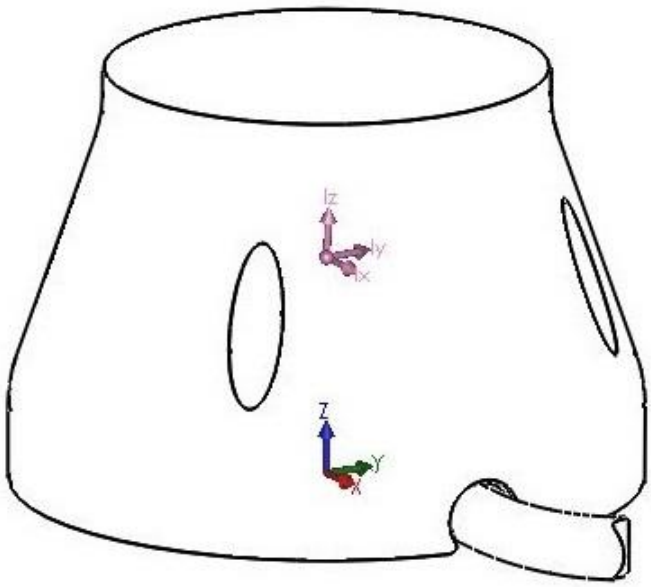
Sono stati individuati per ogni link le posizioni dei centri di massa rispetto alla terna di riferimento D-H modificata e CAD tramite opportune considerazioni geometriche (tabella 14); sono stati poi stimati i valori dei tensori centrali d'inerzia (quindi rispetto alla terna calcolata nel baricentro) utilizzando i componenti CAD conoscendo con esattezza la geometria, le masse e i baricentri.

Tabella 14: centri di massa rispetto le terne di D-H modificata e rispetto le terne CAD

center_of_mass		
	center_of_mass (modify)	**center_of_mass (CAD)**
Base	[0,0,-0.02]	[0,0,0.04595]
Upper Arm	[0.11365,0,0.1157]	[-0.11365,0,0.0617]
Lower arm	[0.16325,0,0.0238]	[-0.16325,0,0.0538]
Wrist 2	[0,-0.01,0]	[0,-0.01,0.04646]
Wrist 3	[0,0.01,0]	[0,0.01,0.0441]
Tool Mounting Bracket	[0,0,-0.02]	[0,0,-0.022]

Si riportano di seguito i singoli link con i sistemi di riferimento opportuni e il calcolo dei tensori centrali d'inerzia e le matrici di orientazione dei sistemi centrali d'inerzia rispetto al sistema di riferimento CAD, stimati tramite il Solidworks.



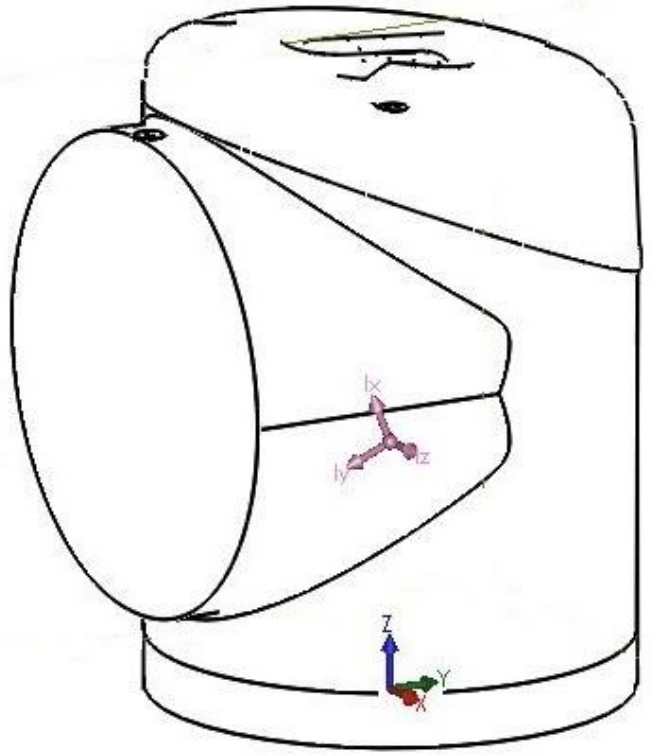


centro di massa sostituito dall'utente: (millimetri)	
X	0,00
Y	0,00
Z	46,22

	asse principale d'inerzia		
Ix	0,99	0,09	-0,08
Iy	-0,09	1	0
Iz	0,08	0	1

momenti principali di inerzia (grammi * millimetri quadrati) nel centro della massa	
Px	3007351,95
Py	3074302,03
Pz	3977399,05

Figura 63: supporto base (link 0)

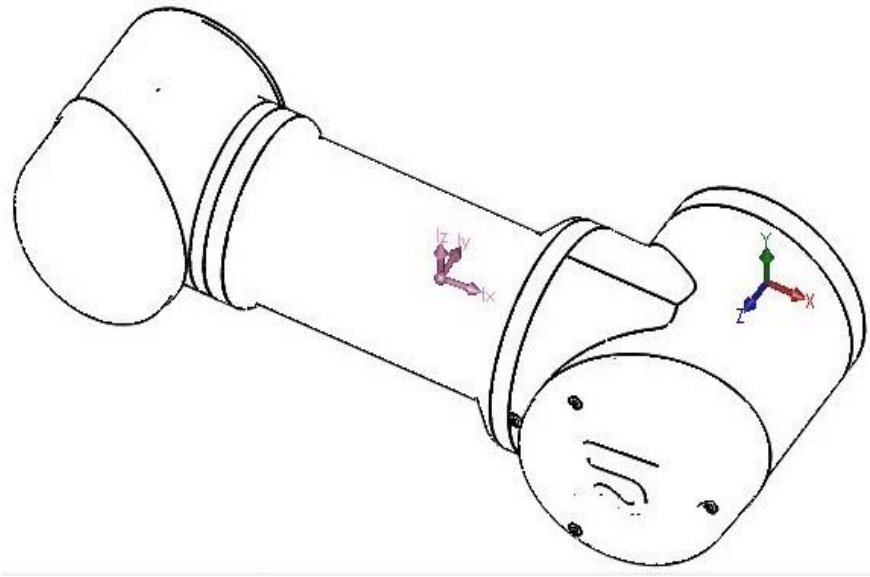


centro di massa sostituito dall'utente: (millimetri)	
X	0,00
Y	0,00
Z	45,95

	asse principale d'inerzia		
Ix	0	-0,38	0,93
Iy	0	-0,93	-0,38
Iz	1	0	0

momenti principali di inerzia (grammi * millimetri quadrati) nel centro della massa	
Px	2307568,63
Py	3078439,42
Pz	3195431,48

Figura 64: base

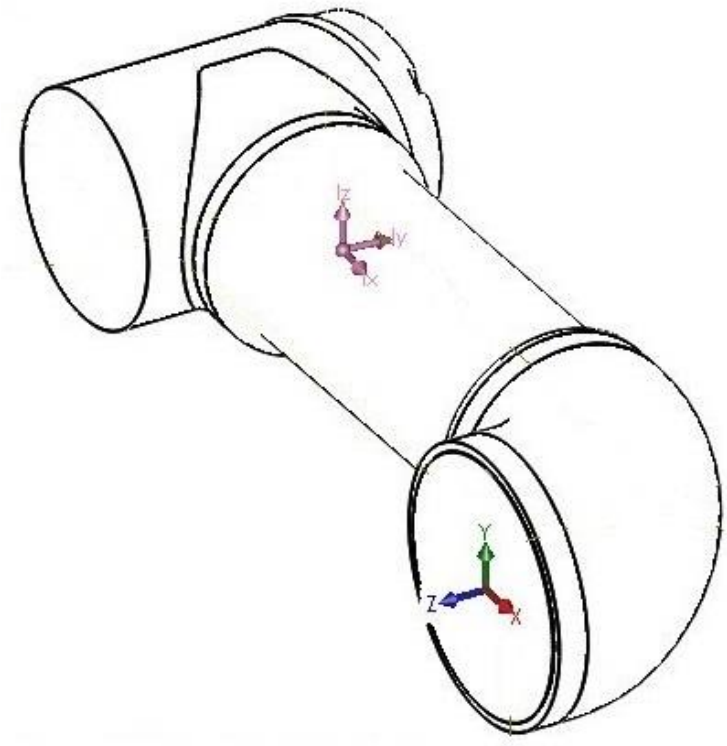


centro di massa sostituito dall'utente: (millimetri)	
X	-133,65
Y	0,00
Z	61,70

	asse principale d'inerzia		
Ix	1,00	0,00	-0,08
Iy	-0,08	0,00	-1,00
Iz	0,00	1,00	0,00

momenti principali di inerzia (grammi * millimetri quadrati) nel centro della massa	
Px	3804371,80
Py	35860952,51
Pz	36746748,18

Figura 65: Upper Arm

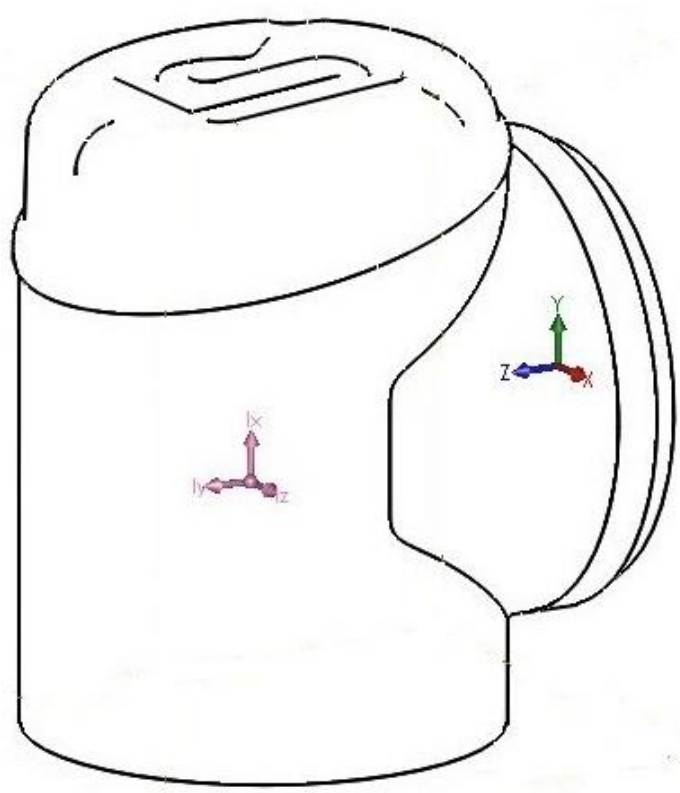


centro di massa sostituito dall'utente: (millimetri)	
X	-163,25
Y	0,00
Z	-53,80

	asse principale d'inerzia		
Ix	1,00	0,00	0,09
Iy	0,09	0,00	-1,00
Iz	0,00	1,00	0,00

momenti principali di inerzia (grammi * millimetri quadrati) nel centro della massa	
Px	809800,99
Py	9376539,48
Pz	9551484,85

Figura 66: Lower Arm

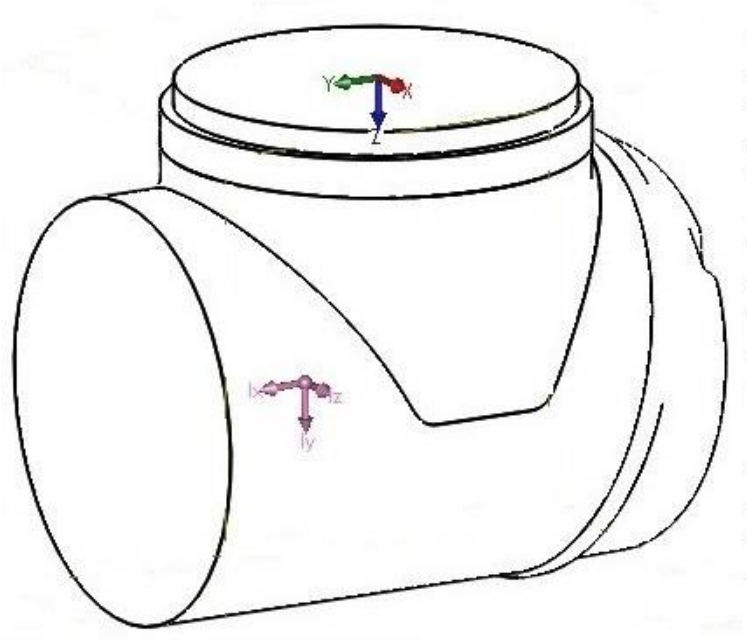


centro di massa sostituito dall'utente: (millimetri)	
X	0,00
Y	-10,00
Z	46,46

	asse principale d'inerzia		
Ix	0,00	1,00	-0,03
Iy	0,00	0,03	1,00
Iz	1,00	0,00	0,00

momenti principali di inerzia (grammi * millimetri quadrati) nel centro della massa	
Px	505334,86
Py	643507,57
Pz	736825,99

Figura 67: Wrist 2

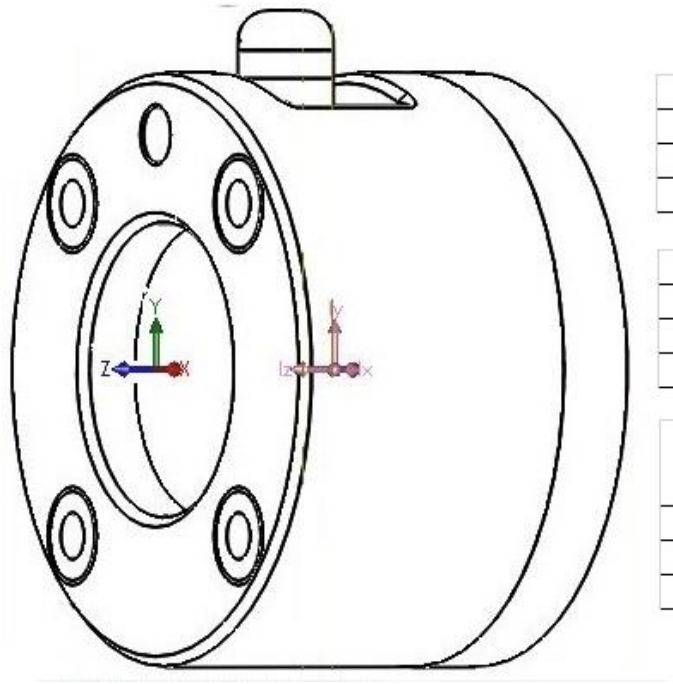


centro di massa sostituito dall'utente: (millimetri)	
X	0,00
Y	10,00
Z	44,10

	asse principale d'inerzia		
Ix	0,00	1,00	0,03
Iy	0,00	-0,03	1,00
Iz	1,00	0,00	0,00

momenti principali di inerzia (grammi * millimetri quadrati) nel centro della massa	
Px	505314,71
Py	643509,30
Pz	736811,49

Figura 68: Wrist 3



centro di massa sostituito dall'utente: (millimetri)	
X	0,00
Y	0,00
Z	-22,00

	asse principale d'inerzia		
Ix	1,00	0,00	0,00
Iy	0,00	1,00	-0,01
Iz	0,00	0,01	1,00

momenti principali di inerzia (grammi * millimetri quadrati) nel centro della massa	
Px	131084,28
Py	131727,13
Pz	179216,80

Figura 69: Tool Mounting Bracket

L'ultimo passaggio è dunque quello di calcolare il tensore principale d'inerzia rispetto alla terna di riferimento di D-H per ogni link.

Per ottenere tale trasformazione bisogna considerare la seguente espressione:

$$^{j}I_{G} = {}^{j}A_{i} * {}^{i}I_{G} * {}^{j}A_{i}^{T}$$

dove:

- $^{j}I_{G}$ = matrice d'inerzia baricentrica rispetto al sistema di riferimento j (rispetto D-H modificata);
- $^{i}I_{G}$ = matrice d'inerzia baricentrica rispetto al sistema di riferimento i (rispetto al baricentro);
- $^{j}A_{i}$ = matrice di trasformazione dal sistema di riferimento i al sistema di riferimento j.

Tale relazione viene ottenuta dalla trasformazione del momento della quantità di moto baricentrico rispetto al sistema di riferimento j nel momento della quantità di moto baricentrico rispetto a i.

Nello specifico caso analizzato la matrice di trasformazione è data da un prodotto di due matrici, poiché bisogna riportare il tensore d'inerzia dal sistema baricentrico a quello di D-H. Una riporta il sistema baricentrico nel sistema CAD e l'altra riporta la grandezza dal sistema CAD al sistema di D-H.

Per cui si può scrivere che:

$$\boldsymbol{^{i}I_{G} = \left(^{i}A_{CAD_i} * {}^{CAD_i}A_{CM_i}\right) * {}^{CM_i}I_{G} * \left(^{i}A_{CAD_i} * {}^{CAD_i}A_{CM_i}\right)^{T}}$$

Nel modello realizzato la dinamica inversa viene risolta tramite la funzione "*rne_mhe_POLITO.m*".

2.4 Rappresentazione virtuale del robot e simulazione V-REP del gripper

Per poter avere una certa visualizzazione grafica del robot viene utilizzato l'ambiente di simulazione *Simulink*. È stato necessario scomporre l'insieme Solidworks dell'UR3 fornito dal sito della casa costruttrice, individuare per ogni link (convertito in formato ".stl") la terna di riferimento CAD e capire quale fosse il legame tra quest'ultima e le terne di riferimento scelte precedentemente con la convenzione di D-H modificata (per quest'ultimo punto si sono effettuate delle misure sull'assieme CAD).

Si riporta di seguito la posizione delle terne CAD per ogni link e le matrici di trasformazione che rimangono costanti qualsiasi sia il movimento del robot e che esplicano il legame tra queste terne e quelle di D-H.

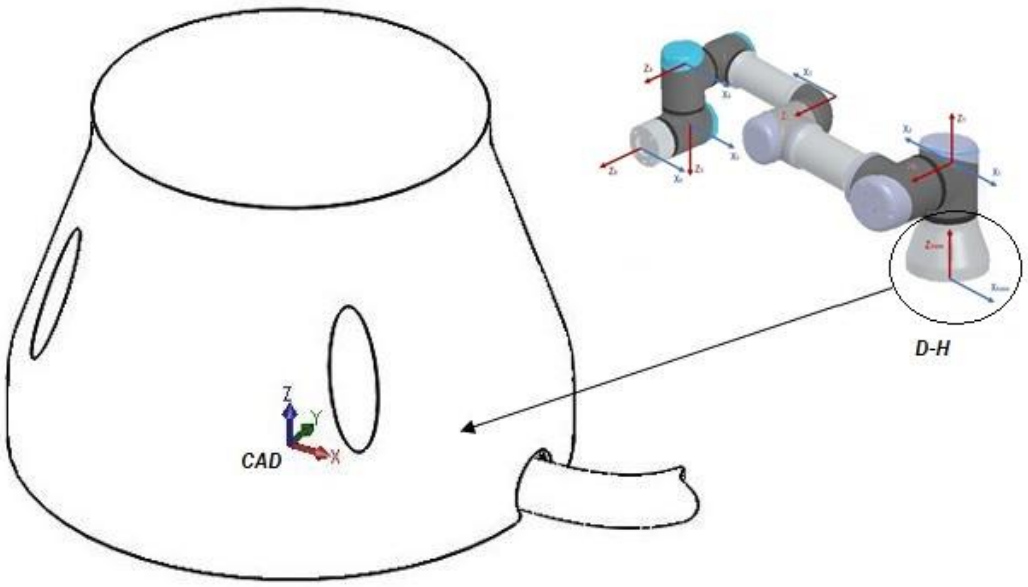


Figura 70: link base, sistema CAD

Tabella 15: matrice di trasformazione del sistema cad rispetto D-H per la base

$^{D-H}A_{CAD}$			
1	**0**	**0**	**0**
0	**1**	**0**	**0**
0	**0**	**1**	**0**
0	**0**	**0**	**1**

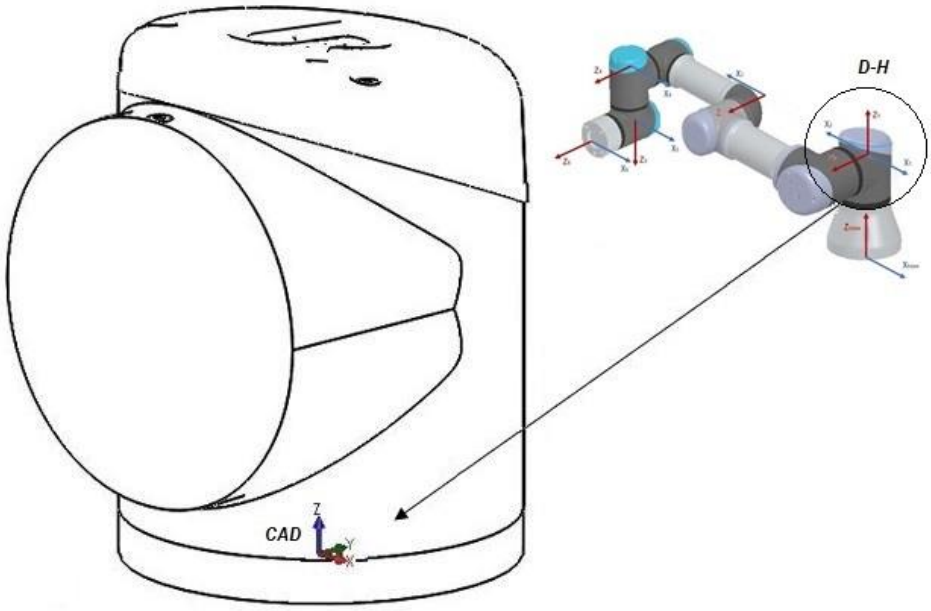


Figura 71: link 1, sistema CAD

Tabella 16: matrice di trasformazione del sistema cad rispetto D-H per il link 1

$^{D-H1}A_{CAD1}$			
1	**0**	**0**	**0**
0	**1**	**0**	**0**
0	**0**	**1**	**-0.06585**
0	**0**	**0**	**1**

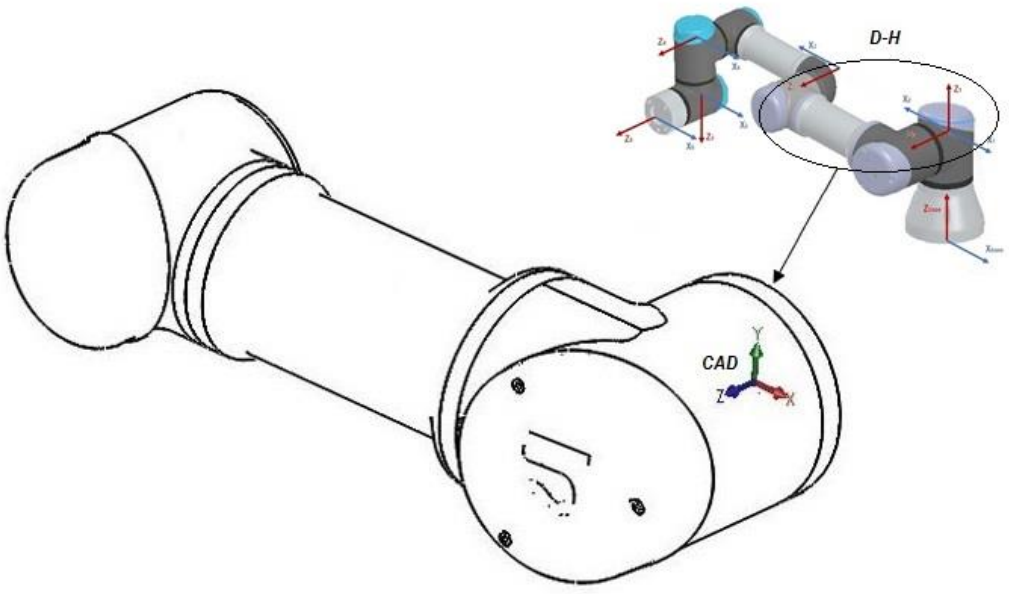


Figura 72: link 2, sistema CAD

Tabella 17: matrice di trasformazione del sistema cad rispetto D-H per il link 2

$^{D-H2}A_{CAD2}$			
-1	**0**	**0**	**0**
0	**-1**	**0**	**0**
0	**0**	**1**	**0.054**
0	**0**	**0**	**1**

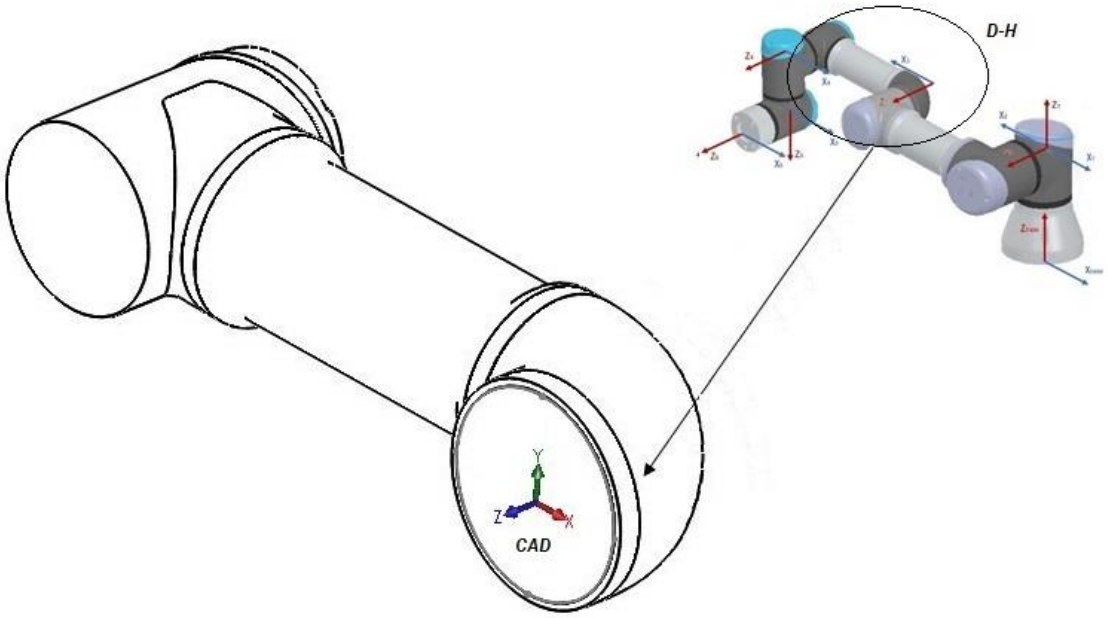


Figura 73: link 3, sistema CAD

Tabella 18: matrice di trasformazione del sistema cad rispetto D-H per il link 3

$^{D-H3}A_{CAD3}$			
-1	**0**	**0**	**0**
0	**-1**	**0**	**0**
0	**0**	**1**	**0.0776**
0	**0**	**0**	**1**

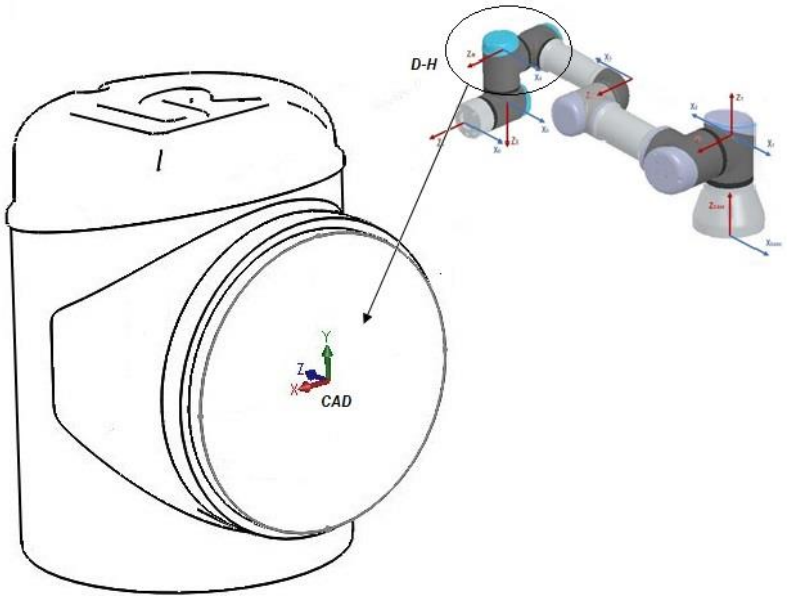


Figura 74: link 4, sistema CAD

Tabella 19: matrice di trasformazione del sistema cad rispetto D-H per il link 4

$^{D-H4}A_{CAD4}$			
1	**0**	**0**	**0**
0	**1**	**0**	**0**
0	**0**	**1**	**-0.04646**
0	**0**	**0**	**1**

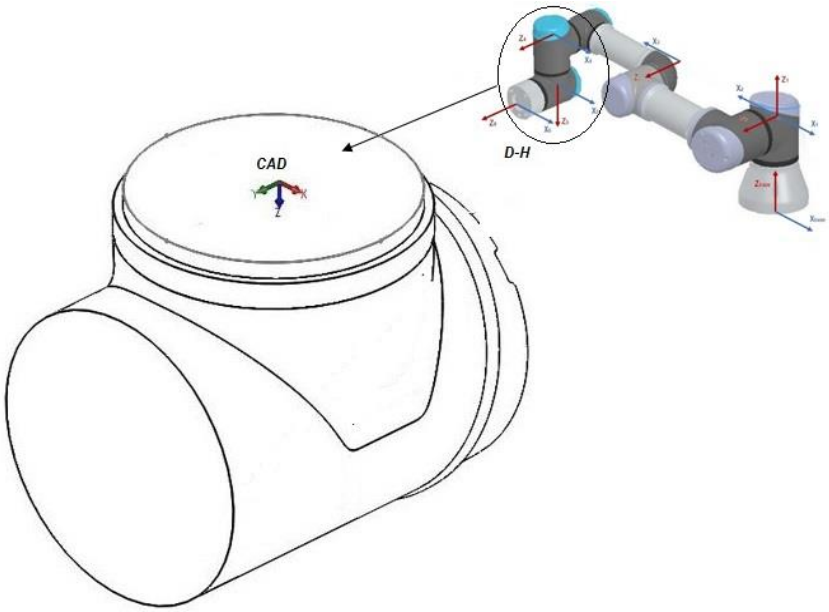


Figura 75: link 5, sistema CAD

Tabella 20 : matrice di trasformazione del sistema cad rispetto D-H per il link 5

$^{D-H5}A_{CAD5}$			
1	**0**	**0**	**0**
0	**1**	**0**	**-0.002**
0	**0**	**1**	**-0.0441**
0	**0**	**0**	**1**

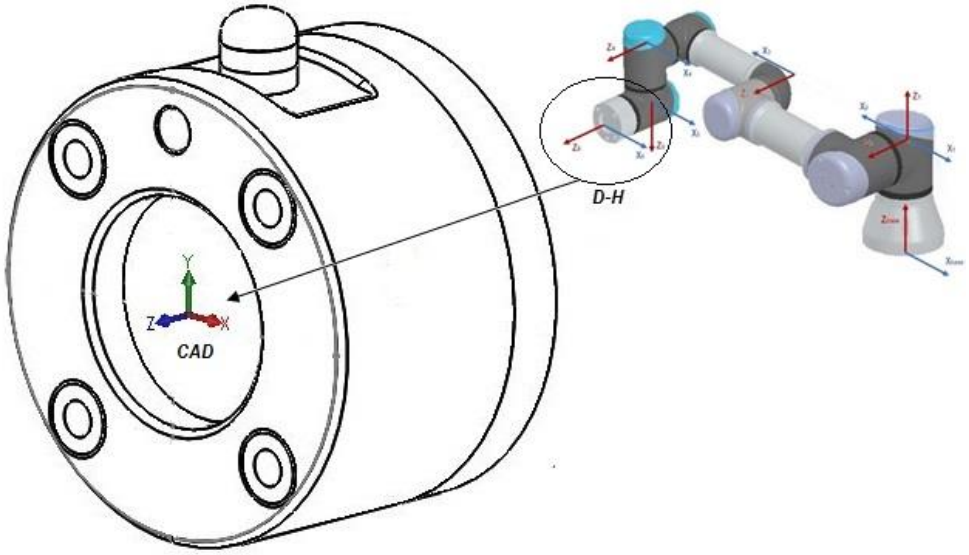


Figura 76: link 6, sistema CAD

Tabella 21 : matrice di trasformazione del sistema cad rispetto D-H per il link 6

$^{D-H6}A_{CAD6}$			
-1	**0**	**0**	**0**
0	**-1**	**0**	**0**
0	**0**	**1**	**-0.002**
0	**0**	**0**	**1**

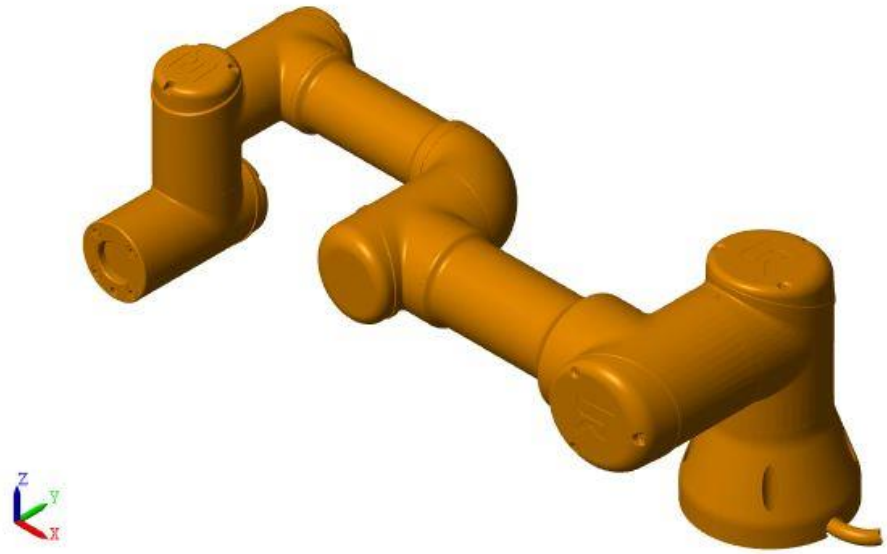


Figura 77: visualizzazione del robot in ambiente simulink in configurazione tutti zeri

Figura 78: visualizzazione del robot in simulink configurazione verticale

Successivamente a tale modello sono stati aggiunti due ulteriori componenti, ovvero una cella di carico e un gripper a due dita e sono stati resi solidali al link 6. Per le analisi svolte successivamente, se si volesse considerare anche la presenza di questi due componenti aggiuntivi, dovrebbero essere rivalutati i valori delle masse, delle inerzie e dei baricentri dell'ultimo link.

Figura 79: cella di carico e gripper montati sul braccio

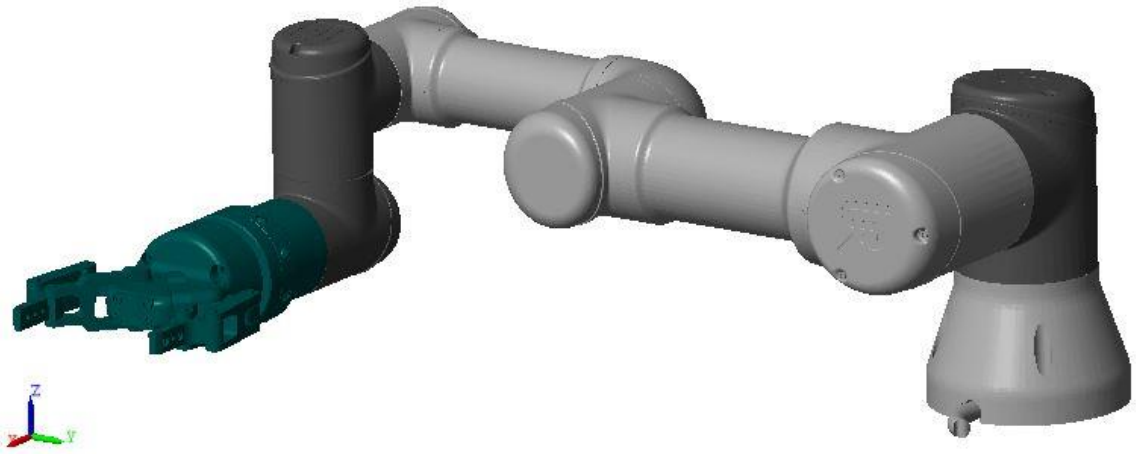


Figura 80: visualizzazione del braccio con gripper e cella di carico in ambiente simulink

Nell'immagine in figura 80 sono stati cambiati alcuni colori dei link, per rendere la visualizzazione della macchina più simile alla composizione reale.

Durante lo studio di questo sistema meccanico è stato utilizzato un ulteriore ambiente di simulazione. Questa esigenza nasce dal desiderio di simulare e visualizzare dei movimenti mediante l'uso del gripper prima che questo componente venisse montato realmente sul robot.

L'ambiente di simulazione V-REP permette di fare tutto questo in maniera facile ed efficace. Bisogna specificare che questo software è stato utilizzato in due modi differenti:

- viene utilizzato da visualizzatore di movimenti per quanto riguarda i giunti del robot, infatti il sistema è comandato direttamente da istruzioni scritte in ambiente URSim, ovvero il simulatore dell'ambiente di programmazione PolyScope di UR, e tramite l'ausilio del Matlab, comunicate al V-REP;
- viene utilizzato da simulatore per quanto riguarda le funzioni della pinza, infatti vengono programmati degli script scritti in V-REP per poter permettere il giusto funzionamento del gripper.

Dunque, l'utilizzo del V-REP è stato utile per poter simulare una sorta di sensoristica e perciò implementare degli script in ambiente PolyScope, in modo che alcuni movimenti avvenissero solamente in seguito alla chiusura e all'apertura della pinza in V-REP. Quindi si può riassumere l'ordine di lancio dei programmi e la logica del movimento nella seguente immagine:

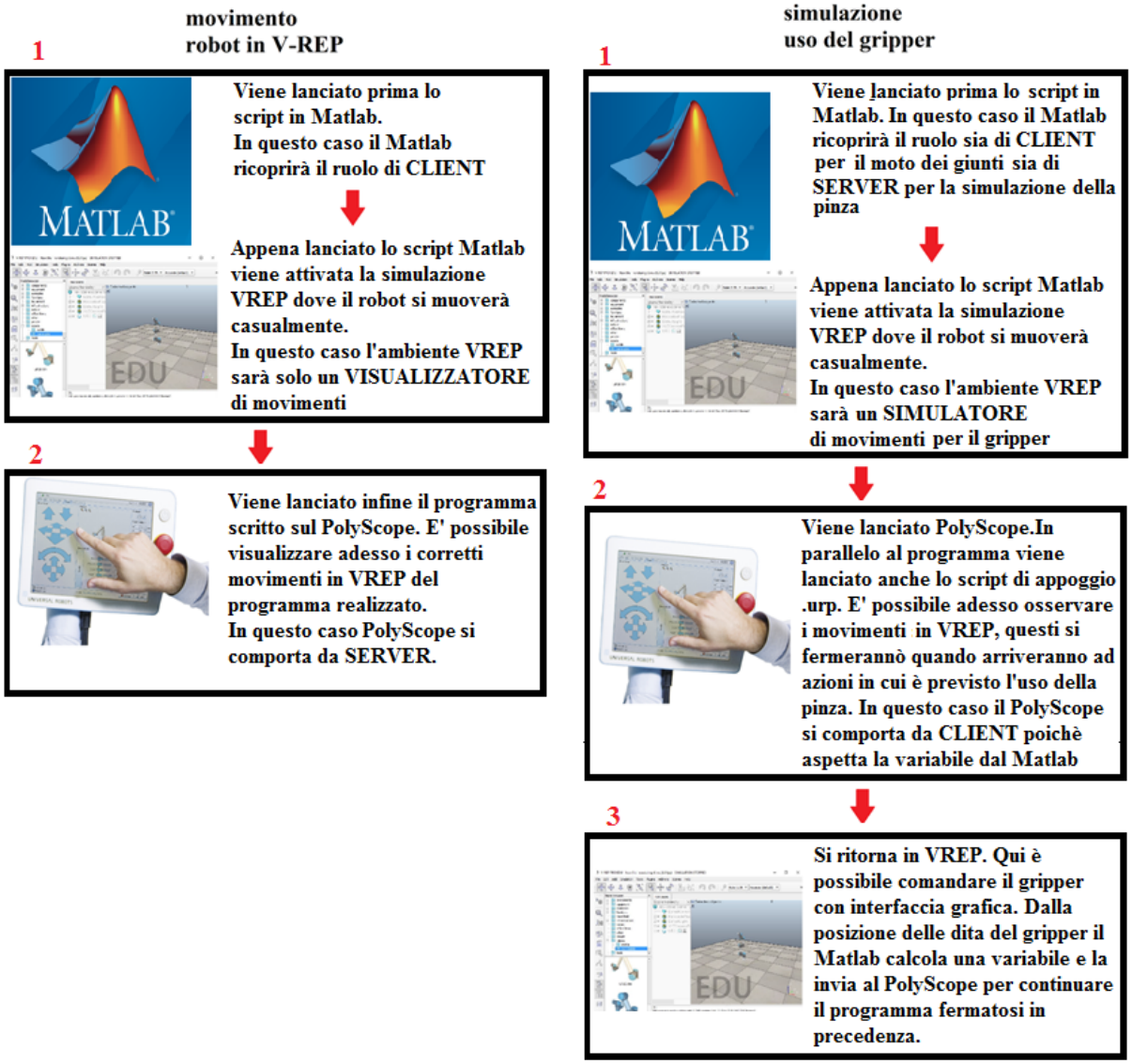


Figura 81: funzione dei software per i diversi movimenti e ordine di lancio

I movimenti riprodotti in V-REP sono stati quelli dell'azione di pick and place di default, presente nella schermata principale del Polyscope e i due movimenti visti nel capitolo precedente, ovvero la palletizzazione e lo stack.

Per far comunicare il PolyScope con il V-REP sono stati scritti degli script in Matlab che permettono di aprire dei socket di comunicazione, dove quest'ultimo software funge una volta da client, quando riceve i dati, e una volta da server quando invia variabili al simulatore.

Infatti, per simulare un sensore nel simulatore del robot si è deciso di inviare una variabile booleana calcolata al Matlab. Questa variabile viene calcolata sfruttando i dati di posizione dei componenti della pinza presenti in V-REP nel seguente modo:

1. si prende in considerazione la posizione delle due dita del gripper su V-REP;
2. se ne calcola la distanza;
3. si considera la larghezza del pallet da prelevare e gli si impone una certa tolleranza dimensionale;
4. se la distanza in modulo calcolata del gripper è leggermente più grande di questa larghezza di target allora la variabile booleana assumerà il valore di 0 e ciò indicherà che il gripper è aperto;
5. se la distanza in modulo calcolata del gripper è leggermente più piccola di questa larghezza di target allora la variabile booleana assumerà il valore di 1 e ciò indicherà che il gripper e chiuso.

Ovviamente durante i movimenti è possibile aprire e chiudere il gripper grazie ad un'interfaccia grafica presente in ambiente V-REP

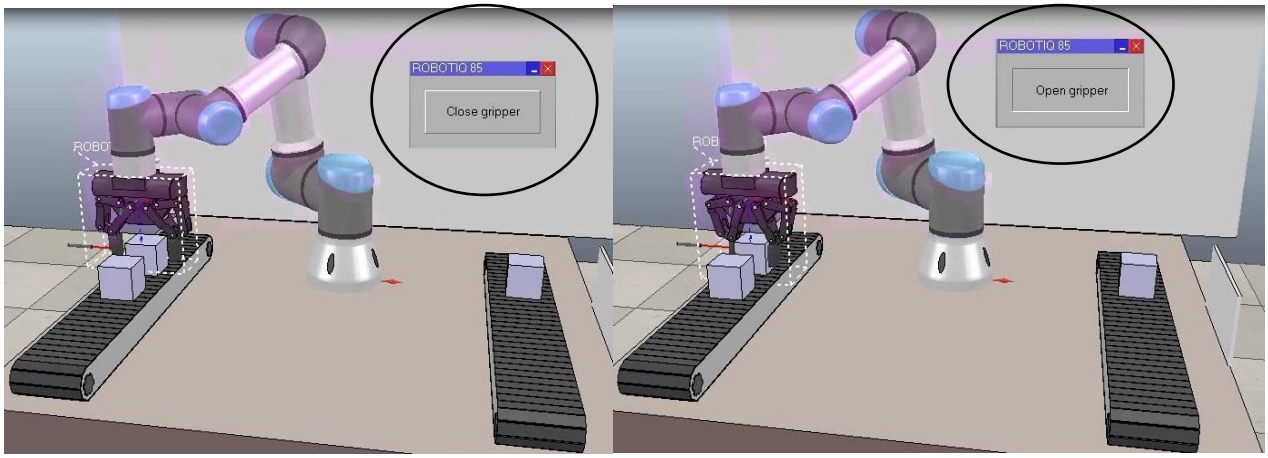


Figura 82: finestra grafica in V-REP per controllo della pinza

Si decide di riportare parte del codice scritto per comprendere come viene creata la connessione e la comunicazione tra i vari software. Sono stati scritti tre programmi in ambiente Matlab in cui a cambiare sono solo alcuni comandi legati al tipo di movimento da riprodurre, sono stati codificati degli script in ambiente V-REP per poter simulare i movimenti della pinza e dei differenti elementi presenti nelle scene

come sensori o nastri trasportatori ed infine, oltre ai singoli movimenti programmati nell'ambiente proprio del robot, uno script di appoggio in questo stesso ambiente da utilizzare nel momento in cui il Matlab si comporta da server.

Le prime righe di codice riportate di seguito spiegano come avviene la connessione tra il software dell'UR e il V-REP tramite Matlab. È possibile trovarle nei programmi scritti "*pick_and_place_command.m*", "*pallet_command.m*" e "*stack_command.m*".

```
%% Communication
close all
IP='192.168.56.101'; %IP simulator
%IP='192.168.56.103'; %IP Robot
```
1

```
% disp('inserire larghezza del pallet in metri'); % 0.05
% size=input('');
size=0.05;
```
2

```
fig = uifigure('Position',[100 500 200 150]);
GUIswitch=uiswitch(fig,'toggle',...
    'Items',{'Stop','Start'},...
    'Position',[95 60 20 45],...
    'ValueChangedFcn',@(GUIswitch,event) switchMoved(event)); %si può usare
anche con fig,'toggle'
```
3

```
… %creazione di vettori vuoti per il salvataggio dei valori calcolati nel
while
```

```
vrep=remApi('remoteApi');  % using the prototype file (remoteApiProto.m)
vrep.simxFinish(-1);       % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19997,true,true,5000,5);
if (clientID>-1)
disp('Connected to remote API server');
% vrep.simxSynchronous(clientID,true);
vrep.simxStartSimulation(clientID,vrep.simx_opmode_blocking);
```
4

```
%handle
[Joint1,Joint2,Joint3,Joint4,Joint5,Joint6]=VREPHandle_UR3(clientID,vrep);
[rtCode,gripper]=vrep.simxGetObjectHandle(clientID,'ROBOTIQ_85',vrep.simx_o
pmode_blocking);

… %comandi di tipo vrep.sim

% code
[rtCode,pos_gripper]=vrep.simxGetObjectPosition(clientID,gripper,-
1,vrep.simx_opmode_streaming);
… %comandi di tipo vrep.sim
```
5

```
input('Premere Start su GUI e poi invio per iniziare ad acquisire.');
Status=switchMoved(GUIswitch);
disp('inserire 0 per muovere soltanto oppure 1 per mandare comandi')
dataorcommand=input('');
if dataorcommand==1
Connection2UR_v1(IP,'on'); %connessione TCP/IP
```
6

```
elseif dataorcommand==0
Connection2UR_v1(IP,'on','D'); %connessione TCP/IP
end
t_tot=tic;
```

```
    while Status == 1
    t_invio=tic;
    Data{i}=URStream_v1;
    vrep.simxPauseCommunication(clientID,1);
```

7

```
… %comandi per la movimentazione del robot sfruttando i comandi dal
software     dell'UR
```

```
[rtCode,pos_dito_sinistro]=vrep.simxGetObjectPosition(clientID,dito_sinistr
o,-1,vrep.simx_opmode_buffer);
[rtCode,pos_dito_destro]=vrep.simxGetObjectPosition(clientID,dito_destro,-
1,vrep.simx_opmode_buffer);
pospianoditoS(i)=sqrt(((pos_dito_sinistro(1,1))^2)+((pos_dito_sinistro(1,2)
)^2))
pospianoditoD(i)=sqrt(((pos_dito_destro(1,1))^2)+((pos_dito_destro(1,2))^2)
);

   if abs(pospianoditoS(i)-pospianoditoD(i))<=(size+0.026)
      sensorePresa=1;
   else
      sensorePresa=0;
   end
```

8

```
   if dataorcommand==1
             VAR2URa(sensorePresa);
              disp(sensorePresa);
   elseif dataorcommand==0
              disp(sensorePresa);
   end
```

9

```
   deltat(i,1)=toc(t_invio);
   i=i+1;
   pause(0.0000001)
   Status=switchMoved(GUIswitch);
      end
      vrep.simxStopSimulation(clientID,vrep.simx_opmode_blocking);
```

10

11

```
% Now close the connection to V-REP:
        vrep.simxFinish(clientID);
    else
        disp('Failed connecting to remote API server');
end
TOT=toc(t_tot);
```

12

```
if dataorcommand==1
Connection2UR_v1(IP,'off'); %connessione TCP/IP
elseif dataorcommand==0
Connection2UR_v1(IP,'off','D'); %connessione TCP/IP
end
```

13

Il significato dei vari blocchi è il seguente:

1 comunicare l'IP del robot o simulatore;
2 specificare la dimensione (larghezza) del pallet da prelevare;
3 si crea come visto precedentemente una GUI con un tasto on off per poter avviare la simulazione;
4 uso delle funzioni remote API che permettono di utilizzare comandi di V-REP in ambienti differenti;
5 si creano delle variabili riferite a dei particolari componenti del robot di V-REP e delle variabili in cui vengono salvate informazioni come la posizione;
6 si apre la connessione in seguito ad un input da tastiera e dalla GUI. La connessione viene aperta mediante la funzione "*Connection2UR_v1*". Si apre un if per decidere se usare questa funzione solo per mandare comandi al V-REP (senza D nell'argomento della funzione) oppure anche per usare come Matlab come server (con la D nell'argomento della funzione);
7 si apre un ciclo while che permette di eseguire l'operazione di acquisizione e di invio dati finché non viene spostato il pulsante della GUI su off. Appena si entra nel ciclo viene utilizzata la funzione "*URStream_v1.m*" che permette di salvare per ogni iterazione una structure con i dati acquisibili;
8 viene calcolata la distanza tra le due dita del gripper e si impone una condizione nell'if legata a questa distanza. Viene creata una variabile booleana "sensorePresa": quando assume valore 1 significa che la pinza è chiusa ed ha prelevato l'oggetto mentre se assume valore 0 la pinza è aperta ed ha rilasciato l'oggetto;
9 se la funzione "*Connection2UR_v1.m*" viene utilizzata anche per inviare dati dal Matlab (quindi con la D nell'argomento) viene lanciata la funzione "*VAR2URa.m*" che apre un socket di comunicazione tra Matlab, utilizzato come server, e il software dell'UR, utilizzato come client;
10 viene utilizzato il comando pause per far compiere al ciclo while delle brevissime soste (un intervallo di tempo ben più piccolo del tempo di acquisizione dati) per poter verificare ad ogni iterazione lo stato della GUI;
11 interruzione della simulazione su V-REP;
12 chiusura connessione con V-REP;
13 chiusura socket di comunicazione con il software dell'UR.

