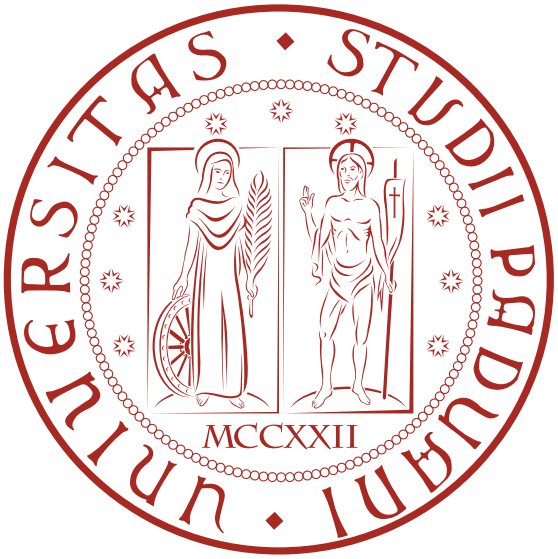


Università degli Studi di Padova
Dipartimento di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica

# Software per la modellazione ed emulazione della dinamica di robot antropomorfi

Relatore:
Prof. Enrico Pagello

Laureando:
Luca Lazzarini

Anno Accademico 2014-2015

Ai miei genitori e ai miei amici, per avermi sostenuto durante tutti questi anni.
A Silvia, per i suoi consigli e incoraggiamenti.
Ai miei coinquilini, per tutto il divertimento e l'amicizia.
Ad Anna, la mia nipotina speciale.

# Abstract

La tesi descrive lo studio per l'emulazione al computer dei movimenti PTP di robot antropomorfi. Nel caso specifico per la tesi è stato usato il robot KUKA Agilus 1100 Sixx. Lo scopo della tesi è quello di realizzare un simulatore in grado di stimare in maniera accurata il tempo necessario richiesto da un manipolatore per eseguire un movimento PTP (Point to Point).

Questo genere di informazioni permettono all'azienda che usa il robot di ottimizzare il tempo/ciclo, scegliendo i movimenti più veloci.

La tesi è stata realizzata in collaborazione con la ditta Euclid Labs, che ha messo a disposizione le conoscenze su robot e programmazione, oltre che fornire un robot reale sul quale effettuare le prove.

Il lavoro della tesi si concentra principalmente nello studio del manipolatore fornito, al fine di realizzare un modello da implementare nel simulatore della ditta.

L'obiettivo è quello di fornire un software per modellare un manipolatore senza richiedere la costruzione di un modello dinamico.

# Indice

# Elenco delle figure

# Capitolo 1

# Introduzione

Nell'ambito della realizzazione di sistemi industriali automatizzati, uno degli aspetti fondamentale risiede nell'ottimizzazione del tempo ciclo, ovvero eseguire le operazioni richieste nel minor tempo possibile. Questa tempistica dipende sia dal tipo di lavoro da svolgere, sia dal robot utilizzato.

Per conoscere quanto tempo effettivamente una certa procedura impiega è necessario utilizzare un robot reale, programmarlo ed eseguire il codice al fine di calcolare il tempo totale. Spesso però non si ha sempre a disposizione un robot, quindi nasce la necessità di emulare al computer il funzionamento del robot, al fine di avere una stima più o meno affidabile di quanto tempo è necessario per compiere un movimento.

Questa tesi si propone di studiare il comportamento di un robot al fine di costruire un modello affidabile per emulare al computer il comportamento del robot reale. Questo modello non è un modello dinamico, quindi non tiene in considerazione delle forze che agiscono sul sistema per effettuare il movimento.

In un modello dinamico è necessario conoscere tutte le forze in gioco nel sistema, l'inerzia del sistema stesso (il robot in questo caso) assieme alla catena cinematica che lega le velocità alle forze in gioco.

L'obiettivo di questa tesi è di realizzare un modello accurato utilizzando dei metodi diversi dai modelli dinamici.

Il sistema proposto, per quanto semplice, si pone di ovviare questo problema, cercando di ottimizzare la parte di realizzazione, rendendola uniforme per ogni robot utilizzato.

Verrà innanzitutto effettuata un'analisi del robot, al fine di comprenderne le caratteristiche. In seguito si studierà il comportamento del robot nel movimento PTP (punto-punto) sincrono completo.

La tesi è stata realizzata in collaborazione con la ditta Euclid Labs, la quale ha messo a disposizione le proprie conoscenze sui robot e programmazione, utili in questo progetto.


Figura 1.1: Logo Euclid Labs

Euclid Labs, è una ditta che progetta e sviluppa soluzioni ad alta tecnologia per robotica e automazione industriale.

Fondata nel 2005, con lo scopo di incrementare l'efficienza di sistemi robotici riducendo il tempo di programmazione e aggiungendo capacità adattive.