Viene riportato nella pagina seguente lo script di appoggio scritto nell'ambiente di programmazione dell'UR:

```
"BeforeStart"
ip=-'192.168.56.1'
port=30004
delta_t=0.008
t_out=0.2

"Robot Program"
ip="192.168.56.1"
port=30004
delta_t=0.008
t_out=0.2
socket_open(ip,port,"COMMAND")
socket_send_string("x","COMMAND")
  while (Loop_2 < 1000000)
     command=socket_read_ascii_float(1,"COMMAND",t_out)
     data_input=[command[1]]
       if (command[1] == 1):
          SENSORE=1
       ElseIf command[1]=0"
          SENSORE=0
Wait: 0.0
socket_close("COMMAND")
```

Questo script risulta essere molto semplice e richiama l'IP del server e il nome della porta di comunicazione. Viene aperto il socket di comunicazione "COMMAND" in cui Matlab scrive ad ogni iterazione una variabile che viene letta tramite questo programma, con il comando "socket_read_ascii_float". In questo stesso ambiente tale lettura fornisce una valida condizione all'if che permette di creare la variabile booleana "SENSORE", che viene salvata nel workspace delle variabili di UR. Quando questa assume valore 1 la pinza è aperta, al contrario risulta essere chiusa. Tale script viene lanciato in parallelo ad ogni programma scritto sul teach pendant, attraverso il comando visto in precedenza "THREAD". In questo modo si può simulare il segnale di apertura e chiusura della pinza e far attendere il programma principale quando il robot è in determinati waypoint di prelievo o rilascio.

Nei seguenti esempi vengono riportati alcuni script codificati in V-REP per illustrare brevemente la logica con cui viene programmata la pinza in questo ambiente di simulazione.

- *MOVIMENTO DI PICK AND PLACE:*

In questo tipo di movimento vengono simulati dei componenti aggiuntivi che sono:

- il sensore laser di posizione utilizzato come sensore "presenza pezzo" che legge una certa distanza quando il pezzo è arrivato nella posizione di prelievo mentre ne legge delle altre, più grandi, quando il pezzo è spostato;
- i due nastri trasportatori i quali sono stati programmati per muoversi e fermarsi in un certo modo e con una certa velocità in base ai segnali provenienti dal sensore e dalla posizione del gripper.

La logica con cui è stato simulato tale movimento è semplicemente far muovere i nastri fino a quando il sensore laser non legge una distanza minima, in caso contrario entrambi restano fermi e in attesa di riprendere la loro corsa solo nel momento in cui il pallet è spostato dal robot.

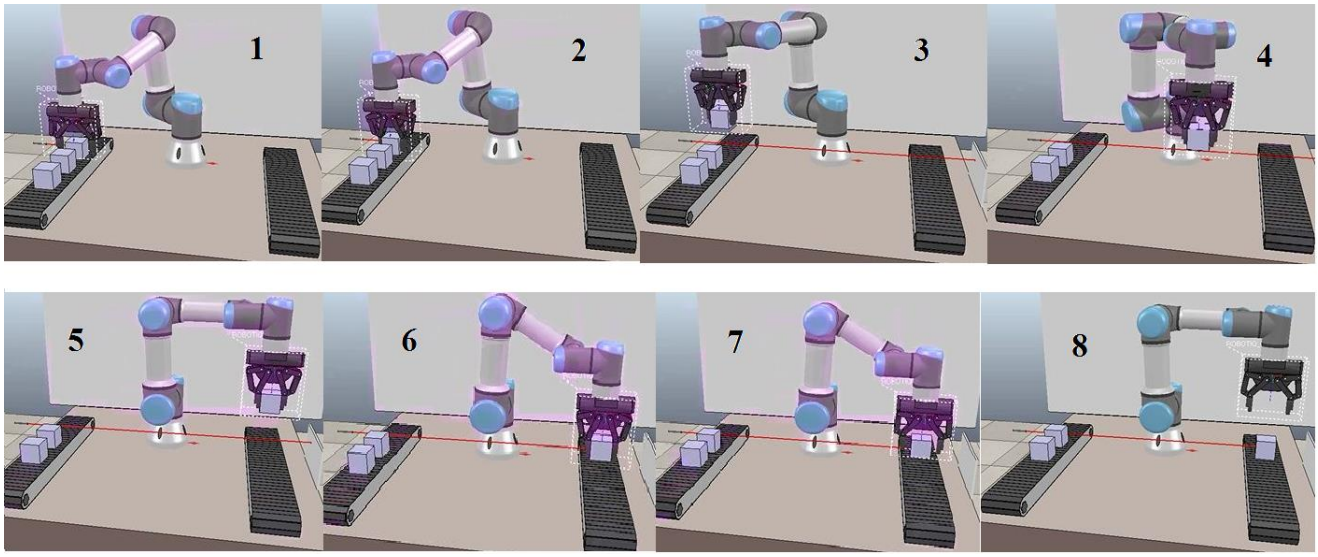


Figura 83: wireframe dello spostamento del primo pallet

Il movimento è simile per gli altri pallet l'unica cosa che varia è la posizione di quest'ultimi sui nastri che si spostano per:

- il primo per poter portare in posizione di prelievo il successivo pallet;
- il secondo per poter creare dello spazio di deposito per il successivo pallet.

Viene riportato, solo per questo movimento, lo script scritto per realizzare l'apertura e chiusura della pinza (*Gripper pick and place.txt*). È possibile consultare gli altri script in formato digitale (.txt).

```
function sysCall_init()
    ui=simGetUIHandle('ROBOTIQ_85_UI')
    j1=sim.getObjectHandle('ROBOTIQ_85_active1')
    j2=sim.getObjectHandle('ROBOTIQ_85_active2')
    pallet1=sim.getObjectHandle('Cuboid1')
    pallet2=sim.getObjectHandle('Cuboid2')
    pallet3=sim.getObjectHandle('Cuboid3')
    table=sim.getObjectHandle('customizableTable')
end
```

Inizializzazione delle variabili:

Alcuni componenti della scena vengono codificati e inizializzati come variabili tramite la funzione sim.getObjectHandle

```
function sysCall_actuation()
    closing=sim.boolAnd32(simGetUIButtonProperty(ui,3),sim.buttonproperty_isdown)~=0
    p1=sim.getJointPosition(j1)
    p2=sim.getJointPosition(j2)
    P1=sim.getObjectPosition(pallet1,3)
    P2=sim.getObjectPosition(pallet2,3)
    P3=sim.getObjectPosition(pallet3,3)
```

Con le due funzioni sim.getJointPosition e sim.getObjectPosition è possibile salvare in delle variabili le posizioni di giunti o componenti del sistema

```
    --1 pallet
      if (closing) and (P2[2]>=0.23) and (P3[2]>=0.334) then
      sim.setObjectParent(pallet1,connector,true)
      end
      if (closing==false) and (P3[2]<=0.334) then
      sim.setObjectParent(pallet1,table,true)
      end

    --2 pallet
      if (closing) and (P2[2]<=0.14) and (P3[2]>=0.23) and (P1[1]>=0.275) then
      sim.setObjectParent(pallet2,connector,true)
      end
      if (closing==false) and (P3[2]<=0.234) then
      sim.setObjectParent(pallet2,table,true)
      end

    --3 pallet
      if (closing) and (P1[1]>=0.275) and (P2[1]>=0.275) and (P3[2]<=0.14) then
      sim.setObjectParent(pallet3,connector,true)
      end
      if (closing==false) then
      sim.setObjectParent(pallet3,table,true)
      end
```

È illustrata la logica di programmazione con cui i pallet vengono mossi. La funzione sim.setObjectParent simula il collegamento tra due oggetti della scena. Quindi sotto determinate condizioni (legate alla posizione sul piano dei pallet) a seconda che la pinza sia chiusa o aperta (comandabile con la GUI codificata in queste righe) si simula la presa dell'oggetto. Nel momento del rilascio il pallet viene "collegato" sempre tramite la stessa funzione al piano.

```
      if (closing) then
        if (p1<p2-0.008) then
        sim.setJointTargetVelocity(j1,-0.01)
        sim.setJointTargetVelocity(j2,-0.04)
        else
        sim.setJointTargetVelocity(j1,-0.04)
        sim.setJointTargetVelocity(j2,-0.04)
        end
      else
        if (p1<p2) then
        sim.setJointTargetVelocity(j1,0.04)
        sim.setJointTargetVelocity(j2,0.02)
        else
        sim.setJointTargetVelocity(j1,0.02)
        sim.setJointTargetVelocity(j2,0.04)
        end
end
```

Viene codificata infine la velocità di apertura e chiusura della pinza.

Figura 84: programmazione chiusura e apertura della pinzain V-REP

Gli altri due file programmati sono:

- laser pick and place .txt;
- nastro trasportatore n1 pick and place .txt;
- nastro trasportatore n2 pick and place .txt.

- *MOVIMENTO DI PALLETIZZAZIONE*

In questo movimento è previsto lo spostamento dei pallet che arrivano su di un nastro traportatore e devono essere collocati in un pallet (in figura 85 le condizioni iniziali e finali del movimento).

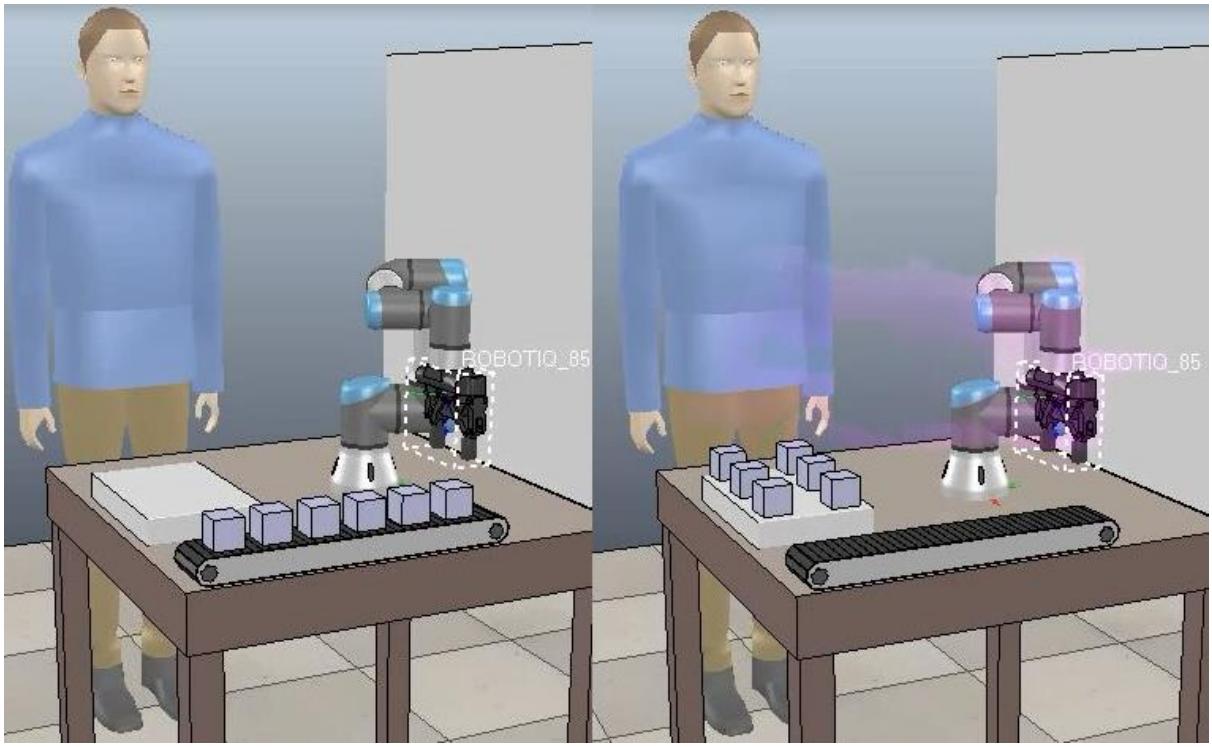


Figura 85: posizione iniziale e finale movimento

In questo tipo di movimento troviamo gli stessi elementi che sono stati usati in precedenza, quindi il nastro trasportatore, la cui velocità è stata codificata in funzione della posizione dei pallet che vengono spostati.
Inoltre, la pedana su cui vengono spostati i pallet ha delle dimensioni specifiche in modo che in fase di deposito pezzo sul piano di riferimento, la pinza non urta gli altri elementi.

Nell'immagine successiva viene fatto il wire-frame del deposito del pallet centrale, dove si può osservare sia come man mano il nastro trasportatore si muova e posizioni gli altri elementi per l'operazione di prelievo, sia come la pinza, scendendo per il deposito sul pallet, non urti quelli già posizionati.

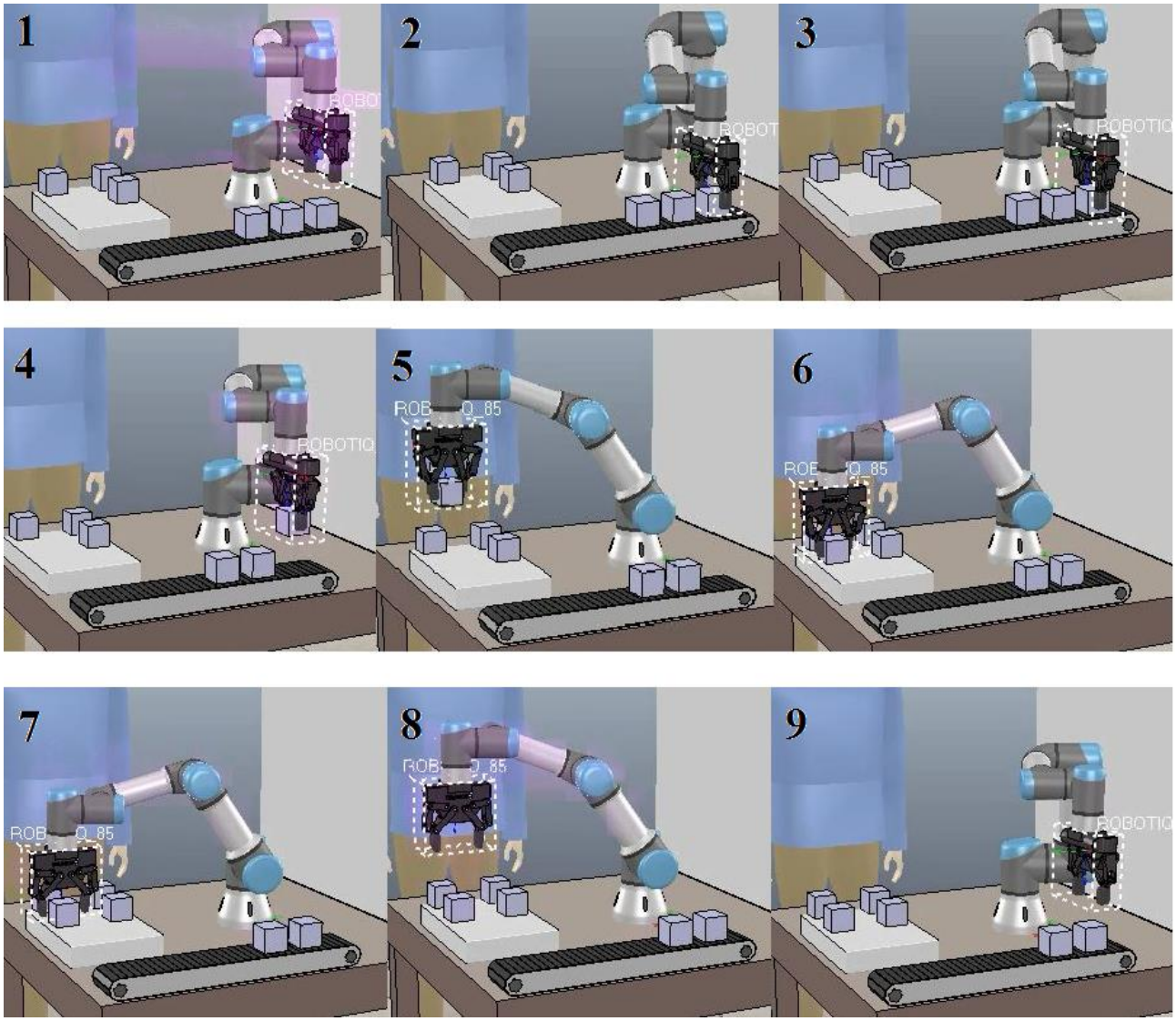


Figura 86: palletizzazione spostamento del quarto pallet

I file utilizzati per questo movimento sono:

- gripper pallet .txt;
- nastro trasportatore pallet .txt.

- *MOVIMENTO DI STACK*

Si ricorda che questo movimento può permettere sia di prelevare sia di depositare oggetti posizionati uno sull'altro. Non vi è alcun nuovo elemento programmato. Di seguito vengono riportate gli stati iniziali e finali del movimento e un wire-frame dello spostamento del secondo pallet.

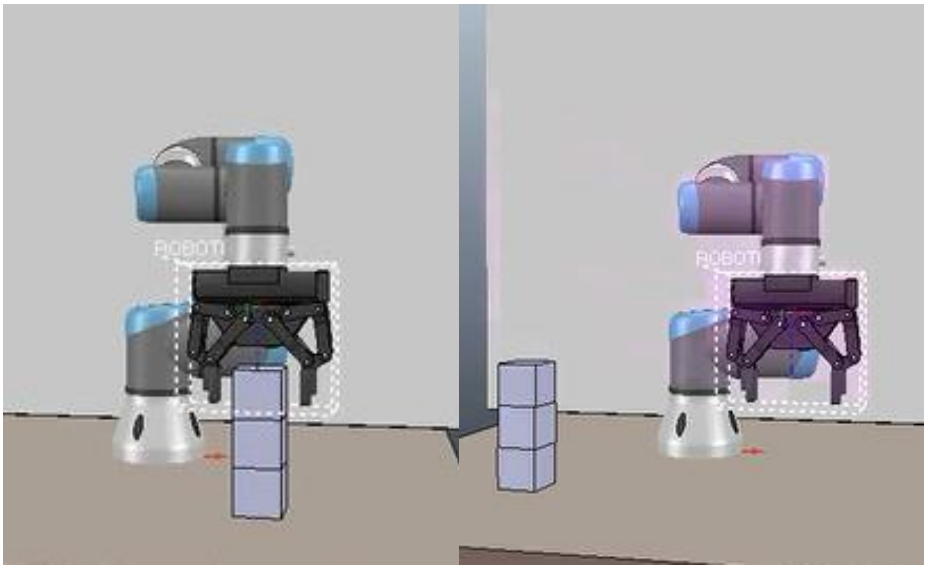


Figura 87: condizione iniziale e finale spostamento

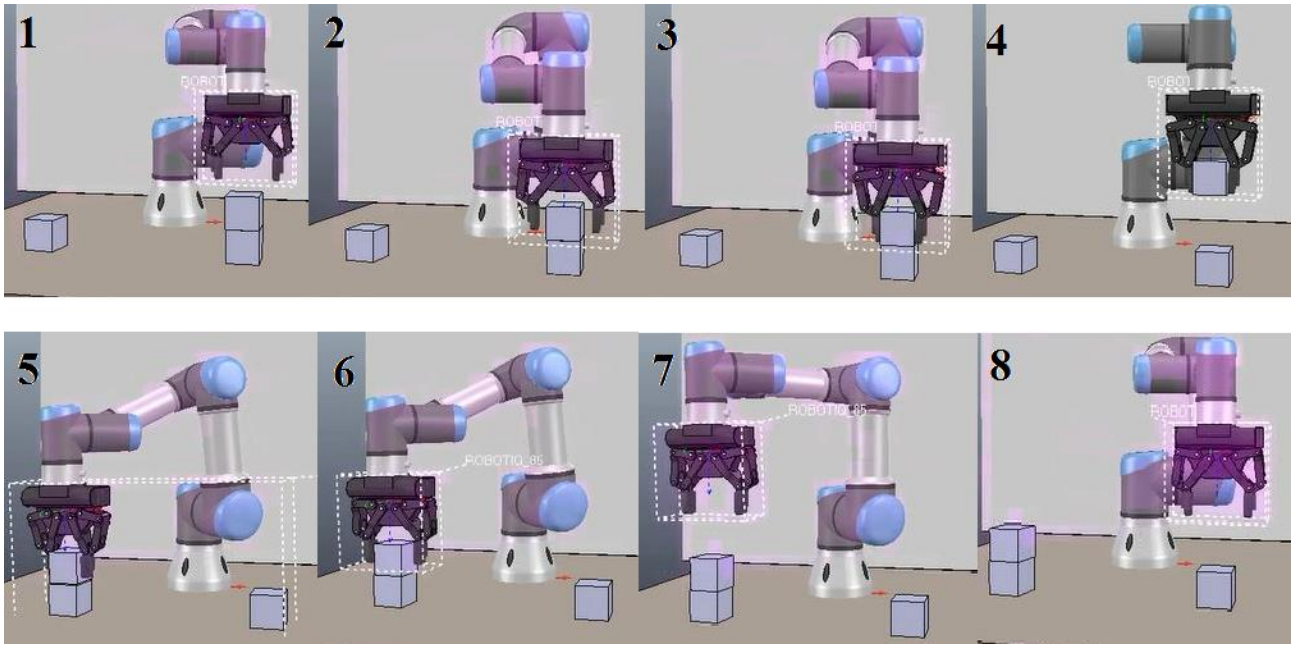


Figura 88: wire-frame stake, spostamento secondo pallet

In questo movimento è stato utilizzato il seguente script in V-REP:

- gripper stack .txt.

3. PROVE E CONFRONTO TRA IL MODELLO ED ACQUISIZIONE

Per poter affrontare un'analisi più approfondita dei movimenti e del modello realizzato viene creata una libreria di tutte le acquisizioni effettuate relative ai programmi fatti ed eseguiti dal robot (per individuare la posizione delle prove e la descrizione visualizzare gli indici in appendice). Tra tutte queste prove, alcune sono state analizzate e confrontate con i risultati ottenuti dal modello multi-body sia dal punto di vista cinematico sia dal punto di vista dinamico. È importante effettuare questo studio per garantire il corretto funzionamento di questo modello e, così facendo, validarlo.

I diversi risultati ottenuti attraverso l'analisi fatta vengono riportati di seguito.

- **Programma scritto:** PROVA_1_1_moveJ_velocità_default_giunto1.urp
  Acquisizione corrispondente: ACQUISIZIONE_1_1_moveJ_velocità_default_giunto1.mat

Questa prova consiste nel far muovere, mediante un moveJ, solo il primo giunto da una configurazione iniziale q_iniziale [0 ,0 ,0 ,0 ,0 ,0] ad una finale q_finale [90° ,0 ,0 ,0 ,0 ,0]

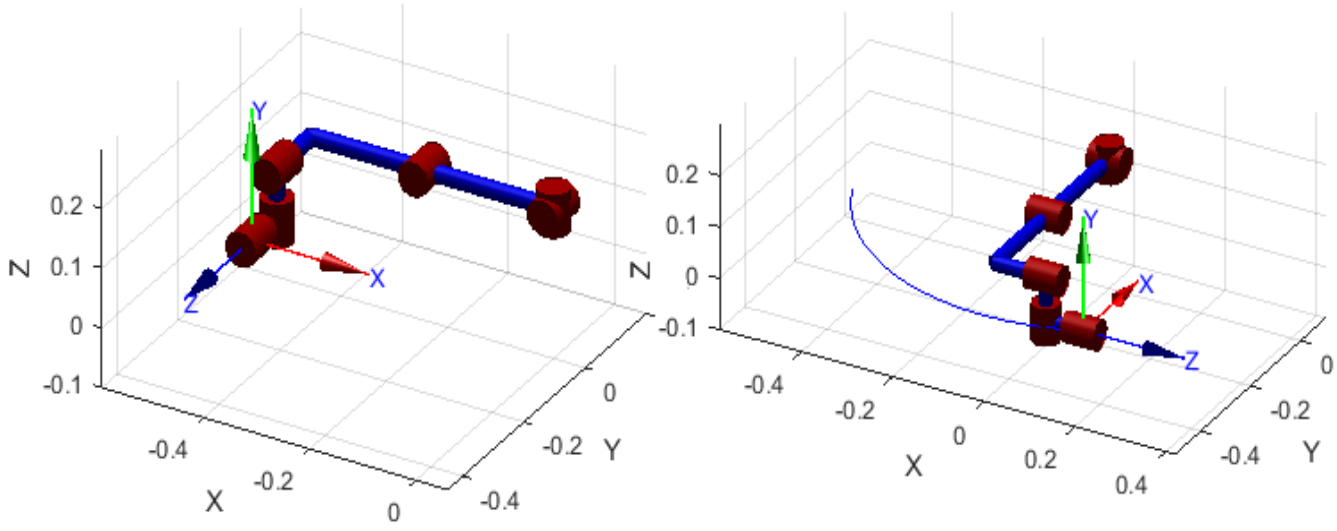


Figura 89: wire frame traiettoria: configurazione iniziale e finale del movimento

Come prima operazione viene fatto un confronto tra gli andamenti dei valori acquisiti di target, calcolati dal software del robot, con quelli invece misurati dai sensori del robot stesso.

Si specifica che l'accelerazione actual riportata nel grafico è stata ottenuta (poiché non data come output dalla macchina) derivando la velocità actual nel tempo.

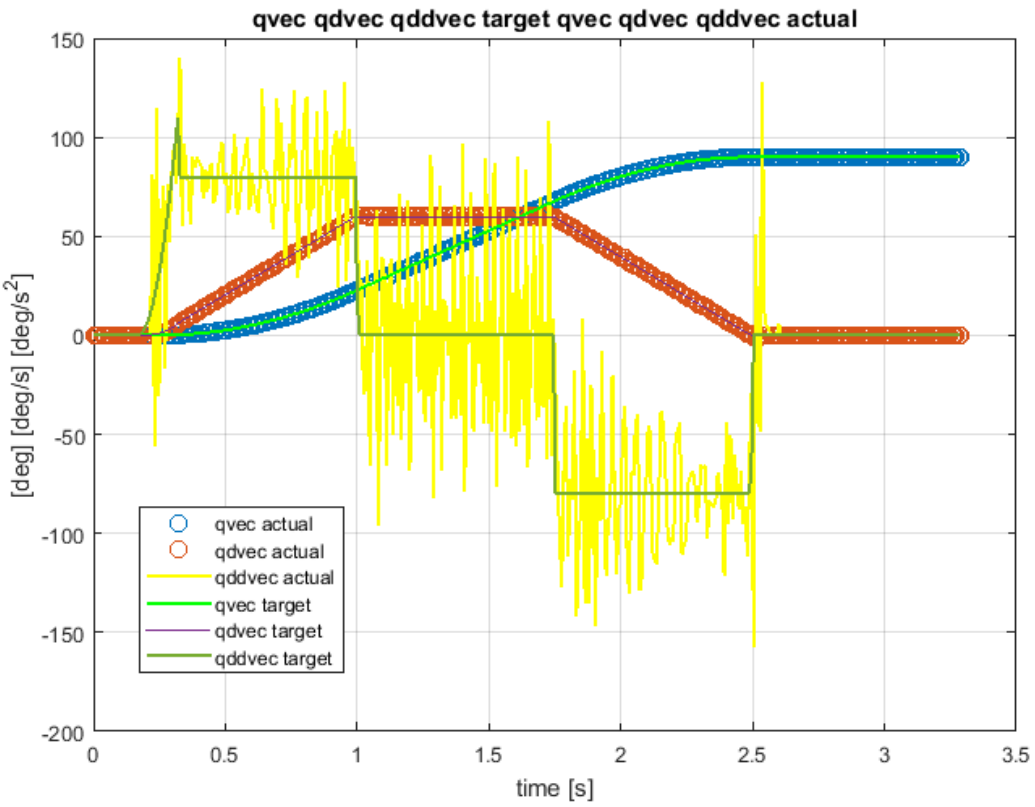


Figura 90: qvec qdvec qddvec prova dove si muove solo il primo giunto

Si riporta un ingrandimento delle varie grandezze nel punto iniziale e finale del trapezio di velocità: si osserva come la grandezza actual di velocità sia oscillante intorno alla corrispettiva grandezza di target. Questo risulta essere il motivo per cui l'andamento dell'accelerazione derivata presenta tutti questi picchi e irregolarità.

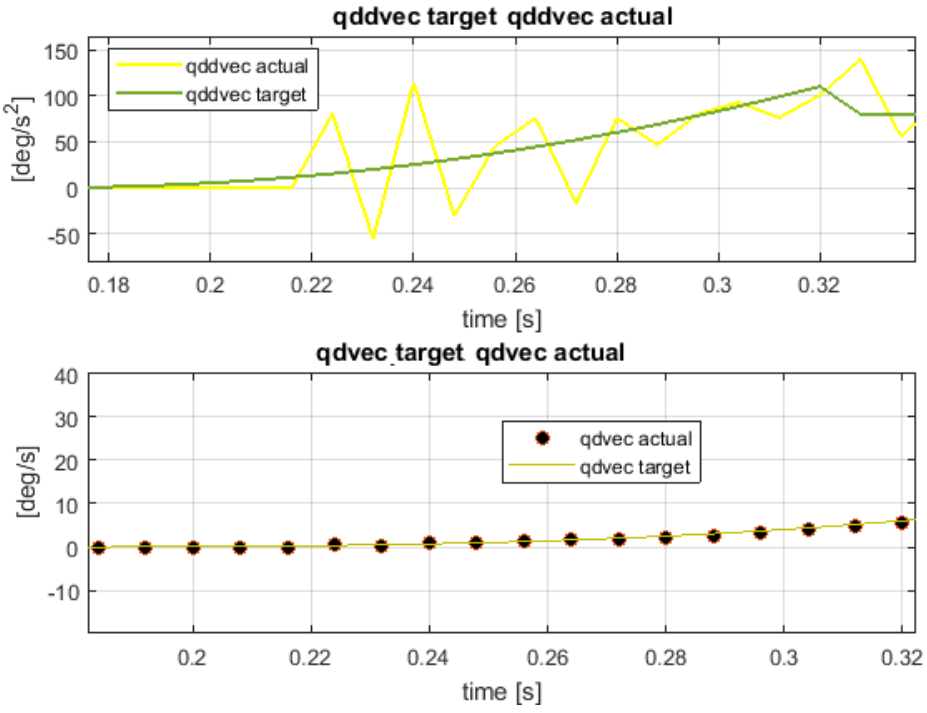


Figura 91: rampa di salita di velocità e accelerazione

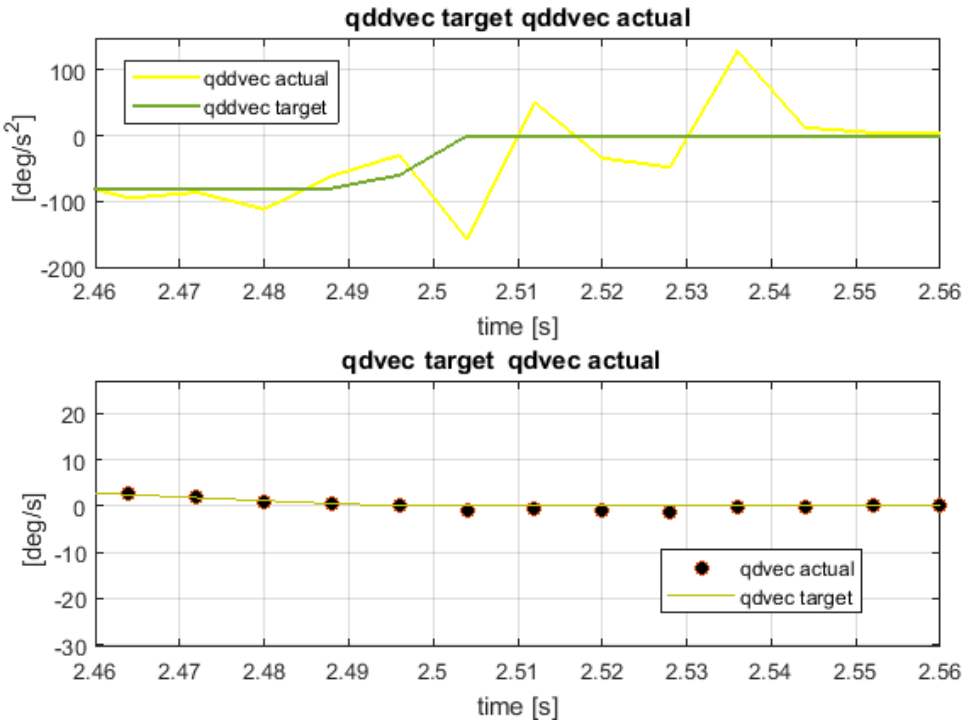


Figura 92: rampa di discesa di velocità e accelerazione

Tra alcune delle misure effettuate si ritiene significativo analizzare gli andamenti della corrente e della coppia applicata al giunto. I due andamenti sono collegati tra loro dalle leggi dell'elettromagnetismo, infatti sono piuttosto simili (poiché le coppie acquisite in output sono di target allora il confronto è stato fatto con la corrente di target). Inoltre, tra i possibili dati da acquisire, si individua una corrente actual e una di controllo (quest'ultima probabilmente viene utilizzata per poter chiudere un anello di controllo di coppia).

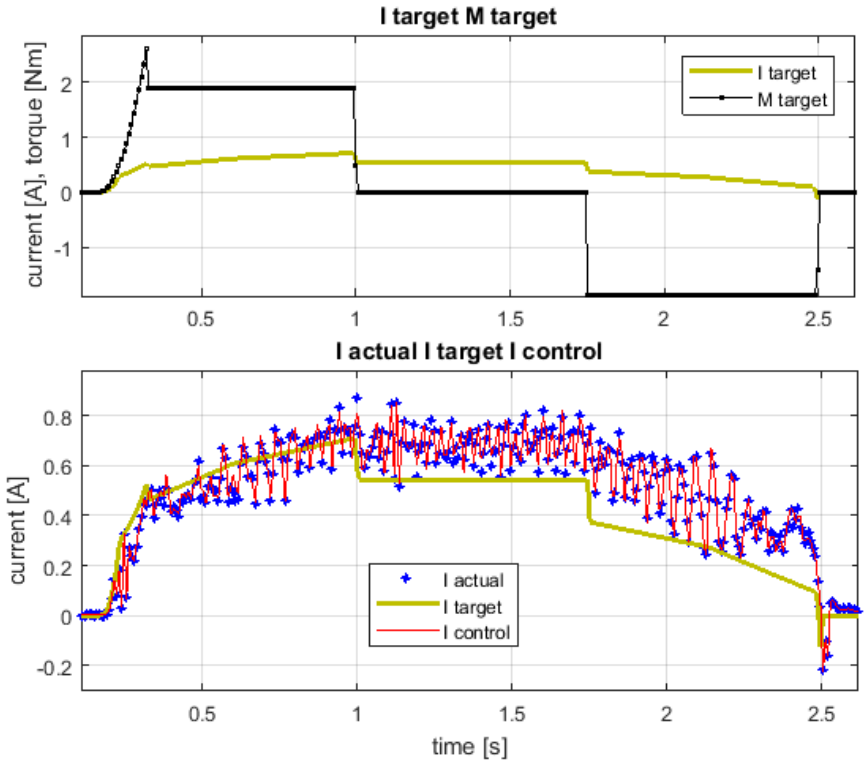


Figura 93: coppie e correnti

3.1 Analisi cinematica e dinamica: confronto modello e acquisizione della prova 1

Per poter verificare l'attendibilità del modello Matlab-Simulink realizzato precedentemente si decide di confrontare i valori misurati relativi al tool con quelli calcolabili dal modello.

Nel modello vengono dati in input i valori di qvec actual misurati e acquisiti durante il movimento, successivamente viene fatto un ciclo *for* dove, per ogni istante del vettore tempo acquisito, viene calcolata la matrice Jacobiana per ogni configurazione e, moltiplicata per i valori di velocità dei giunti misurati e acquisiti, si calcola il vettore velocità generalizzata del TCP. La posizione del tool, invece, viene calcolata semplicemente andando, ad ogni iterazione, a risolvere la cinematica diretta.

Vengono riportati di seguito i risultati ottenuti:

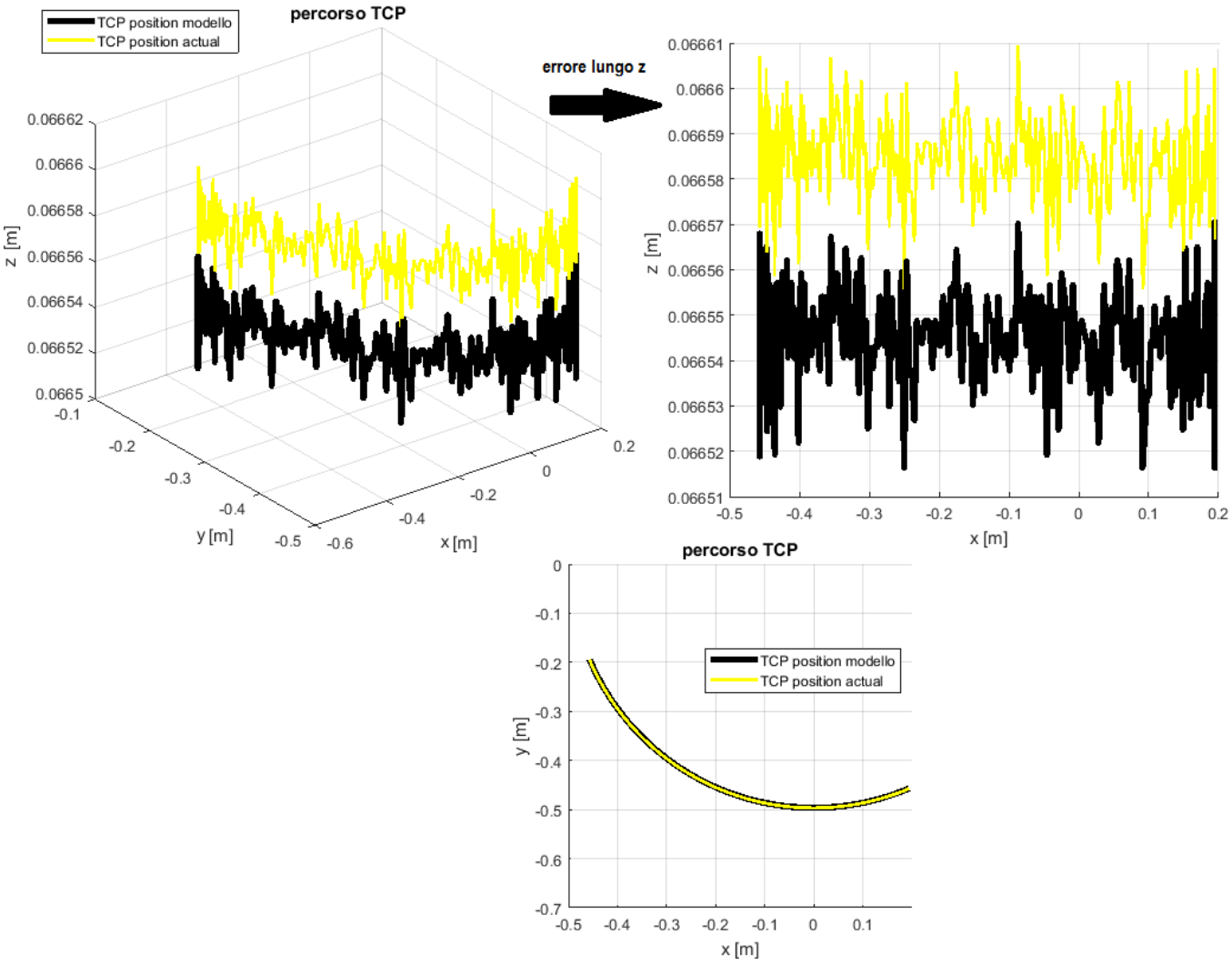


Figura 94: posizione TCP acquisita e calcolata con modello matlab

Si può osservare che l'errore di posizione è piccolissimo e trascurabile e lo si misura lungo l'asse verticale. Per il resto il modello risulta molto attendibile poiché il percorso ottenuto al Matlab risulta essere coincidente con quello che si misura nella realtà e lo si può ben vedere sul piano X-Y.

Si riportano anche gli andamenti della velocità lineare e angolare del tool. È possibile osservare che il modello riproduce molto bene gli andamenti sia in termini di velocità lineare che angolare.

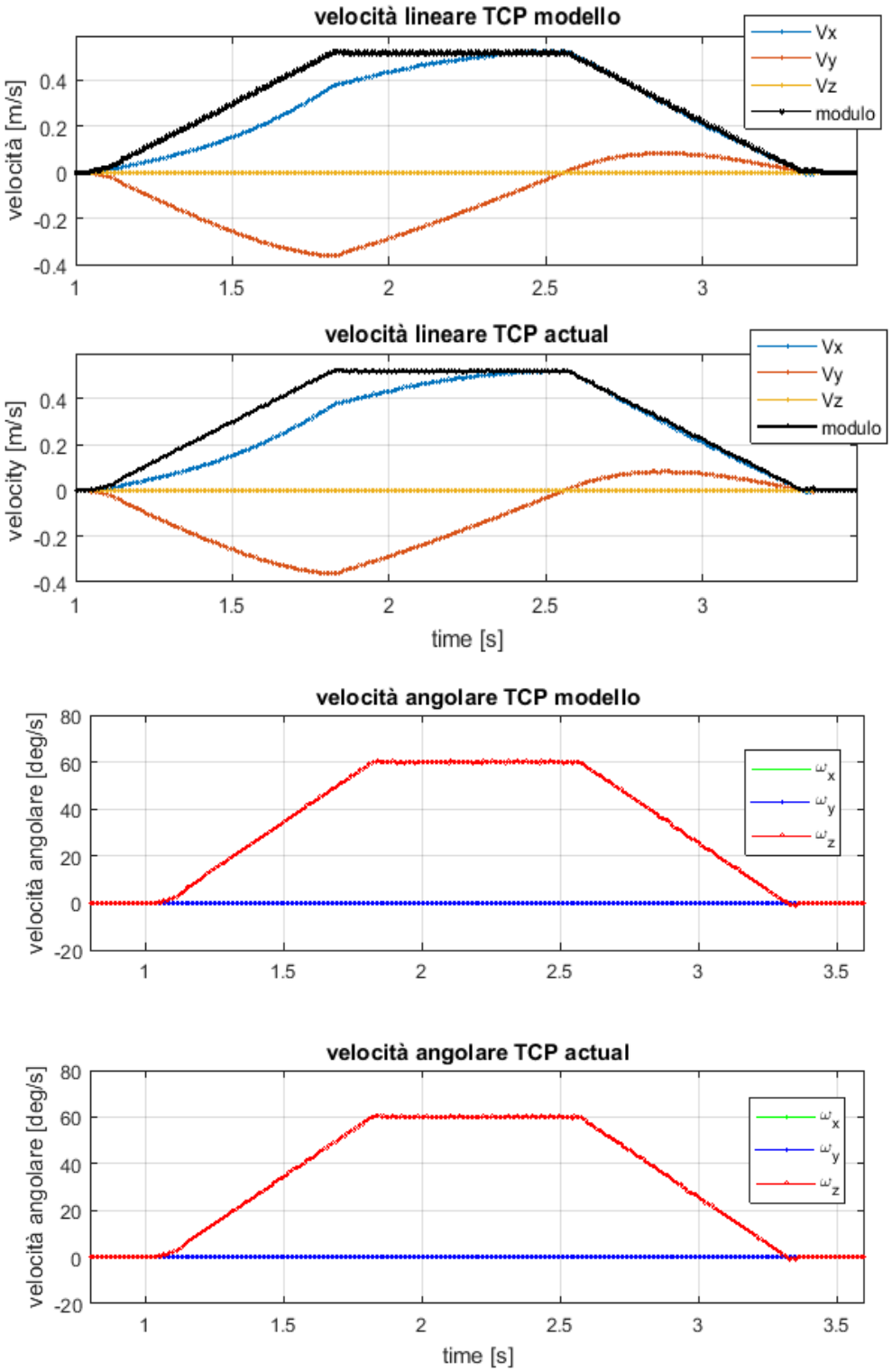


Figura 95: velocità lineare e angolare del TCP acquisita e calcolata con modello matlab

Sono di sotto riportati gli andamenti delle prove effettuate attuando lo stesso movimento ma con parametri cinematici di velocità del giunto differenti:

Tabella 22: altre prove effettuate variando la velocità

prova	velocità giunto [deg/s]
PROVA_1_1_moveJ_velocità_default_giunto1.urp	60
PROVA_1_2_moveJ_velocità_dimezzata_giunto1.urp	54
PROVA_1_3_moveJ_velocità_al10%_giunto1.urp	30
PROVA_1_4_moveJ_velocità_al90%_giunto1.urp	6

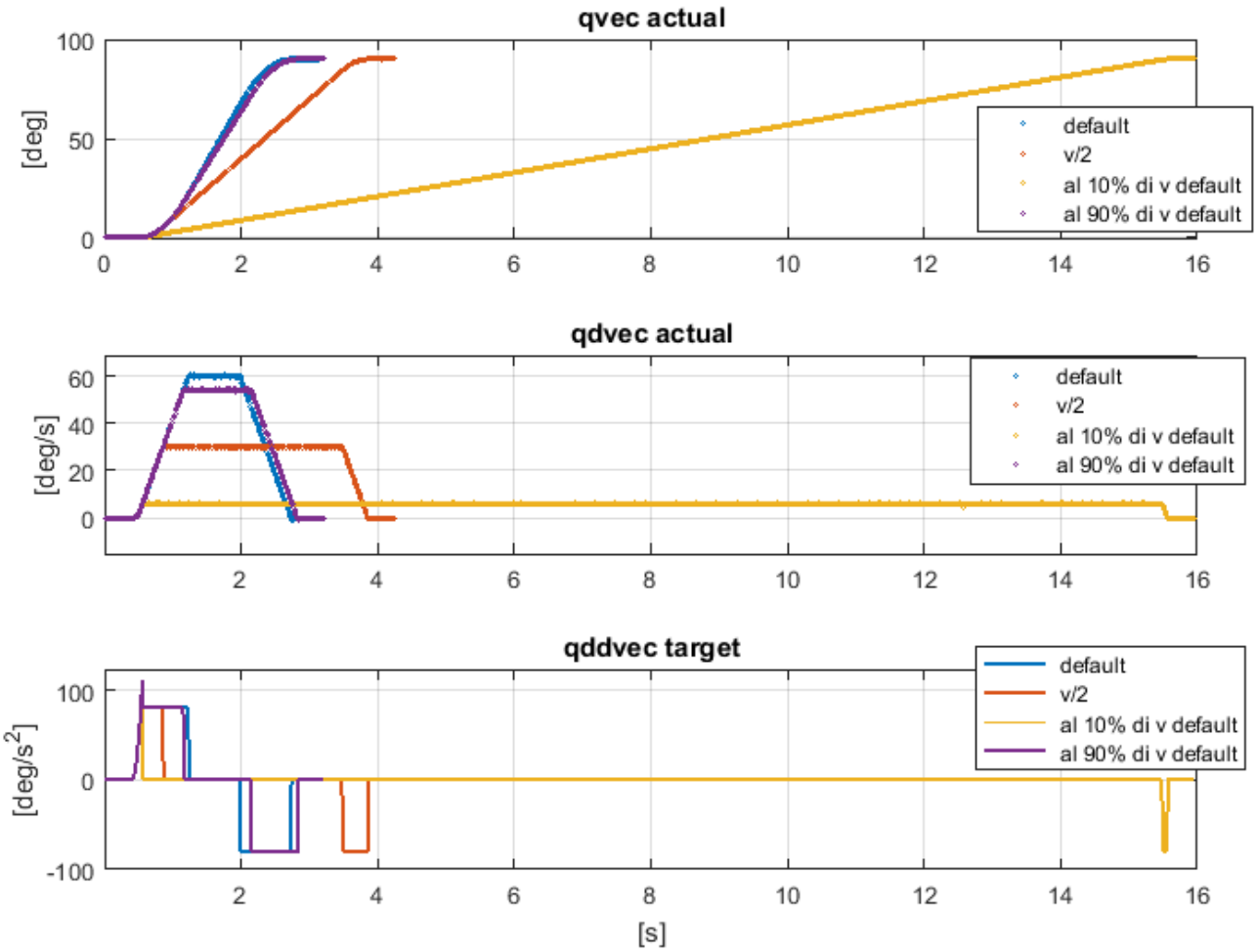


Figura 96: andamenti qvec, qdvec e qddvec al variare della velocità del giunto 1

Si osserva naturalmente che andando a diminuire il valore della velocità del primo giunto, il target di posizione viene raggiunto in un tempo maggiore. Inoltre, il tratto orizzontale del trapezio di velocità tende ad aumentare mentre l'inclinazione dei due lati rimane la stessa poiché si mantiene sempre costante l'accelerazione (di 80deg/s^2).

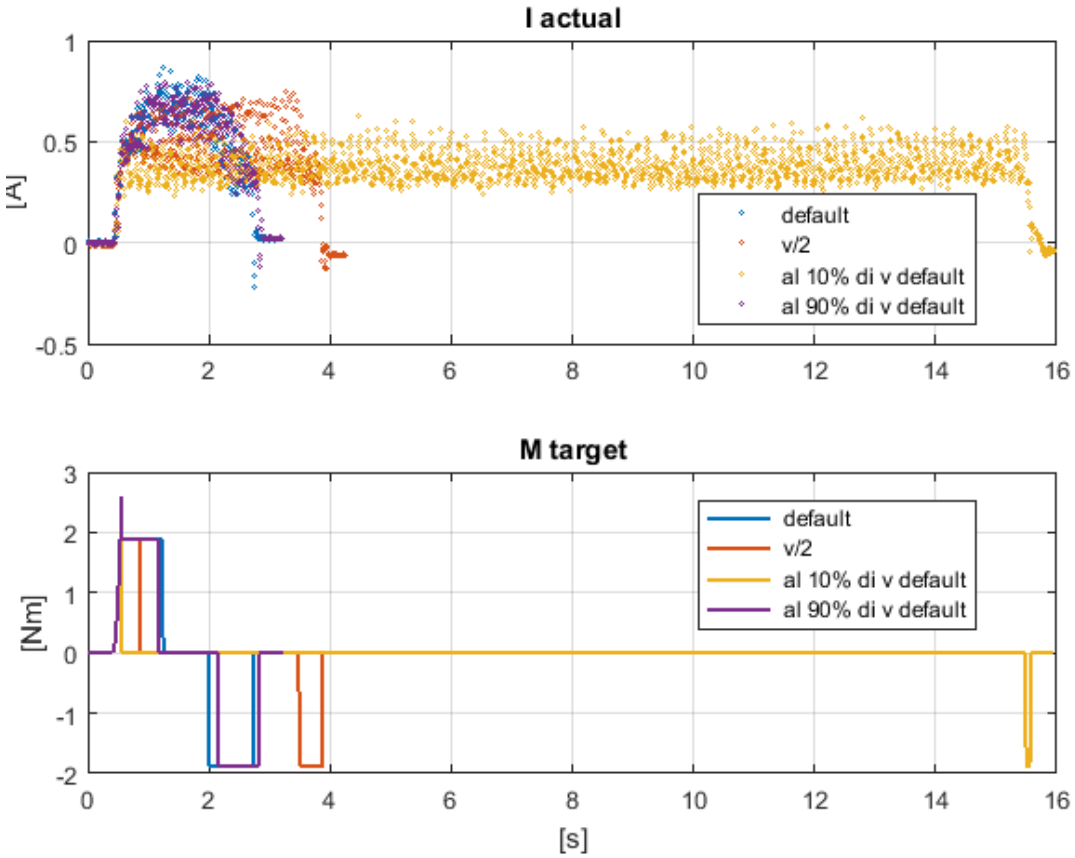


Figura 97: coppie di target e correnti actual al variare della velocità del giunto 1

Come osservato anche precedentemente l'andamento delle coppie tende a seguire gli andamenti delle correnti assorbite. Più le correnti assorbite sono basse più queste devono essere assorbite per un tempo lungo per garantire quello stesso movimento; la larghezza delle onde quadre di coppia dipende dai valori di corrente massimi assorbiti, più i valori di corrente sono elevati più la larghezza sarà grande.

Nell'immagine successiva (fig. 98) si può osservare, invece, che il modello dinamico è abbastanza fedele alla realtà poiché ci restituisce dati molto attendibili.

Resta da osservare però un piccolo errore presente sull'andamento del primo giunto, probabilmente poiché il modello Matlab non tiene conto delle forze d'attrito da vincere e delle dinamiche dei motori. Sui restanti andamenti gli errori percentuali risultano ancora più piccoli e dunque trascurabili.

Il modello realizzato fino a questo punto verrà utilizzato successivamente anche per le analisi delle altre prove riportate nella libreria.

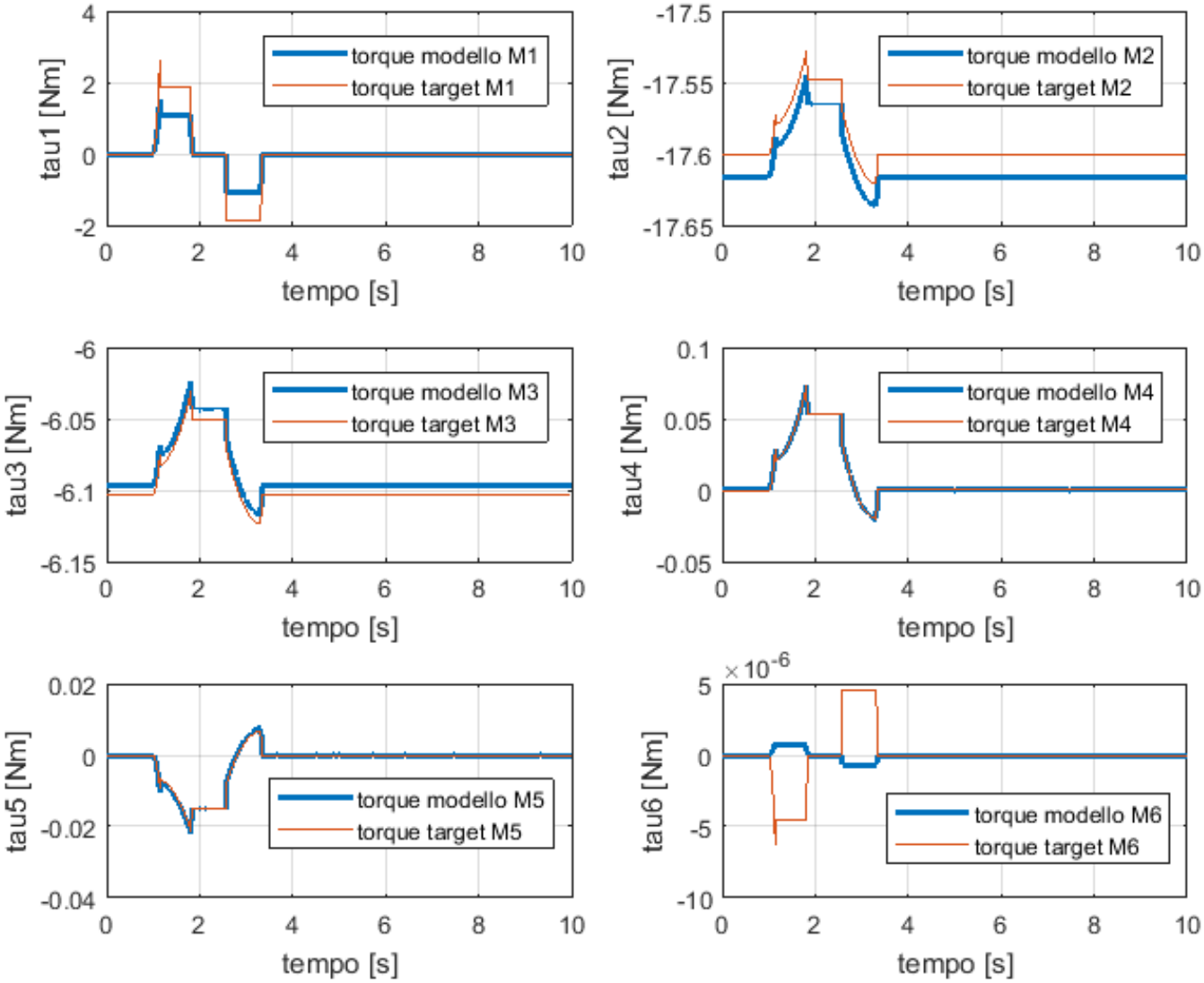


Figura 98: confronto coppie di target e coppie del modello dinamico

3.2 Analisi di altre prove della libreria

Nel seguente schema vengono riassunte tutte le prove fatte in questo lavoro (riportate nell'appendice) ed il motivo per cui sono state analizzate:

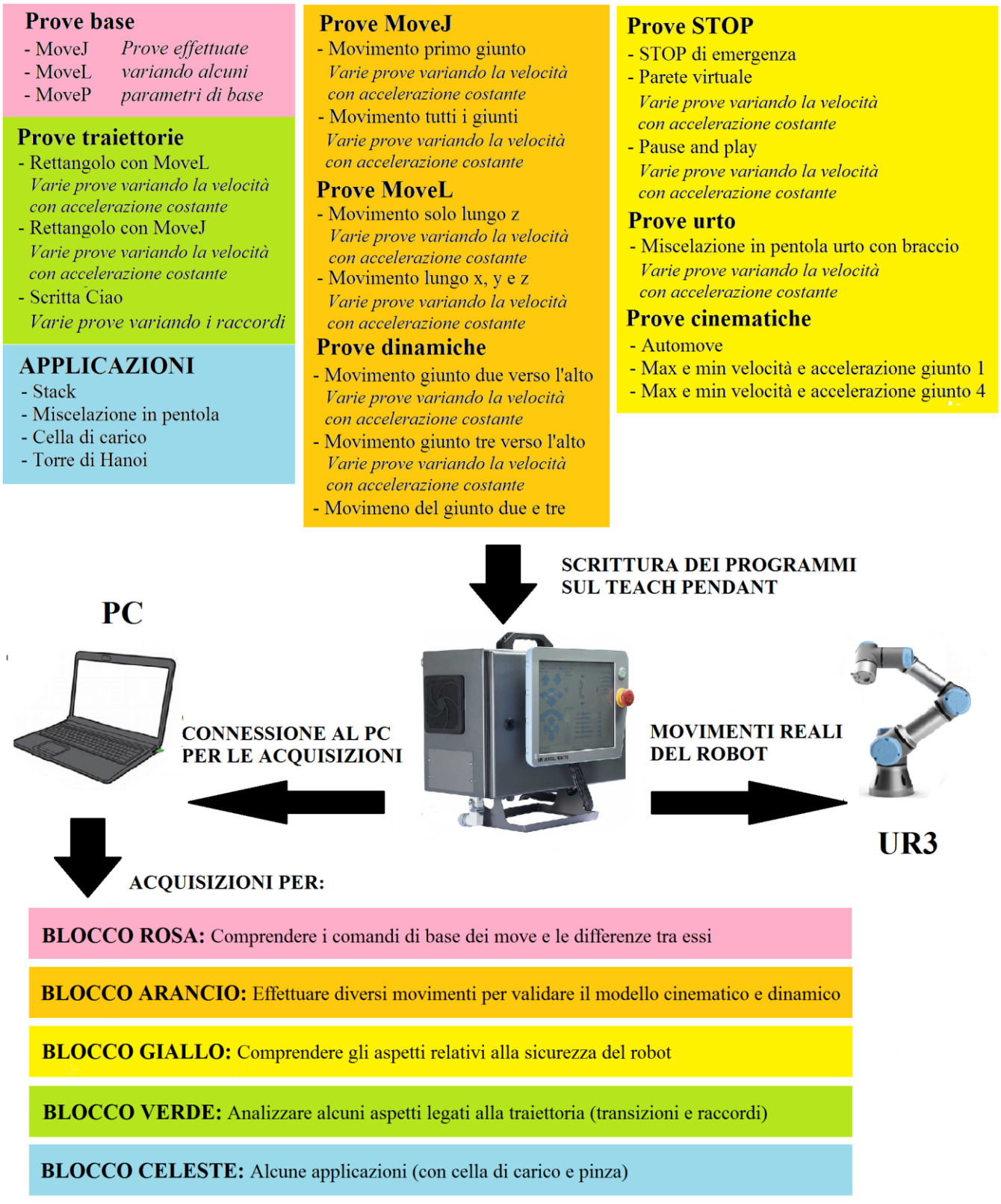


Figura 99: schema riassuntivo di tutte le prove svolte

- ***Programma scritto:*** PROVA_3_1_moveL_velocità_default_lungo_z.urp

  Acquisizione corrispondente: ACQUISIZIONE_3_1_moveL_velocità_default_lungo_z.mat

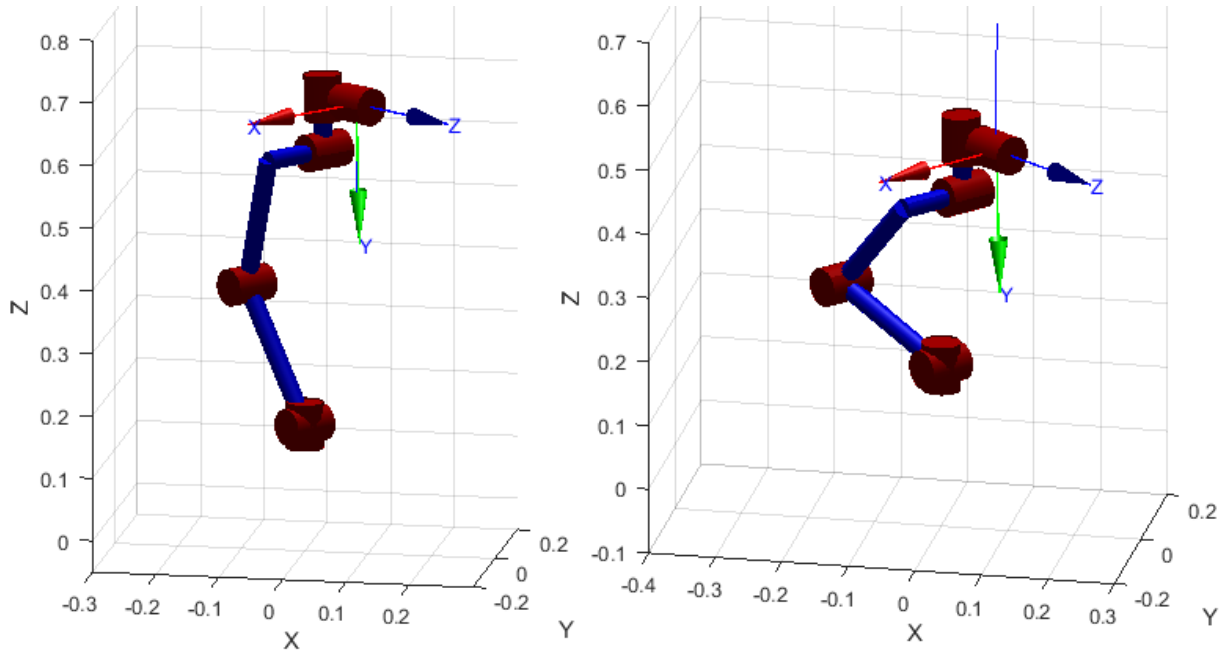


Figura 100: Configurazioni iniziali e finali del movimento

Tale movimento prevede uno spostamento lungo la direzione z di 200mm. (da 660mm a 460mm d'altezza). Vengono riportati gli andamenti acquisiti di: qvec, qdvec, qddvec, delle correnti e dei momenti target.

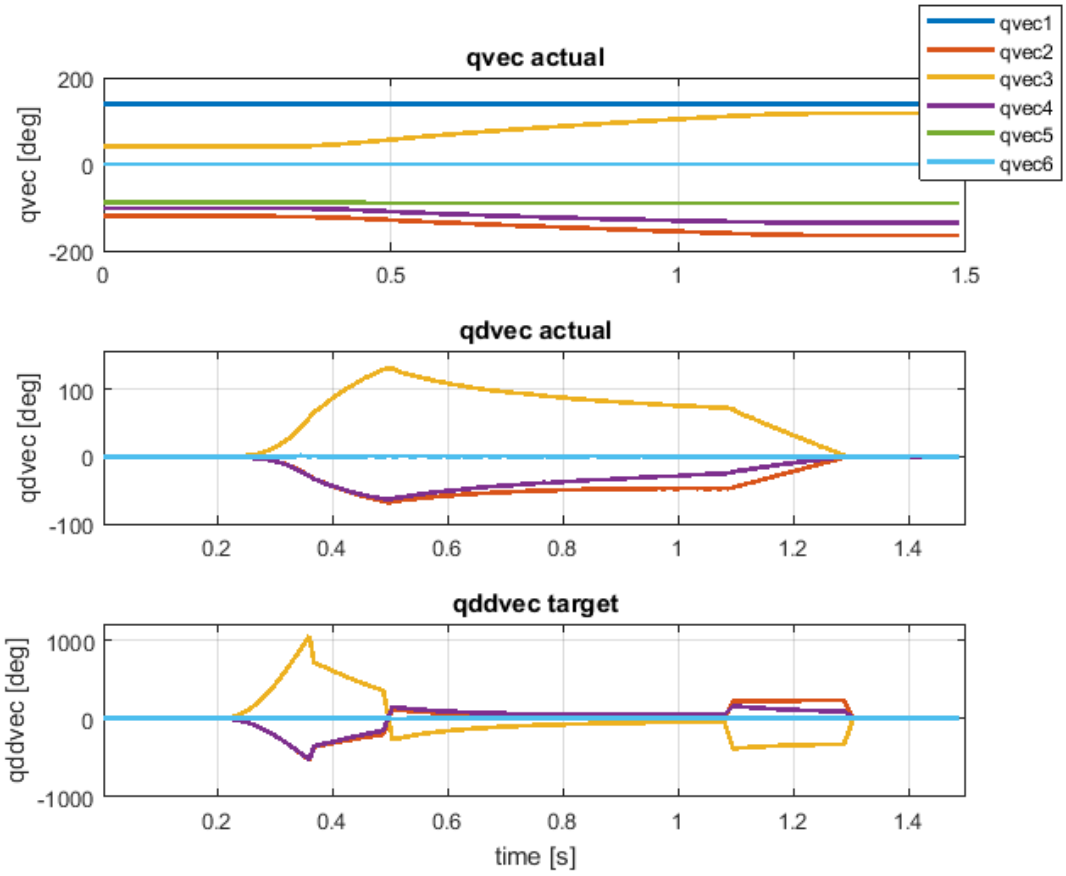


Figura 101: qvec, qdvec e qddvec di questa prova in condizioni di default

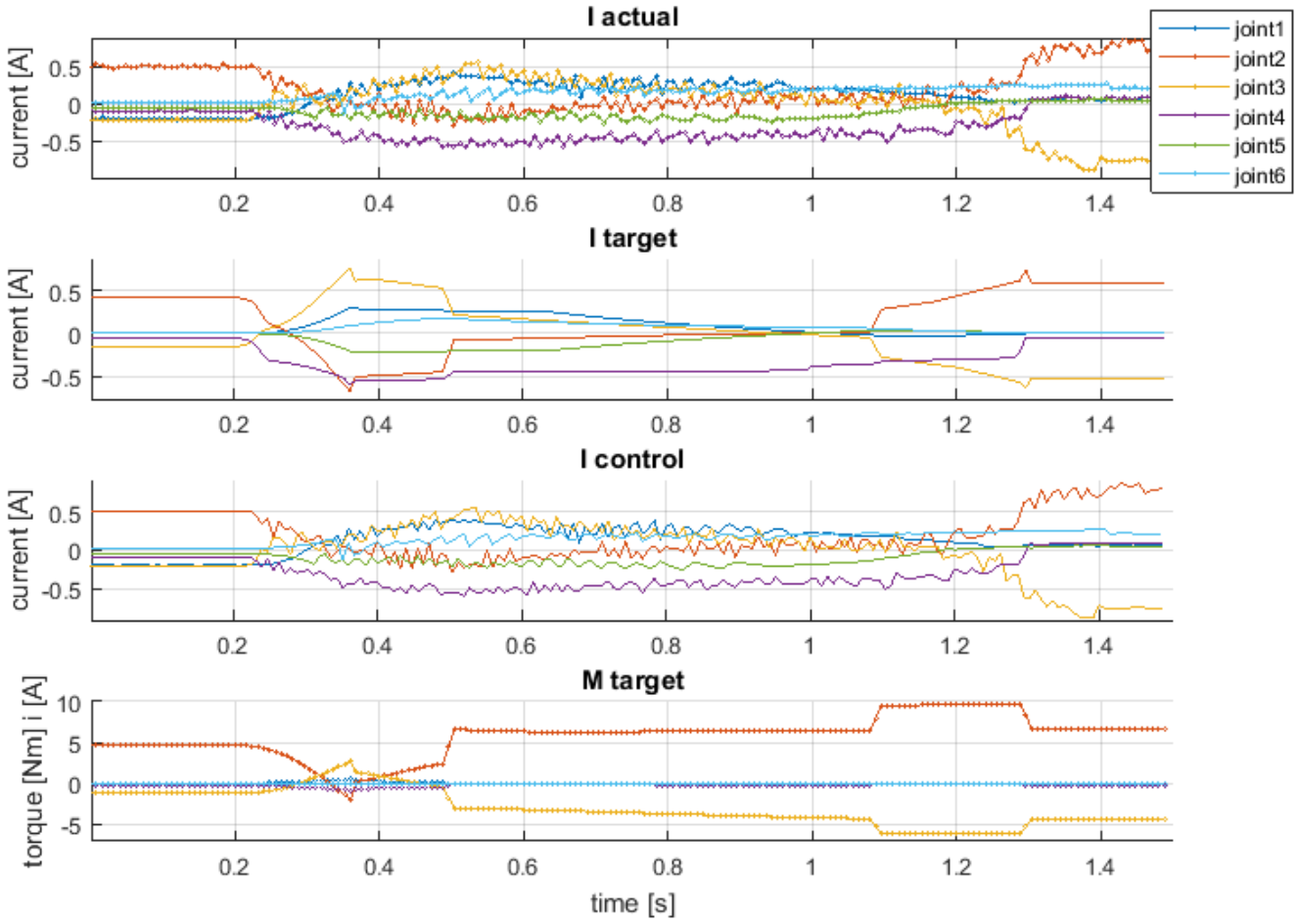


Figura 102: i target, i control, i actual e m target della prova di default

La presenza di una I di controllo nel sistema del robot è dovuta probabilmente alla presenza di un anello chiuso per il controllo della coppia erogata. Inoltre, si osservi come gli andamenti di coppia di target risultino essere molto simili a quelli della corrente di target, come visto in precedenza.

Anche per questa prova viene fatto un confronto tra l'acquisizione e il modello del sistema realizzato in ambiente Matlab, che mostra la sovrapposizione dei percorsi del tool. Si effettuano le stesse precedenti considerazioni, ovvero che è presente un errore, del tutto trascurabile, dell'ordine del decimo di millimetro durante il percorso effettuato.

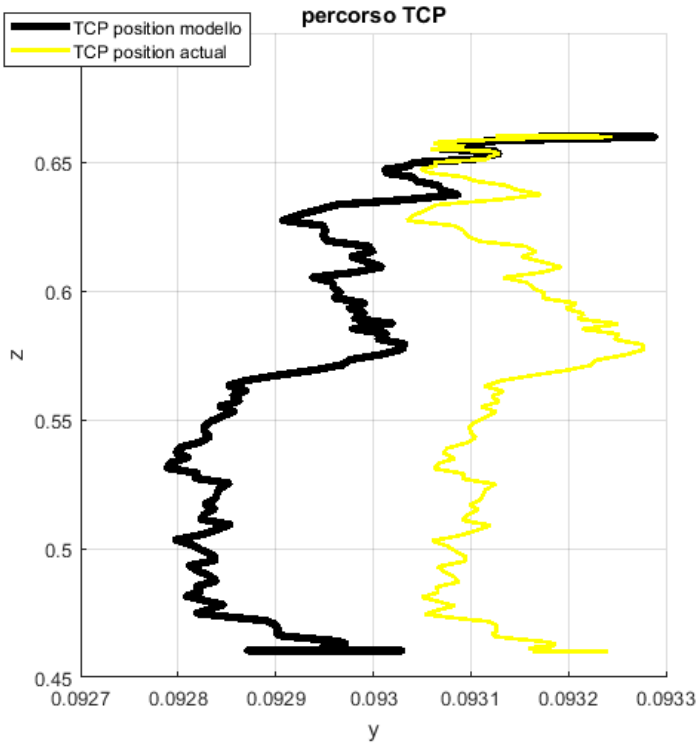


Figura 103: posizione del TCP

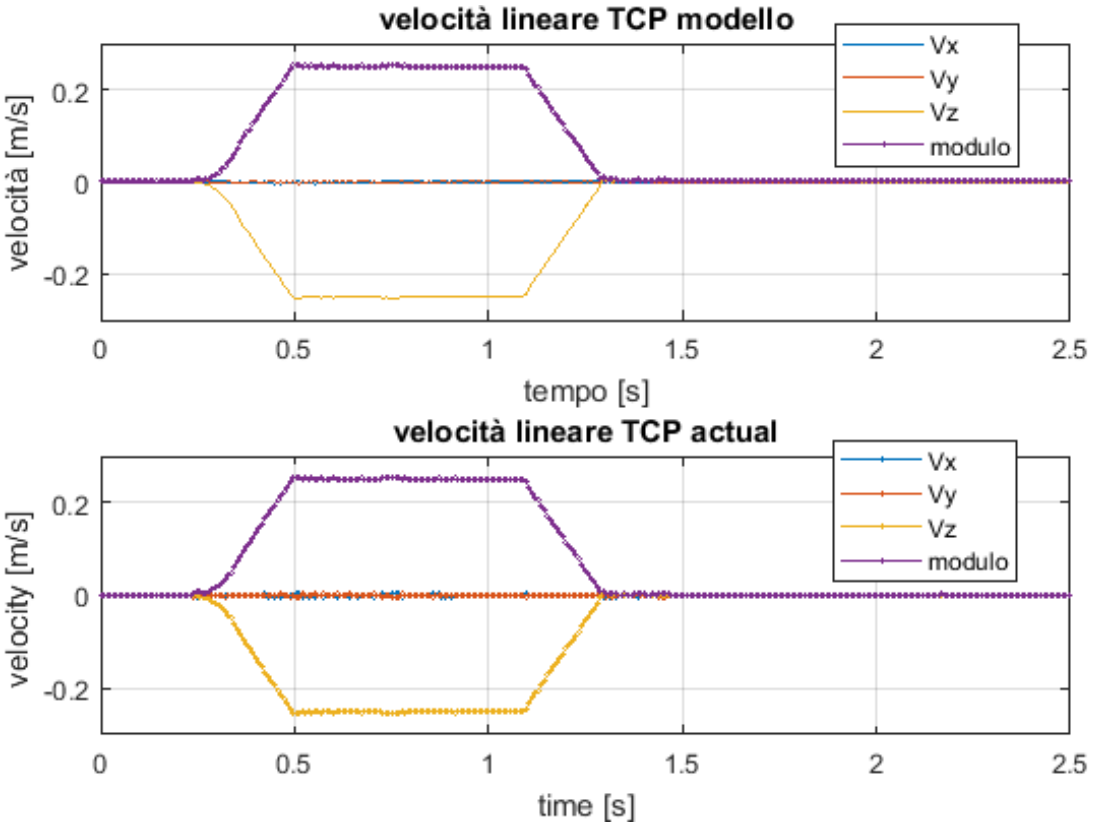


Figura 104: velocità lineare del TCP confronto modello e acquisizione

La prova viene ripetuta più volte ma variando la velocità del tool. Vengono riportati i valori della posizione e della velocità sovrapposti delle diverse prove:

Tabella 23: altre prove effettuate variando la velocità

prova	velocità TCP [m/s]
PROVA_3_1_moveL_velocità_default_lungo_z.urp	0.250
PROVA_3_2_moveL_velocità_dimezzata_lungo_z.urp	0.225
PROVA_3_3_moveL_velocità_al10%_lungo_z.urp	0.125
PROVA_3_4_moveL_velocità_al90%_lungo_z.urp	0.025

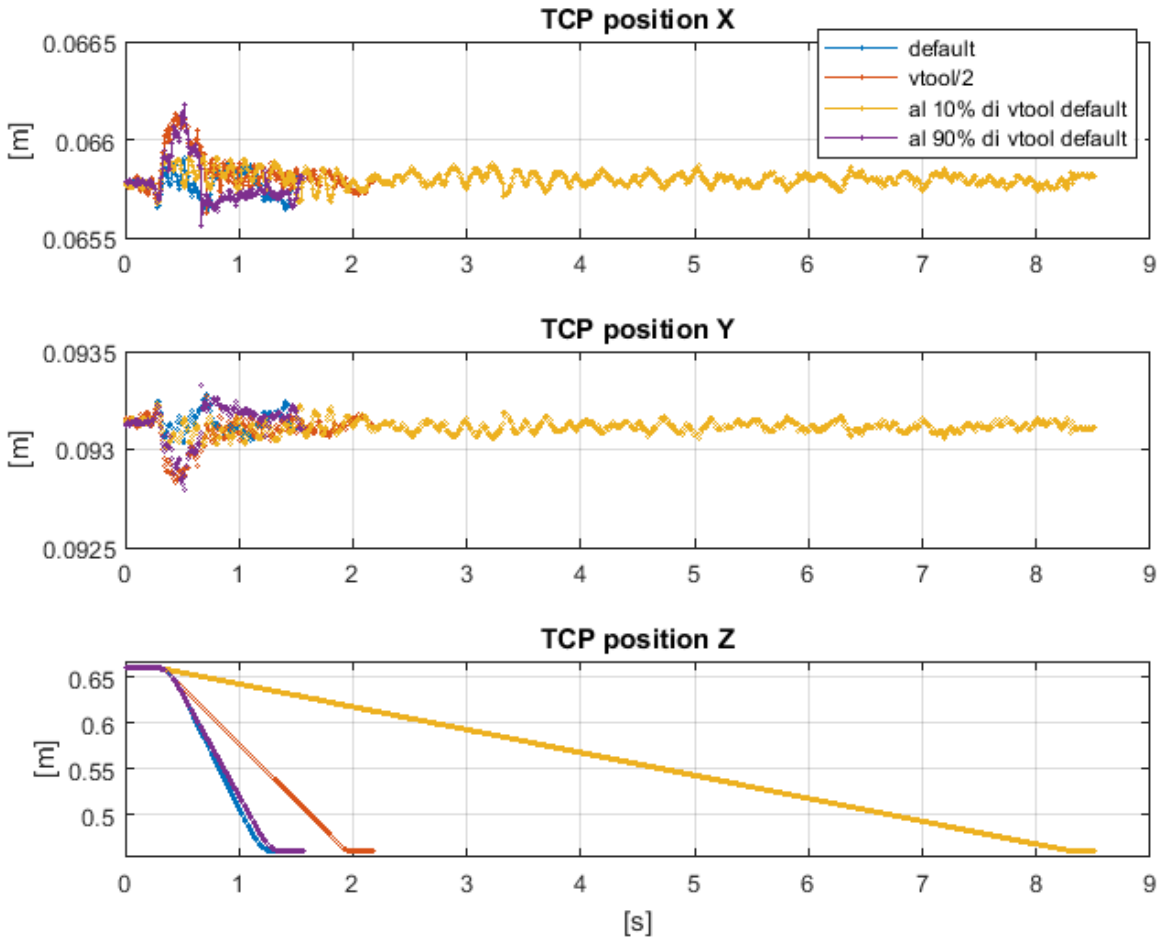


Figura 105: posizione TCP al variare della velocità impostata

Naturalmente si osserva che variazioni significative si verificano solamente lungo la componente z, poiché il tool si sta muovendo solo in quella direzione. Sulle altre componenti è possibile osservare le oscillazioni trascurabili che disturbano il percorso del TCP.

Poiché non ci sono parametri significativi riguardo la velocità angolare, vengono riportati solo gli andamenti della velocità lineare del tool. Si osservi come anche durante il MoveL, la velocità venga impostata come un trapezio (ovviamente questa volta riferita al TCP), dove la pendenza viene dettata sempre dal valore dell'accelerazione del TCP.

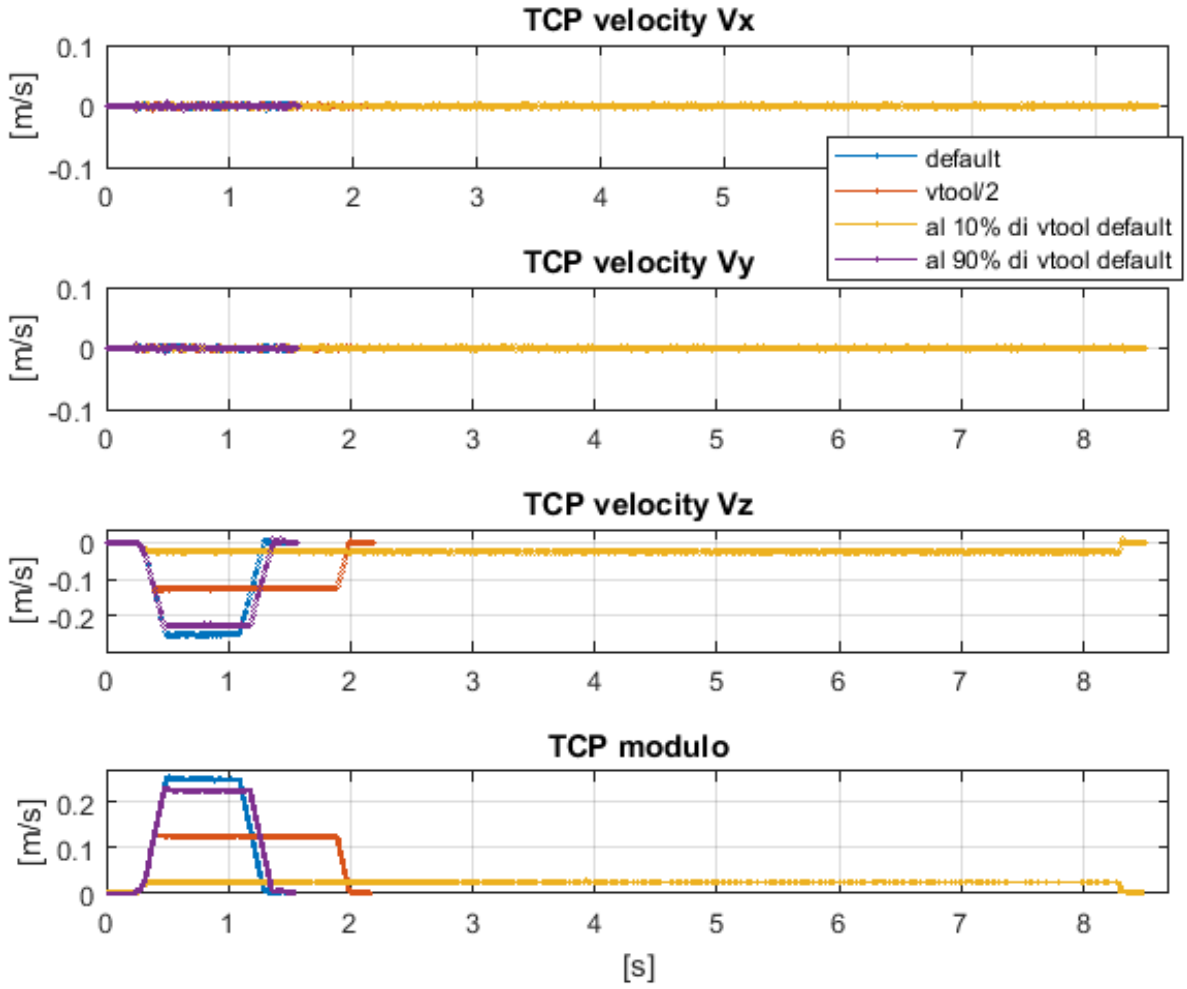


Figura 106: velocità lineare e modulo della velocità del TCP al variare delle prove

Un risultato importante è quello legato all'analisi dinamica (in figura 107). Questa volta risultano significativi sia l'errore di coppia presente sul giunto 2 che sul giunto 4. Proprio nel giunto quattro è possibile osservare che il modello segue i valori di target riportando un andamento opposto a quello aspettato. Questo probabilmente è dovuto ad un offset che nasce per via dell'imprecisione dei dati inerziali stimati con il software Solidworks. Gli errori sugli altri giunti sono invece trascurabili.

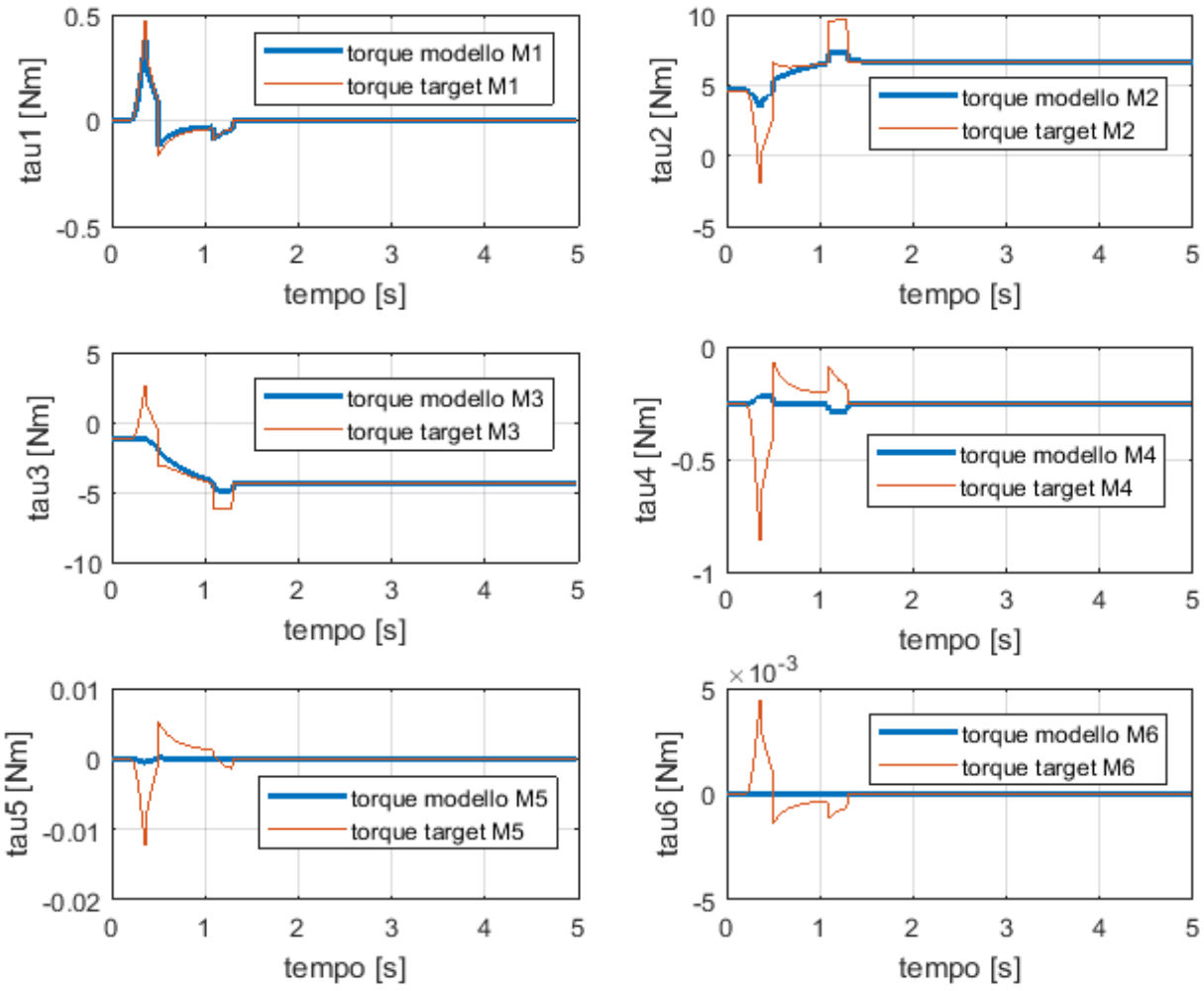


Figura 107: confronto coppie acquisite e coppie calcolate attraverso il modello matlab

- ***Programma scritto:*** PROVA_4_1_moveL_velocità_default_giunti.urp
  Acquisizione corrispondente: ACQUISIZIONE_4_1_moveL_velocità_default_giunti.mat

Con questa prova si decide di far muovere il robot comandandolo in spazio operativo. Viene riportato di seguito il percorso compiuto dal tool:

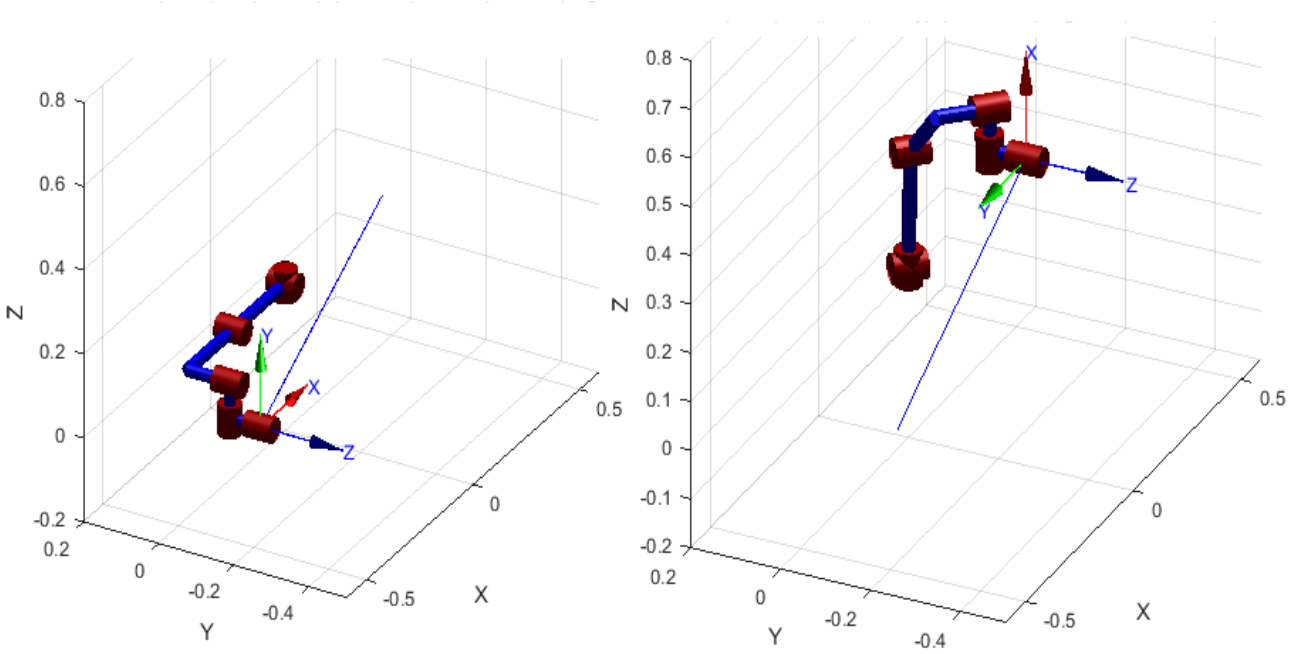


Figura 108: configurazione iniziale e finale del movimento

Tale movimento prevede:

- uno spostamento lungo la direzione x del tool da -0.45 m a -0.045 m;
- uno spostamento lungo la direzione y del tool da -0.195 m a -0.285 m;
- uno spostamento lungo la direzione z del tool da 0.066 m a 0.455 m.

Durante tale movimento programmato in spazio operativo, ovviamente i giunti vengono attuati in un certo modo per consentire lo spostamento lineare pensato per il TCP.

Si osserva dalla figura successiva che tutti i giunti vengono attuati con andamenti di velocità diversi da quelli trapezoidali del MoveJ.

Infine, sono stati riportati gli andamenti delle accelerazioni actual. non però misurate, ma ottenute sempre andando a derivare le velocità acquisite. Si osserva che ci sono dei picchi eccessivi di accelerazione e notevoli irregolarità nell'andamento, questo perché ottenuto come derivata del q actual.

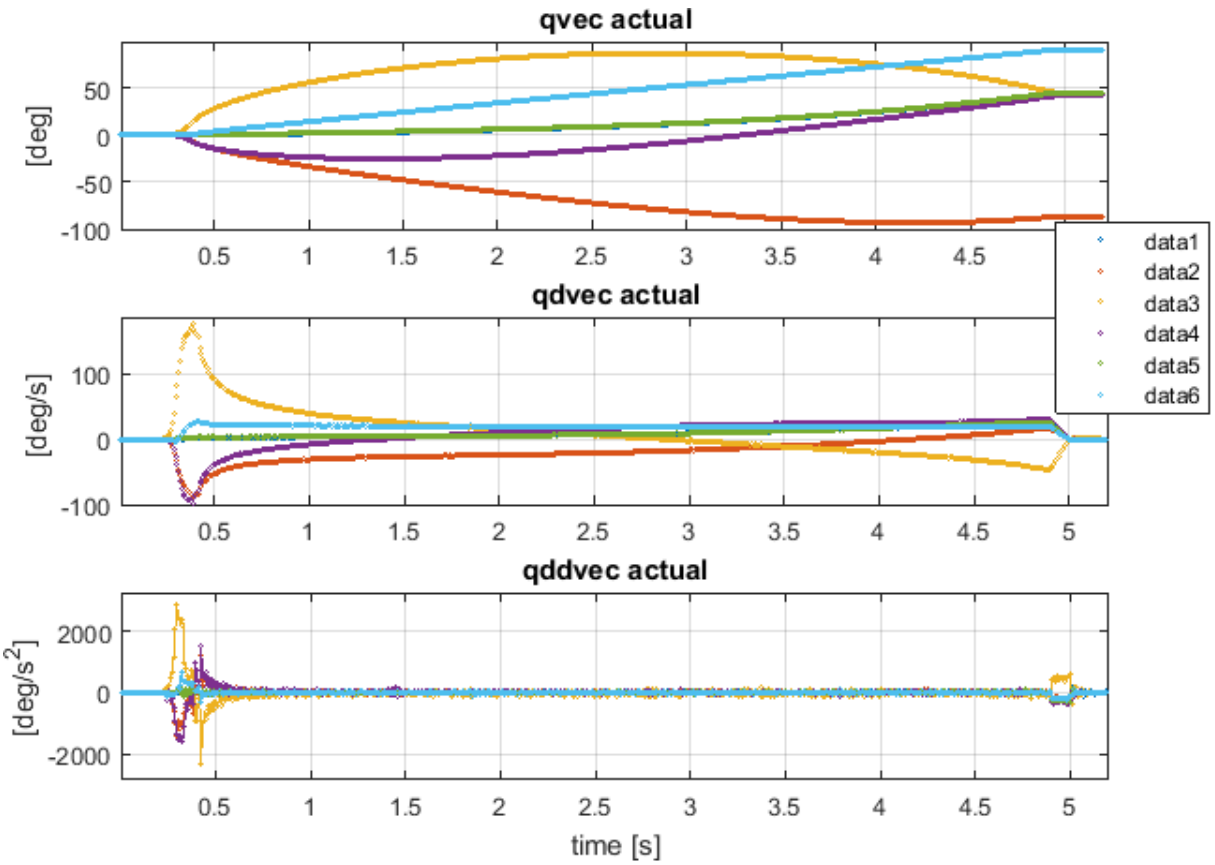


Figura 109: qvec, qdvec, qddvec correnti e momenti della prova di default

Sono poi riportate le misure calcolate e confrontate con quelle del modello relative al TCP come nella prova precedente. In figura 110 grazie ad un certo ingrandimento del percorso non si nota l'errore, come già detto trascurabile, mostrato anche prima dovuto alla piccola oscillazione del tool.

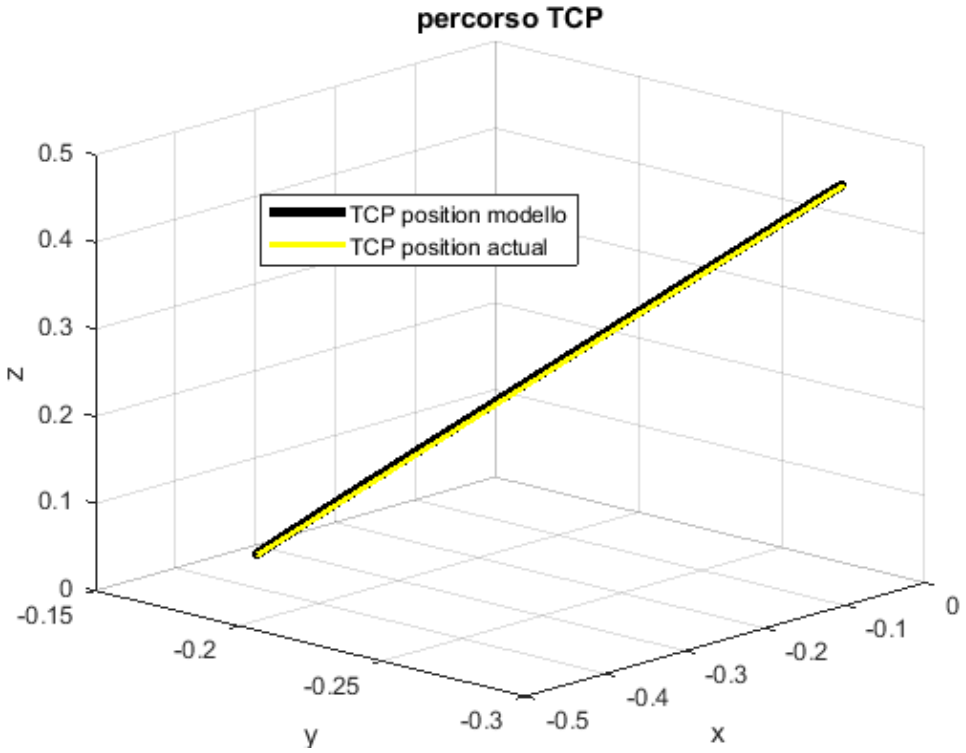


Figura 110: posa del TCP in spazio operativo confronto modello e acquisizione

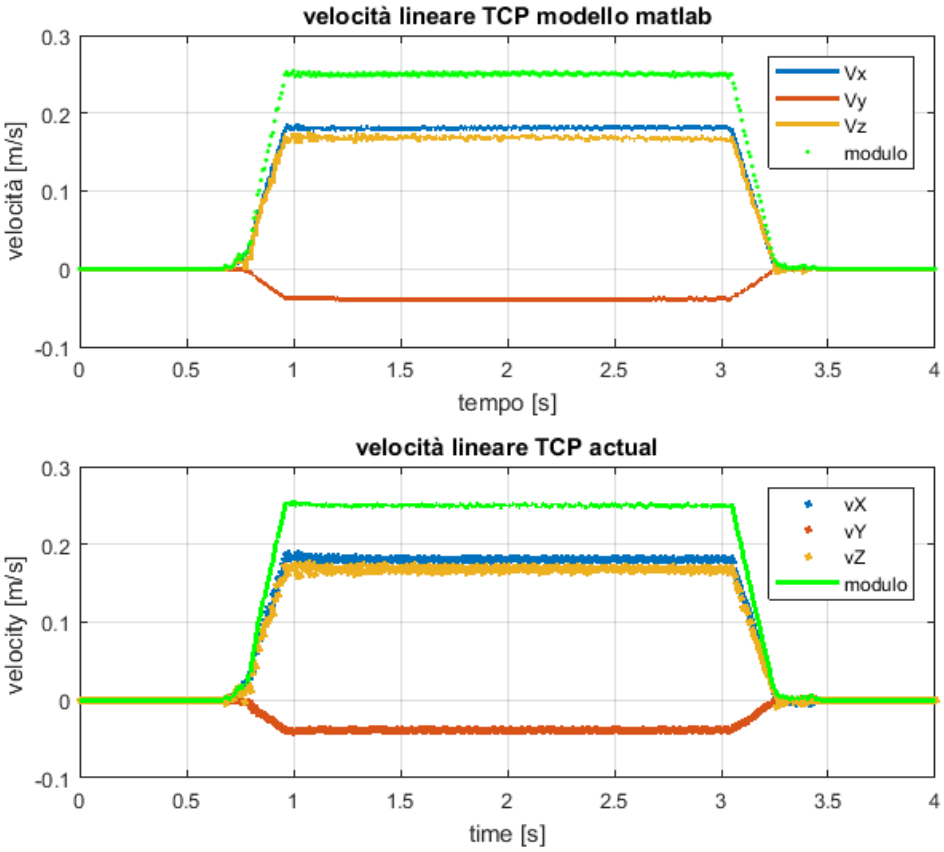


Figura 111: velocità lineare del TCP nella prova di default confronto modello e acquisizione

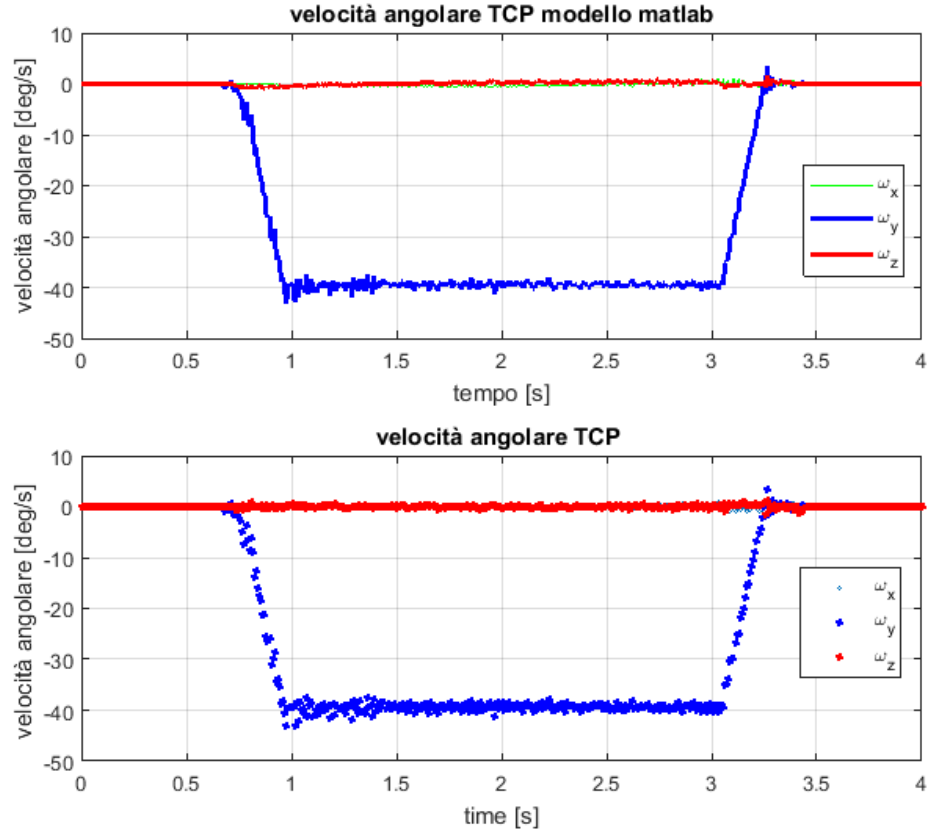


Figura 112: velocità angolare del TCP confronto modello e acquisizione

I valori di spostamento e di velocità del TCP sono calcolati sempre rispetto alla terna di riferimento di base. Si nota che gli andamenti di velocità del tool assumono delle forme trapezoidali (cosi come prima era per i giunti nel MoveJ). Si osserva inoltre che durante il movimento viene attuato anche il giunto 6 che genera una velocità angolare del TCP intorno all'asse y di base. La stessa prova è ripetuta sempre con diversi valori di velocità del TCP.