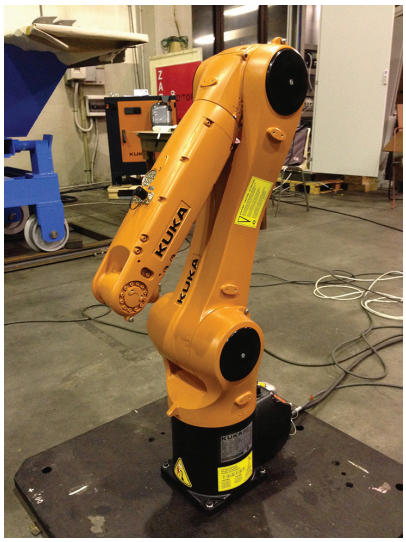


Figura 1.2: KUKA Agilus 1100 Sixx WP

## 1.1 Modello dinamico

Un modello dinamico di un manipolatore comprendo uno studio comprensivo di tutte le forze che agiscono sul sistema per creare movimento.

A differenza della cinematica infatti, la quale considera solo le velocità agenti nel sistema, il modello dinamico considera anche le forze necessarie per creare il movimento.

Dallo studio di questo sistema è quindi possibile sapere come si comporterà il manipolatore reale.

Per costruire un modello dinamico di un manipolatore è necessario conoscere la catena cinematica, assieme alle proprietà fisiche del manipolatore, quali la massa e i tensori di inerzia dei singoli assi.

In letteratura sono presenti due metodi per calcolare il modello dinamico di un manipolatore, il metodo di Eulero-Lagrange, o il metodo di Newton-Eulero. Nel metodo di Eulero-Lagrange, il manipolatore viene considerato come un pezzo unico e il sistema viene analizzato basandosi sull'energia cinetica e potenziale. Nel metodo di Newton-Eulero, invece, ogni link del manipolatore viene considerato singolarmente. Vi è una forward recursion che descrive il moto lineare e angolare, in seguito, la backward recursion, che considera forze e coppie agenti sui singoli link.

Entrambi i metodi portano alla realizzazione del seguente sistema:

$$M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) = \tau \tag{1.1}$$

ove, $M(\Theta)$ è una matrice $n \times n$ delle masse dei giunti, $V(\Theta, \dot{\Theta})$ è un vettore $n \times 1$ contenente i termini di forza centrifuga e di Coriolis, mentre $G(\Theta)$ è un vettore $n \times 1$ contenente i termini di gravità, infine $\tau$ è il vettore delle coppie ai giunti. Questa espressione viene spesso chiamata di stato-spazio per via del termine $V(\Theta, \dot{\Theta})$, il quale contiene termini che dipendono sia dallo stato che dalle velocità. Ogni elemento di $M(\Theta)$ e $G(\Theta)$ è una funzione complessa che dipende da $\Theta$, le posizioni di tutti i giunti. Ogni elemento di $V(\Theta, \dot{\Theta})$ è una funzione complessa di $\Theta$ e $\ddot{\Theta}$.

Nello specifico, il sistema inverso risulta di particolare interesse. Infatti, a partire dai valori di coppie $\tau$, angoli $\Theta$ e velocità $\dot{\Theta}$, è possibile risalire alle accelerazioni $\ddot{\Theta}$ dei singoli assi.

$$\ddot{\Theta} = M(\Theta)^{-1}[\tau - V(\Theta, \dot{\Theta}) - G(\Theta)] \tag{1.2}$$

Da queste accelerazioni è quindi poi possibile ottenere il tempo necessario per spostare il robot da una posizione A ad una posizione B.

Sebbene completo, questo modello non risulta utile ai fini della tesi, dato che l'obiettivo è quello di realizzare un simulatore senza dover ricorrere ad un'analisi approfondita del modello dinamico del robot stesso.

Per stimare il movimento dal modello dinamico è infatti necessario conoscere i valori, istante per istante, di coppie, angoli e velocità angolari di ogni asse coinvolto nel movimento.

Per la ditta è necessario però un simulatore nel quale questi valori non debbano essere specificati, ma anzi, che richieda in ingresso solo i valori dei giunti che corrispondono alla posizione di partenza e di arrivo. Un modello di questo genere quindi non può basarsi su un sistema dinamico, visto che senza i valori citati sopra, non sarebbe possibile stimare il tempo.

## 1.2 Movimento PTP

In questa tesi verrà trattato solamente il movimento PTP del robot. Questo movimento risulta il più semplice di tutti, ma anche il più veloce da eseguire.

Il robot quindi si muove tra i due punti per il percorso più veloce, che spesso differisce dal percorso più breve, quindi non rappresentabile da una linea.

Data la struttura rotativa dei giunti del robot, percorsi curvi vengono eseguiti più velocemente di percorsi lineari.

Va specificato che il percorso finale del robot non può essere previsto.

Dato che il punto finale in una traiettoria non è raggiunto finché tutti gli assi non hanno raggiunto il valore di angolo richiesto, ci sarà un asse che impiegherà più tempo degli altri. Questo asse viene chiamato *leading axis*. Per ridurre l'abrasione tra i giunti, tutti i giunti, escluso il *leading axis*, eseguiranno il loro movimento ad una velocità inferiore della loro velocità massima. Così facendo, il tempo per raggiungere la posizione di destinazione risulta sincronizzato.