Tabella 24: prove al variare della velocità del TCP

prova	velocità TCP [m/s]
PROVA_4_1_moveL_velocità_default_giunti.urp	0.250
PROVA_4_2_moveL_velocità_dimezzata_giunti.urp	0.225
PROVA_4_3_moveL_velocità_al10%_giunti.urp	0.125
PROVA_4_4_moveL_velocità_al90%_giunti.urp	0.025

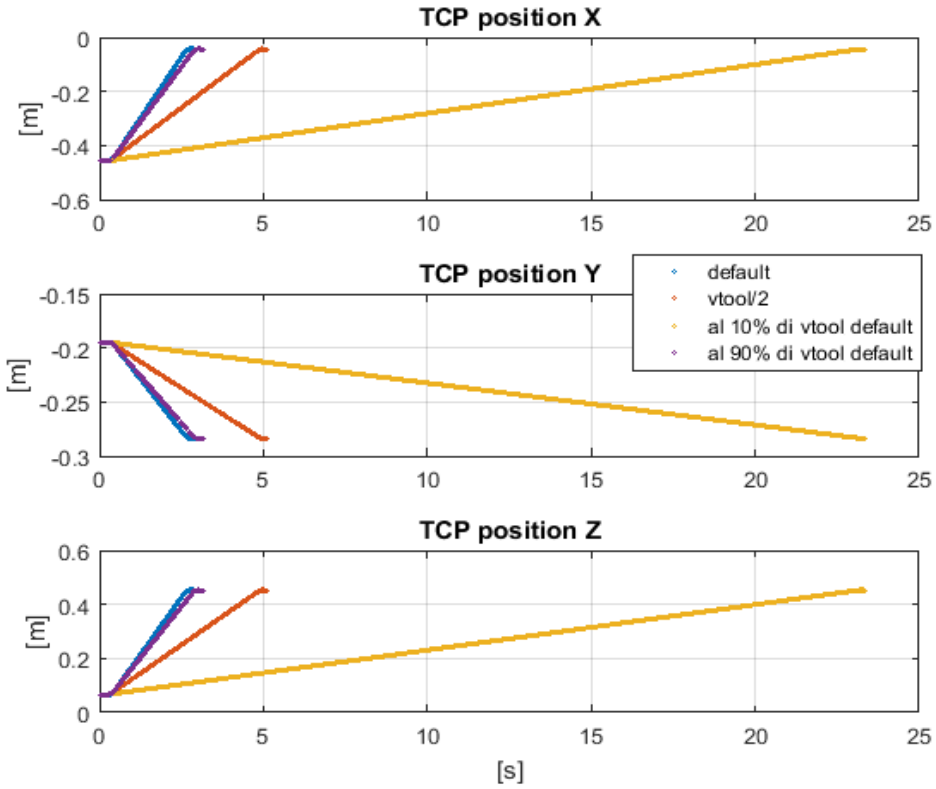


Figura 113: posizione del TCP al variare della velocità

Per quanto riguarda i valori di velocità lineare seguono l'andamento previsto. I valori delle singole componenti restituiscono, se calcolato il modulo, il valore inserito da input nel programma. Ciò è verificato nell'ultimo subplot di figura 114.

Analizzando infine gli andamenti delle coppie sono presenti alcuni picchi che il modello non riesce a seguire bene. L'errore che si percepisce durante questi picchi è abbastanza importante ma la durata del picco in ogni caso è molto breve. Questo fenomeno potrebbe essere correlato alle forze d'attrito, alle dinamiche interne e all'offset dovuto all'approssimazione dei dati inerziali.

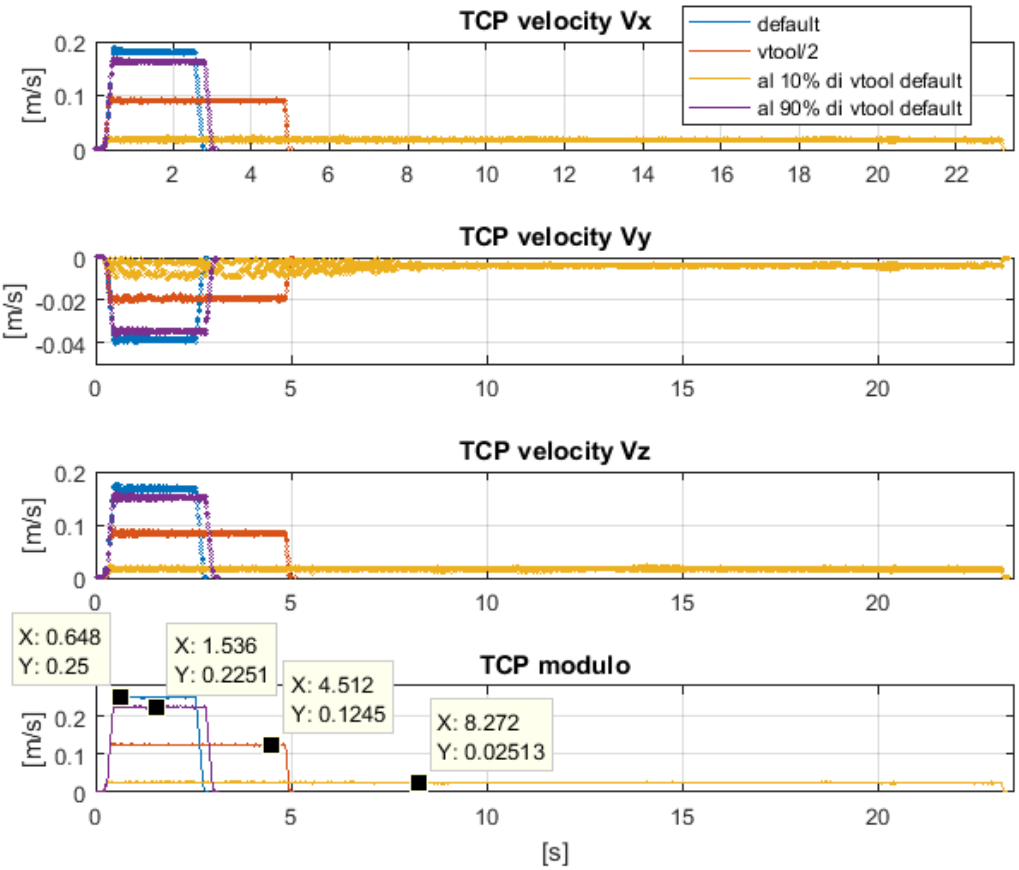


Figura 114: velocità lineare del TCP al variare dell'input di velocità

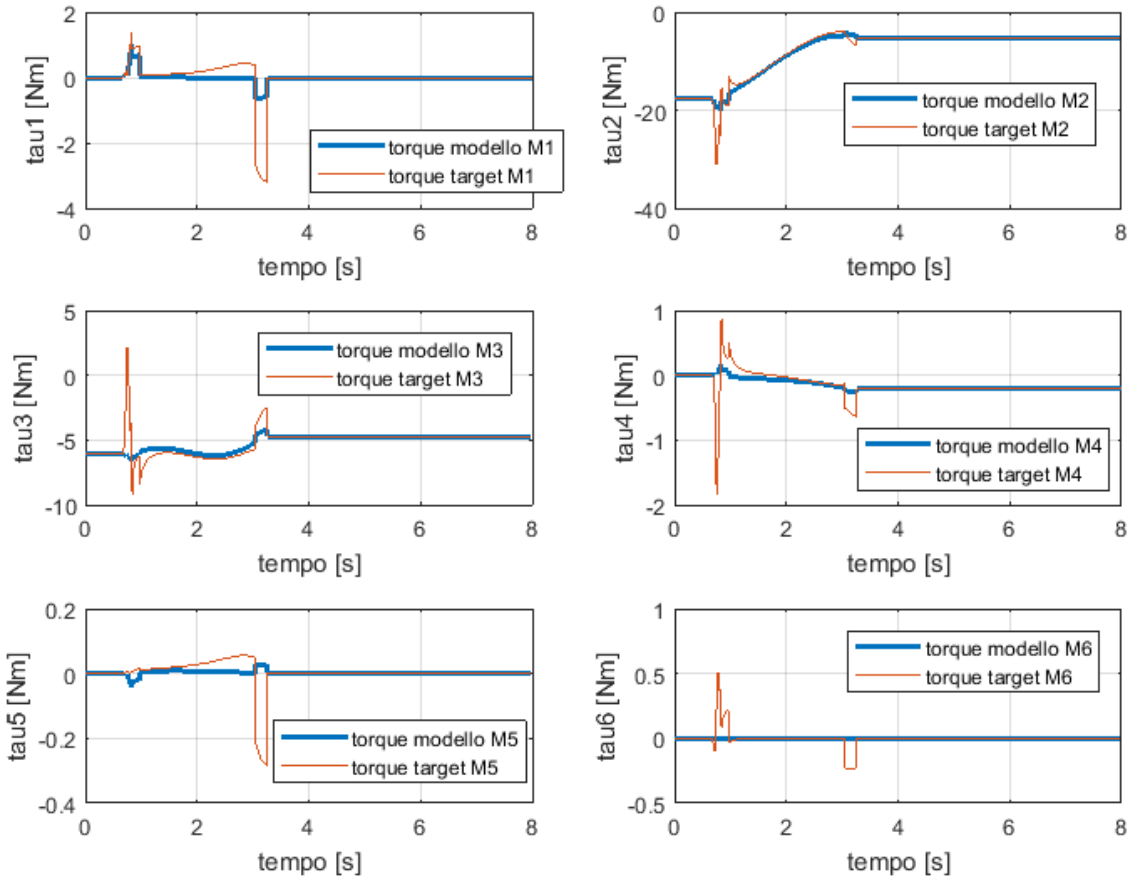


Figura 115: dinamica del movimento: confronto coppie modello e quelle di target

- ***Programma scritto:*** PROVA_11_1_movimento_giunto_3_verso_alto.urp
  Acquisizione corrispondente: ACQUISIZIONE_11_1_movimento_giunto_3_verso_alto.mat

Durante le prove effettuate, si nota che la risoluzione della dinamica solo alcune volte restituisce dei valori leggermente diversi da quelli del modello, in altre situazioni il modello segue perfettamente le dinamiche del robot. A tale proposito viene attuato il giunto tre, in prima prova solo per verificare le dinamiche, poi nella seconda prova, attuandolo con un certo moto uguale ed opposto a quello applicato al giunto due, in modo da far traslare il link tra giunto due e tre. In quest'ultimo caso le dinamiche non risultano corrispondenti per il giunto tre.

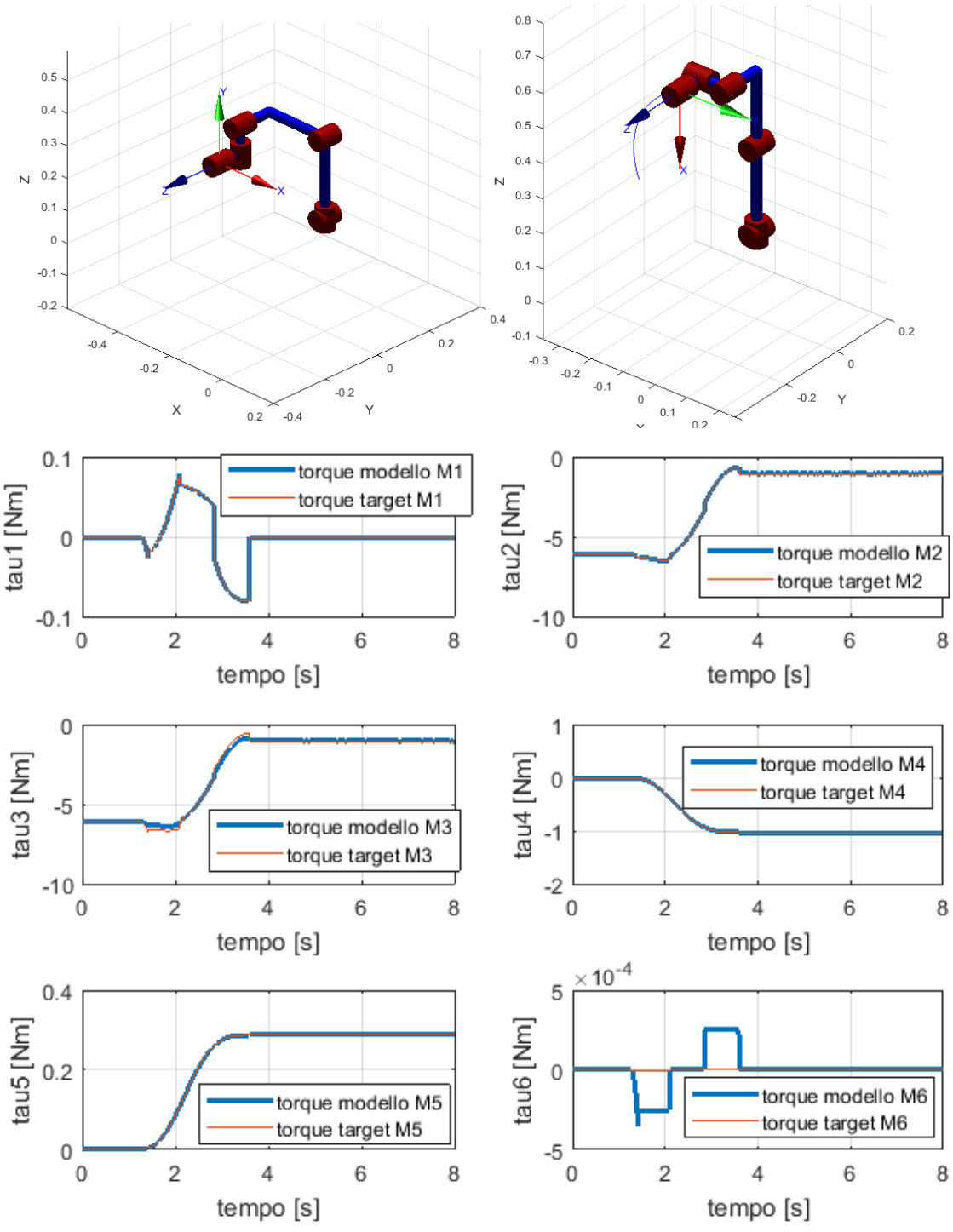


Figura 116: caso 1, attuazione del solo giunto 3

- ***Programmo scritta:*** PROVA_11_2_movimento_giunto_2e3.urp

  Acquisizione corrispondente: ACQUISIZIONE_11_2_movimento_giunto_2e3.mat

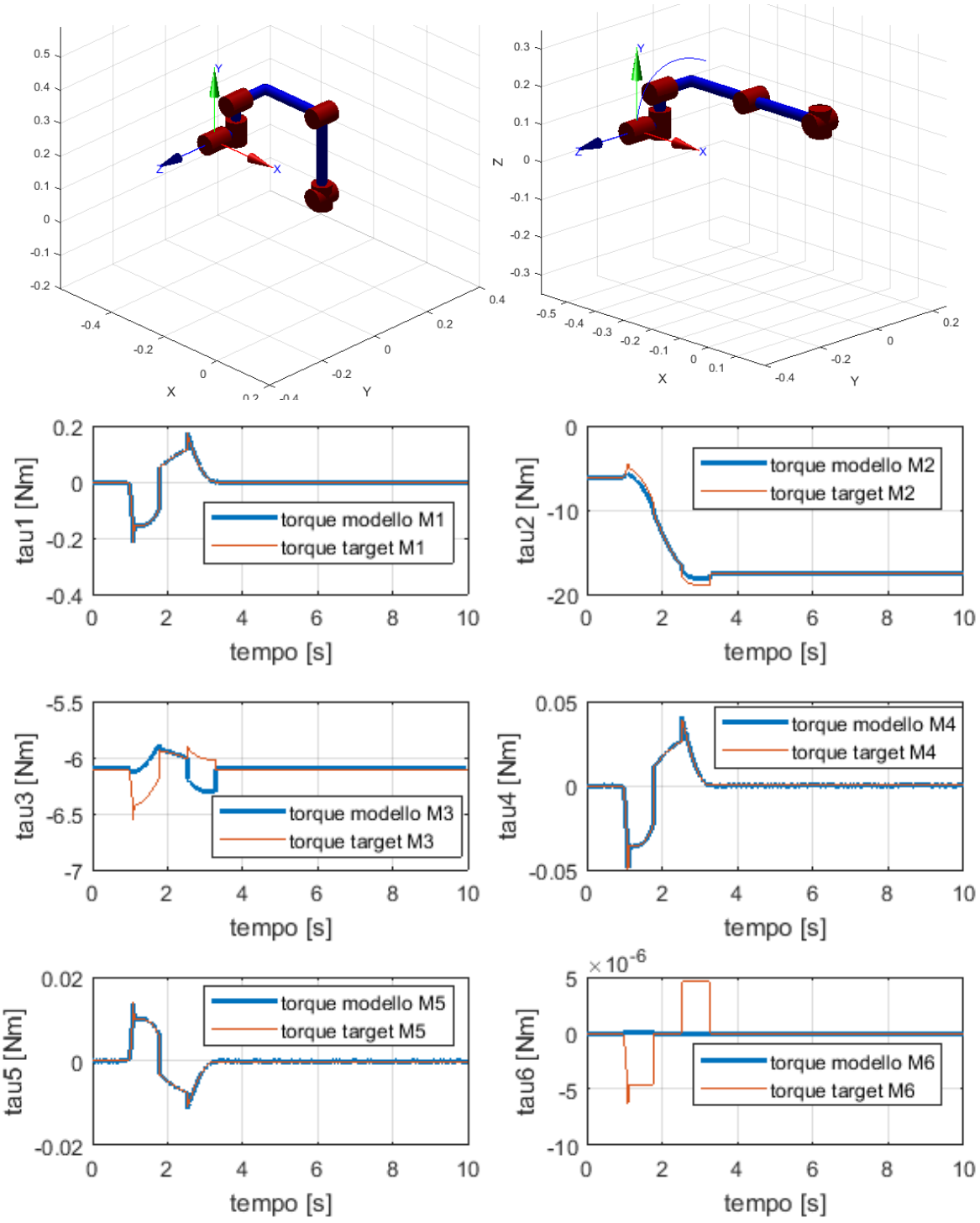


Figura 117: caso 2, attuazione del giunto 2 e 3

Si può osservare come nel primo caso la coppia si riduca man mano che il link tende a raggiungere la posizione verticale, questo è probabilmente dovuto al momento esercitato dalla forza di gravità che diminuisce man mano che il link assume il target verticale. In questo caso le coppie calcolate dal modello sono fedeli a quelle misurate. Nel secondo caso, la coppia rimane costante perché l'azione esercitata dalla gravità è sempre la stessa, poiché il suo braccio rimane inalterato. In questa seconda prova si osserva un errore abbastanza importante proprio sul terzo giunto (come visto precedentemente) che sembra essere dovuto ad un offset legato ai parametri inerziali stimati come già visto in precedenza.

- ***Programma scritto:*** PROVA_5_1_spostamento_in_rapido_movej.urp
  Acquisizione corrispondente: ACQUISIZIONE_5_1_spostamento_in_rapido_movej.mat

Prima di avviare qualsiasi programma che permetta di far muovere il robot, il software richiede di riportare il sistema nella condizione iniziale con una modalità detta *AUTOMOVE.*

A seconda di quale sia stata l'ultima azione eseguita da parte dell'operatore, il robot eseguirà un MoveJ o un MoveL quando il tasto automove è impiegato (anche se è di solito utilizzata dal software la prima modalità poiché più semplice, veloce e meno vincolante). Le impostazioni di sicurezza predefinite di velocità e accelerazione massime sono riportate di seguito [4].

Tabella 25: condizioni limiti di velocità e accelerazione in automove [7]

MOVEJ	vel = 0.524 rad/s (30°/s)	acc = 0.349 rad/s^2 (20°/s^2)
MOVEL	vel = 0.1 m/s	acc = 0.1 m/s^2

In condizione di sicurezza anche tali valori saranno ridotti affinché nessuna impostazione di sicurezza venga violata. Inoltre, tali valori possono essere ridotti durante l'*automove* variando il valore della barra in basso indicante la velocità del movimento. Dall'acquisizione riportata nella libreria è possibile osservare il movimento avvenuto durante un *automove* di tipo MoveJ e verificare i valori massimi indicati in tabella.

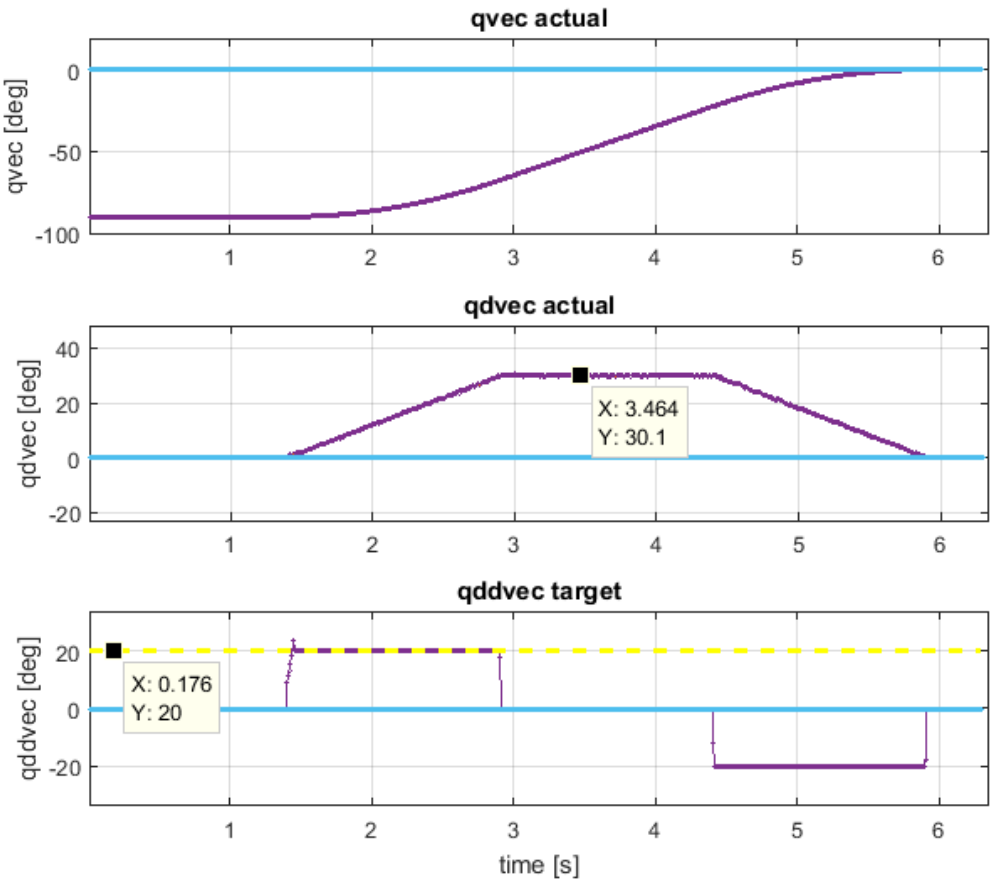


Figura 118: qvec, qdvec e qddvec in automove

4. SICUREZZA E MODALITA' DI ARRESTO DEL ROBOT

4.1 Impostazioni di sicurezza

I robot UR sono dotati di funzioni speciali di sicurezza concepite specificamente per permettere il funzionamento collaborativo in cui il sistema robotico opera senza barriere e/o affiancando una persona.
Uno dei compiti più importanti di cui si deve occupare l'operatore è l'esecuzione di una valutazione del rischio.
Alcune funzioni di sicurezza sono specificatamente progettate per applicazioni collaborative del robot. Le impostazioni di sicurezza consentono di applicare tali funzioni, fondamentali per eliminare pericoli specifici riportati nella valutazione del rischio effettuata dall'operatore [4].
È possibile attuare le seguenti limitazioni: *limitazione di forza, limitazione di potenza, limitazione di velocità, limitazione della quantità di moto* e *limitazione di posizione e orientamento del TCP e dell'utensile/attuatore finale.*

Le impostazioni di sicurezza sono composte da un certo numero di valori utilizzati per limitare i movimenti del braccio robotico e da impostazioni legate agli ingressi e alle uscite configurabili.
Si possono individuare dunque le seguenti opzioni:

1 la sotto-scheda **General limits,** determina forza, potenza, velocità e quantità di moto massimi del robot. Quando il rischio di colpire una persona o di entrare in collisione con un componente nel proprio ambiente è particolarmente alto, si devono selezionare valori bassi per queste impostazioni. Se il rischio è ridotto, dei limiti generali più alti permettono al robot di muoversi più rapidamente ed esercitare maggiore forza sull'ambiente circostante;

2 la sotto-scheda **Joint limits,** include i limiti di velocità giunto e posizione giunto. I limiti di velocità giunto determinano la velocità angolare massima dei singoli giunti e servono a limitare ulteriormente la velocità del braccio robotico. I limiti di posizione giunto determinano la gamma di posizioni ammesse dei singoli giunti (nello spazio giunto).

3 la sotto-scheda **Boundaries,** definisce i piani di sicurezza (nello spazio cartesiano) e un limite di orientamento per il centro utensile del robot. I piani di sicurezza possono essere configurati come limiti inflessibili per la posizione del centro utensile robot, o come inneschi per l'attivazione dei limiti di sicurezza in "modalità Ridotta". Il limite di orientamento utensile stabilisce un limite inflessibile sull'orientamento del centro utensile robot;

4 la sotto-scheda **Safety I/O,** definisce le funzioni di sicurezza per gli ingressi e le uscite configurabili. Ad esempio, l'arresto di emergenza può essere configurato come ingresso.

Nel momento in cui vengono modificate tali condizioni di sicurezza, viene sospesa l'alimentazione del braccio e non è possibile rifornirla fino a quando le modifiche non siano state applicate o annullate.

Sono disponibili due metodi per configurare i limiti di sicurezza generali all'interno dell'installazione: *Impostazioni di base* e *Impostazioni avanzate*.

Definendo i limiti di sicurezza generali però, si determinano solo i limiti per l'utensile e non i limiti complessivi del robot. Ciò significa che sebbene si definisca un limite di velocità, ciò non garantisce che altre parti del braccio si mantengano entro lo stesso limite [4].

Di seguito si approfondiscono i punti elencati precedentemente.

4.1.1 General Limits: impostazioni di base e avanzate

Il pannello dei limiti generali iniziali, visualizzato come schermata base, presenta un cursore con quattro set di valori predefiniti per i limiti di forza, alimentazione (Potenza), velocità e quantità di moto, in modalità Normale e Ridotta (dove la ridotta è sempre uguale a quella più restrittiva e precedente alla configurazione scelta come normale ed è uguale alla normale solo se si seleziona "Very restricted")

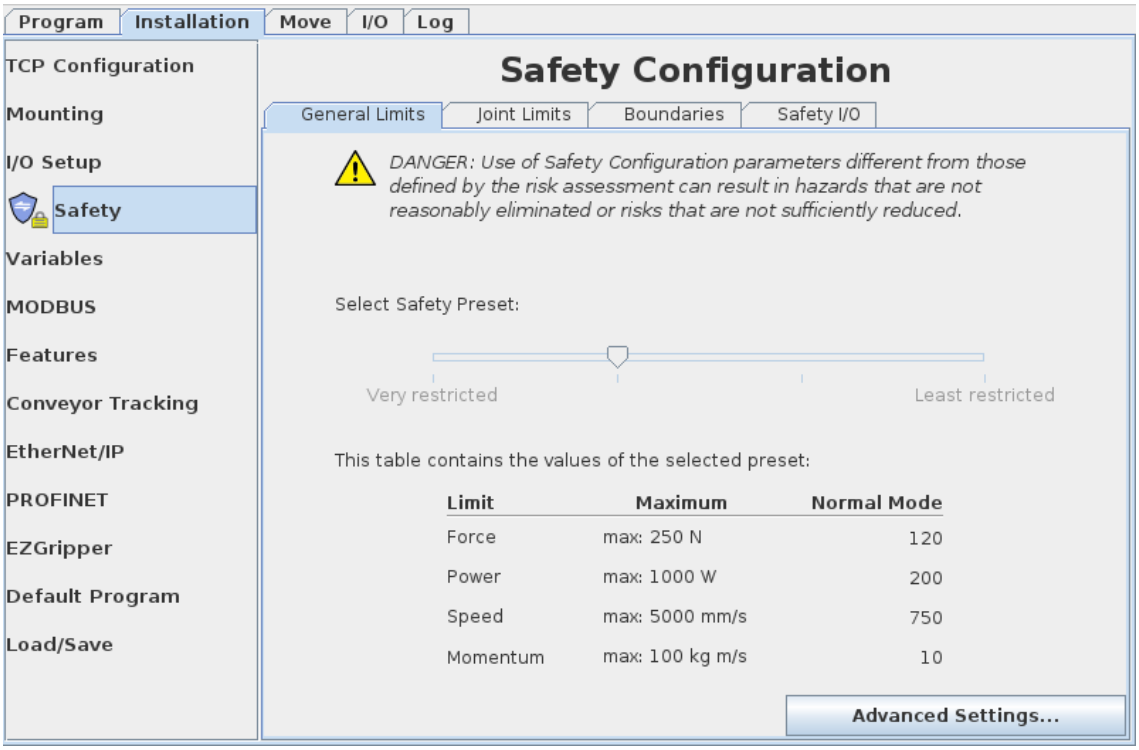


Figura 119: General Limits, pagina base

Tabella 26: set predefinito di sicurezza

	very restricted	restricted I	restricted II	Least restricted	MAX	MIN
FORCE [N]	100	120	150	250	250 [N]	50 [N]
POWER [W]	80	200	300	1000	1000 [W]	80 [W]
SPEED [mm/s]	250	750	1500	5000	5000 [mm/s]	160 [mm/s]
MOMENTUM [Kg m/s]	5	10	25	100	100 [Kg m/s]	4 [Kg m/s]

Nel momento in cui nessuno dei set predefiniti soddisfa le condizioni necessarie, è possibile scegliere le impostazioni avanzate e modificare ogni parametro indipendentemente, ovviamente mantenendo il valore nei limiti massimi e minimi consentiti.

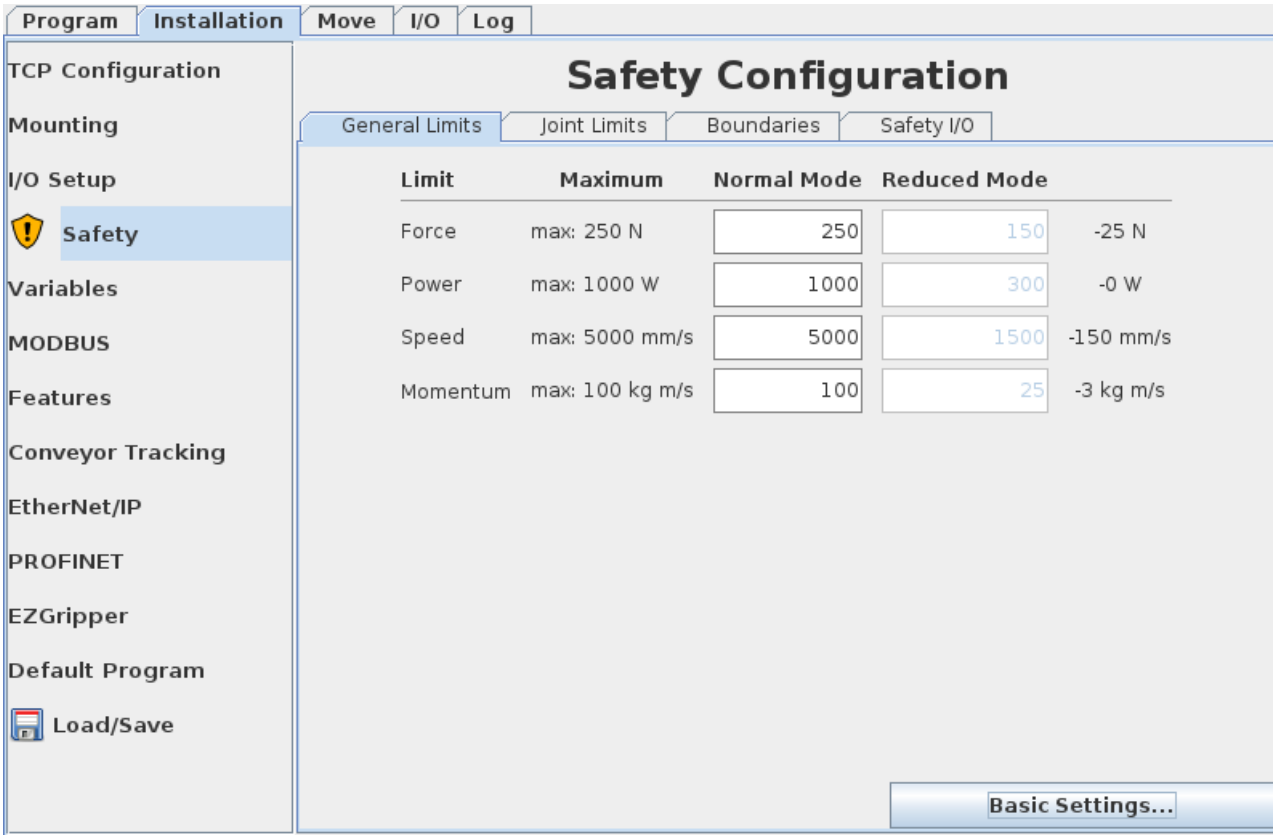


Figura 120: General limits, impostazioni avanzate

La modalità ridotta può essere attivata in caso di superamento da parte del TCP di un piano di sicurezza ridotto oppure tramite un ingresso configurabile. In questo caso la colonna Reduced mode non può essere modificata perché non vi è alcuna modalità ridotta attiva.

La tolleranza e l'unità di misura di ciascun limite sono elencati alla fine della rispettiva riga. Quando un programma è in esecuzione, la velocità del braccio robotico viene regolata automaticamente per evitare di superare i valori immessi meno la tolleranza. Notare che il segno meno visualizzato con il valore di tolleranza

è presente solo per indicare che la tolleranza va sottratta dal valore effettivo immesso. Il sistema di sicurezza esegue una categoria di arresto 0 (approfondita nel paragrafo successivo) se il braccio robot supera il limite (senza tolleranza).

4.1.2 Joint Limits: impostazioni di base e avanzate

I Joint Limits restringono il movimento dei singoli giunti nello spazio, ovvero non fanno riferimento allo spazio cartesiano ma piuttosto alla posizione interna (rotazionale) dei giunti e alla loro velocità di rotazione.

- VELOCITA' MASSIMA

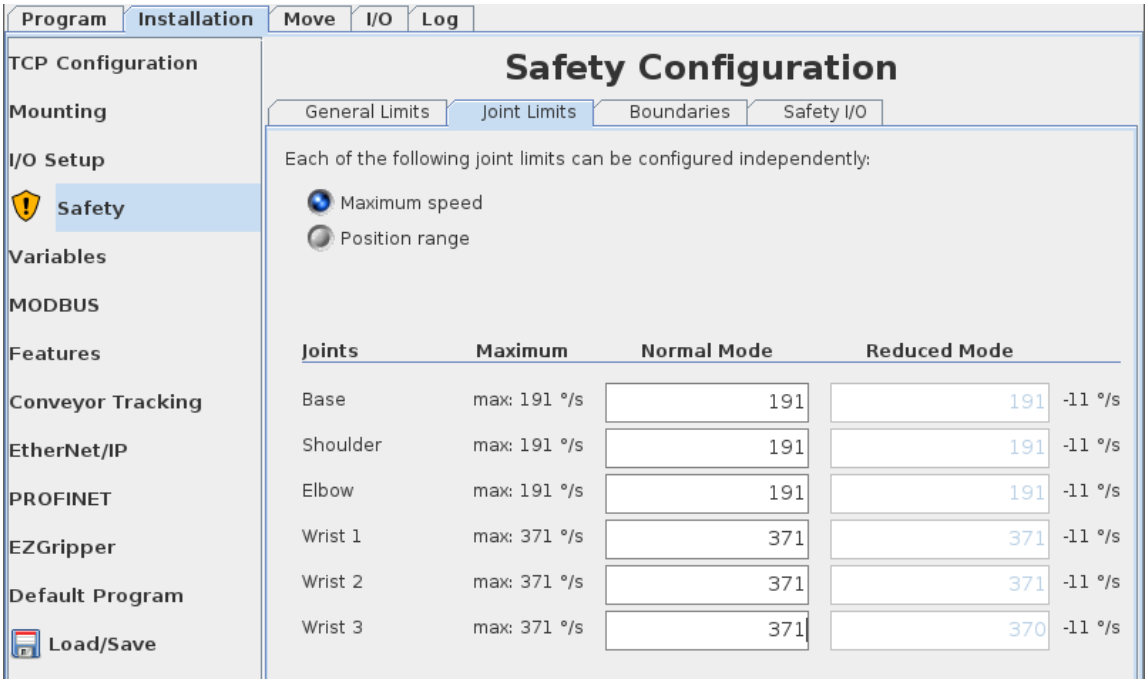


Figura 121: Joint Limit,: schermata base velocità

Questa opzione determina la velocità angolare massima di ciascun giunto. Ciò si esegue selezionando il campo di testo appropriato e immettendo un nuovo valore. Il valore massimo ammesso è elencato nella colonna Massimo, mentre la tolleranza non può essere cambiata ed è riportata sulla destra.

Per verificare ulteriormente questi valori e comprendere quindi le velocità massime e minime alla quale possono muoversi i giunti, sono state fatte delle prove riportate nella libreria e di seguito presentate.

Nel momento in cui si impostano i parametri di velocità e accelerazione possiamo scegliere dei valori arbitrari compresi in determinati range.

Ipotizzando di selezionare la voce Least restricted nelle impostazioni di sicurezza (ovvero quella meno restrittiva di tutte) si può osservare che:

Tabella 27: velocità e accelerazione limite per giunto

velocità minima	**velocità massima**
per tutti i giunti 0.1	○ Per i giunti 1 2 e 3 180°/s ○ Per i giunti 4 5 e 6 360°/s
accelerazione minima	**accelerazione massima**
per tutti i giunti 0.1	per tutti i giunti 4584°/s

Nel caso in cui venisse impostata una condizione più restrittiva la velocità viene saturata mentre i limiti di accelerazione variano a seconda del giunto mosso e dei parametri di sicurezza. Questo fenomeno può essere osservato nei seguenti grafici, che mostrano inoltre anche i limiti riportati su in tabella 27.

- ***Programma scritto:*** PROVA_12_1_simulatore_giunto1_vel_acc_max.urp
  Acquisizione corrispondente: ACQUISIZIONE_12_1_simulatore_giunto1_vel_acc_max.mat

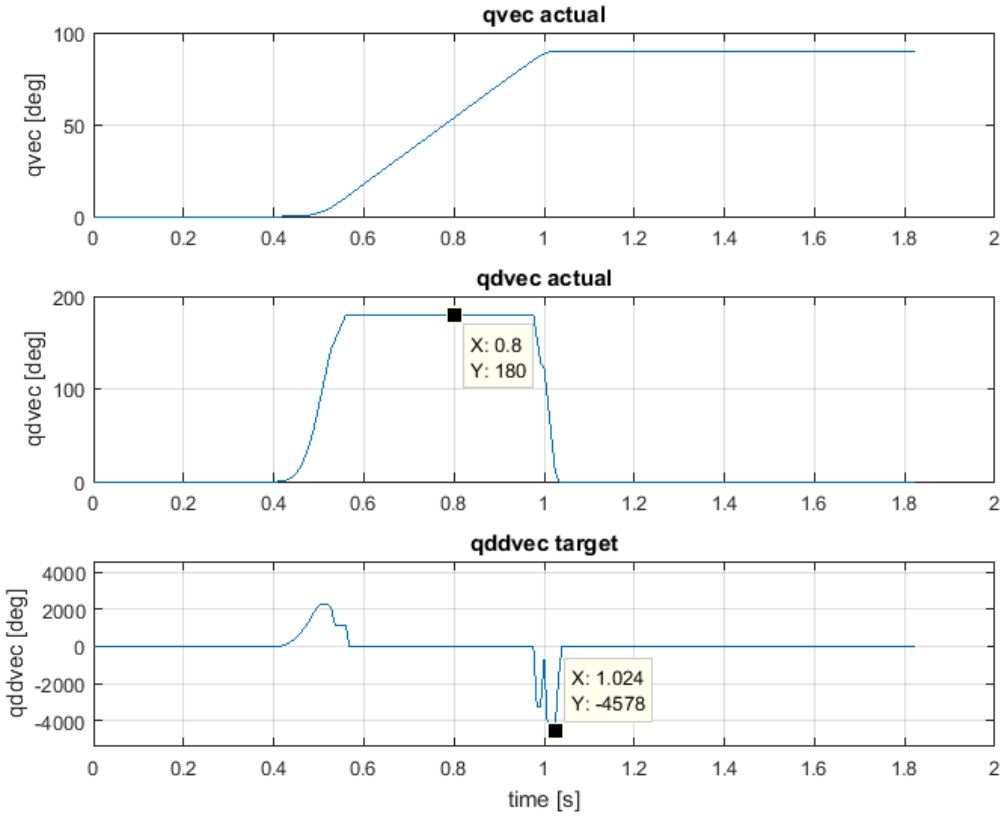


Figura 122: movimento giunto uno nelle condizioni di sicurezza meno restrittive

- ***Programma scritto:*** PROVA_12_2_simulatore_giunto1_vel_acc_max_condizioniristrette.urp

  Acquisizione corrispondente:

  ACQUISIZIONE_12_2_simulatore_giunto1_vel_acc_max_condizioniristrette.mat

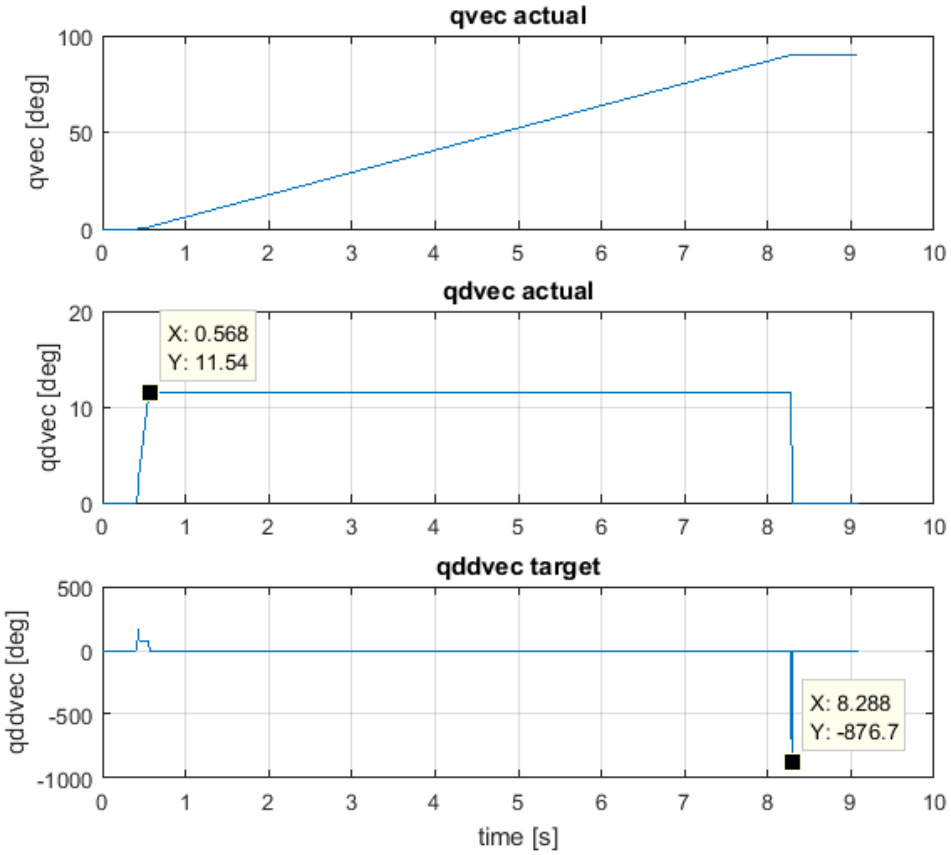


Figura 123: movimento giunto uno nelle condizioni di sicurezza più restrittiva

Si può osservare una particolarità legata al trapezio di velocità, che risulta in entrambi i casi non simmetrico.

Questo probabilmente è dovuto al fatto che in fase di salita, per raggiungere quelle determinate velocità, non è necessario giungere all'accelerazione massima; proprio per questo la salita è molto meno ripida dell'andamento che è invece presente in fase di discesa. Sembra che nel momento dell'arresto il robot preferisca fermarsi sfruttando il massimo valore dell'accelerazione fermandosi il prima possibile, annullando quindi la velocità.

- ***Programma scritto:*** PROVA_13_1_simulatore_giunto4_vel_acc_max.urp

  Acquisizione corrispondente: ACQUISIZIONE_13_1_simulatore_giunto4_vel_acc_max.mat

Le stesse considerazioni possono essere fatte per le due figure successive dove avviene un movimento del giunto quattro di 90°. Si può osservare la differenza di velocità massima sottolineata nella tabella 27 tra il giunto uno, che raggiunge un massimo di 180°/s, e il giunto quattro che raggiunge i 360°/s.

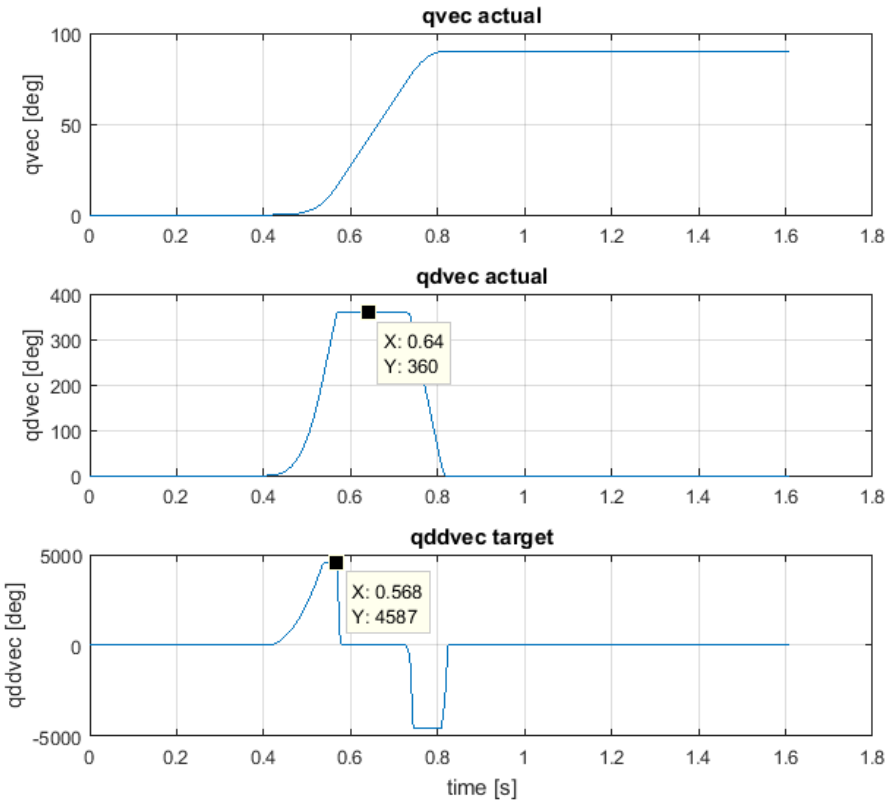


Figura 124: movimento giunto quattro nelle condizioni di sicurezza meno restrittive

- ***Programma scritto:*** PROVA_13_1_simulatore_giunto4_vel_acc_max_ condizionirestrittive.urp
  Acquisizione corrispondente:
  ACQUISIZIONE_13_1_simulatore_giunto4_vel_acc_max_ condizionirestrittive.mat

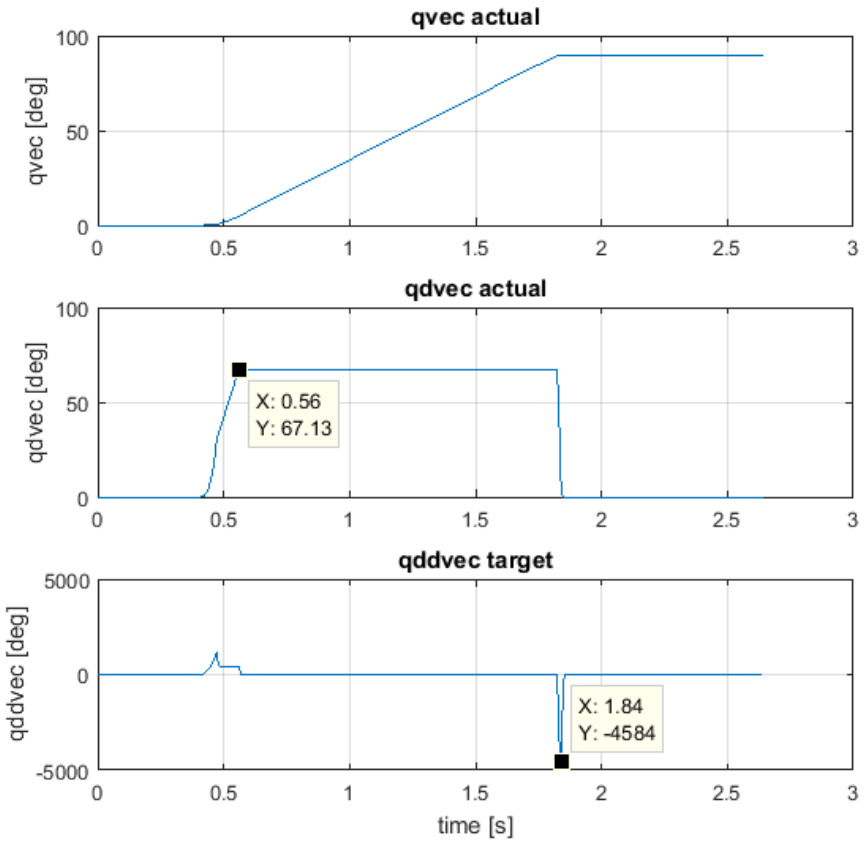


Figura 125: movimento giunto quattro nelle condizioni di sicurezza più restrittive

- GAMMA POSIZIONI

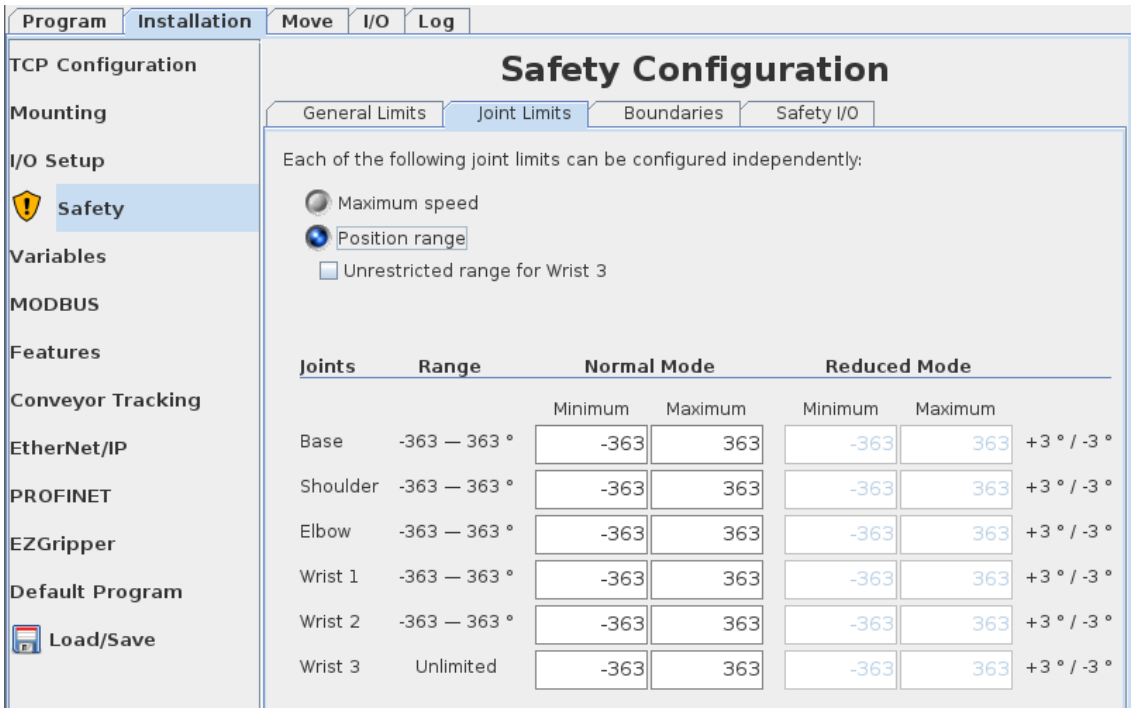


Figura 126: Joint Limits, gamma posizioni

La schermata in figura 126 determina la gamma di posizioni per ciascun giunto. Questa si imposta selezionando i campi di testo corrispondenti e digitando nuovi valori per i limiti di posizione inferiore e superiore del giunto. L'intervallo immesso deve rientrare nei valori elencati nella colonna intestata Range ed il limite inferiore non può essere maggiore del limite superiore.

Si noti che in entrambi i casi, i campi per i limiti in modalità Ridotta sono disabilitati quando non sono definiti ne piani di sicurezza ne ingressi configurabili per innescarla. Inoltre, anche in questo caso valgono le stesse considerazioni fatte in precedenza sulle tolleranze e sulla regolazione automatica della velocità da parte del sistema. Ci si imbatte nella modalità di arresto 0 (nel paragrafo successivo) in caso di superamento della soglia di sicurezza [4].

4.1.3 Boundaries

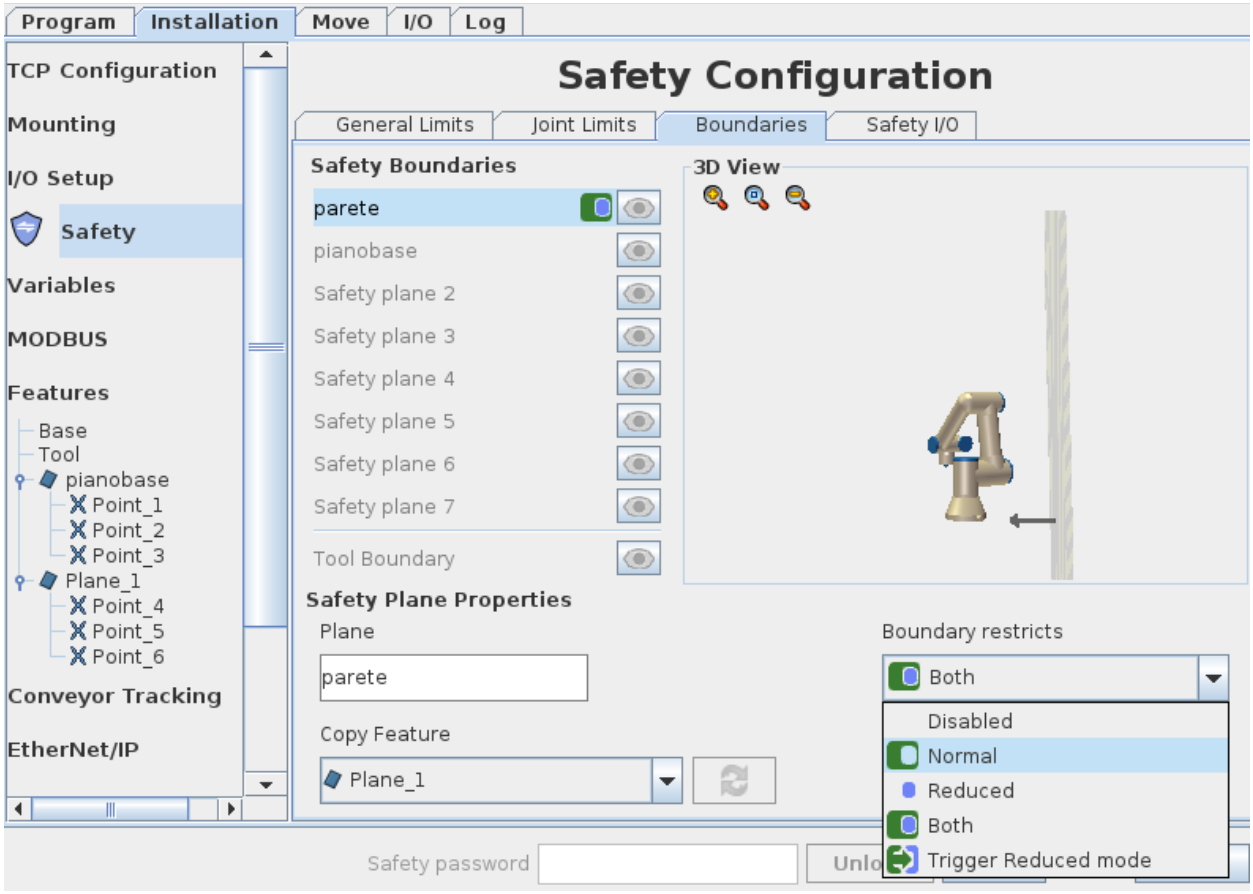


Figura 127: Boundaries, schermata piani di sicurezza

Per poter applicare delle condizioni di sicurezza opportune è possibile impostare dei piani in cui il movimento del robot sia limitato in velocità oppure sia bloccato.

In questo modo è possibile limitare i movimenti della macchina nel caso in cui sia presente un operatore, compensando in parte il problema dell'urto. Come già sottolineato, se il robot dovesse urtare una sua qualsiasi parte, eccetto il TCP, non è detto che si blocchi. Applicando invece una parete virtuale è possibile maggiorare le condizioni di sicurezza. Tra le possibili opzioni è possibile settare le seguenti:

- modalità di sicurezza normale, con la quale il robot si ferma al contatto con la parete virtuale in caso di condizioni di sicurezza normali;
- modalità di sicurezza ridotta, con la quale il robot si ferma al contatto con la parete virtuale in caso di condizioni di sicurezza ridotte;
- entrambi, con la quale il robot si ferma a contatto con la parete virtuale in qualsiasi caso di condizione di sicurezza;
- modalità trigger, con la quale il robot transita attraverso la parete ma con parametri di potenza e velocità ridotte.

È possibile inoltre, impostare un limite all'orientazione del TCP per evitare che questo possa assumere particolari posizioni.

Nel caso del robot in laboratorio è stato necessario aggiungere 2 pareti virtuali:

- la prima per la base, in questo modo durante i movimenti non si rischia che il robot vada a sbattere sul tavolo;
- la seconda invece, è stata impostata per il bordo posteriore del piano di lavoro poiché in caso di movimento non controllato il robot rischierebbe di urtare la parete della stanza.

Figura 128: piani di sicurezza impostati

Ovviamente nel momento in cui sono state aggiunte la pinza e la cella di carico questi piani sono stati traslati di una certa distanza, poiché anche il TCP del robot è stato spostato nel punto terminale della pinza.

4.1.4 *Safety I/O*

Possono essere infine configurate anche delle condizioni di sicurezza per eventuali segnali di input o output gestiti dal software. Ogni funzione può controllare una sola coppia di I/O, se si prova a selezionare la stessa funzione di sicurezza una seconda volta, la si rimuove dalla coppia precedentemente impostata.

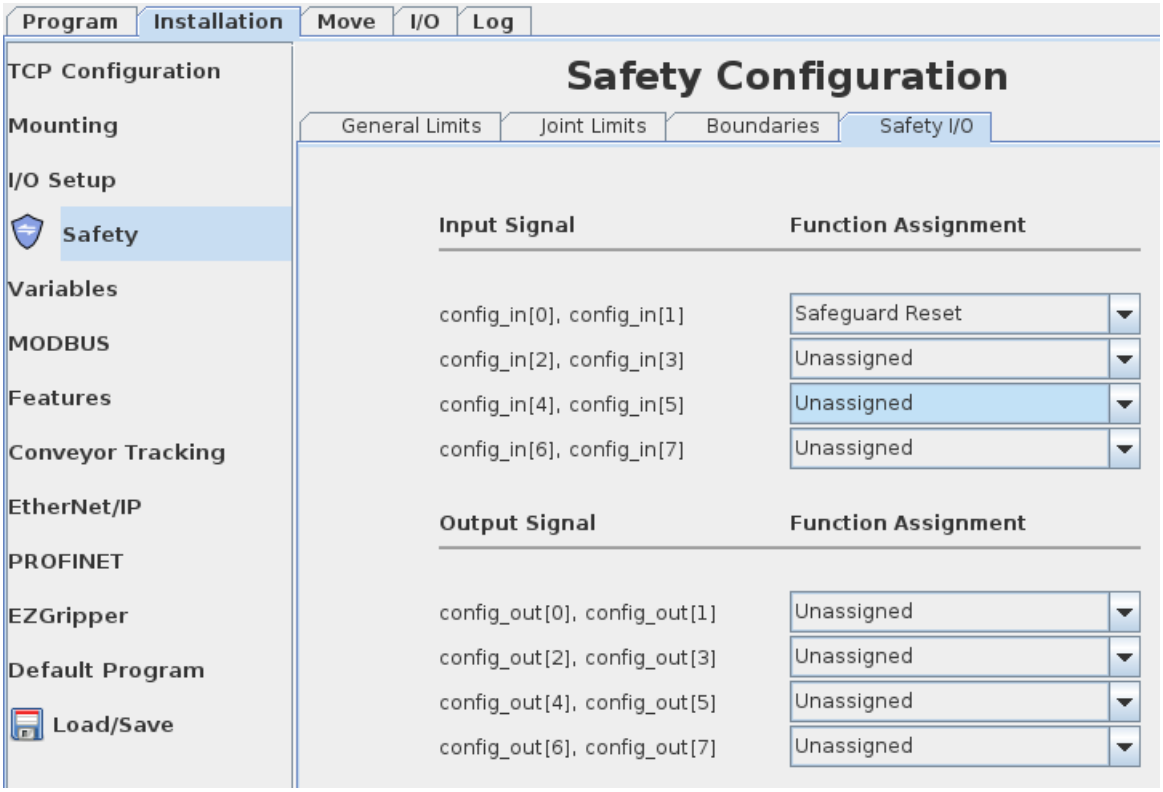


Figura 129: schermata base Safety I/O

Sicurezza per i segnali input

Per i segnali in ingresso, si possono selezionare le seguenti funzioni di sicurezza [4]:

- *Arresto di emergenza sistema,* consente di aggiungere un altro pulsante per l'arresto di emergenza;
- *Modalità ridotta,* permette il passaggio, durante il funzionamento del robot, dalla modalità normale a quella ridotta, facendo rallentare così la macchina;
- *Reset protezione,* garantisce lo stato di protezione fino ad un eventuale ripristino;
- *Dispositivo di abilitazione a 3 posizioni,* permette di far entrare il robot in due configurazioni particolari, la "modalità di esecuzione" in cui il robot compie solo particolari azioni e la "modalità di programmazione" nella quale vengono annullare le precedenti limitazioni;
- *Modalità di funzionamento.*

Sicurezza per i segnali output

Per tali segnali è possibile implementare tutte le precedenti funzioni.

4.2 Tipologie di arresti di sicurezza del robot

Sfruttando alcune prove della libreria è stato possibile analizzare alcuni aspetti legati alle diverse tipologie di fermata del robot. Con il termine "stop category" si individua una classificazione di come il movimento del robot è interrotto in maniera sicura. Ci sono tre tipi di stop differenti [4]:

- **Stop category 0**, il movimento è interrotto tramite rimozione immediata di alimentazione al robot. È un arresto incontrollato, in cui il robot può deviare dal percorso programmato con i freni di ogni giunto che intervengono nel modo più veloce possibile. Questo arresto di protezione viene utilizzato se viene superato un limite relativo alla sicurezza o in caso di guasto nelle parti del sistema di controllo relative alla sicurezza;

- **Stop category 1**, il moto viene fermato con la potenza disponibile del robot per ottenere l'arresto e poi la rimozione di potenza avviene quando l'arresto è raggiunto. Si tratta di un arresto controllato, dove il robot continuerà lungo il percorso programmato. L'alimentazione viene rimossa non appena il robot si ferma. Lo stop deve avvenire in un massimo di 0.6s altrimenti il sistema attuerebbe uno stop di categoria 0;

- **Stop category 2**, un arresto controllato con potenza lasciata disponibile al robot. Il sistema di controllo relativo alla sicurezza controlla che il robot rimanga in posizione di stop.

I tipi di arresto studiati in questo caso sono i seguenti [4]:

- Impatto con una parete virtuale, è un arresto di tipo 2 poiché successivamente all'impatto viene lasciata potenza disponibile al robot;

- Premere pulsante di emergenza, è un arresto di tipo 1. Il robot viene frenato e privato di alimentazione;

- Pausa e ripartenza, è semplicemente un comando dato dall'operatore. Non è uno stop di emergenza.

Si osserva che quando viene rilevata una condizione di sicurezza di stop, la decelerazione inizia dopo 0.024s

4.2.1 *Stop category 0*

Non è stata analizzata una situazione simile ma per certo, se durante un qualche tipo di stop non venissero rispettate le sue condizioni standard, il robot andrebbe in modalità zero e le sue condizioni per la modalità di arresto, nel peggiore dei casi (con carico massimo al tool), sarebbero le seguenti:

- rilevazione del problema in 0.25s;
- tempo di depotenziamento 1s (tempo che intercorre tra la rilevazione e l'abbassamento della tensione del robot fino a 7.3 V);
- tempo totale di reazione 1.25s.

4.2.2 *Stop category 1*

Per approfondire la conoscenza di questo arresto si è pensato di analizzare una delle prove della libreria. Durante il movimento del solo giunto 1, ad un tratto si decide di premere il pulsante rosso di emergenza per poter analizzare i valori acquisiti dal robot. Si nota che:

- ***Programma scritto:*** PROVA_8_1_stop_button.urp
  Acquisizione corrispondente: ACQUISIZIONE_PROVA_8_1_stop_button.mat

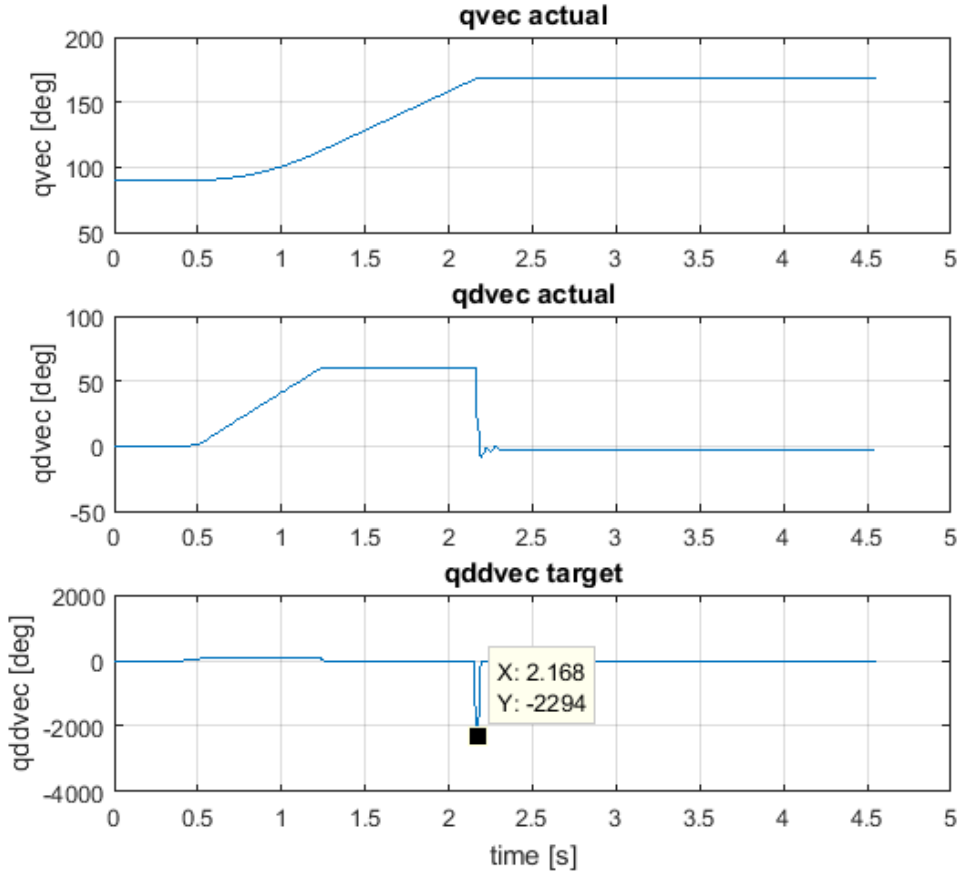


Figura 130: qvec, qdvec e qddvec in emergency stop

Si ha una certa decelerazione che porta ad avere una velocità nulla in un intervallo di tempo molto ridotto. È possibile osservare questo dettaglio dalla figura successiva. Si nota inoltre che la decelerazione, come da default, deve verificarsi entro un tempo di 0.024s dalla manifestazione dell'evento. In questo caso viene verificato (infatti si incomincia ad avere una riduzione di velocità dopo 0.008s), ma se non fosse stato così, il sistema sarebbe andato in modalità stop category 0.

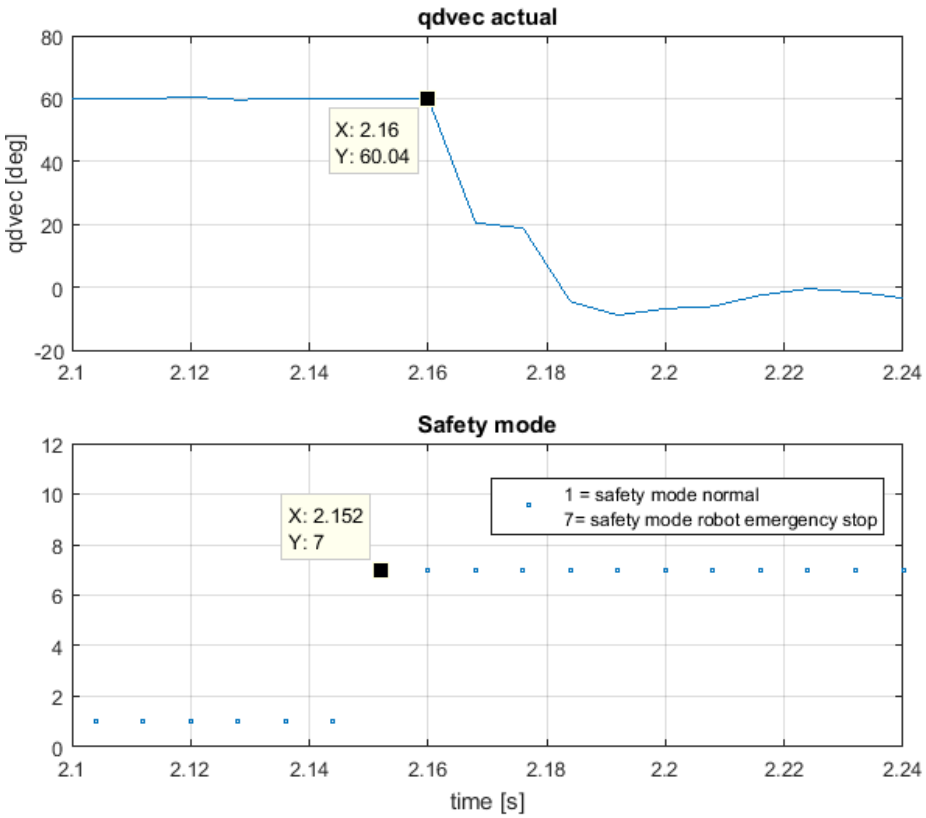


Figura 131: inizio decelerazione

Questo tipo di stop prevede la rimozione della potenza di alimentazione del robot. A tale proposito c'è una condizione sulla tensione del robot. Questo subisce un certo "depotenziamento" ovvero deve subire una variazione di tensione fino a 7 Volt entro un secondo (figura 132). Se non viene rispettata si va in category 0.

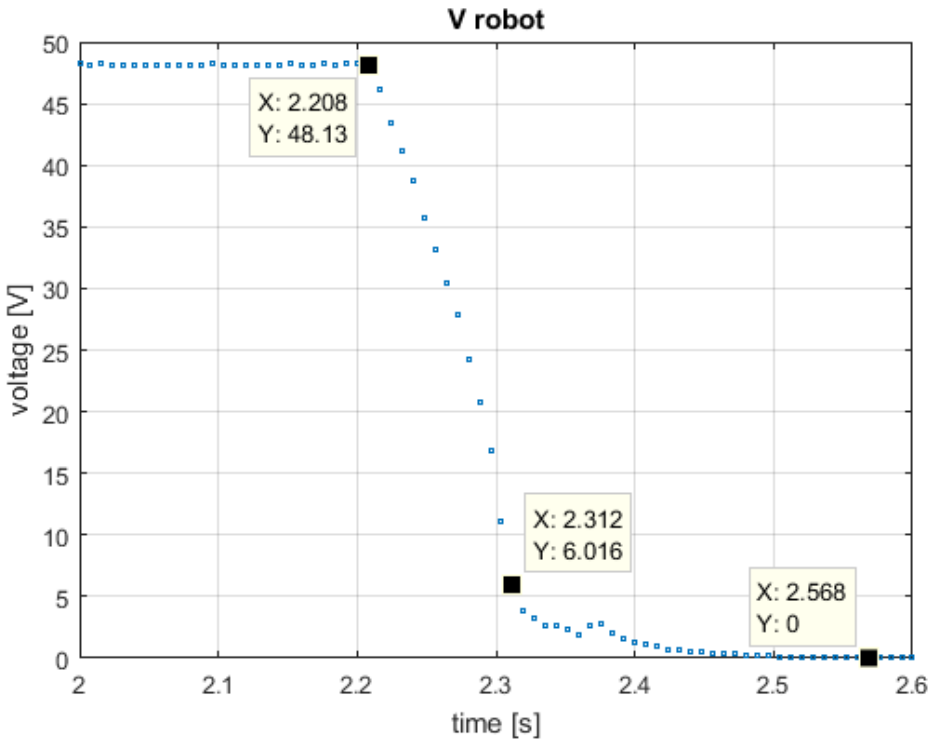


Figura 132: depotenziamento [7]

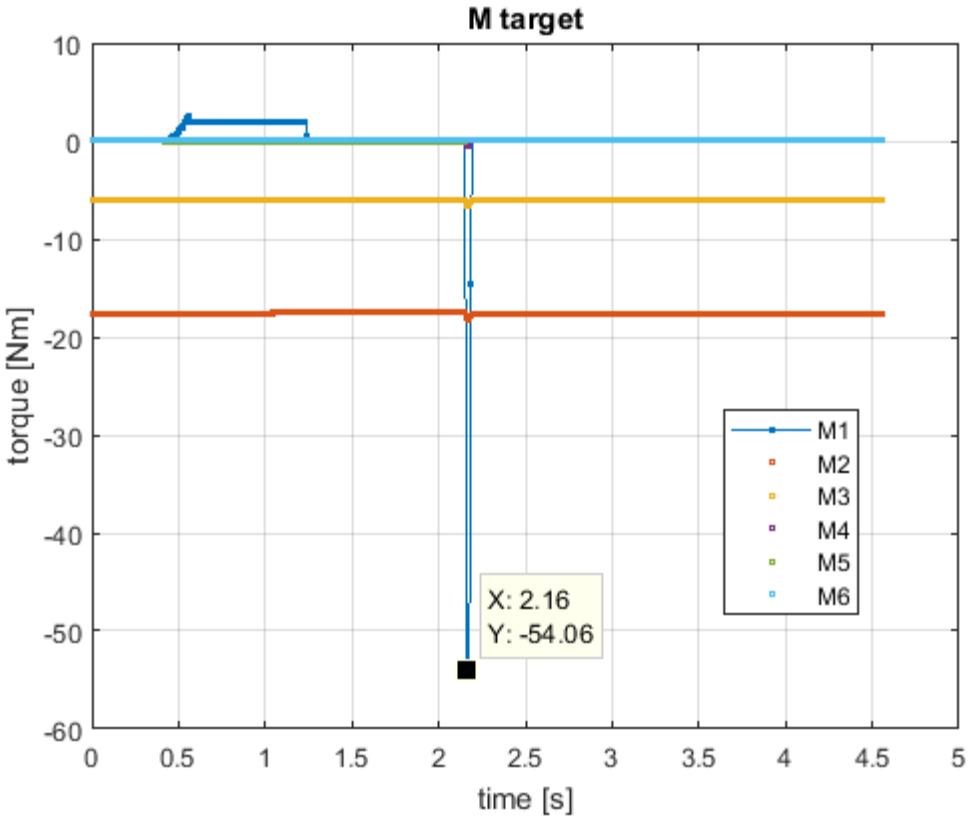


Figura 133: coppie applicate ai giunti

Vengono riportati infine gli andamenti delle coppie applicate ai giunti. Si nota che durante questa tipologia di stop, il robot frena con la coppia massima relativa al giunto che si sta muovendo (in questo caso giunto 1 con coppia massima di circa 56 Nm).

4.2.3 *Stop category 2*

Anche in questo caso viene analizzata una prova per osservare il fenomeno. Durante il movimento del solo giunto 1, ad un tratto si verifica l'impatto con una parete virtuale impostata in modo tale che il robot si fermi al momento dell'urto (poiché è possibile far passare il robot all'interno della parete con una modalità ridotta).

A tale proposito la prova è stata effettuata con diversi valori di velocità, lasciando invariata l'accelerazione.

Si osserva che la modalità di stop dipende dal tipo di velocità con cui si muove il giunto, infatti per velocità ridotte, le accelerazioni e soprattutto le coppie da applicare per bloccare il robot sono minori di quelle massime dello stesso giunto.

Si nota che le modalità di stop sono analoghe a quelle del pulsante rosso, con l'unica differenza relativa alla potenza poiché in questo caso non è annullata.

Tabella 28: Prove utilizzate per lo studio dell'impatto con parete

programma	**acquisizione**	**velocità [deg/s]**
PROVA_6_1_parete_vel_default.urp	ACQUISIZIONE_6_1_parete_vel_default.mat	60
PROVA_6_2_parete_vel_dimezzata.urp	ACQUISIZIONE_6_2_parete_vel_dimezzata.mat	30
PROVA_6_3_parete_vel_al10%.urp	ACQUISIZIONE_6_3_parete_vel_al10%.mat	6

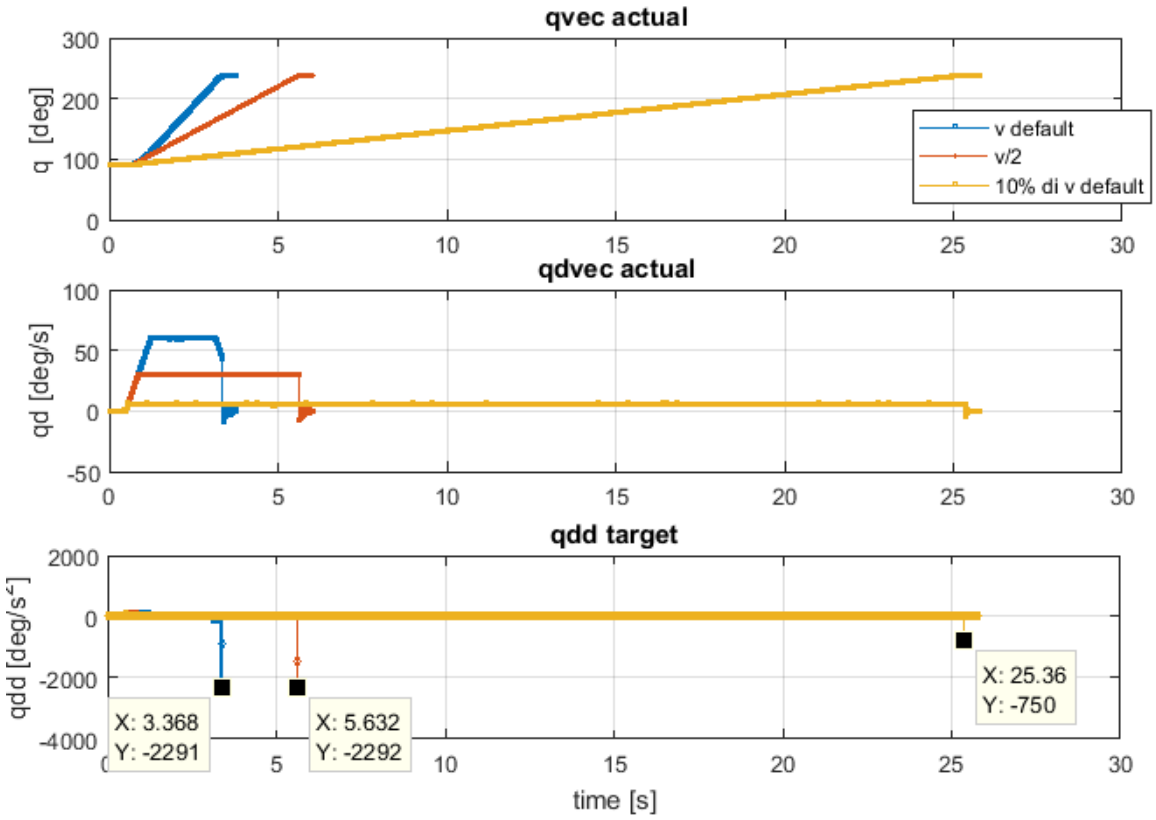


Figura 134: qvec, qdvec e qddvec al variare della velocità del movimento

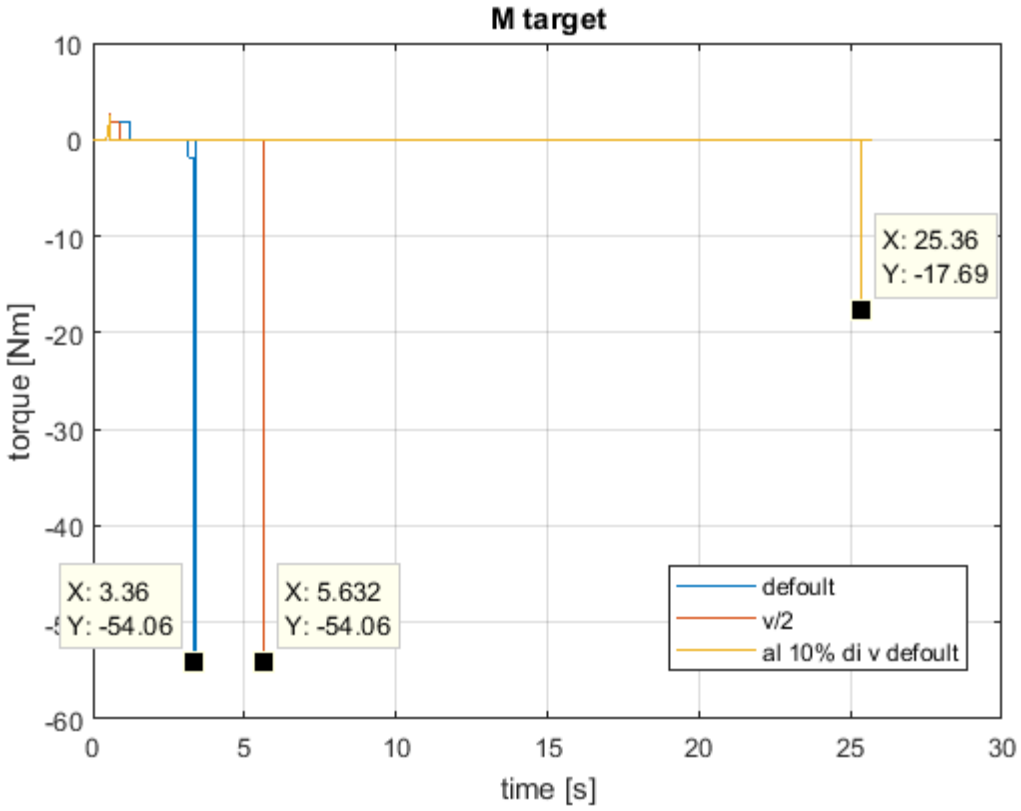


Figura 135: coppie applicate al giunto

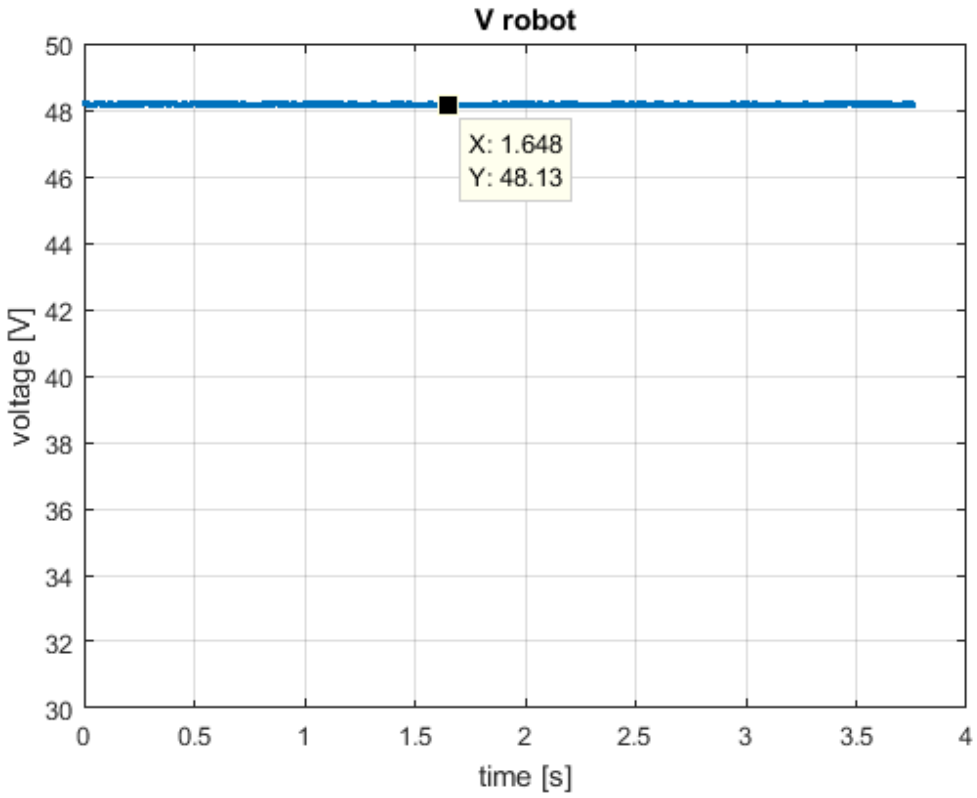


Figura 136: tensione di alimentazione del robot

In questo caso la tensione non si annulla perché il robot non viene privato della potenza.

4.2.4 *Pause and play*

Questo tipo di arresto non rientra in nessuna modalità di fermata di sicurezza poiché viene data in input dall'utente stesso.

Tabella 29: Prove utilizzare per comprendere tale modalità di stop

programma	**acquisizione**	**velocità [deg/s]**
PROVA_7_1_pause_play_vel_default.urp	ACQUISIZIONE_7_1_pause_play_vel_default.mat	60
PROVA_7_2_pause_play_vel_dimezzata.urp	ACQUISIZIONE_7_2_pause_play_vel_dimezzata.mat	30
PROVA_7_3_pause_play_vel_al10%.urp	ACQUISIZIONE_7_3_pause_play_vel_al10%.mat	6

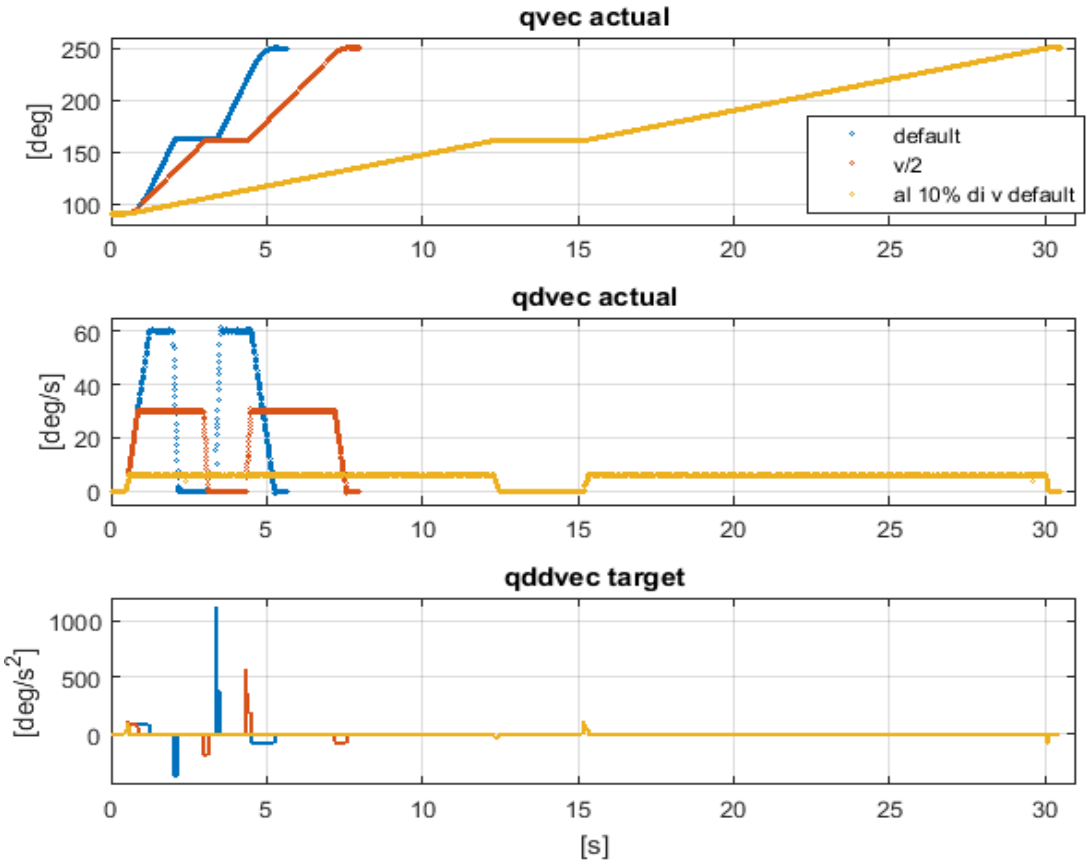


Figura 137: qvec, qdvec e qddvec al variare della velocità di input

Si può osservare che le variazioni di velocità avvengono tutte nello stesso intervallo di tempo, infatti, il modo di arresto dipende dall'energia cinetica che il giunto presenta prima di fermarsi (ovvero dipende dalla sua velocità). In funzione di questa energia, affinchè il tempo di pausa sia lo stesso, bisogna applicare delle coppie maggiori. Riducendo la velocità, la coppia da apportare si dimezza, nel caso di riduzione della velocità al 10% anche la coppia segue lo stesso andamento.

Nelle immagini successive il program state indica quando avvengono le diverse azioni:

- da valore 1 a 2 = inizio movimento;
- valore 3 = è stato pigiato pause;
- valore 4 = si solleva il dito dal pulsante di pause e il robot rimane fermo;
- valore 5 = si preme il pulsante play;
- valore 2 = ripristino movimento fino al raggiungimento del target (1).

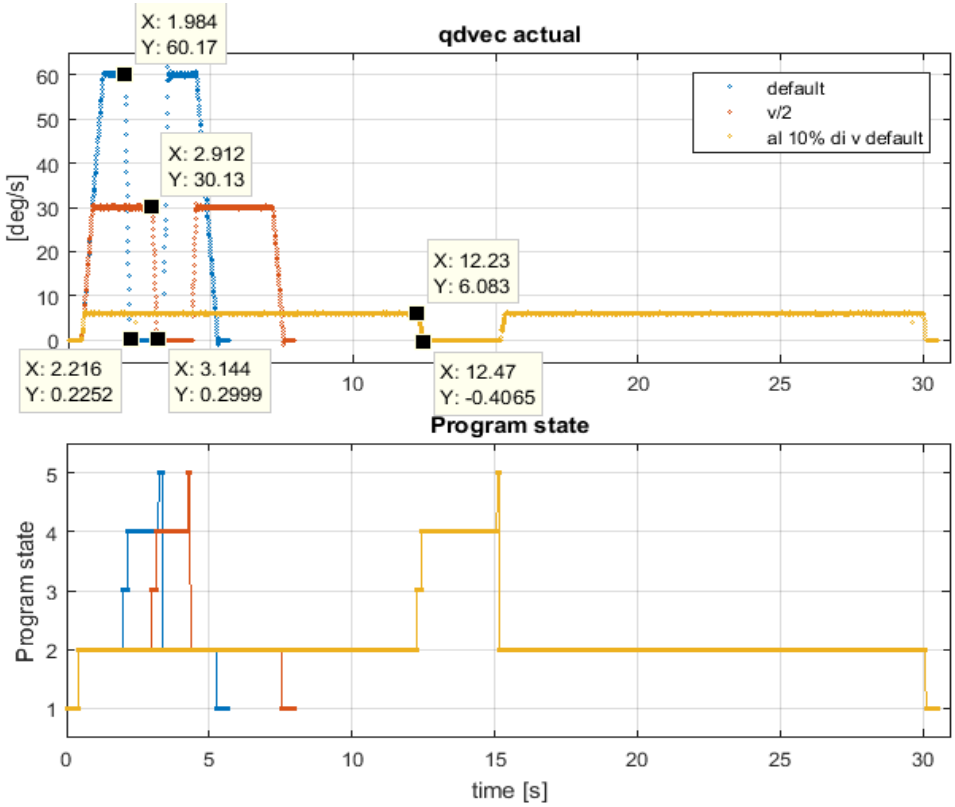


Figura 138: variazioni di qdvec al variare della velocità di input con program state

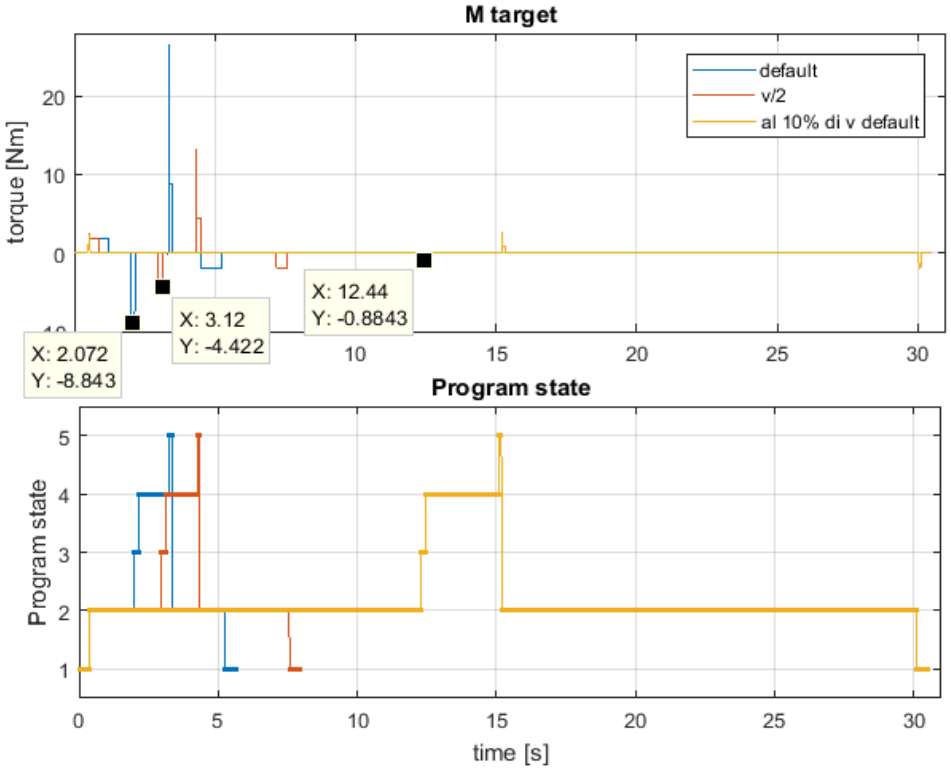


Figura 139: coppie da applicare per la pausa nei diversi casi di velocità del movimento

➢ CONFRONTO TRA LE DIVERSE MODALITA' DI ARRESTO

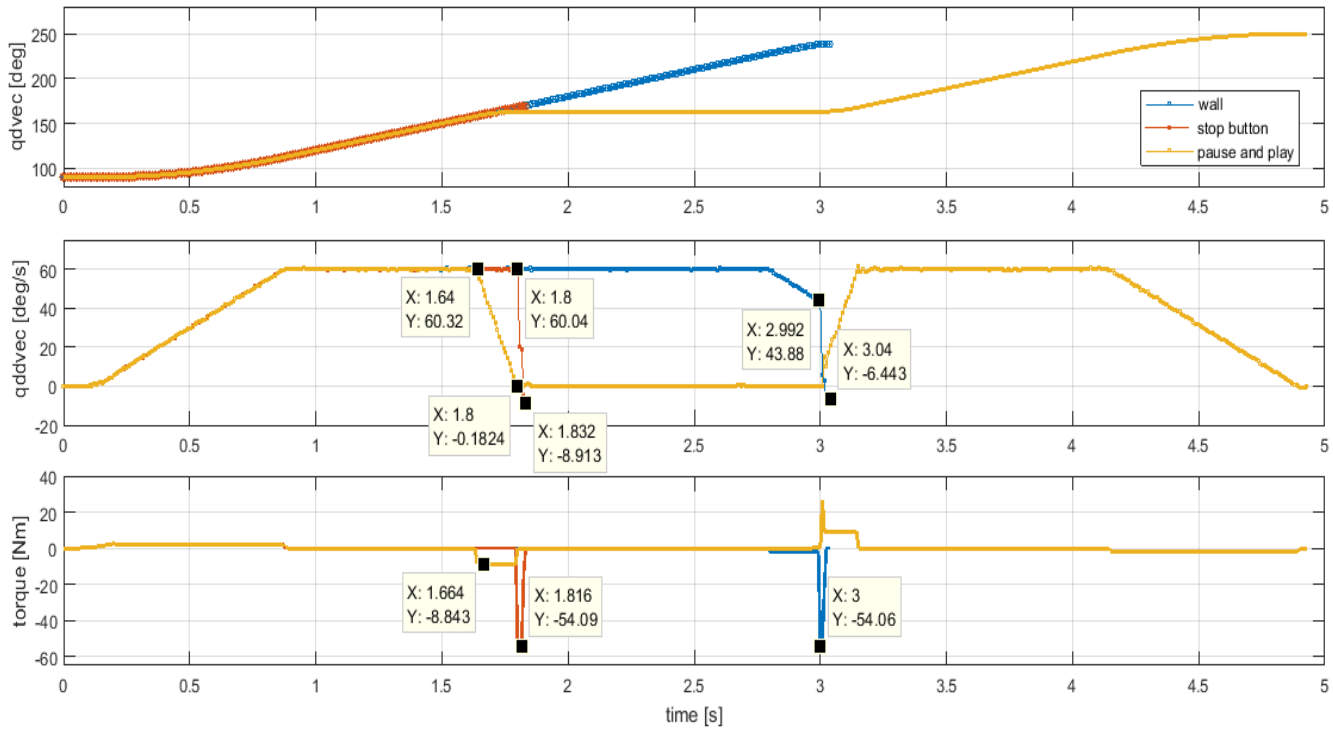


Figura 140: sovrapposizione caso di stop, caso di parete virtuale e caso di stop d'emergenza tutti default

In questo grafico vengono sovrapposte le tre prove default per i tre tipi di stop.

Si evince che per lo stop di category 1 e 2 le coppie esercitate sono le stesse e uguali a quelle massime applicabili. La stessa considerazione può essere fatta per i tempi (c'è una piccola differenza perché l'impatto a parete avviene già durante la fase di decelerazione). Per quanto riguarda "pause and play" le velocità variano in maniera più lenta nella zona di stop e dunque le coppie da applicare sono meno elevate rispetto agli altri due casi.

5. ESEMPI DI APPLICAZIONI

In questa parte conclusiva vengono riportati alcuni programmi realizzati a scopo di analisi di alcune delle funzioni e di applicazioni concrete nella realtà.

5.1 Programmi per l'analisi dei raccordi

Sono stati effettuati diversi movimenti che illustrano le differenze tra i MoveJ e MoveL nelle fasi di transizione con e senza raccordi. Si decide di scrivere un programma che faccia realizzare al robot un rettangolo sul piano. Sono stati valutati quattro casi:

1. tutti i waypoint vengono raggiunti con dei movimenti di tipoL mantenendo la velocità di default. I movimenti vengono effettuati più volte con raggio di raccordo prima nullo poi di 50, 100 e 150mm;
2. tutti i waypoint vengono raggiunti con dei movimenti di tipoL variando la velocità (in alcuni movimenti viene mantenuta quella di default in altri viene utilizzata una velocità e accelerazione ridotta al 10% di quella di default). I movimenti vengono effettuati più volte con raggio di raccordo prima nullo poi di 50, 100 e 150mm;
3. tutti i waypoint vengono raggiunti con dei movimenti di tipoJ mantenendo la velocità di default. I movimenti vengono effettuati più volte con raggio di raccordo prima nullo poi di 50, 100 e 150mm;
4. tutti i waypoint vengono raggiunti con dei movimenti di tipoJ variando la velocità (in alcuni movimenti viene mantenuta quella di default in altri viene utilizzata una velocità e accelerazione ridotta al 10% di quella di default). I movimenti vengono effettuati più volte con raggio di raccordo prima nullo poi di 50, 100 e 150mm;
5. Applicazione: scritta "CIAO".

Le prove infine sono state confrontate tra loro e sono state individuate differenze e particolarità.

1) Per la prima analisi le prove utilizzate sono:

Tabella 30: Prove utilizzate in questa prima analisi

programma	acquisizione
PROVA_14_1_moveL_rettangolo_no_raccordi.urp	ACQUISIZIONE_14_1_moveL_rettangolo_no_raccordi.mat
PROVA_14_2_moveL_rettangolo_r50.urp	ACQUISIZIONE_14_2_moveL_rettangolo_r50.mat
PROVA_14_3_moveL_rettangolo_r100.urp	ACQUISIZIONE_14_3_moveL_rettangolo_r100.mat
PROVA_14_4_moveL_rettangolo_r150.urp	ACQUISIZIONE_14_4_moveL_rettangolo_r150.mat

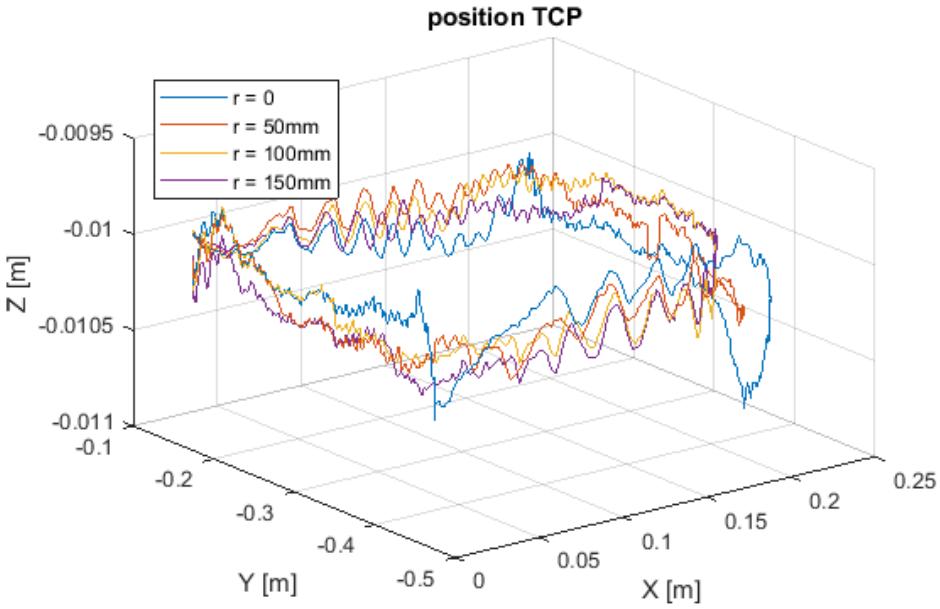


Figura 141: percorso del TCP nello spazio

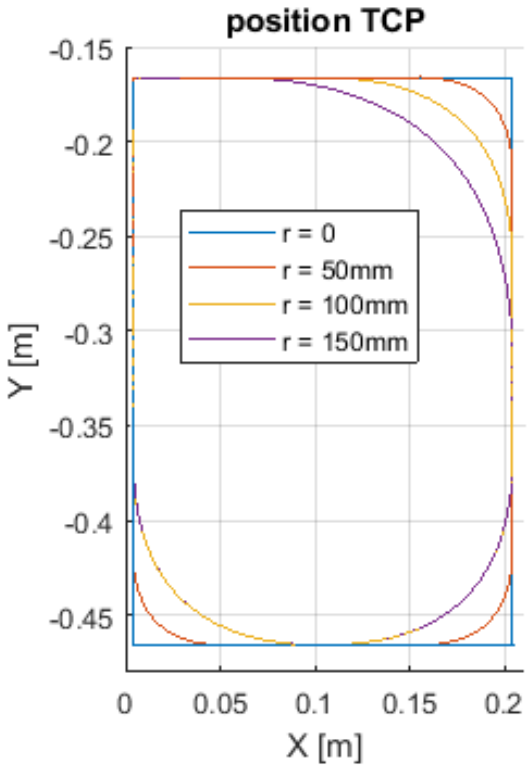


Figura 142: posa del TCP vista sul piano X-Y

In questo caso i raccordi sono puliti ed eseguiti in maniera corretta. Si osservano delle piccole oscillazioni lungo la direzione z che tuttavia sono dell'ordine di decimi di millimetro, quindi trascurabili. In ultima osservazione, poiché non è stato impostato un loop infinito, il punto di partenza coincide con quello finale e per questo non presenta dei raggi di transizione come negli altri casi.

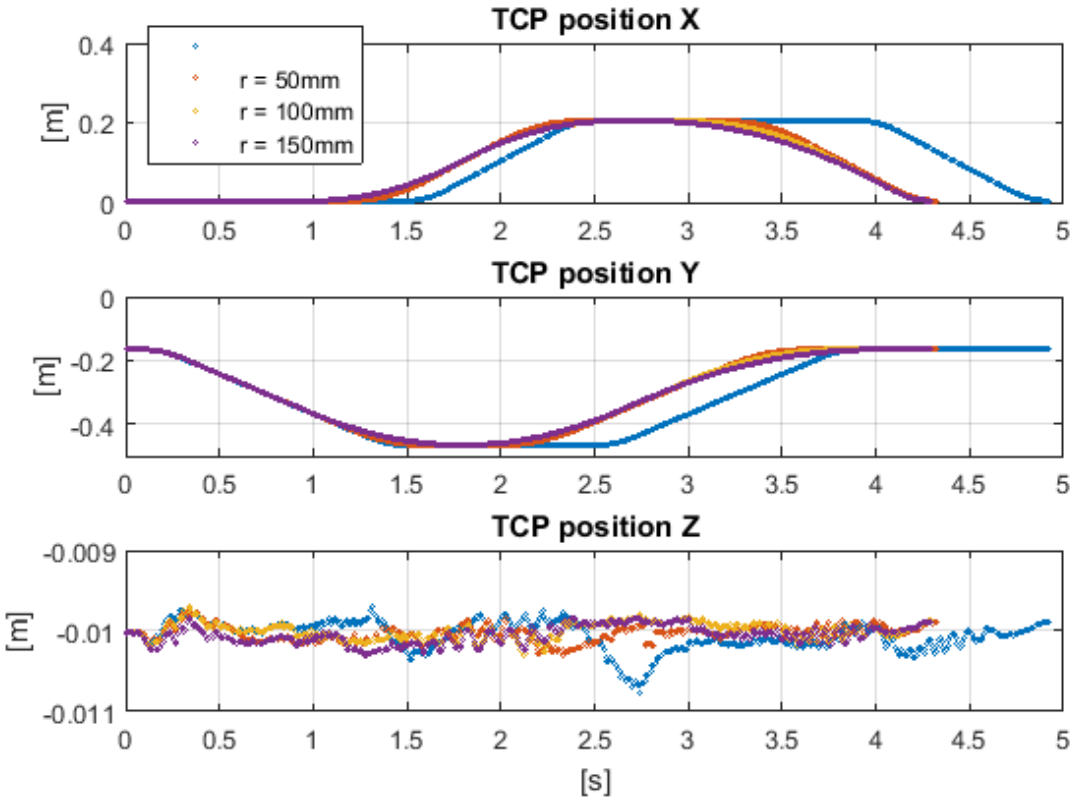


Figura 143: traiettoria del TCP nella direzione X, Y e Z

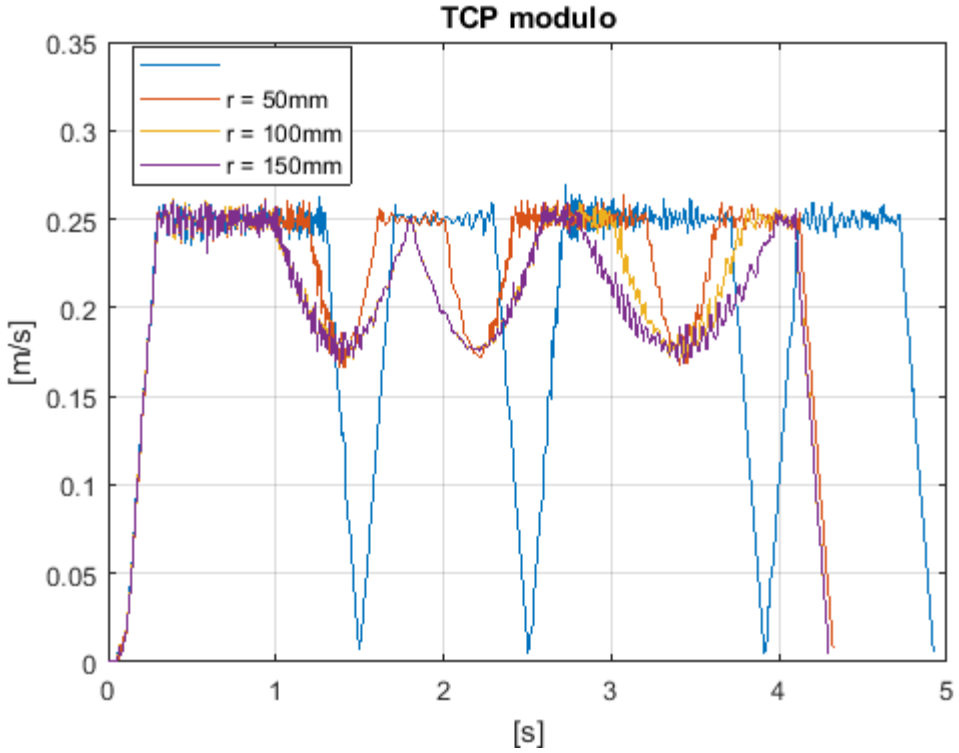


Figura 144: modulo della velocità del TCP

Da questo grafico si nota la particolarità del raccordo che permette di eseguire una serie di movimenti evitando che, in ogni transizione, la velocità del tool vada a zero.

2) Le prove utilizzate in questa seconda analisi sono:

Tabella 31: programmi utilizzati in questa seconda analisi

programma	acquisizione
PROVA_14_1_moveL_rettangolo_no_raccordi.urp	ACQUISIZIONE_14_1_moveL_rettangolo_no_raccordi.mat
PROVA_14_5_moveL_rettangolo_r50_vel_variabili_def_10_10_def.urp	ACQUISIZIONE_14_5_moveL_rettangolo_r50_vel_variabili_def_10_10_def.mat
PROVA_14_6_moveL_rettangolo_r100_vel_variabili_def_10_10_def.urp	ACQUISIZIONE_14_6_moveL_rettangolo_r100_vel_variabili_def_10_10_def.mat
PROVA_14_7_moveL_rettangolo_r150_vel_variabili_def_10_10_def.urp	ACQUISIZIONE_14_7_moveL_rettangolo_r150_vel_variabili_def_10_10_def.mat

In questo secondo caso sono state considerate delle variazioni di velocità tra un movimento e l'altro.

Tabella 32: variazioni di velocità tra i diversi waipoint

	velocità [mm/s]	**accelerazione [mm/s^2]**
waypoint1 a waypoint2	250	1200
waypoint2 a waypoint3	25	120
waypoint3 a waypoint4	25	120
waypoint4 a waypoint1	250	1200

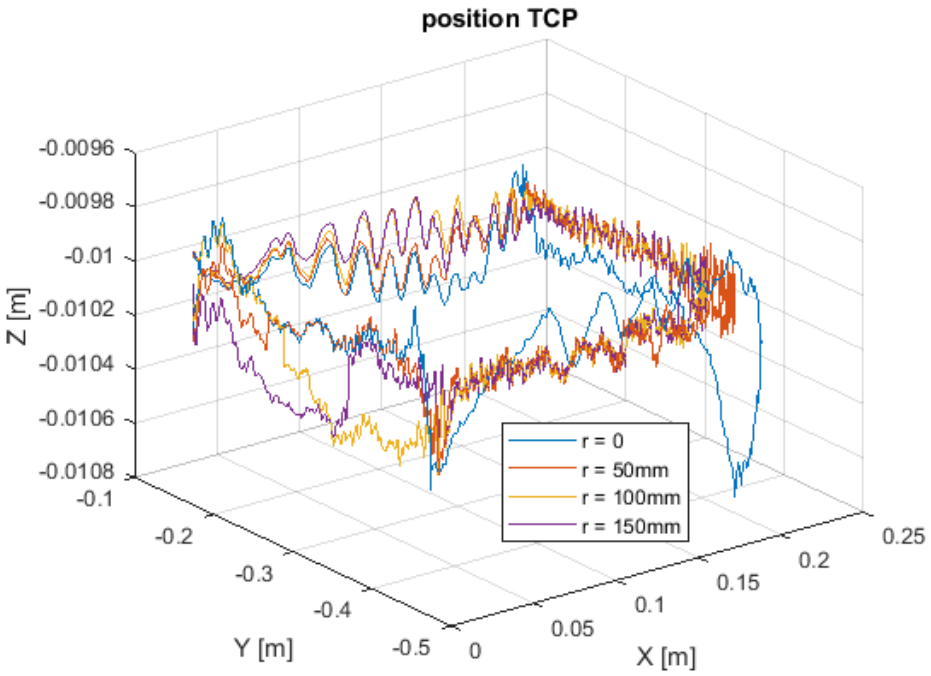


Figura 145: percorso del TCP nello spazio

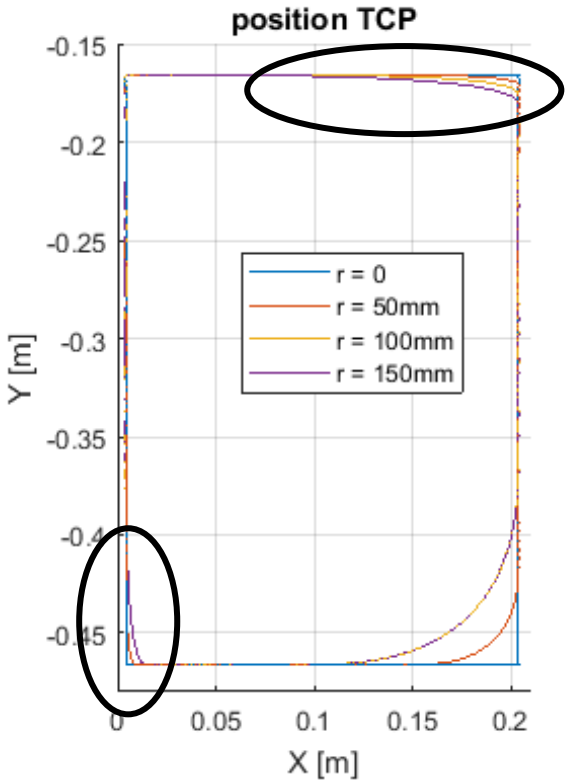


Figura 146: percorso del TCP nel piano X-Y

Da questa immagine si può osservare che in caso di MoveL non si rischia di andare fuori traiettoria nelle transizioni da movimenti veloci a movimenti lenti e viceversa. I raccordi però non vengono realizzati della grandezza desiderata, infatti vengono effettuati o con anticipo (nel secondo raccordo) o con ritardo (nel quarto raccordo).

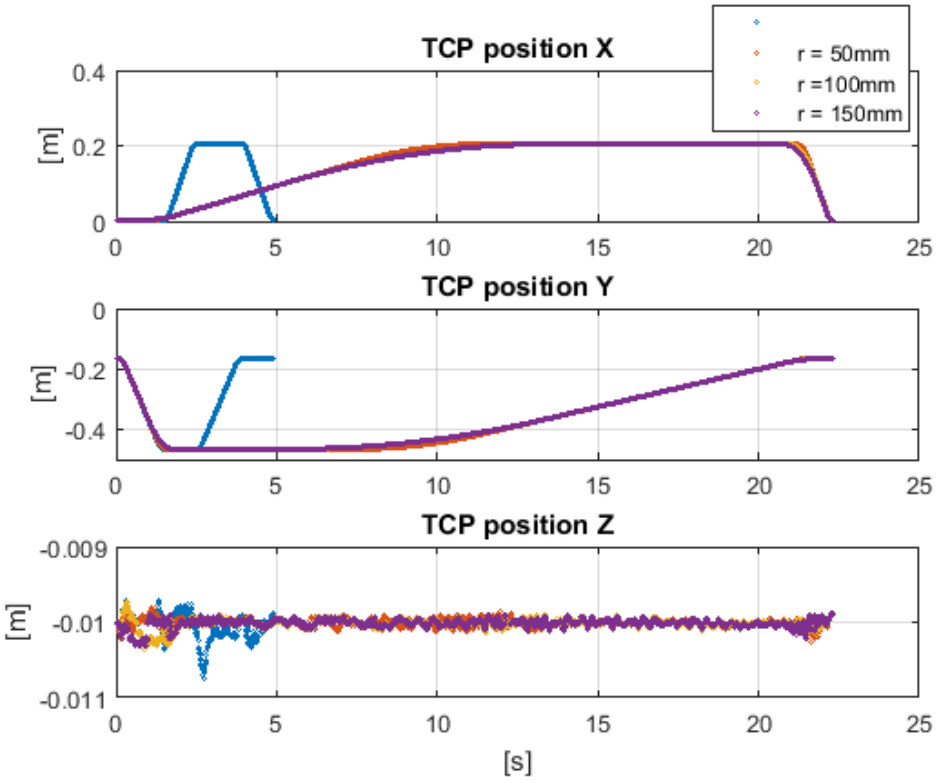


Figura 147: traiettoria TCP lungo X, Y e Z

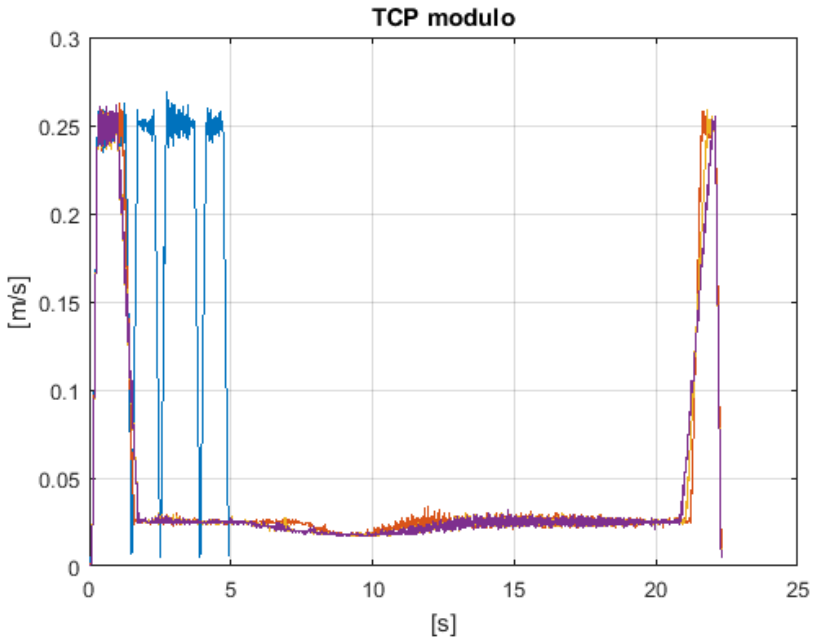


Figura 148: modulo della velocità del TCP

3) Realizzazione della traiettoria mediante MoveJ.

Vengono effettuate delle prove simili alle precedenti questa volta utilizzando dei movimenti di tipo J.

Tabella 33: programmi utilizzati per quest'ulteriore analisi

programma	acquisizione
PROVA_15_1_moveJ_rettangolo_no_raccordi.urp	ACQUISIZIONE_15_1_moveJ_rettangolo_no_raccordi.mat
PROVA_15_2_moveJ_rettangolo_r50.urp	ACQUISIZIONE_15_2_moveJ_rettangolo_r50.mat
PROVA_15_3_moveJ_rettangolo_r100.urp	ACQUISIZIONE_15_3_moveJ_rettangolo_r100.mat
PROVA_15_4_moveJ_rettangolo_r150.urp	ACQUISIZIONE_15_4_moveJ_rettangolo_r150.mat

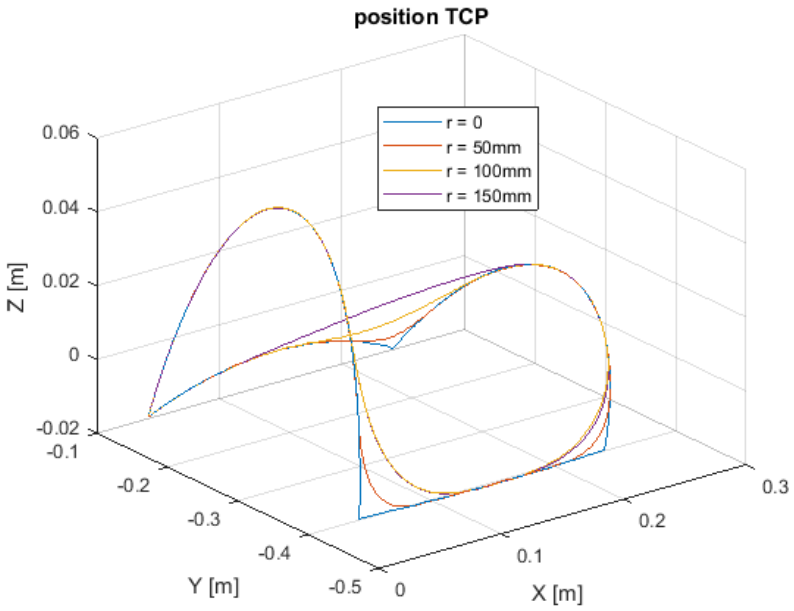


Figura 149: percorso TCP nello spazio

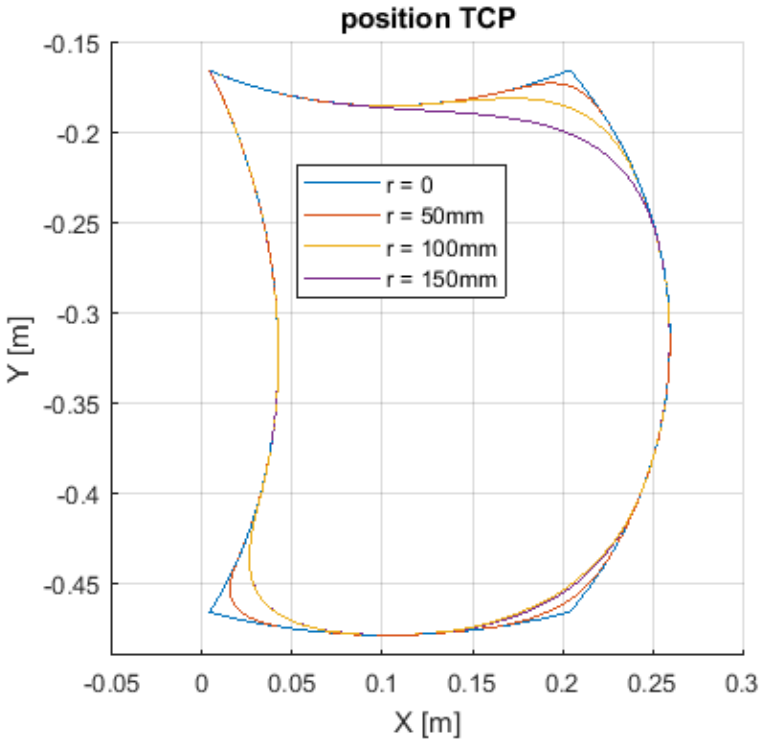


Figura 150: percorso del TCP nel piano X-Y

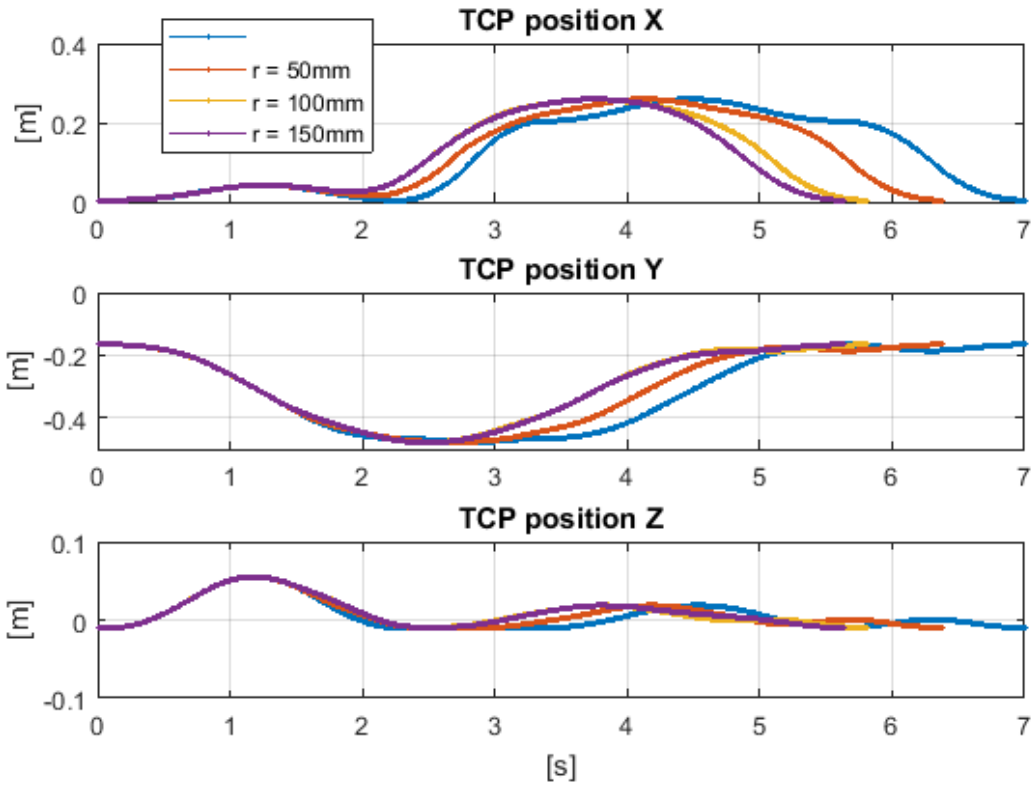


Figura 151: traiettoria del TCP lungo X, Y e Z

Si può osservare che non sempre i raccordi vengono effettuati. Nel caso in questione si osserva che durante la seconda e la terza transizione il raccordo con raggio maggiore ovvero di 150mm non viene effettuato per motivi geometrici: il software infatti comprende che lo spazio a disposizione tra un waypoint e l'altro non è sufficiente, cosa che non accade durante l'ultima transizione. Si osserva inoltre,

sia dal percorso sia dalla traiettoria, che più si tende a raccordare una traiettoria più il percorso sarà meno fedele all'originale e il tempo in cui viene effettuato il movimento si riduce.

4) Stessa prova del punto precedente variando le velocità tra un waypoint e l'altro.

In questo secondo caso sono state considerate delle variazioni di velocità tra un movimento e l'altro.

Tabella 34: prove utilizzate in questo caso

programma	acquisizione
PROVA_15_1_moveJ_rettangolo_no_raccordi.urp	ACQUISIZIONE_15_1_moveJ_rettangolo_no_raccordi.mat
PROVA_15_5_moveJ_rettangolo_r50_vel_variabili_def_10_10_def.urp	ACQUISIZIONE_15_5_moveJ_rettangolo_r50_vel_variabili_def_10_10_def.mat
PROVA_15_6_moveJ_rettangolo_r100_vel_variabili_def_10_10_def.urp	ACQUISIZIONE_15_6_moveJ_rettangolo_r100_vel_variabili_def_10_10_def.mat
PROVA_15_7_moveJ_rettangolo_r150_vel_variabili_def_10_10_def.urp	ACQUISIZIONE_15_7_moveJ_rettangolo_r150_vel_variabili_def_10_10_def.mat

Tabella 35: variazioni di velocità tra un waypoint e l'altro

	velocità [deg/s]	accelerazione [deg/s^2]
waypoint1 a waypoint2	60	80
waypoint2 a waypoint3	6	8
waypoint3 a waypoint4	6	8
waypoint4 a waypoint1	60	80

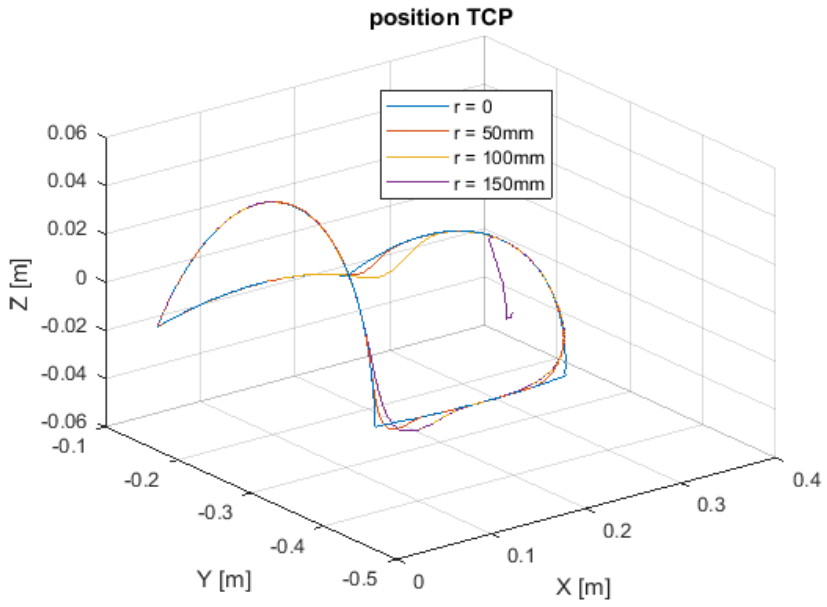


Figura 152: percorso del TCP nello spazio

Negli ingrandimenti successivi è possibile osservare sul piano x-y il percorso compiuto e quali particolarità si notano nelle zone di transizione.

Dalle figure 154 e 155 si osserva che nel momento in cui c'è un gran salto di velocità tra un waypoint e l'altro si potrebbero creare dei piccoli errori di percorso.

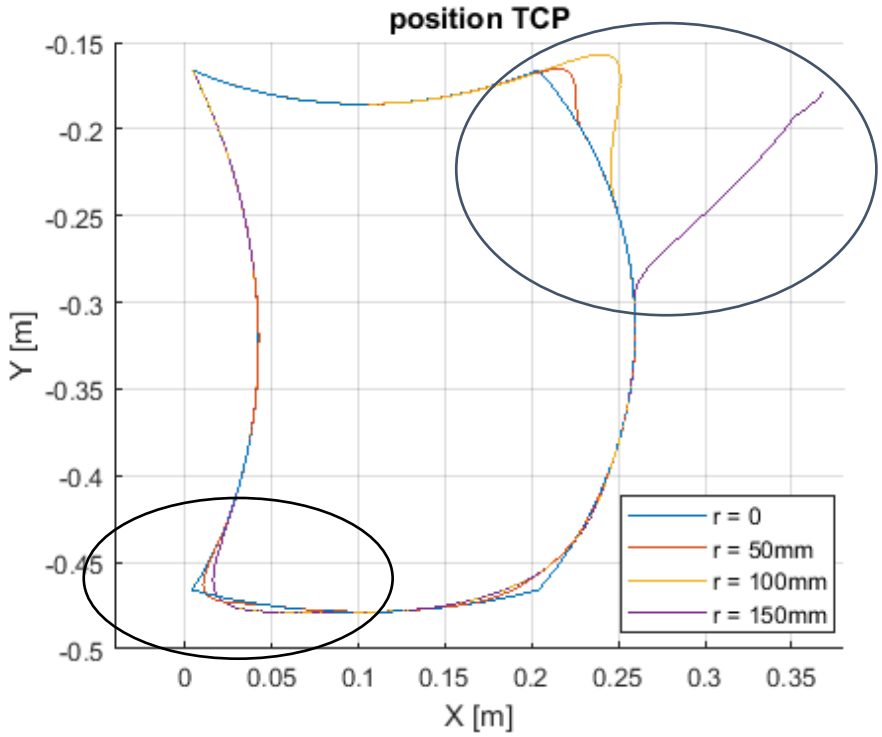


Figura 153: percorso TCP sul piano X-Y

Nella figura 154 si sta passando dalla velocità di default a quella ridotta al 10% e si manifesta un errore di percorso dopo il waypoint.

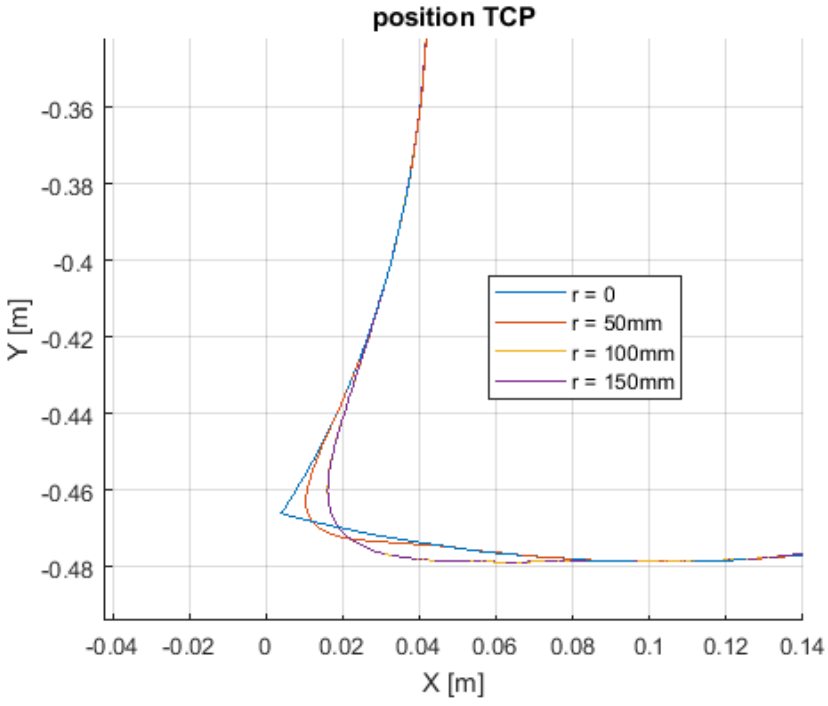


Figura 154: ingrandimento primo raccordo

Nella figura 155 si sta passando da una velocità ridotta a quella di default e l'errore si manifesta prima del waypoint. Può anche capitare che per raggi troppo elevati il robot non riesca a ritornare sul giusto percorso programmato e la posizione diverge indesideratamente.

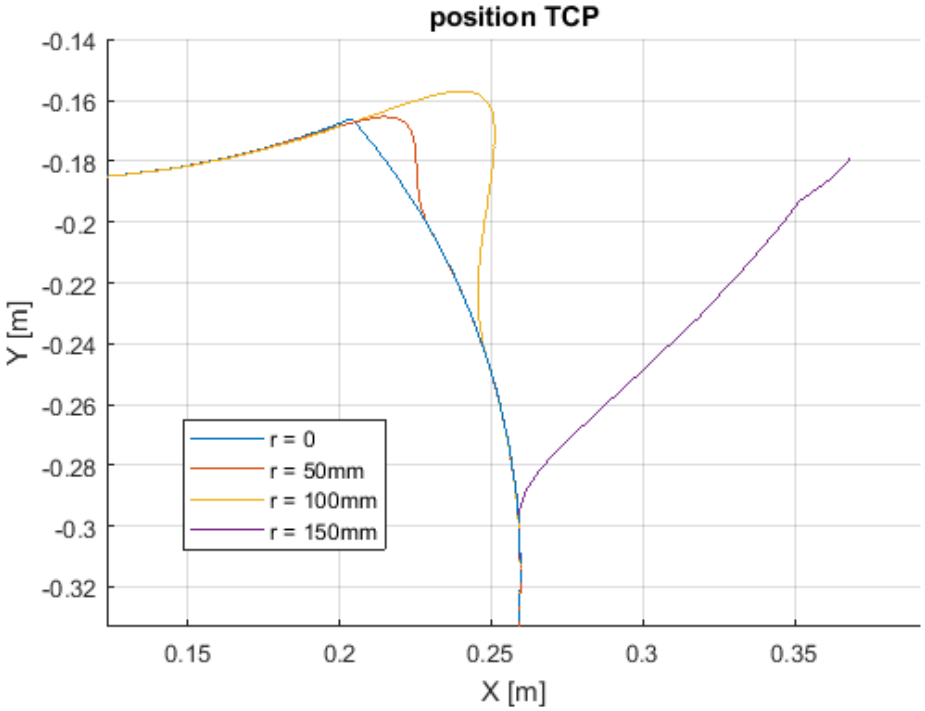


Figura 155: ingrandimento ultimo raccordo

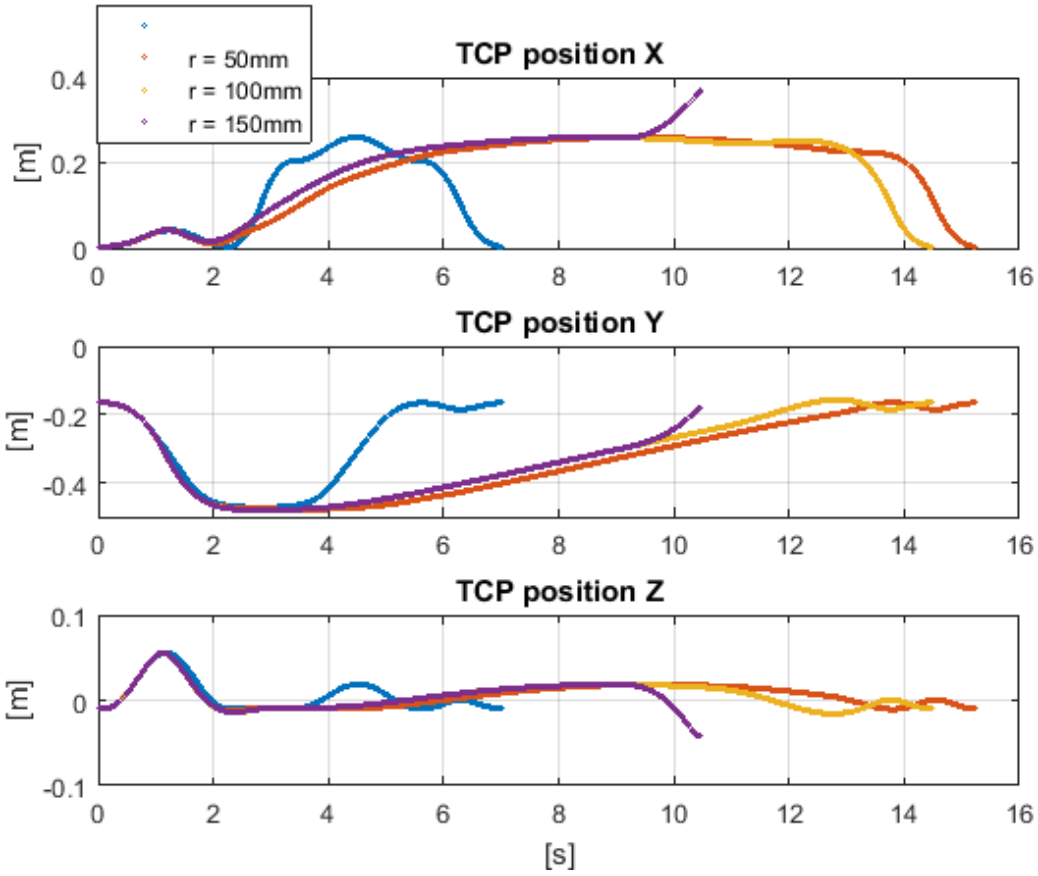


Figura 156: traiettoria TCP lungo X, Y e Z

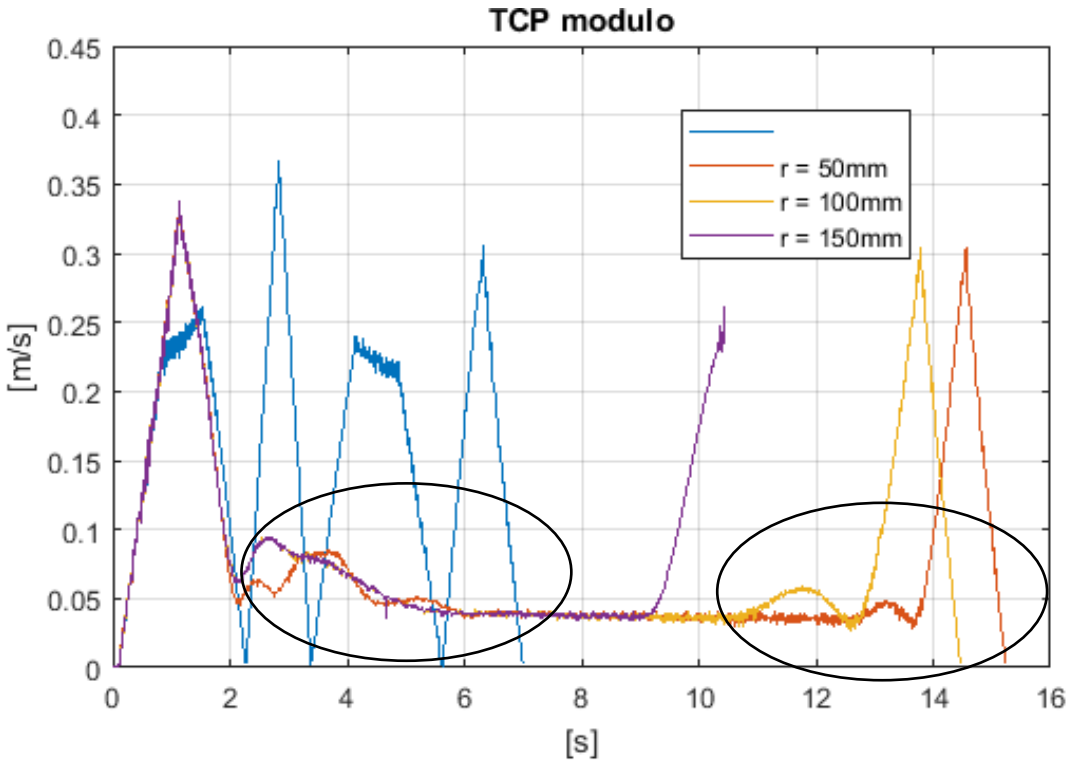


Figura 157: modulo velocità TCP

Dal grafico del modulo della velocità si individuano le zone in cui avviene la transizione tra i movimenti veloci e quelli lenti e viceversa. Si osserva come il movimento che prevede il raccordo massimo ad un tratto si interrompe poiché il robot va fuori traiettoria e si scontra contro la parete virtuale "base" bloccandosi.

5) ANALISI DELLA SCRITTA CIAO

Come ultima analisi per quanto riguarda i raccordi, vengono fatte delle osservazioni sulla traiettoria ottenuta durante i movimenti per far scrivere al robot la parola "CIAO" (non era disponibile la pinza).

Figura 158: scritta CIAO

- ***Programma scritto:*** PROVA_16_1_CIAO_default.urp

  Acquisizione corrispondente: ACQUISIZIONE_16_1_CIAO_default.mat

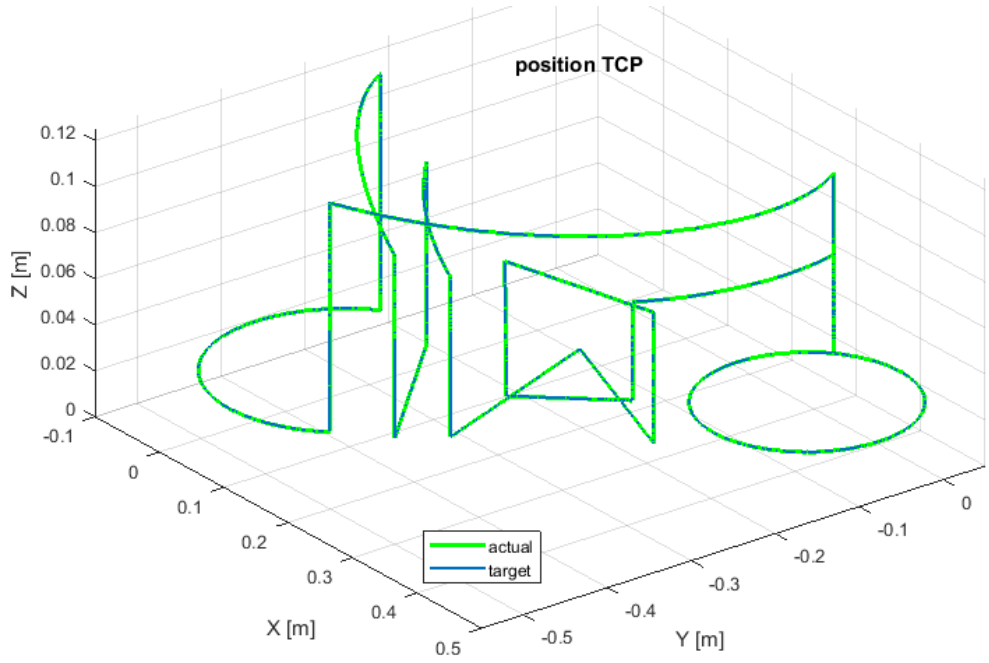


Figura 159: percorso TCP

Con questa prima pianificazione si osserva che da un movimento all'altro il passaggio avviene senza alcun raccordo. Tale dettaglio si può notare anche dal grafico successivo, quello delle velocità, in particolare dal modulo della velocità lineare del TCP che ad ogni transizione tende ad annullarsi.

```
Program
 Robot Program
  MoveL
   Waypoint_1
  C
   MoveL
    discesa_c
   MoveP
    CircleMove
     ViaPoint_1
     EndPoint_1
   MoveL
    salita_c
  I
   MoveJ
    rapido_i
   MoveL
    discesa_i
    Waypoint_6
   MoveL
    salita_i
  A
    MoveJ
     rapido_a
    MoveL
     discesa_a
     Waypoint_10
     Waypoint_11
    MoveL
     salita_a
    MoveL
     Waypoint_13
    MoveL
     discesa_trattoA
     Waypoint_15
     salita_trattoA
  O
   MoveJ
    rapido_o
   MoveL
    discesa_o
   MoveP
    CircleMove
     ViaPoint_2
     EndPoint_2
   MoveP
    CircleMove
     ViaPoint_3
     EndPoint_3
   MoveL
    salita_o
  MoveJ
   rapido_c
```

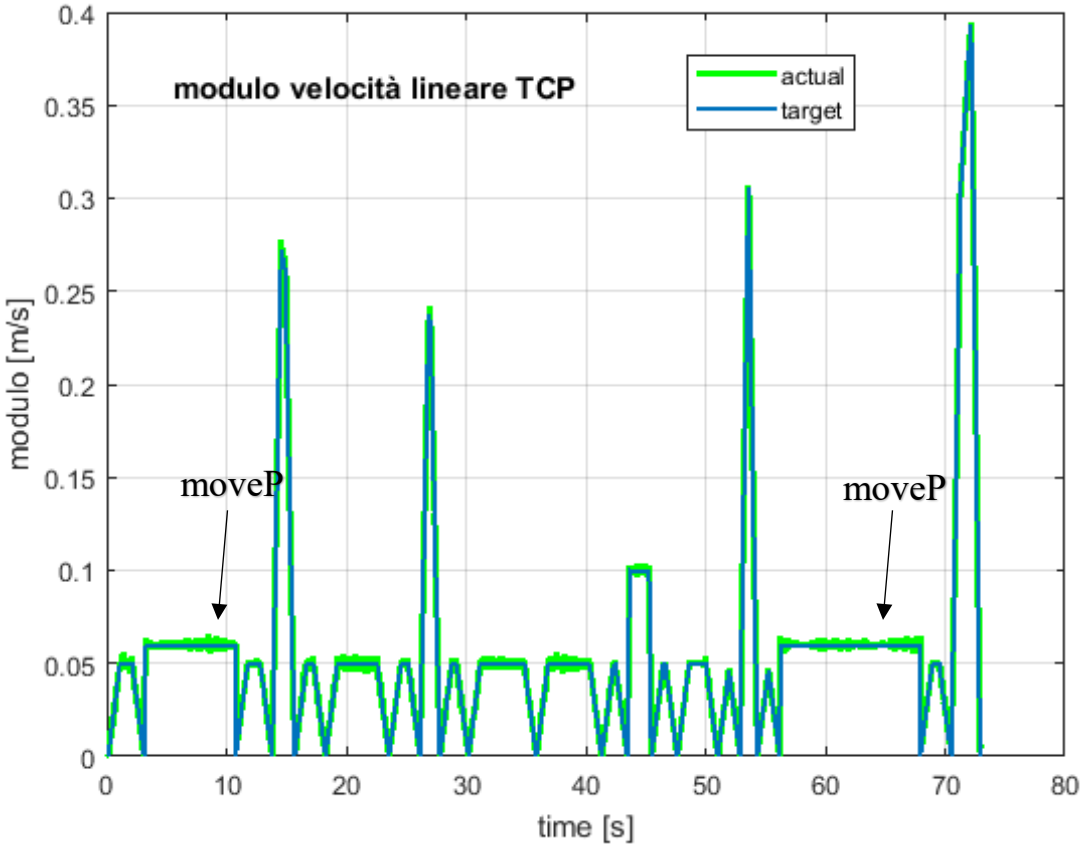


Figura 160: modulo velocità TCP

Si osserva inoltre che per i due MoveP la velocità del TCP durante tutto il movimento viene mantenuta costante (proprio come prevede il comando), mentre durante i MoveL vengono realizzati degli spostamenti che prevedono velocità con andamento trapezoidale (tutti gli altri trapezi al di fuori dei MoveP sono i movimenti di MoveL), differentemente dai MoveJ dove tale andamento è pianificato nello spazio giunti. Per ogni MoveJ inoltre si raggiungono dei picchi maggiori poiché tale movimento è stato utilizzato per poter compiere degli spostamenti in rapido.

- ***Programma scritto:*** PROVA _16_2_CIAO_raccordo_soloP.urp
  Acquisizione corrispondente: ACQUISIZIONE_16_2_CIAO_raccordo_soloP.mat

Si decide di cambiare leggermente la scrittura del programma e di raccordare i due MoveP con il movimento lineare successivo. Questa operazione garantisce una transizione meno brusca e più dolce. È stato necessario il cambio della scrittura del programma perché non è possibile raccordare il MoveP con movimenti successivi differenti, ma è possibile raccordare tale movimento solamente con un altro successivo MoveP. L'entità del raggio di raccordo è pari a 25mm.

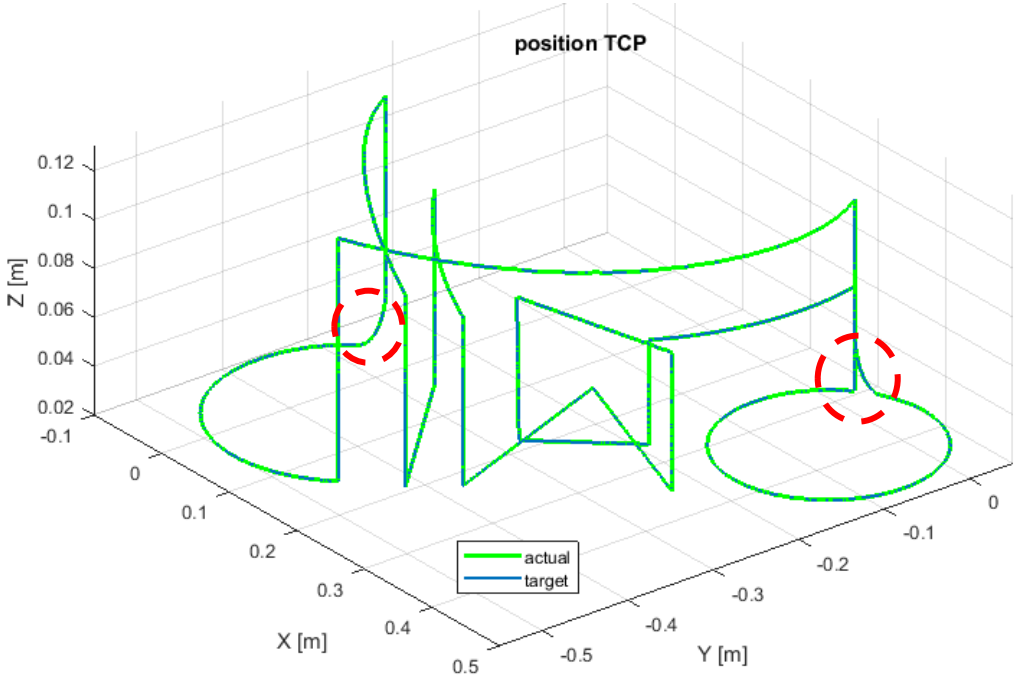


Figura 161: percorso TCP

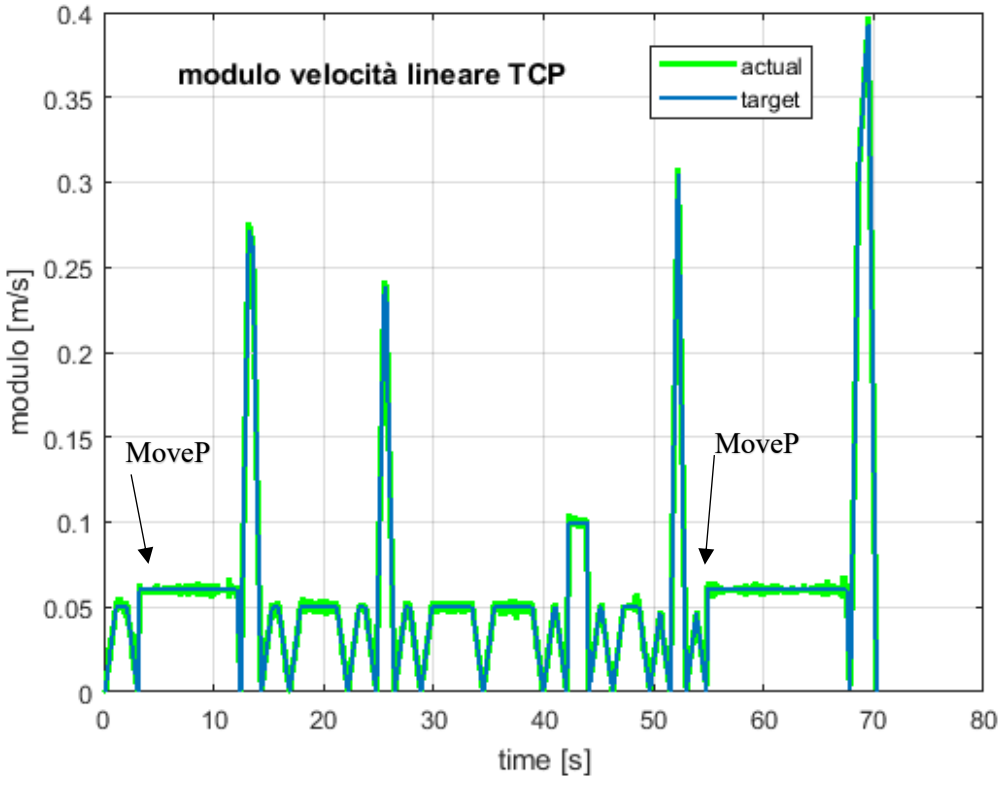


Figura 162 modulo velocità TCP

```
Program
 Robot Program
  MoveL
   Waypoint_1
  C
   MoveL
    discesa_c
   MoveP
    CircleMove
     ViaPoint_1
     EndPoint_1
    salita_c
  I
   MoveJ
    rapido_i
   MoveL
    discesa_i
    Waypoint_6
   MoveL
    salita_i
  A
    MoveJ
     rapido_a
    MoveL
     discesa_a
     Waypoint_10
     Waypoint_11
    MoveL
     salita_a
    MoveL
     Waypoint_13
    MoveL
     discesa_trattoA
     Waypoint_15
     salita_trattoA
  O
   MoveJ
    rapido_o
   MoveL
    discesa_o
   MoveP
    CircleMove
     ViaPoint_2
     EndPoint_2
   MoveP
    CircleMove
     ViaPoint_3
     EndPoint_3
    salita_o
 MoveJ
  rapido_c
```

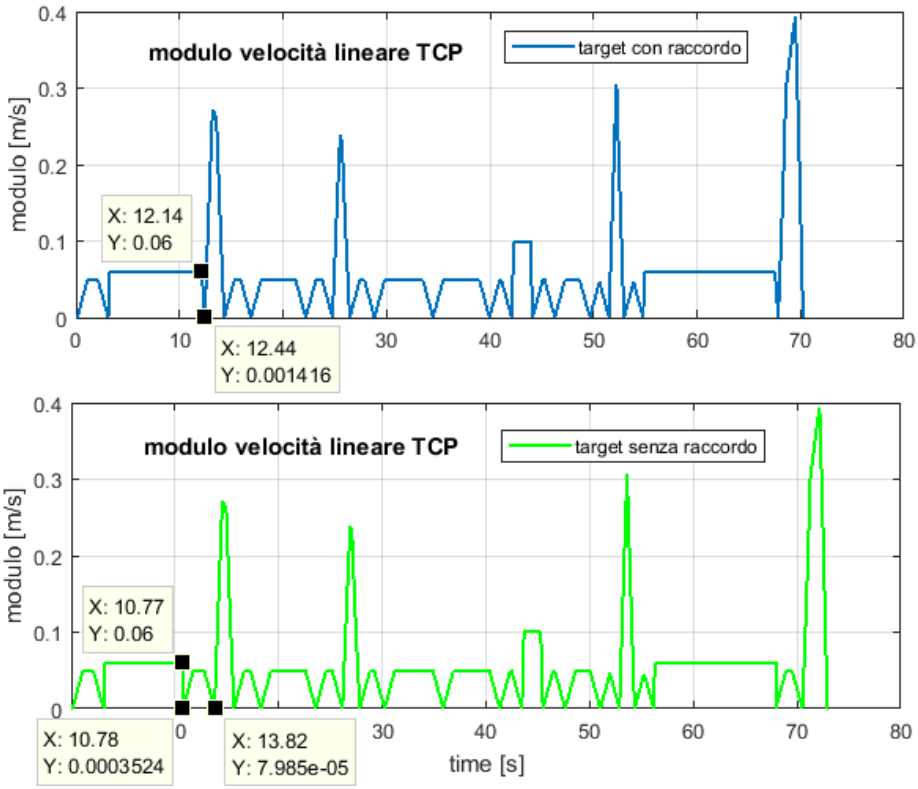


Figura 163: modulo velocità TCP confronto

In questa immagine è possibile osservare alcuni dettagli importanti. Si intuisce, guardando il grafico nel suo complesso, che la durata dell'intera traiettoria è minore nel caso in cui si usi il raccordo rispetto alla traiettoria senza.

Nell'immagine con il raccordo si osserva che dopo l'intero movimento di tipo P (quindi sia quello circolare sia il lineare per la risalita), il breve arresto viene raggiunto in maniera meno brusca. Nel secondo caso, il passaggio dal movimento lineare al MoveJ successivo continua ad essere poco brusco, mentre non lo è il passaggio dal movimento circolare di tipo P allo spostamento lineare di salita. Quest'ultima costatazione può essere fatta osservando sia il tempo di arresto, che è (con opportuni arrotondamenti) pari a un centesimo di secondo nel caso senza raccordi e in fase di salita, e sia dal grafico delle coppie. Dalla figura 164 infatti si può dedurre che nel momento del passaggio da MoveP a MoveL vi è un istantaneo aumento di coppia e quindi sollecitazione ai motori; cosa che non succede affatto nel secondo caso in cui sia la lettera C sia la salita successiva vengono effettuate con un MoveP e quindi è stato possibile raccordare le due traiettorie.

Le stesse considerazioni possono essere fatte per il raccordo successivo che avviene tra i 55 e i 70 secondi.

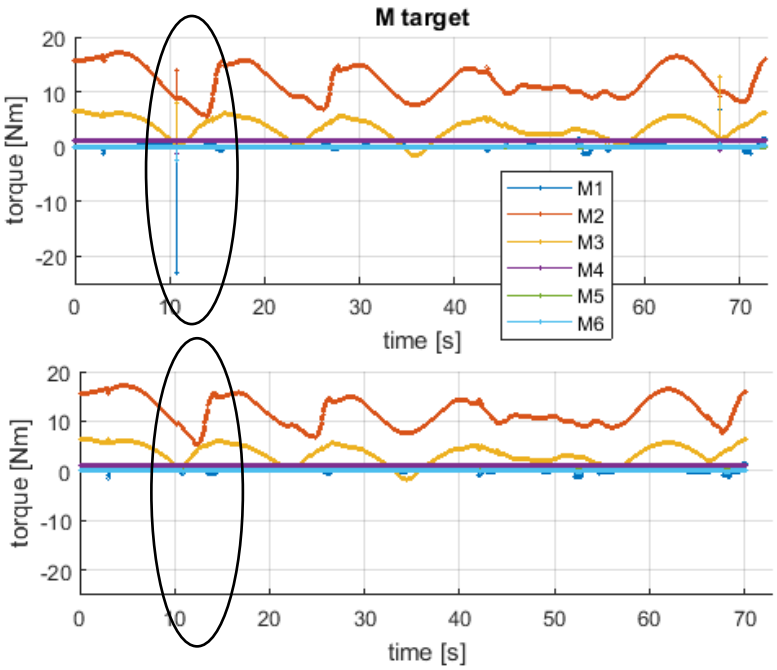


Figura 164: coppie nel caso senza e con raccordo dei moveP

- **Analisi effettuata al variare del raggio di raccordo**

Tabella 36: programmi utilizzati per quest'analisi

Programmi	Acquisizione
PROVA_16_3_CIAO_tutto_raccordato_5.urp	ACQUISIZIONE_16_3_CIAO_tutto_raccordato_5.mat
PROVA _16_4_CIAO_tutto_raccordato_25.urp	ACQUISIZIONE_16_4_CIAO_tutto_raccordato_25.mat
PROVA _16_5_CIAO_tutto_raccordato_50.urp	ACQUISIZIONE_16_5_CIAO_tutto_raccordato_50.mat

In quest'ultimo caso si decide di raccordare la traiettoria ovunque sia possibile con dei raccordi a raggio variabile. Si osserva che in alcuni casi i raccordi con una certa grandezza non vengono realizzati dal robot poiché il software UR calcola a seconda delle posizioni dei waypoint, dei diversi Move e della geometria, quali siano i possibili raggi di raccordo accettabili.

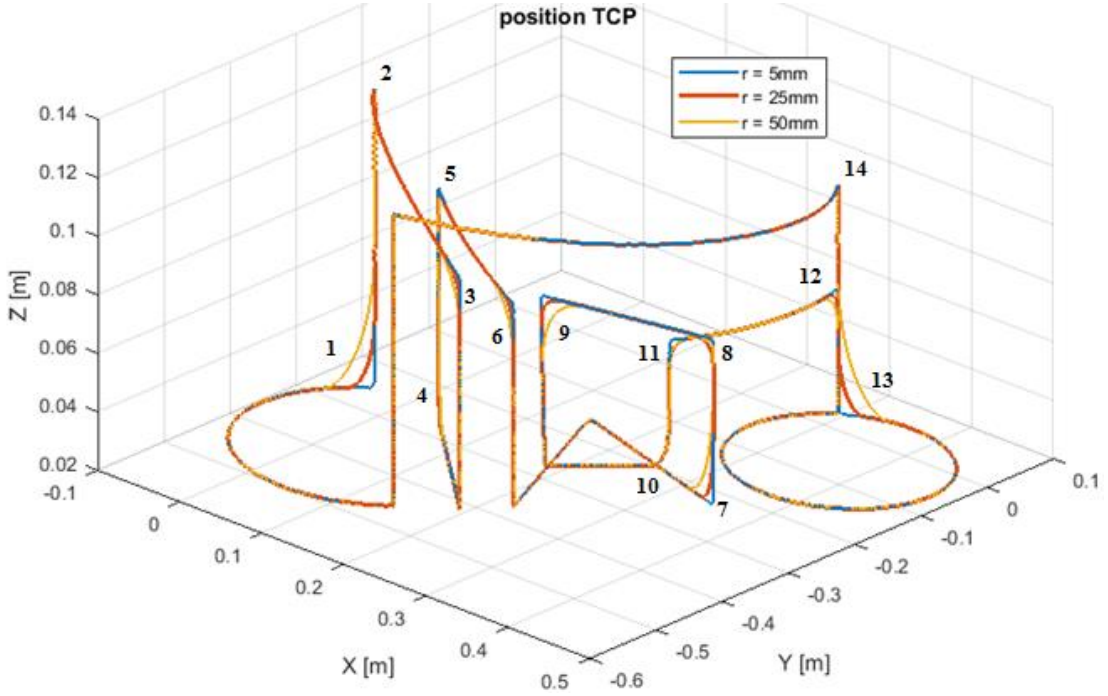


Figura 165: percorso TCP raccordi

Soffermandosi su alcuni di questi punti illustrati in figura si osserva, come anticipato, che in alcuni casi non è stato possibile realizzare il raccordo voluto. Nelle seguenti immagini la forma dei raccordi non è perfettamente circolare, ma risultano essere leggermente deformati, poiché la traiettoria non avviene completamente su un unico piano.

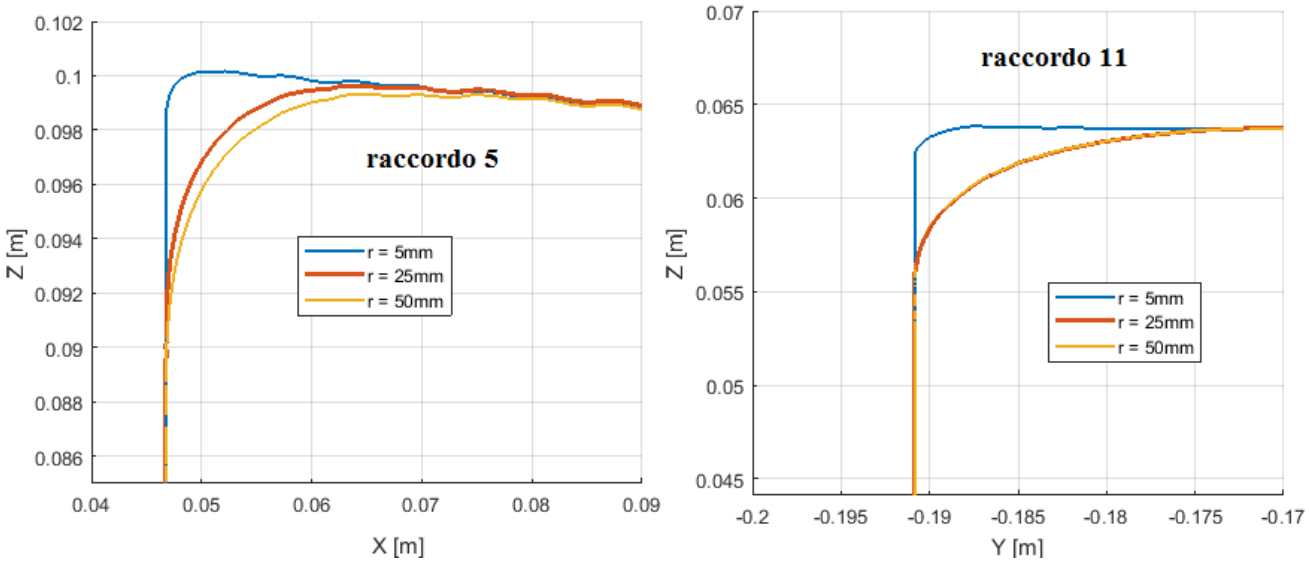


Figura 166: raccordo 5 e 11

In questa immagine sono rappresentati i raccordi 5 e 11 dove per motivi legati alla geometria non è stato possibile realizzare il raccordo di raggio massimo, ovvero 50mm. Nel raccordo 5 il raggio massimo che può essere applicato è leggermente superiore al valore di quello intermedio, ovvero 25mm; nel secondo caso i due tendono proprio a coincidere.

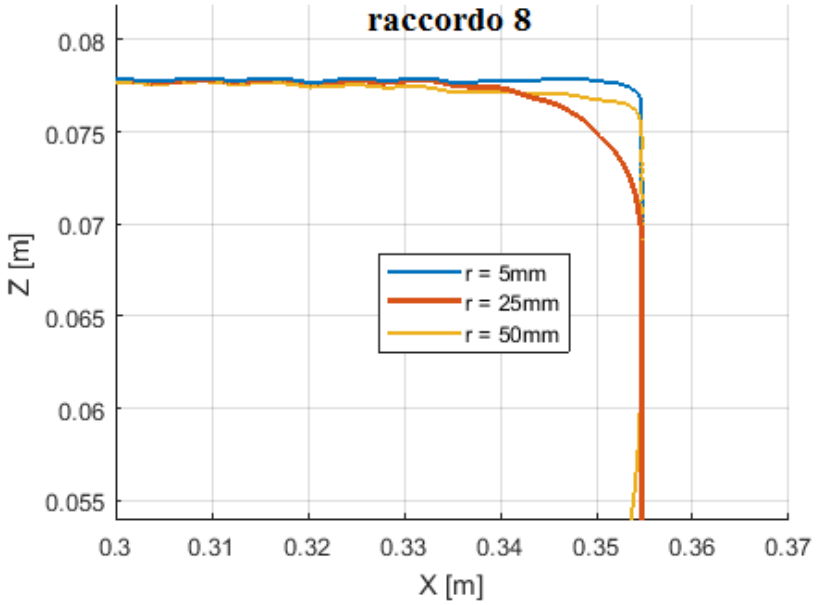


Figura 167: raccordo 8

Nel raccordo 8 si nota un fenomeno particolare: il raggio di dimensioni non solo non viene eseguito correttamente ma tende ad essere anche più piccolo di quello intermedio. Questo fenomeno è dovuto probabilmente all'eccessiva velocità di esecuzione tra il movimento precedente e quello successivo che può comportare un'alterazione della traiettoria.

Altre osservazioni che possono essere fatte:

- nel punto 2 e 14 non vi sono raccordi poiché si ha il passaggio da un MoveP ad un altro tipo di Move;
- nelle fasi di discesa e posizionamento pennarello è stato deciso di non utilizzare nessun raccordo per una maggiore precisione di posizione dell'utensile grafico;
- il raccordo 10 è stato mantenuto uguale per tutte le prove per un valore pari a 17mm.

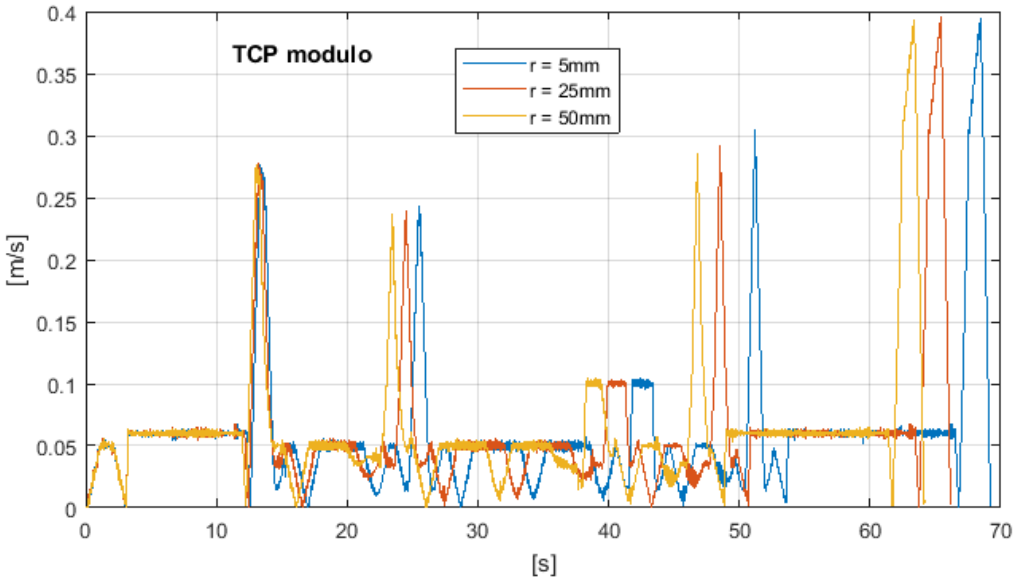


Figura 168: confronto tra i moduli delle diverse prove con raccordi

Si può osservare da questi grafici di velocità che l'intera traiettoria è influenzata dal raggio dei raccordi: maggiore sarà il raggio, meno durerà il movimento. Ovviamente con raccordi maggiori è più probabile imbattersi in errori maggiori legati alla traiettoria, ci si allontana infatti sempre di più da quella desiderata.

Dall'immagine seguente si nota come in una traiettoria più raccordata rispetto a una priva di raccordi, i passaggi per velocità nulle si riducono, tra una posizione e l'altra il robot non si ferma.

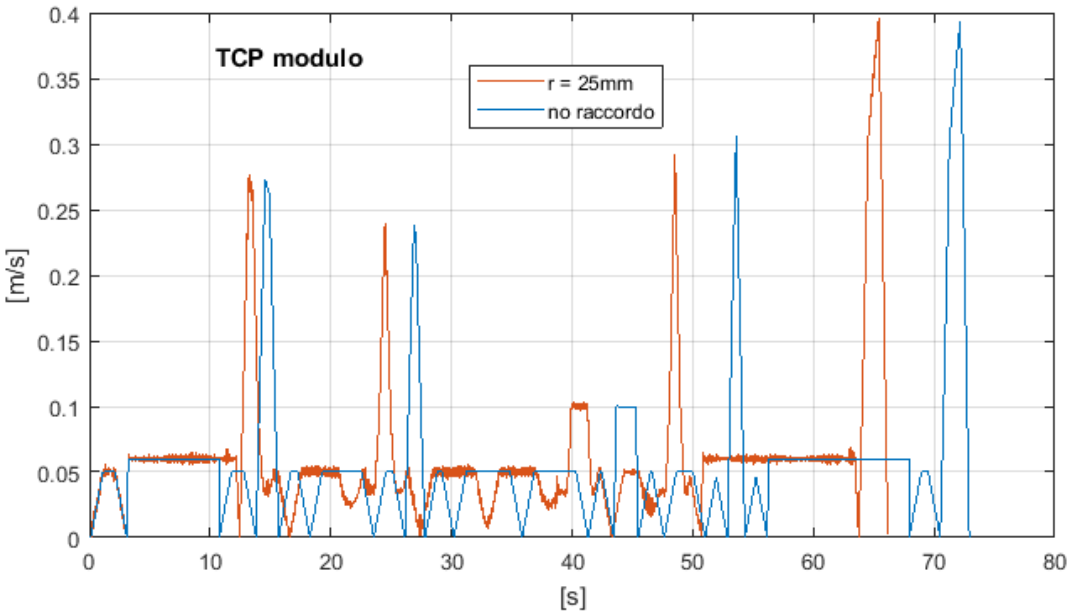


Figura 169: confronto modulo velocità tra percorso raccordato e non

5.2 Programmi con l'utilizzo della pinza e della cella di carico

5.2.1. Miscelazione in pentola e studio dell'urto

Un primo programma scritto riguarda un'applicazione collaborativa tra ente e robot in cui la persona passa al robot un cucchiaio di legno e il robot, una volta avuta la conferma di presa, incomincia a miscelare un liquido in una pentola.

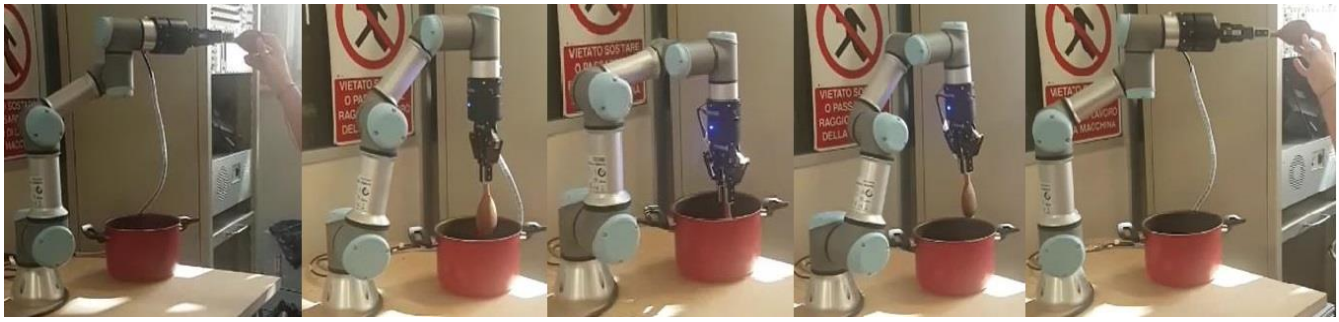


Figura 170: wire-frame miscelazione

Il movimento di miscelazione è stato realizzato in due varianti: una in cui il tool è fisso e l'altra in cui viene dato al tool un certo movimento di rollio. Ovviamente il rollio dato è stato ottenuto tramite una rotazione del giunto sei alternata, per evitare di avvolgere eccessivamente il cavo di collegamento del gripper e della cella di carico all'unità di controllo.
I due programmi, con relative acquisizioni, sono sotto riportati.

Tabella 37: programmi rollio e non rollio

Programma	Acquisizione
PROVA _18_1_miscelazione_no_rollio.urp	ACQUISIZIONE_18_1_miscelazione_no_rollio.mat
PROVA _18_2_miscelazione_rollio.urp	ACQUISIZIONE_18_2_miscelazione_rollio.mat

Questa differenza può essere osservata dal grafico successivo dove vengono riportati gli andamenti dei giunti.

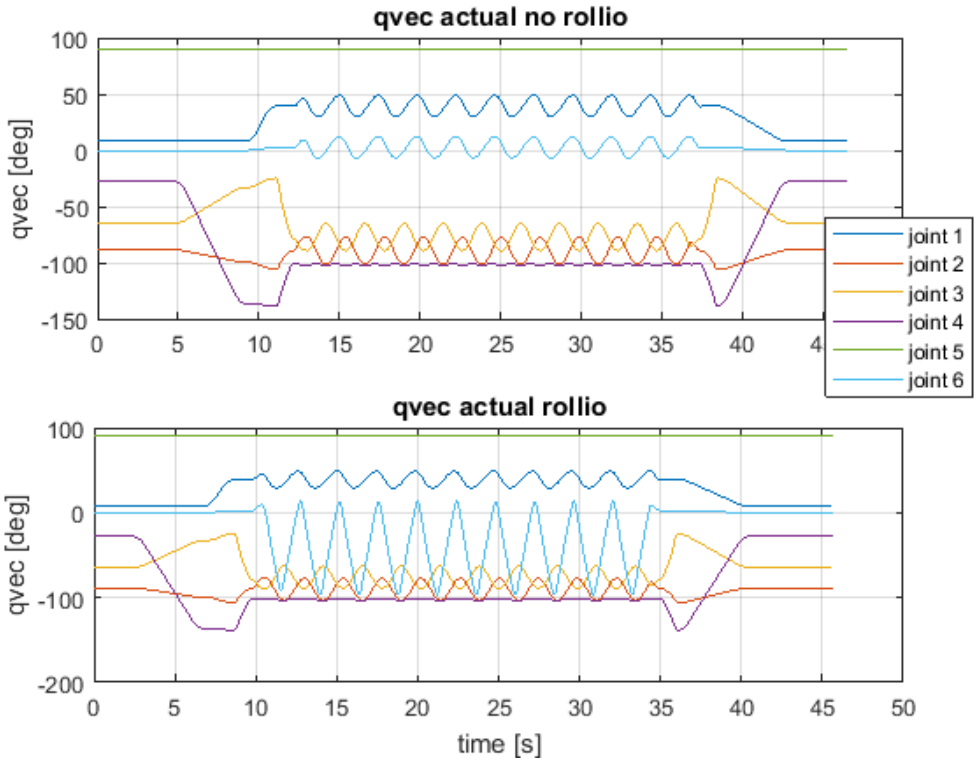


Figura 171: giunti del movimento senza e con rollio

Si osserva come, mentre nel caso senza rollio il giunto sei oscilla leggermente intorno alla stessa posizione, nel secondo caso è possibile osservare l'oscillazione di 90°. Dare un rollio non varia il percorso che il tool effettua nello spazio operativo. Quest'ultimo viene riportato nel seguente grafico con i rispettivi movimenti che il robot compie. Si osserva che prima del programma vi è una sezione beforestart in cui vengono inseriti i comandi per introdurre i differenti input e le varie pose parametrizzate in funzione del raggio e della posizione del centro della circonferenza.

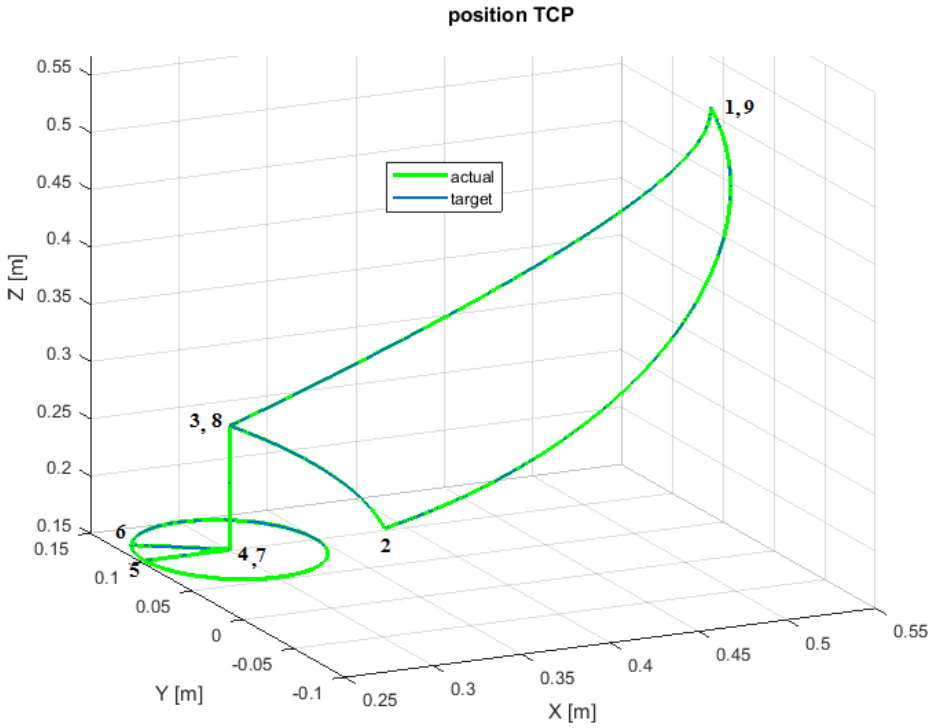


Figura 172: percorso TCP

```
Program
  BeforeStart
    raggio='inserisci il raggio della pentola in metri'
    X='inserisci coordinata x del centro della pentola in metri'
    Y='inserisci coordinata y del centro della pentola in metri'
    Z='inserisci altezza pentola in metri'
    vel_mescolare='inserire velocita con cui si vuole mescolare in m/s'
    acc_mescolare='inserire accelerazione con cui si vuole mescolare in m/s^2'
    posa_centro_su=p[X,Y,Z+0.108,2.8228,-1.4111,0.0113]
    posa_centro_giu=p[X,Y,Z,2.8228,-1.4111,0.0113]
    posa_inizio=p[X-raggio+0.002,Y,Z,2.8228,-1.4111,0.0113]
    posa_primoq=p[X,Y-raggio+0.002,Z,3.149,-0.226,0.002]
    posa_mezzo=p[X+raggio-0.002,Y,Z,3,0.981,-0.008]
    posa_ultimoq=p[X,Y+raggio-0.002,Z,3.149,-0.226,0.002]
  Robot Program
    MoveJ
      Waypoint_1
    Popup
    Gripper Close (1)
1➡2 MoveJ
      Waypoint_2
2➡3 movej(posa_centro_su,0.6,0.523,0,0)
3➡4 movel(posa_centro_giu,0.6,0.125,0,0)
4➡5 movel(posa_inizio,0.6,0.125,0,0)
    Loop 10 times
5➡6   movec(posa_primoq,posa_mezzo,acc_mescolare,vel_mescolare,0.025,0)
      movec(posa_ultimoq,posa_inizio,acc_mescolare,vel_mescolare,0.025,0)
6➡7 movel(posa_centro_giu,0.6,0.125,0,0)
7➡8 movel(posa_centro_su,0.6,0.125,0,0)
8➡9 MoveJ
      Waypoint_1
    Popup
    Gripper Open (1)
```

Figura 173: scrittura programma

Il programma illustrato nell'immagine 173 viene scritto interamente in maniera parametrica. I dati in input che possono essere inseriti sono essenzialmente il raggio e la posizione del centro della pentola, l'altezza della pentola e la velocità e l'accelerazione con cui si desidera miscelare.

Si è deciso di effettuare alcune prove per poter osservare il comportamento del robot in caso di urto con un operatore o un eventuale ente esterno.

Sono state effettuate tre prove nella quale il robot si muove durante l'azione di miscelazione con parametri differenti, per osservare quale sia la sua reazione nel momento dell'urto. I parametri con cui sono state svolte le prove vengono riportate nella seguente tabella:

Tabella 38: prove di urto effettuate

PROVA	**velocità tool miscelazione [m/s]**	**accelerazione tool miscelazione [m/s^2]**
PROVA _18_3_miscelazione_inseritomano_default.urp	0.125	0.6
PROVA_18_4_miscelazione_inseritomano_default_mezzi.urp	0.0625	0.3
PROVA_18_5_miscelazione_inseritomano_default_quarti.urp	0.03125	0.15

Per poter effettuare tali prove sono stati impostati i parametri di sicurezza più restrittivi [4].

FORCE	**POWER**	**SPEED**	**MOMENTUM**
100[N]	80[W]	250[mm/s]	5[Kg m/s]

Sono stati verificati tali parametri:

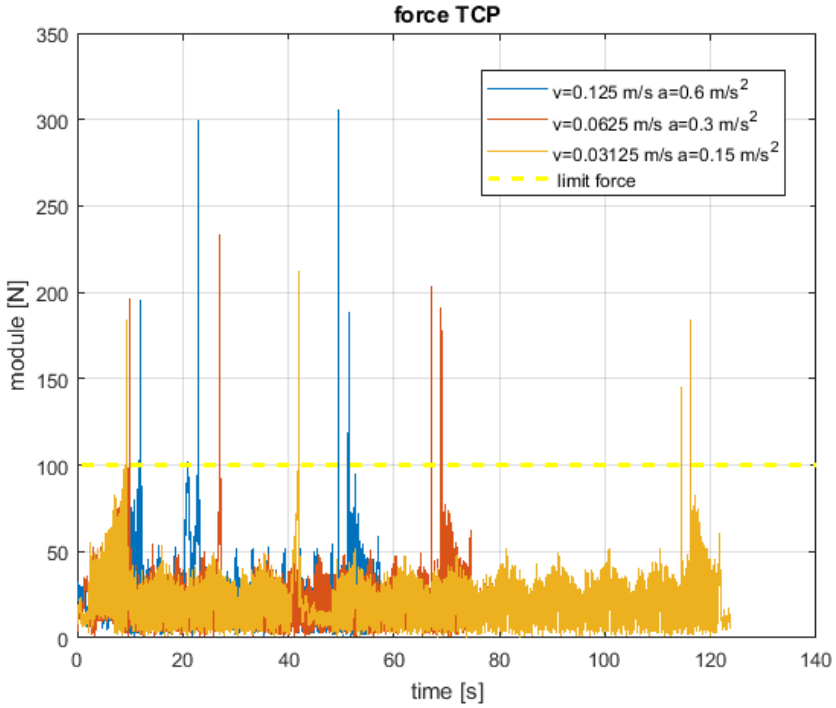


Figura 174: valori di forza al TCP

In questo grafico si nota che il modulo delle componenti di forza applicate al TCP rispettano il vincolo assegnato eccetto in alcuni punti specifici in cui si verificano dei picchi: questi si manifestano in tutte e tre le prove sia in corrispondenza dell'urto con l'operatore e sia quando si passa dal movimento circolare di tipo P a quello lineare di tipo L.

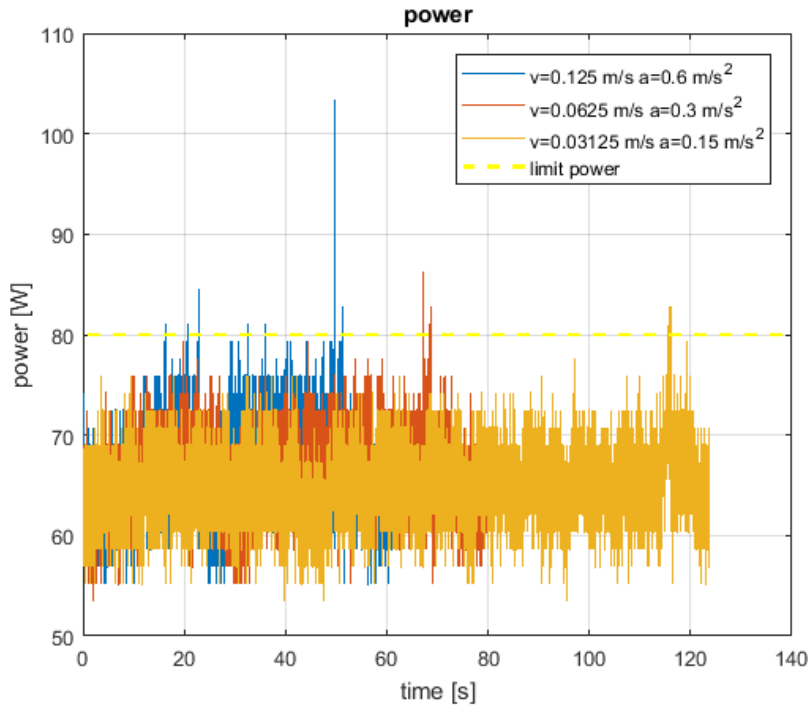


*Figura 175: andamento della forza calcolata come V*I*

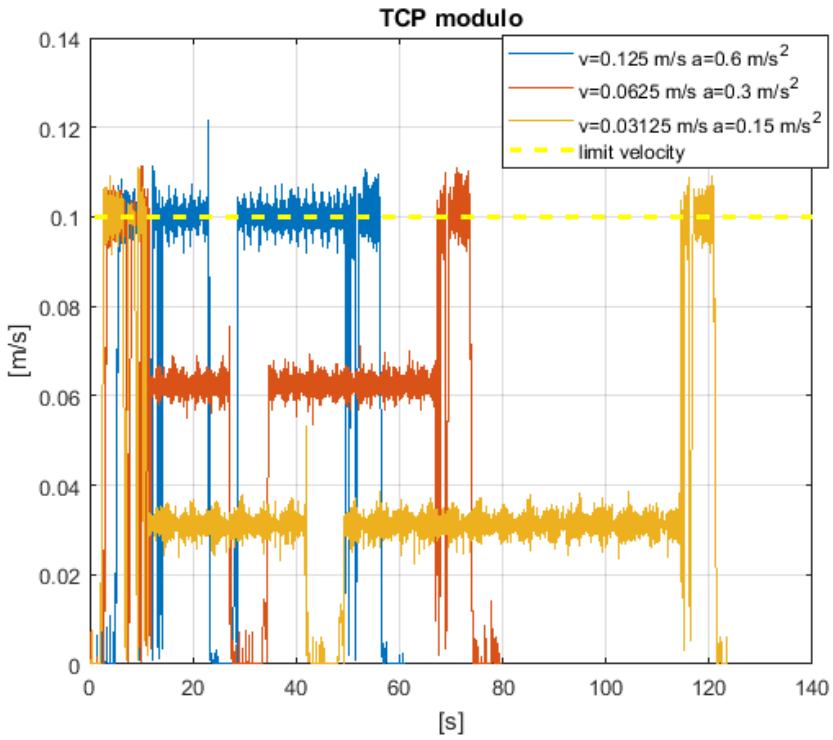


Figura 176: modulo della velocità del TCP

Per quanto riguarda le velocità si può osservare che, non considerando il rumore dell'acquisizione, il tool si muove sempre entro i limiti impostati. Lo si può osservare per tutte e tre le prove quando si considera il movimento di ritorno alla posizione iniziale, poiché tale movimento è stato impostato in tutte le prove con velocità maggiori di quella di sicurezza, ma dal grafico si osserva come vengano sempre saturate al valore limite.

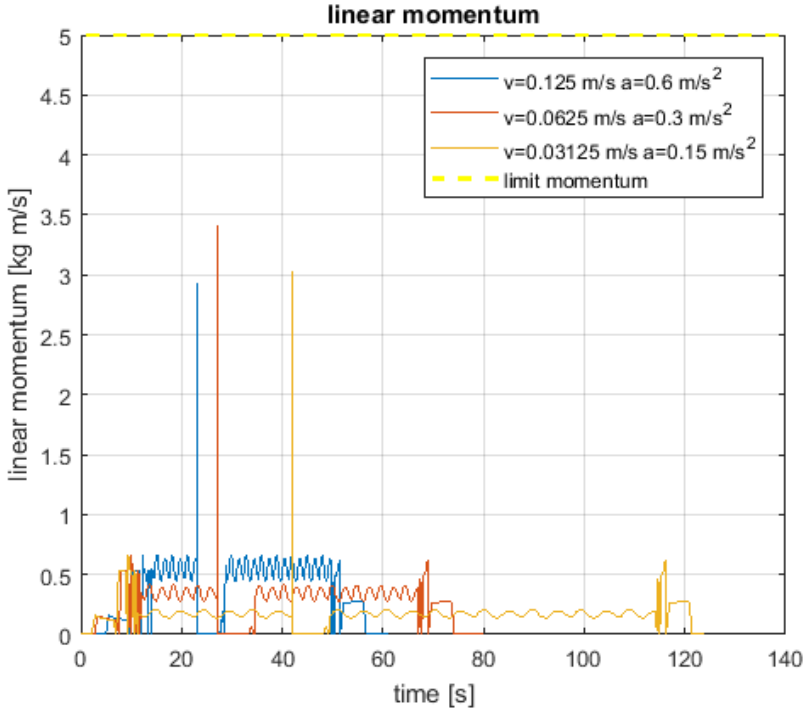


Figura 177: quantità di moto

Dalle prove effettuate si è osservato che nel momento in cui si ha l'interruzione del movimento per mezzo di un utente, il robot si ferma con uno stop di tipologia 2, è come se il programma si stoppasse ma con presenza permanente dell'alimentazione elettrica del robot. Per poter riprendere il movimento basta semplicemente premere il tasto pause/play sul tablet di controllo.

Viene riportato di seguito il grafico rappresentante le coppie applicate ai giunti nei vari casi.

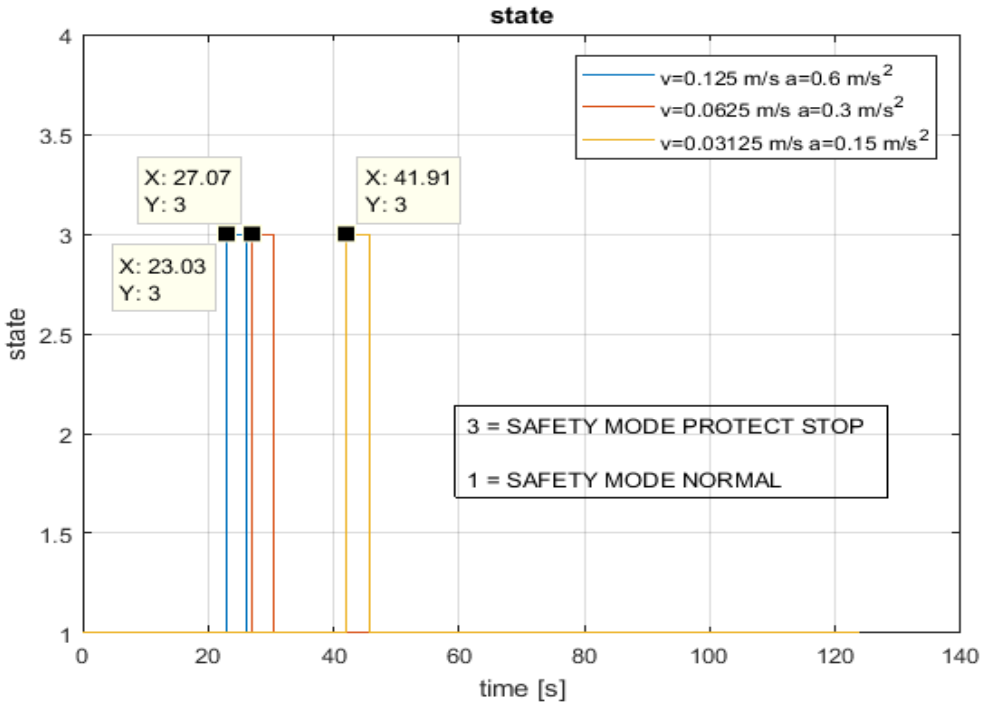


Figura 178: stato del programma

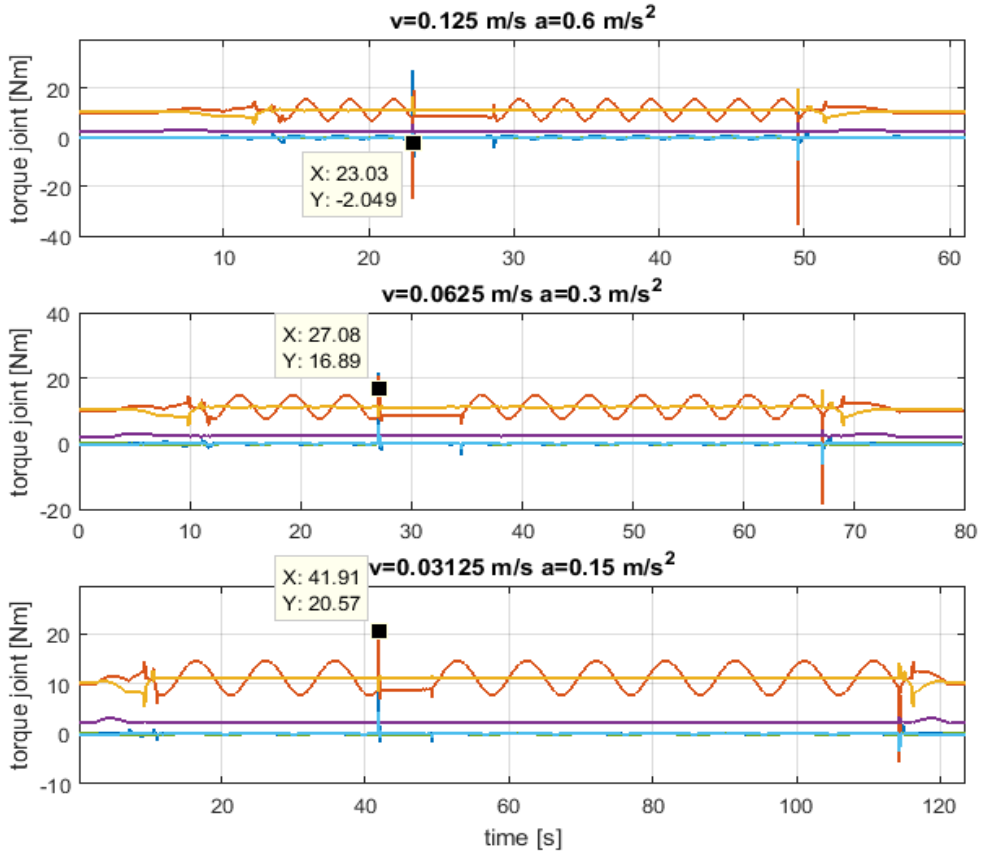


Figura 179: andamento delle coppie nei diversi casi

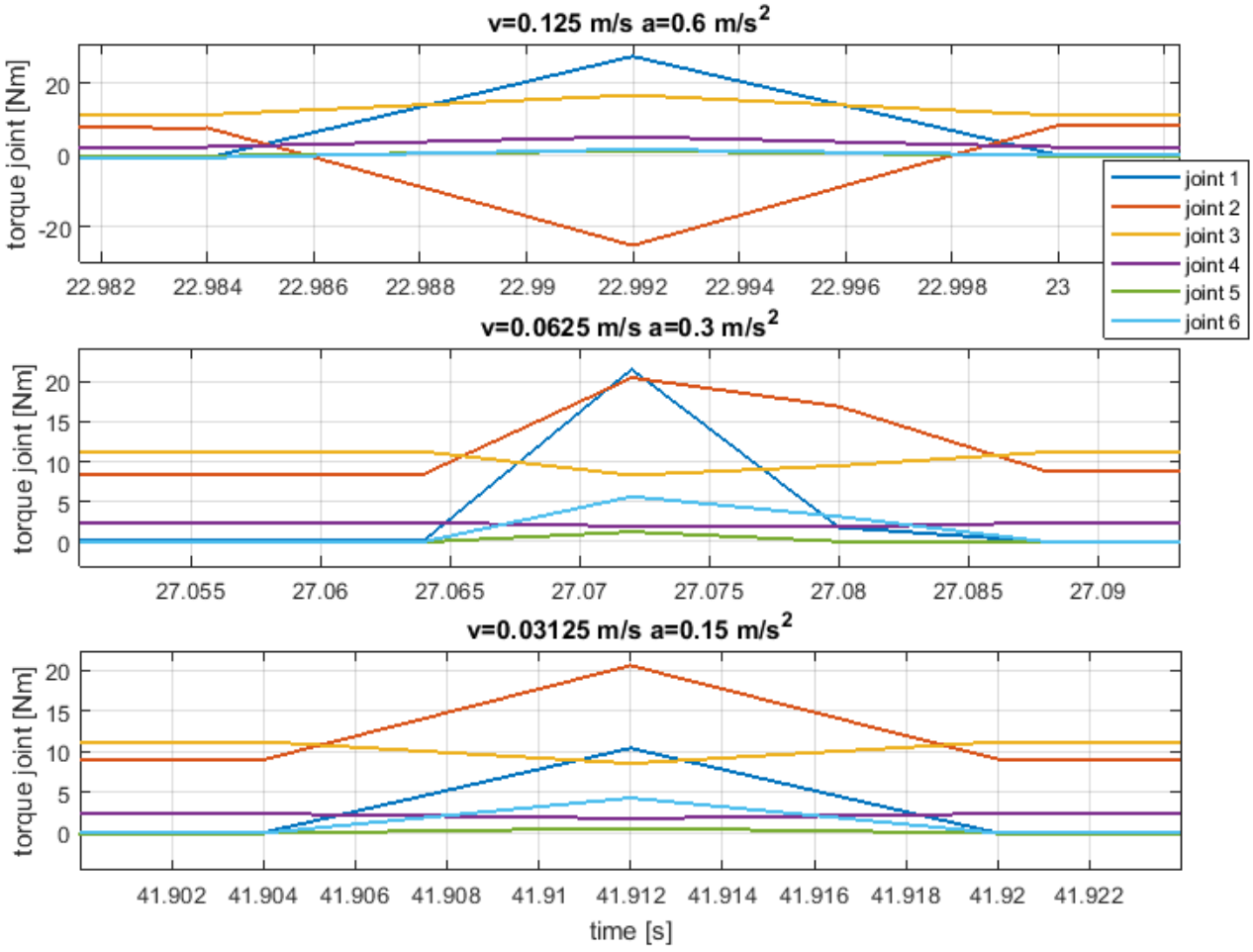


Figura 180: ingrandimento dell'andamento delle coppie al momento dell'urto

Quest'ultimo grafico mostra come durante l'impatto con la mano dell'operatore i giunti che risentono di più di altri dell'urto sono il primo e il secondo. Questi due infatti vengono fermati con una coppia applicata in 16 centesimi di secondo maggiore rispetto a quella applicata agli altri giunti. Si può osservare inoltre che mentre il giunto due è fermato, in tutti i casi, con una coppia in modulo sempre uguale (nel primo grafico è negativa poiché la mano è stata messa in un momento diverso rispetto agli altri due casi); la coppia applicata al primo giunto invece, è funzione della velocità e dell'accelerazione del movimento. Per cui si può dedurre che nel momento in cui un operatore si trova a lavorare a stretto contatto con questi tipi di robot, è consigliabile ed opportuno far muovere la macchina con velocità basse. Vengono riportati di seguito anche i valori delle correnti.

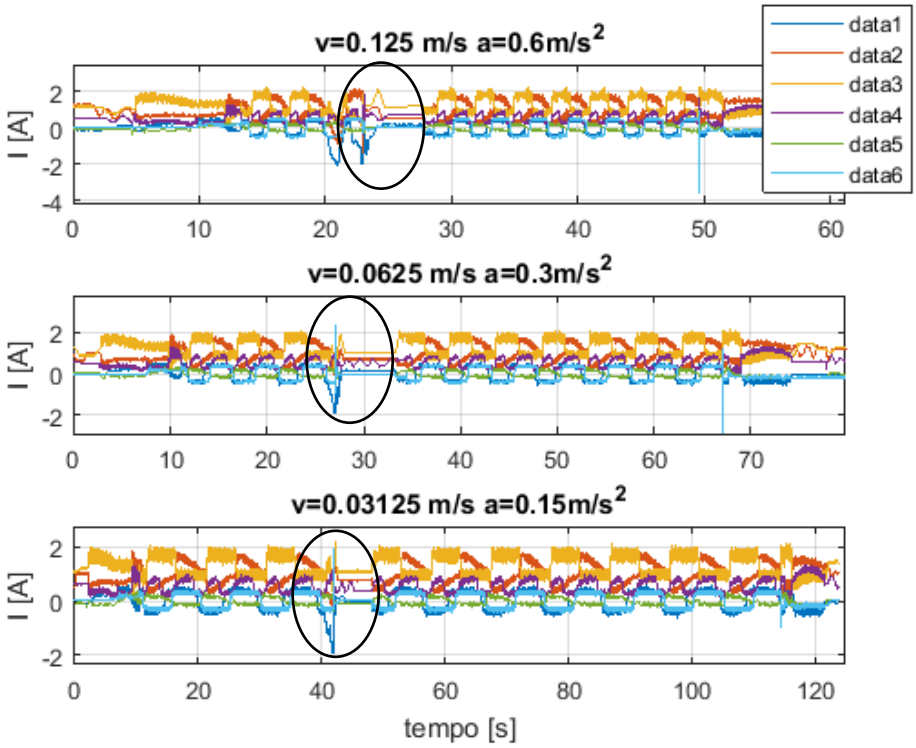


Figura 181: andamento delle correnti

Si osserva che ci sono dei corrispondenti picchi di corrente nel momento in cui si verificano i picchi di coppia. Nel caso delle correnti sembra che il picco si registri qualche istante dopo aver applicato la coppia di stop. Questo fenomeno lo si può osservare confrontando i tempi nel grafico in figura 180 con quello in figura 182. Si può osservare come nel caso delle correnti ci siano picchi non solo per i giunti frenati maggiormente, ma anche per altri, come ad esempio il giunto 6.

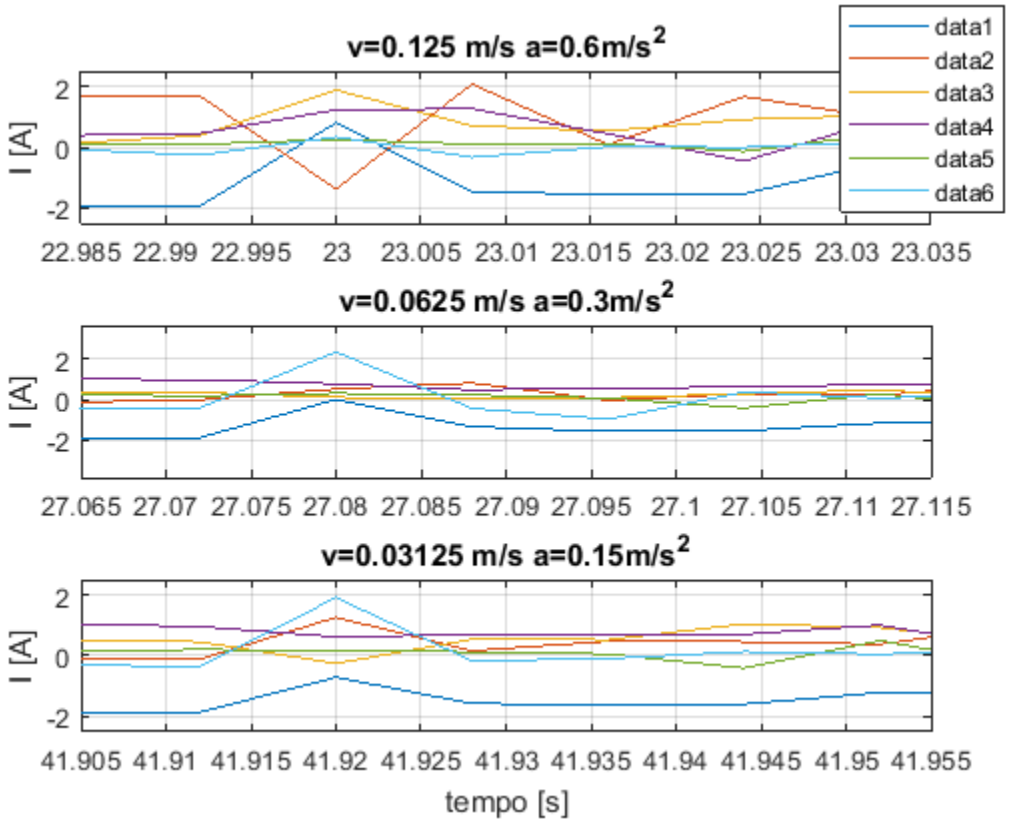


Figura 182: ingrandimento negli istanti dell'urto

5.2.2. *Stack*

- ***Programma scritto:*** PROGRAMMA_stack_acciaio_pinza.urp

Questo movimento è stato già spiegato in precedenza nel capitolo 1. In questo caso viene riprodotto in laboratorio il movimento facendo spostare dei componenti reali di acciaio per mezzo della pinza.

Figura 183: wire-frame movimento stack

Viene riportato di seguito il programma scritto per lo stack. Questo risulta essere simile a quello visto in precedenza ma con la presenza dei comandi della pinza implementati. Si osserva che anche in questo caso si preferisce utilizzare la completa chiusura e apertura della pinza per i movimenti di presa e rilascio.

```
Program
  Robot Program
    Loop 3 times
      Stack
        StartPos_1
        Direction
          FromPos_1
          ToPos_1
        PlaceSequence
          StackPos_1
          Gripper Close (1)
          Exit_1
      Destack
        StartPos_2
        Direction
          FromPos_2
          ToPos_2
        PickSequence
          StackPos_2
          Gripper Open (1)
          Exit_2
      MoveL
        Waypoint_1
    MoveL
      Waypoint_1
```

Figura 184: scrittura del programma di stack

5.2.3. *Applicazione leggero-pesante*

- ***Programma scritto:*** PROVA _17_cella_di_carico_leggero_pesante.urp

L'applicazione pensata per verificare il giusto funzionamento del sensore consiste semplicemente nel prendere due oggetti (con geometria differente dal classico parallelepipedo per mostrare la capacità di adattamento del gripper) e verificare la lettura della componente di forza verticale differente nei due casi. L'oggetto prelevato in questione è sempre una bottiglietta d'acqua di plastica, ma una volta piena e una vuota.

Figura 185: deposito finale movimento

Tabella 39: proprietà bottiglia

	peso [Kg]	lettura cella lungo z [N]
bottiglia piena (destra)	0.5	26.4
bottiglia vuota (sinistra)	0.02	21.4

Viene impostato un if in cui la condizione soglia viene posta pari a 23.5N: se il valore letto dalla cella di carico è inferiore, la macchina capisce che l'oggetto è "leggero", mostra all'utente un pop up in cui trasmette un messaggio di riconoscimento del peso e si sposta a sinistra, altrimenti l'oggetto viene considerato "pesante" e sempre in seguito ad un pop up viene spostato a destra.

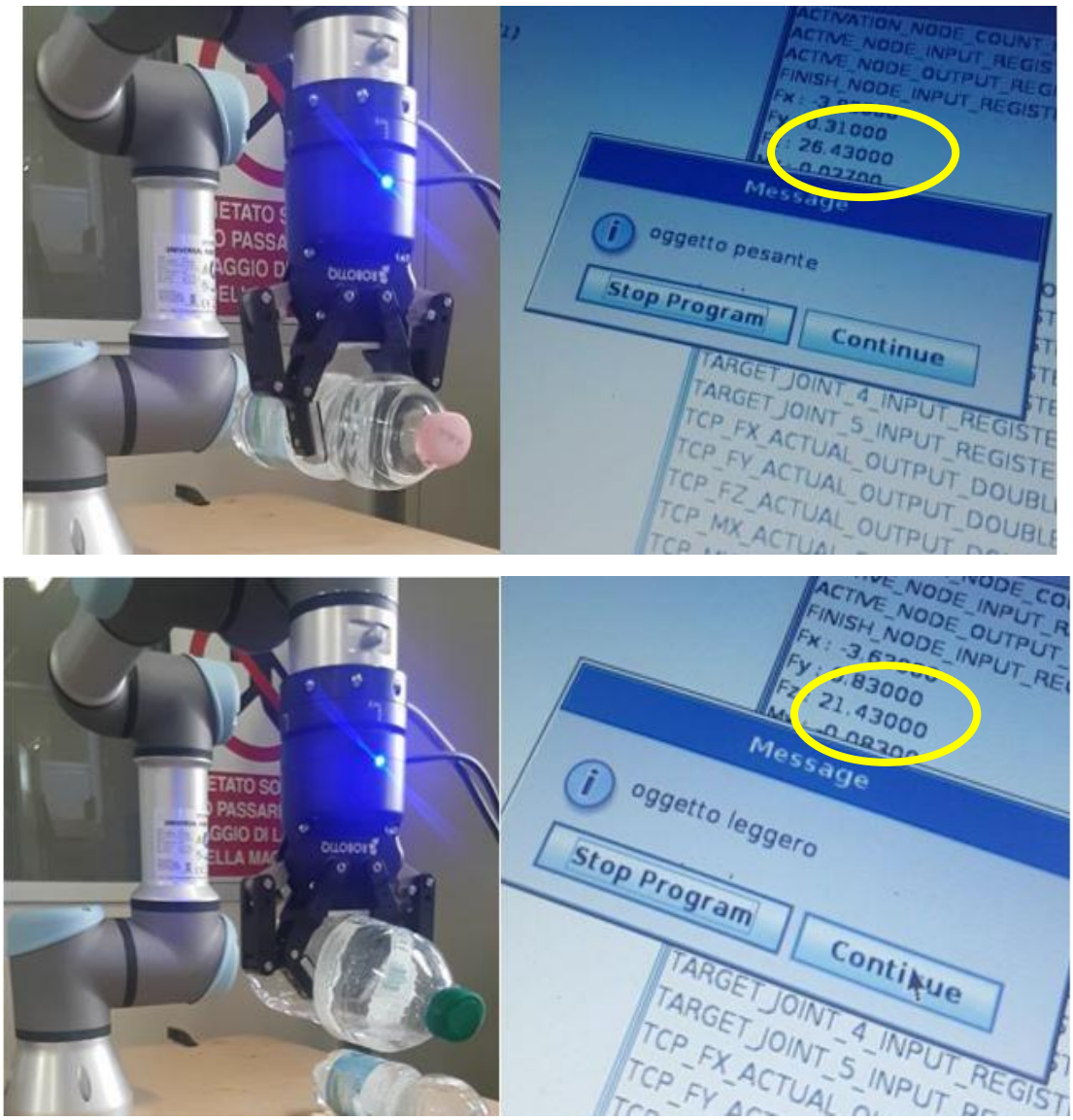


Figura 186: presa degli oggetti

```
Program
  Robot Program
    Loop 2 times
      MoveL
        Waypoint_1
        Waypoint_2
        Gripper Close (1)
        Waypoint_3
      If  Fz<23.5
        Popup
        MoveJ
          Waypoint_4
        MoveL
          Waypoint_5
          Gripper Open (1)
          Waypoint_6
      Else
        Popup
        MoveJ
          Waypoint_7
        MoveL
          Waypoint_9
          Gripper Open (1)
          Waypoint_8
    MoveJ
      Waypoint_10
```

Figura 187 scrittura del programma

5.2.4. *Torre di Hanoi*

- ***Programma scritto:*** PROGRA MMA_torre_hanoi.urp

Come ultimo movimento si decide di programmare la macchina in modo tale che realizzi la torre di Hanoi. Questa applicazione consiste nello spostare dei dischi di diverso diametro da una prima asta ad una terza. Durante gli spostamenti non bisogna collocare dischi di dimensione più grande su altri di dimensione più piccola e spostare un singolo disco alla volta. Nel caso affrontato viene realizzata la torre utilizzando tre dischi di dimensione 70, 65 e 60 mm. Viene illustrato un wire-frame di uno spostamento intermedio e successivamente spiegata la logica di programmazione del movimento.

Figura 188: configurazione torre inizio – fine

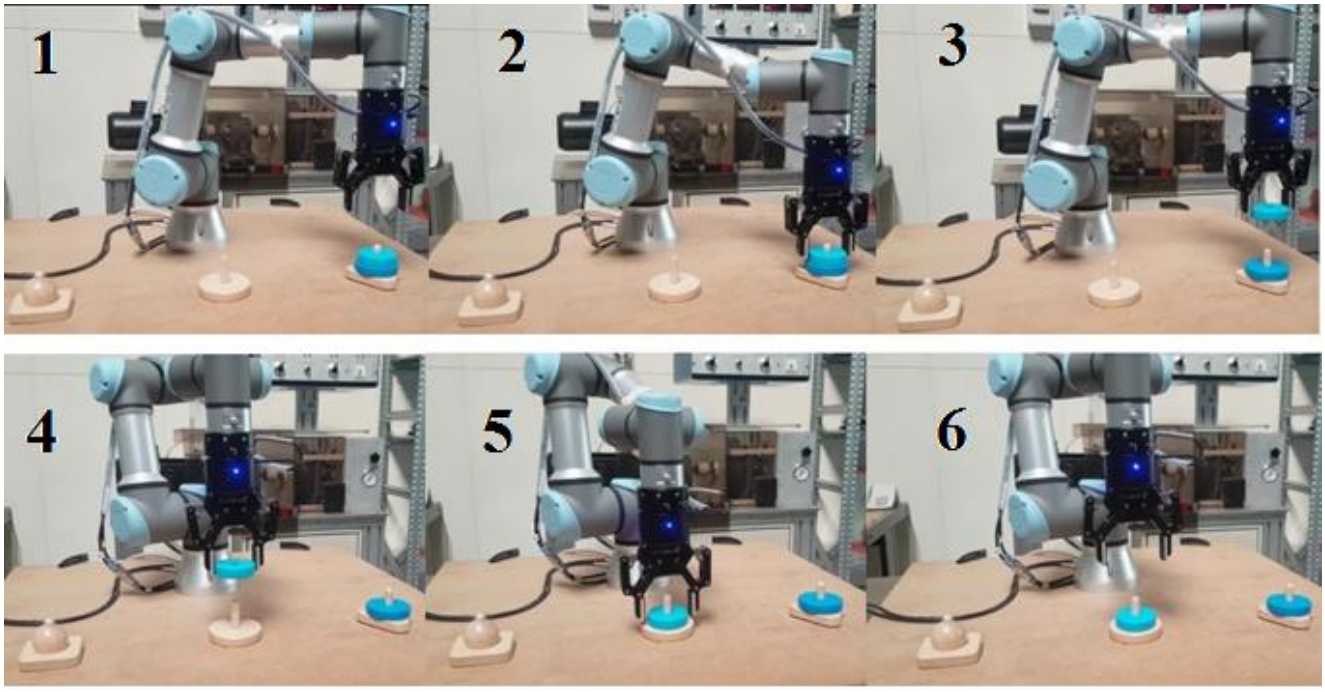


Figura 189: wire-frame terzo movimento

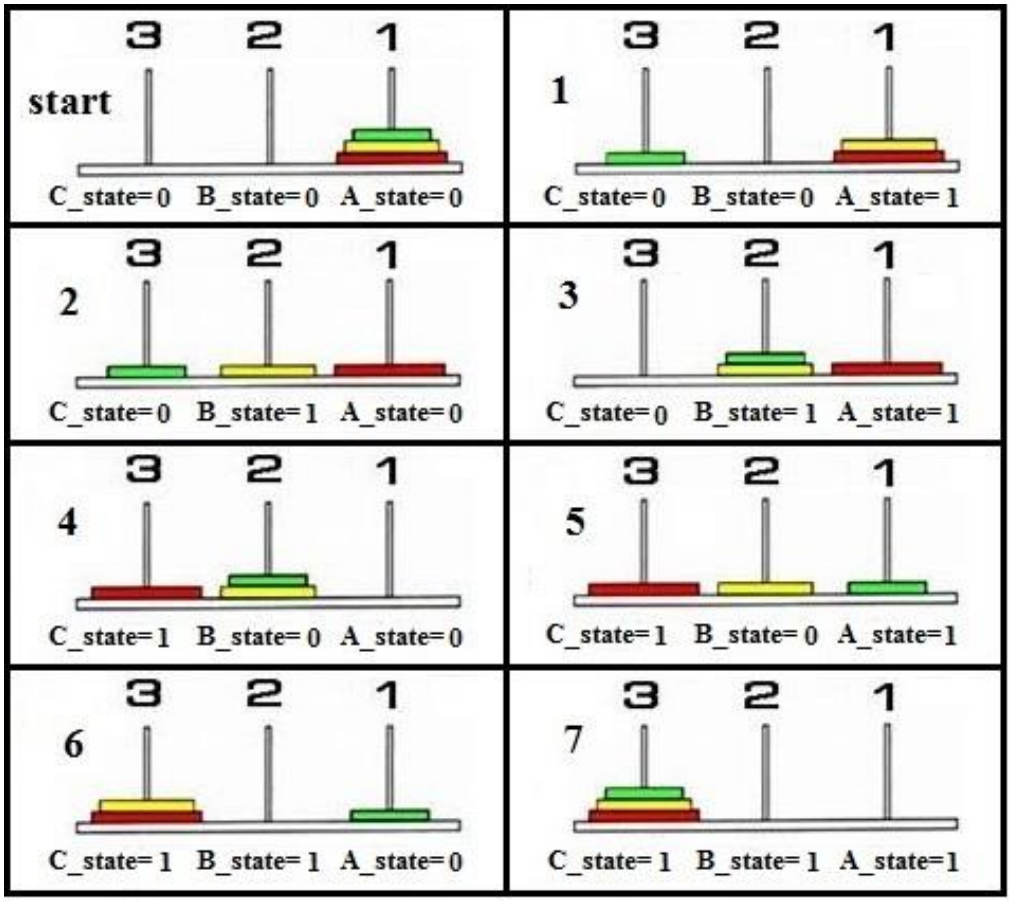


Figura 190: logica di risoluzione della torre: sequenza di spostamenti

Il programma sembra essere molto lungo in realtà può essere suddiviso in due sezioni in cui:

- nella sezione "BeforeStart" vengono scritte tutte le variabili che corrispondono alle diverse posizioni relative ad ogni asse e alle velocità e accelerazioni che si desidera avere durante i movimenti. Inoltre, vengono riportate delle variabili: A_state (riferita alla prima asta), B_state (riferita all'asta centrale) e C_state (riferita alla terza asta) che assumono sempre valore 0 o 1 e servono a capire, a seconda del numero binario che viene letto, quali movimenti il robot deve compiere. Infatti, i numeri binari che si possono generare con tre sole posizioni sono 8, come il numero di eventi da considerare in questa risoluzione. In questo modo ogni movimento viene codificato univocamente con un numero binario;

- nella sezione "StartProgram" vengono riportati i vari comandi parametrizzati collocati all'interno di un loop con un if.. else (con 7 casistiche differenti). A seconda del numero che assumono le variabili A_state, B_state e C_state, il robot comprende sia in quale if deve entrare, leggendo la giusta condizione legata alle variabili, sia il movimento esatto che deve fare spostando in maniera corretta i dischi e rispettando la regola alla base della torre di Hanoi

Questo metodo di risoluzione risulta essere rigido, poiché se si sposta la colonna di partenza, il robot non è più in grado di realizzare la torre. Si potrebbe pensare di implementare l'algoritmo e renderlo più flessibile usando dei sensori visivi e sfruttare la cella di carico per capire quale disco si sta sollevando in base al peso.

Conclusioni

Nel presente elaborato sono stati illustrati con un certo dettaglio i componenti dell'intero sistema meccanico e l'ambiente di programmazione proprio della macchina. Con questo lavoro è stato possibile ottenere una sorta di manuale per comprendere il funzionamento del robot e dei suoi comandi.

Grazie allo sviluppo di alcune funzioni di analisi e acquisizione dati, sono state effettuate differenti osservazioni sui principali movimenti del robot capendone il corretto funzionamento. Successivamente è stato possibile implementare un modello virtuale dell'UR3 per avviare diverse simulazioni. Questo aspetto non è da sottovalutare perché permette ad eventuali utenti di comprendere, prima di compiere un importante investimento per l'acquisto della macchina, come il robot funzioni, la sua area di lavoro e come questo si muova nello spazio con le relative proprietà cinematiche e dinamiche; tale modello, perciò, è stato validato attraverso dati fisici realistici. Per questa operazione tornano nuovamente utili i file di acquisizione, infatti raccogliendo dati sui movimenti reali del robot, è stato possibile verificare la validità del modello Matlab. L'analisi di tutti questi dati consente di poter programmare, con una certa cautela, alcuni movimenti del robot e implementare dei programmi per alcune semplici applicazioni.

Tale modello, inoltre, potrebbe essere utilizzato in progetti futuri o già in sviluppo qui in questo ateneo, che riguardano temi molto importanti e strettamente collegati alla robotica collaborativa come la collision avoidance. Si potrebbe pensare di sviluppare degli algoritmi che, applicati al modello, possano far muovere il robot in ambiente di simulazione in maniera tale da evitare eventuali ostacoli in movimento, quali arti o altre parti del corpo di un eventuale operatore che lavora nei pressi della macchina. Validati nelle simulazioni, questi codici potrebbero essere infine, applicati anche nella realtà, utilizzando degli opportuni sistemi visivi per riconoscere gli ostacoli e permettere al robot di evitarli.

Si deduce che questa macchina pensata da Universal Robot presenti ottime proprietà a scopo collaborativo ed sia in grado, se programmata in maniera corretta, di svolgere opportune funzioni anche al fianco dell'operatore stesso con una grande precisione nei movimenti. Sebbene presenti questi vantaggi e ottime proprietà di sicurezza, si potrebbe lavorare ancora sullo sviluppo di ulteriori sistemi di questo tipo poiché quelli già presenti sono abbastanza efficaci ma talvolta troppo poco restrittivi.

Si potrebbe aggiungere ed implementare una sensoristica leggermente più complessa ed efficace per rendere ancora più sicuro lo stato di salute dell'operatore riducendo in questo modo ulteriormente il rischio.

La scrittura di alcuni programmi consente di sottolineare la grande opportunità che questi robot offrono nel mondo del lavoro, soprattutto poiché permettono di automatizzare e velocizzare numerosi processi ma sempre fianco a fianco con l'operatore. La collaborazione tra uomo e macchina è in continua evoluzione e l'UR3, analizzando tutto ciò che è presente in questo testo, risulta un elemento di spicco di questa nuova crescita verso il futuro.

INDICE PROGRAMMI UR

PER VISUALIZZARE TUTTI I PROGRAMMI, PERCORSO: TESI DIGITALE\UR SCRIPT

PER VISUALIZZARE SUL CONTROLLER, PERCORSO: 241958\ProgrammiUR3

NOME DEL PROGRAMMA (formati ".urp")	UTILIZZO (acquisizioni in formato ".mat")
PROVA_BASE_moveJ_prova_1	ACQUISIZIONE_moveJ_prova_1
PROVA_BASE_moveJ_prova_2	ACQUISIZIONE_moveJ_prova_2
PROVA_BASE_moveJ_prova_3.1	ACQUISIZIONE_moveJ_prova_3.1
PROVA_BASE_moveJ_prova_3.2	ACQUISIZIONE_moveJ_prova_3.2
PROVA_BASE_moveJ_prova_4	ACQUISIZIONE_moveJ_prova_4
PROVA_BASE_moveJ_prova_5	ACQUISIZIONE_moveJ_prova_5
PROVA_BASE_moveL_prova_1	ACQUISIZIONE_moveL_prova_1
PROVA_BASE_moveL_prova_2	ACQUISIZIONE_moveL_prova_2
PROVA_BASE_moveP_prova_1	ACQUISIZIONE_moveP_prova_1
PROVA_1_1_moveJ_velocità_default_giunto1	ACQUISIZIONE_1_1_moveJ_velocità_default_giunto1
PROVA_1_2_moveJ_velocità_dimezzata_giunto1	ACQUISIZIONE_1_2_moveJ_velocità_dimezzata_giunto 1
PROVA_1_3_moveJ_velocità_al10%_giunto1	ACQUISIZIONE_1_3_moveJ_velocità_al10%_giunto1
PROVA_1_4_moveJ_velocità_al90%_giunto1	ACQUISIZIONE_1_4_moveJ_velocità_al90%_giunto1
PROVA_2_1_moveJ_velocità_default_giunti	ACQUISIZIONE_2_1_moveJ_velocità_default_giunti
PROVA_2_2_moveJ_velocità_dimezzata_giunti	ACQUISIZIONE_2_2_moveJ_velocità_dimezzata_giunti
PROVA_2_3_moveJ_velocità_al10%_giunti	ACQUISIZIONE_2_3_moveJ_velocità_al10%_giunti
PROVA_2_4_moveJ_velocità_al90%_giunti	ACQUISIZIONE_2_4_moveJ_velocità_al90%_giunti
PROVA_3_1_moveL_velocità_default_lungo_z	ACQUISIZIONE_3_1_moveL_velocità_default_lungo_z
PROVA_3_2_moveL_velocità_dimezzata_lungo_z	ACQUISIZIONE_3_2_moveL_velocità_dimezzata_lungo _z
PROVA_3_3_moveL_velocità_al10%_lungo_z	ACQUISIZIONE_3_3_moveL_velocità_al10%_lungo_z
PROVA_3_4_moveL_velocità_al90%_lungo_z	ACQUISIZIONE_3_4_moveL_velocità_al90%_lungo_z
PROVA_4_1_moveL_velocità_default_giunti	ACQUISIZIONE_4_1_moveL_velocità_default_giunti
PROVA_4_2_moveL_velocità_dimezzata_giunti	ACQUISIZIONE_4_2_moveL_velocità_dimezzata_giunti
PROVA_4_3_moveL_velocità_al10%_giunti	ACQUISIZIONE_4_3_moveL_velocità_al10%_giunti
PROVA_4_4_moveL_velocità_al90%_giunti	ACQUISIZIONE_4_4_moveL_velocità_al90%_giunti
PROVA_5_1_spostamento_in_rapido_movej	ACQUISIZIONE_5_1_spostamento_in_rapido_movej

PROVA_5_2_spostamento_in_rapido_moveL	ACQUISIZIONE_5_2_spostamento_in_rapido_moveL
PROVA_6_1_parete_vel_default	ACQUISIZIONE_6_1_parete_vel_default
PROVA_6_2_parete_vel_dimezzata	ACQUISIZIONE_6_2_parete_vel_dimezzata
PROVA_6_3_parete_vel_al10%	ACQUISIZIONE_6_3_parete_vel_al10%
PROVA_7_1_pause_play_vel_default	ACQUISIZIONE_7_1_pause_play_vel_default
PROVA_7_2_pause_play_vel_dimezzata	ACQUISIZIONE_7_2_pause_play_vel_dimezzata
PROVA_7_3_pause_play_vel_al10%	ACQUISIZIONE_7_3_pause_play_vel_al10%
PROVA_8_1_stop_button	ACQUISIZIONE_8_1_stop_button
PROVA_9_1_giunto_due_vel_default	ACQUISIZIONE_9_1_giunto_due_vel_default
PROVA_9_2_giunto_due_vel_dimezzata	ACQUISIZIONE_9_2_giunto_due_vel_dimezzata
PROVA_9_3_giunto_due_vel_al10%	ACQUISIZIONE_9_3_giunto_due_vel_al10%
PROVA_10_1_giunto_tre_vel_default	ACQUISIZIONE_10_1_giunto_tre_vel_default
PROVA_10_2_giunto_tre_dimezzata	ACQUISIZIONE_10_2_giunto_tre_vel_dimezzata
PROVA_10_3_giunto_tre_vel_al10%	ACQUISIZIONE_10_3_giunto_tre_vel_al10%
PROVA_11_1_movimento_giunto_3_verso_alto	ACQUISIZIONE_11_1_movimento_giunto_3_verso_alto
PROVA_11_2_movimento_giunto_2e3	ACQUISIZIONE_11_2_movimento_giunto_2e3
PROVA_12_1_simulatore_giunto1_vel_acc_max	ACQUISIZIONE_12_1_simulatore_giunto1_vel_acc_max
PROVA_12_2_simulatore_giunto1_vel_acc_max_condizioniristrette	ACQUISIZIONE_12_2_simulatore_giunto1_vel_acc_max_condizioniristrette
PROVA_13_1_simulatore_giunto4_vel_acc_max	ACQUISIZIONE_13_1_simulatore_giunto4_vel_acc_max
PROVA_13_2_simulatore_giunto4_vel_acc_max_condizionirestrittive	ACQUISIZIONE_13_2_simulatore_giunto4_vel_acc_max_condizionirestrittive
PROVA_14_1_moveL_rettangolo_no_raccordi	ACQUISIZIONE_14_1_moveL_rettangolo_no_raccordi
PROVA_14_2_moveL_rettangolo_r50	ACQUISIZIONE_14_2_moveL_rettangolo_r50
PROVA_14_3_moveL_rettangolo_r100	ACQUISIZIONE_14_3_moveL_rettangolo_r100
PROVA_14_4_moveL_rettangolo_r150	ACQUISIZIONE_14_4_moveL_rettangolo_r150
PROVA_14_5_moveL_rettangolo_r50_vel_variabili_def_10_10_def	ACQUISIZIONE_14_5_moveL_rettangolo_r50_vel_variabili_def_10_10_def
PROVA_14_6_moveL_rettangolo_r100_vel_variabili_def_10_10_def	ACQUISIZIONE_14_6_moveL_rettangolo_r100_vel_variabili_def_10_10_def
PROVA_14_7_moveL_rettangolo_r150_vel_variabili_def_10_10_def	ACQUISIZIONE_14_7_moveL_rettangolo_r150_vel_variabili_def_10_10_def
PROVA_15_1_moveJ_rettangolo_no_raccordi	ACQUISIZIONE_15_1_moveJ_rettangolo_no_raccordi

PROVA_15_2_moveJ_rettangolo_r50	ACQUISIZIONE_15_2_moveJ_rettangolo_r50
PROVA_15_3_moveJ_rettangolo_r100	ACQUISIZIONE_15_3_moveJ_rettangolo_r100
PROVA_15_4_moveJ_rettangolo_r150	ACQUISIZIONE_15_4_moveJ_rettangolo_r150
PROVA_15_5_moveJ_rettangolo_r50_vel_variabili_def_10_10_def	ACQUISIZIONE_15_5_moveJ_rettangolo_r50_vel_variabili_def_10_10_def
PROVA_15_6_moveJ_rettangolo_r100_vel_variabili_def_10_10_def	ACQUISIZIONE_15_6_moveJ_rettangolo_r100_vel_variabili_def_10_10_def
PROVA_15_7_moveJ_rettangolo_r150_vel_variabili_def_10_10_def	ACQUISIZIONE_15_7_moveJ_rettangolo_r150_vel_variabili_def_10_10_def
PROVA_16_1_CIAO_default	ACQUISIZIONE_16_1_CIAO_default VIDEO_CIAO
PROVA _16_2_CIAO_raccordo_soloP	ACQUISIZIONE_16_2_CIAO_raccordo_soloP
PROVA_16_3_CIAO_tutto_raccordato_5	ACQUISIZIONE_16_3_CIAO_tutto_raccordato_5
PROVA _16_4_CIAO_tutto_raccordato_25	ACQUISIZIONE_16_4_CIAO_tutto_raccordato_25
PROVA _16_5_CIAO_tutto_raccordato_50	ACQUISIZIONE_16_5_CIAO_tutto_raccordato_50
PROVA _17_cella_di_carico_leggero_pesante	ACQUISIZIONE_17_cella_di_carico_leggero_pesante VIDEO_leggero_pesante
PROVA _18_1_miscelazione_no_rollio	ACQUISIZIONE_18_1_miscelazione_no_rollio VIDEO_pentola_norollio
PROVA _18_2_miscelazione_rollio	ACQUISIZIONE_18_2_miscelazione_rollio VIDEO_pentola_rollio
PROVA _18_3_miscelazione_inseritomano_default	ACQUISIZIONE_18_3_miscelazione_inseritomano_default
PROVA_18_4_miscelazione_inseritomano_default_mezzi	ACQUISIZIONE_18_4_miscelazione_inseritomano_default_mezzi
PROVA_18_5_miscelazione_inseritomano_default_quarti	ACQUISIZIONE_18_5_miscelazione_inseritomano_default_quarti
PROVA _19_acquisizione_torre_hanoi	ACQUISIZIONE_19_acquisizione_torre_hanoi
PROGRAMMA_pick_and_place_nopinza	VIDEO_pick_and_place_nopinza
PROGRAMMA_pallet_nopinza	VIDEO_pallet_nopinza
PROGRAMMA_stack_nopinza	VIDEO_stack_nopinza
PROGRAMMA_stack_acciaio_pinza	VIDEO_stack_acciaio_pinza
PROGRAMMA_if_nopinza	VIDEO_if_nopinza
PROGRAMMA_switch_nopinza	VIDEO_switch_nopinza
PROGRAMMA_torre_hanoi	VIDEO_torre_hanoi

INDICE ACQUISIZIONI

PER VISUALIZZARE TUTTE LE ACQUISIZIONI, PERCORSO: TESI DIGITALE\Matlab\PROVE

Prove per la spiegazione dei MOVE (paragrafo 1.3) (formato ".mat")	Descrizione
ACQUISIZIONE_moveJ_prova_1	prova utilizzata per verificare il tempo di crociera del moveJ (impostando velocità ed accelerazione)
ACQUISIZIONE_moveJ_prova_2	prova utilizzata per verificare la velocità ed accelerazione di crociera (impostando il tempo)
ACQUISIZIONE_moveJ_prova_3.1	prova utilizzata per analizzare gli "shared parameters"
ACQUISIZIONE_moveJ_prova_3.2	prova utilizzata per analizzare gli "shared parameters"
ACQUISIZIONE_moveJ_prova_4	prova per comprendere il leader axis
ACQUISIZIONE_moveJ_prova_5	prova effettuata per verificare i waypoint relativi
ACQUISIZIONE_moveL_prova_1	movimenti consecutivi moveL senza raccordo
ACQUISIZIONE_moveL_prova_2	movimenti consecutivi moveL con raccordo
ACQUISIZIONE_moveP_prova_1	movimenti consecutivi moveP con raccordo

	Nome delle acquisizioni della libreria (nei capitoli 3, 4 e 5) (formato ".mat")	Descrizione
MoveJ movimento di un solo giunto (primo giunto)	ACQUISIZIONE_1_1_moveJ_velocità_default_giunto1	Prova con velocità default (60°/s) e accelerazione default (80°/s)
	ACQUISIZIONE_1_2_moveJ_velocità_dimezzata_giunto1	Prova con velocità dimezzata (30°/s) ad accelerazione di default
	ACQUISIZIONE_1_3_moveJ_velocità_al10%_giunto1	Prova con velocità ridotta del 10% ad accelerazione di default
	ACQUISIZIONE_1_4_moveJ_velocità_al90%_giunto1	Prova con velocità ridotta del 90% ad accelerazione di default
MoveJ movimento di tutti i giunti	ACQUISIZIONE_2_1_moveJ_velocità_default_giunti	Prova con velocità default (60°/s) e accelerazione default (80°/s)
	ACQUISIZIONE_2_2_moveJ_velocità_dimezzata_giunti	Prova con velocità dimezzata (30°/s) ad accelerazione di default
	ACQUISIZIONE_2_3_moveJ_velocità_al10%_giunti	Prova con velocità ridotta del 10% ad accelerazione di default
	ACQUISIZIONE_2_4_moveJ_velocità_al90%_giunti	Prova con velocità ridotta del 90% ad accelerazione di default
MoveL movimento in spazio operativo (spostamento solo lungo la direzione verticale z)	ACQUISIZIONE_3_1_moveL_velocità_default_lungo_z	Prova con velocità TCP default (250mm/s) e accelerazione TCP default (1200mm/s^2)
	ACQUISIZIONE_3_2_moveL_velocità_dimezzata_lungo_z	Prova con velocità TCP dimezzata (125mm/s) ad accelerazione di default

	ACQUISIZIONE_3_3_moveL_velocità_al10%_lungo_z	Prova con velocità TCP ridotta del 10% ad accelerazione di default
	ACQUISIZIONE_3_4_moveL_velocità_al90%_lungo_z	Prova con velocità TCP ridotta del 90% ad accelerazione di default
MoveL movimento in spazio operativo (spostamento lungo la direzione x, sia y sia z)	ACQUISIZIONE_4_1_moveL_velocità_default_giunti	Prova con velocità TCP default (250mm/s) e accelerazione TCP default (1200mm/s^2)
	ACQUISIZIONE_4_2_moveL_velocità_dimezzata_giunti	Prova con velocità TCP dimezzata (125mm/s) ad accelerazione di default
	ACQUISIZIONE_4_3_moveL_velocità_al10%_giunti	Prova con velocità TCP ridotta del 10% ad accelerazione di default
	ACQUISIZIONE_4_4_moveL_velocità_al90%_giunti	Prova con velocità TCP ridotta del 90% ad accelerazione di default
prove in rapido	ACQUISIZIONE_5_1_spostamento_in_rapido_movej	Prove in rapido (AUTOMOVE)
	ACQUISIZIONE_5_2_spostamento_in_rapido_moveL	Prove in rapido (AUTOMOVE)
impatto parete giunto 1	ACQUISIZIONE_6_1_parete_vel_default	Prova con velocità default (60°/s) e accelerazione default (80°/s)
	ACQUISIZIONE_6_2_parete_vel_dimezzata	Prova con velocità dimezzata (30°/s) ad accelerazione di default
	ACQUISIZIONE_6_3_parete_vel_al10%	Prova con velocità ridotta del 10% ad accelerazione di default

"pause and play" giunto 1	ACQUISIZIONE_7_1_pause_play_vel_default	Prova con velocità default (60°/s) e accelerazione default (80°/s)
	ACQUISIZIONE_7_2_pause_play_vel_dimezzata	Prova con velocità dimezzata (30°/s) ad accelerazione di default
	ACQUISIZIONE_7_3_pause_play_vel_al10%	Prova con velocità ridotta del 10% ad accelerazione di default
tasto rosso di emergenza	ACQUISIZIONE_8_1_stop_button	Prova premendo il tasto di emergenza rosso
MoveJ movimento un solo giunto contro gravità (giunto 2)	ACQUISIZIONE_9_1_giunto_due_vel_default	Prova con velocità default (60°/s) e accelerazione default (80°/s)
	ACQUISIZIONE_9_2_giunto_due_vel_dimezzata	Prova con velocità dimezzata (30°/s) ad accelerazione di default
	ACQUISIZIONE_9_3_giunto_due_vel_al10%	Prova con velocità ridotta del 10% ad accelerazione di default
MoveJ movimento un solo giunto contro gravità (giunto tre)	ACQUISIZIONE_10_1_giunto_tre_vel_default	Prova con velocità default (60°/s) e accelerazione default (80°/s)
	ACQUISIZIONE_10_2_giunto_tre_vel_dimezzata	Prova con velocità dimezzata (30°/s) ad accelerazione di default
	ACQUISIZIONE_10_3_giunto_tre_vel_al10%	Prova con velocità ridotta del 10% ad accelerazione di default
prove dinamiche	ACQUISIZIONE_11_1_movimento_giunto_3_verso_alto	Prova attuando solo il giunto tre (rotazione link 3 verso l'alto)
	ACQUISIZIONE_11_2_movimento_giunto_2e3	Prova attuando giunto due e tre (traslazione del link 3)
movimento giunto uno per max e min velocità	ACQUISIZIONE_12_1_simulatore_giunto1_vel_acc_max	In condizioni di sicurezza meno restrittive

	ACQUISIZIONE_12_2_simulatore_giunto1_vel_acc_max_condizioniristrette	In condizioni di sicurezza più restrittive
movimento giunto 4 per max e min velocità	ACQUISIZIONE_13_1_simulatore_giunto4_vel_acc_max	In condizioni di sicurezza meno restrittive
	ACQUISIZIONE_13_2_simulatore_giunto4_vel_acc_max_condizionirestrittive	In condizioni di sicurezza più restrittive
movimento per studio dei raccordi: rettangolo realizzato con movimenti di tipo moveL	ACQUISIZIONE_14_1_moveL_rettangolo_no_raccordi	Prova senza utilizzo di raccordi
	ACQUISIZIONE_14_2_moveL_rettangolo_r50	Prova con raccordi di 50mm mantenendo le velocità costanti
	ACQUISIZIONE_14_3_moveL_rettangolo_r100	Prova con raccordi di 100mm mantenendo le velocità costanti
	ACQUISIZIONE_14_4_moveL_rettangolo_r150	Prova con raccordi di 150mm mantenendo le velocità costanti
	ACQUISIZIONE_14_5_moveL_rettangolo_r50_vel_variabili_def_10_10_def	Prova con raccordi di 50mm variando le velocità
	ACQUISIZIONE_14_6_moveL_rettangolo_r100_vel_variabili_def_10_10_def	Prova con raccordi di 100mm variando le velocità
	ACQUISIZIONE_14_7_moveL_rettangolo_r150_vel_variabili_def_10_10_def	Prova con raccordi di 150mm variando le velocità
movimento per studio dei raccordi: rettangolo realizzato con movimenti di tipo moveJ	ACQUISIZIONE_15_1_moveJ_rettangolo_no_raccordi	Prova senza utilizzo di raccordi
	ACQUISIZIONE_15_2_moveJ_rettangolo_r50	Prova con raccordi di 50mm mantenendo le velocità costanti
	ACQUISIZIONE_15_3_moveJ_rettangolo_r100	Prova con raccordi di 100mm mantenendo le velocità costanti
	ACQUISIZIONE_15_4_moveJ_rettangolo_r150	Prova con raccordi di 150mm mantenendo le velocità costanti
	ACQUISIZIONE_15_5_moveJ_rettangolo_r50_vel_variabili_def_10_10_def	Prova con raccordi di 50mm variando le velocità
	ACQUISIZIONE_15_6_moveJ_rettangolo_r100_vel_variabili_def_10_10_def	Prova con raccordi di 100mm variando le velocità
	ACQUISIZIONE_15_7_moveJ_rettangolo_r150_vel_variabili_def_10_10_def	Prova con raccordi di 150mm variando le velocità

APPLICAZIONE: scritta CIAO	ACQUISIZIONE_16_1_CIAO_default	Prova default
	ACQUISIZIONE_16_2_CIAO_raccordo_soloP	Prova raccordando solo i moveP (utilizzata la funzione circolar move)
	ACQUISIZIONE_16_3_CIAO_tutto_raccordato_5	Prova con raccordi di tutta la traiettoria di 5mm
	ACQUISIZIONE_16_4_CIAO_tutto_raccordato_25	Prova con raccordi di tutta la traiettoria di 25mm
	ACQUISIZIONE_16_5_CIAO_tutto_raccordato_50	Prova con raccordi di tutta la traiettoria di 50mm
APPLICAZIONE. cella di carico	ACQUISIZIONE_17_cella_di_carico_leggero_pesante	Prova default
APPLICAZIONE: miscelazione in pentola	ACQUISIZIONE_18_1_miscelazione_no_rollio	Prova di miscelazione no rollio
	ACQUISIZIONE_18_2_miscelazione_rollio	Prove per la sicurezza: inserimento della mano con velocità default
	ACQUISIZIONE_18_3_miscelazione_inseritomano_default	Prove per la sicurezza: inserimento della mano con velocità default diviso due
	ACQUISIZIONE_18_4_miscelazione_inseritomano_default_mezzi	Prove per la sicurezza: inserimento della mano con velocità default diviso due
	ACQUISIZIONE_18_5_miscelazione_inseritomano_default_quarti	Prove per la sicurezza: inserimento della mano con velocità default diviso quattro
APPLICAZIONE: torre di Hanoi	ACQUISIZIONE_19_acquisizione_torre_hanoi	Prova default

INDICE FUNZIONI MATLAB

PER VISUALIZZARE TUTTE LE FUNZIONI, PERCORSO: TESI DIGITALE\Matlab\FILE PER L'ACQUISIZIONE o TESI DIGITALE\Matlab\MODELLO MULTIBODY

FUNZIONI MATLAB	DESCRIZIONE:
File per l'acquisizione	
Main.m	programma principale in cui vengono lanciati gli altri file utili per l'acquisizione
file_acquisizione.m	file analizzato nel testo, permette l'acquisizione dei dati durante un qualsiasi movimento
file_estrazione.m	file che permette l'estrazione delle singole grandezze dalle acquisizioni fatte
Connection2UR_ON_35.m	apre il socket di comunicazione tra UR3 e pc
Connection2UR_OFF.m	chiude il socket di comunicazione tra UR3 e pc
switchMoved.m	funzione di supporto per la GUI
UR3Stream30003_V6_35.m	funzione che permette la lettura del socket
plot_singoli_tutte_le_grandezze.m	file per fare i grafici di tutte le grandezze estratte
plot_grandezze_importanti.m	file per fare solo alcuni grafici di grandezze più significative
sovrapponi_grafici.m	funzione per fare opportune sovrapposizioni
plot_analisi_urto.m	file per grafica utilizzato per lo studio dei dati acquisiti dalle prove di urto
File D-H modificata	
main_UR3_modificata.m	programma principale del modello. Vengono inseriti i parametri di D_H modificato e i dati dinamici e lanciate le diverse funzioni.
build_robot_UR3_modificata.m	funzione che permette di costruire il robot
forward_kinematics_modificata.m	funzione che permette di calcolare le matrici di trasformazione
plot_UR3.m	funzione utilizzata per riprodurre in ambiente virtuale il robot
rne_mdh_POLITO.m	funzione che risolve la dinamica inversa

r2angvec_POLITO_aggiornata.m	funzione aggiornata per il calcolo delle matrici di rotazione
forward_kinematics_inverse_dynamics.m	risolve cinematica diretta e dinamica inversa e restituisce grafici
graph_UR3.slx	file simulink per la riproduzione virtuale del robot
File D-H standard	
main_UR3_standard.m	programma principale del modello. Vengono inseriti i parametri di D_H modificato e lanciate le diverse funzioni.
build_robot_UR3_standard.m	funzione che permette di costruire il robot
forward_kinematics_standard.m	funzione che permette di calcolare le matrici di trasformazione

INDICE SCRIPT V-REP

PER VISUALIZZARE TUTTI GLI SCRIPT V-REP, PERCORSO: TESI DIGITALE\V-REP\script v-rep

SCRIPT	DESCRIZIONE
pick and place	
Gripper pick and place.txt	codifica del gripper per questo movimento
Laser pick and place.txt	codifica del sensore di posizione laser
Nastro trasportatore n1 pick and place.txt	codifica del movimento del nastro numero 1
Nastro trasportatore n2 pick and place.txt	codifica del movimento del nastro numero 2
pallet	
Gripper pallet.txt	codifica del gripper per questo movimento
Nastro trasportatore pallet.txt	codifica del movimento del nastro
stack	
Gripper stack.txt	codifica del gripper per questo movimento

INDICE VIDEO V-REP

PER VISUALIZZARE I VIDEO, PERCORSO:

TESI DIGITALE\V-REP\V-REP simulation\VIDEO_REALIZZATI

OUTPUT V-REP: VIDEO
1) Pick and place
2) pallet
3) stack

INDICE FUNZIONI MATLAB PER CONNESSIONI SOCKET

PER VISUALIZZARE LE FUNZIONI DI CONNESSIONE, PERCORSO: TESI DIGITALE\V-REP\V-REP simulation\COMANDI_MATLAB_REALIZZATI

Funzione	Descrizione
pick_and_place_command.m	programma principale per gestire la comunicazione tra il software di UR e V-REP per tale movimento
pallet_command.m	programma principale per gestire la comunicazione tra il software di UR e V-REP per tale movimento
stack_command.m	programma principale per gestire la comunicazione tra il software di UR e V-REP per tale movimento
switchMoved.m	funzione di ausilio per la GUI
Connection2UR_v1.m	connessione tra software di UR e matlab: permette apertura e chiusura del socket (sia quello di sola lettura sia quello dove Matlab può scrivere).
URStream_v1.m	funzione di lettura dati dal socket
VREPHandle_UR3.m	funzione di ausilio per muovere UR in ambiente V-REP tramite i comandi mandati dal software di UR
VAR2URa.m	genera una variabile su Matlab e la invia al software di UR
Richiamo_sensore1.urp (script appoggio UR3)	software di appoggio sul software di UR per permettere la connessione con il Matlab (dove il Matlab funge da server).

Versioni software utilizzati

- Matlab R2017a
- VirtualBox versione 5.2.8 r121009
- URSim 3.5.3.10825
- V-REP PRO EDU versione 3.5.0
- Peter Corke toolbox 9.1

Bibliografia

[1] V. F., «La robotica collaborativa,» tecniche nuove.

[2] Robotiq, «Robotiq FT-300 force torque sensor».

[3] Robotiq, «Robotiq 2F-85 & 2F-140 for e-Series Universal Robot».

[4] Universal Robot, «User Manual».

[5] B. Siciliano, L. Sciavicco, L. Villani e G. Oriolo, «Robotica modellistica, pianificazione e controllo,» McGraw-Hill, 2008.

[6] U. Robot, «Service Manual».

[7] U. Robot, «Brochure».

[8] Robotiq, «FT300 FORCE TORQUE SENSOR, quick start guide».

[9] Universal Robot, «The URScript Programming Language,» Aprile 12,2018.

[10] Robotiq, «2F-85 & 2F-140 ADAPTIVE GRIPPER, quick start guide».

[11] Robotiq, «Getting Started with Collaborative Robots, Part I: Kick-Starting Your Project».

[12] S. Bouchard, «A Guide to Making Robots Work in Your Factory».