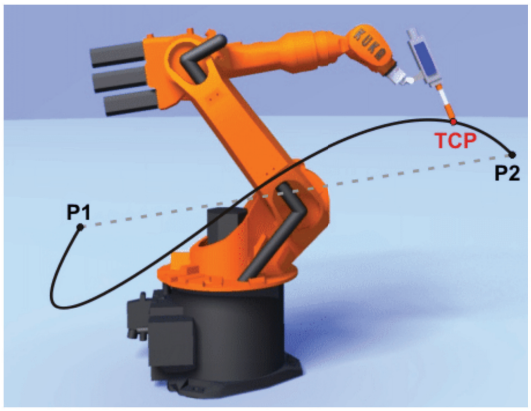


Figura 1.3: Esempio movimento PTP

### 1.2.1 PTP sincrono

In figura 1.4 è possibile vedere l'andamento delle velocità di una coppia di giunti nel caso venga eseguito il movimento PTP sincrono.

In questo caso il tempo di esecuzione di ogni movimento viene adattato in modo da terminare in sincrono con gli altri assi.

Alcuni assi vengono deliberatamente rallentati.

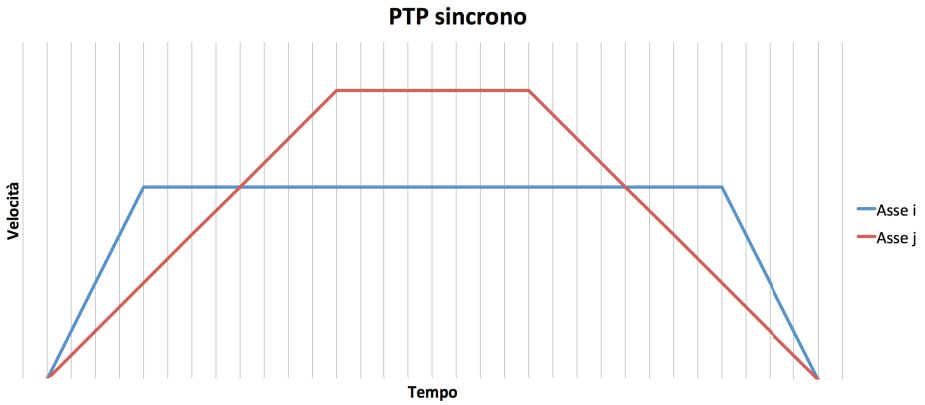


Figura 1.4: PTP sincrono

In questo caso, l'asse j risulta il *leading axis*, l'asse i è viene rallentato, per terminare in sincronia con l'asse j.

### 1.2.2 PTP completamente sincrono

A differenza del PTP sincrono, nel movimento PTP completamente sincrono vengono sincronizzate anche le accelerazioni e decelerazioni. Questo porta un'ulteriore riduzione dell'abrasione dei giunti, aumentando la vita del robot.

In figura 1.5 è possibile come viene modificata l'accelerazione e decelerazione dei vari assi.

Dalle prove effettuate sul robot, si è visto che il KUKA Agilus esegue i movimenti PTP in maniera completamente sincrona.

## 1.3 Struttura della tesi

Il capitolo 1 introduce brevemente il modello dinamico, nella sua formulazione matematica. Viene inoltre descritto il tipo di movimento utilizzato dal robot per questa tesi, ovvero il movimento PTP (Point to Point), nella versione sincrona e sincrona completa.

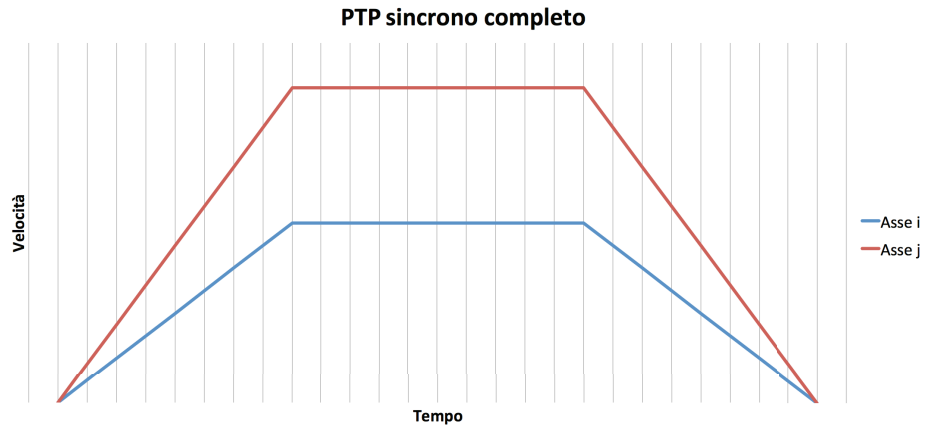


Figura 1.5: PTP sincrono completo

Il capitolo 2 mostra le caratteristiche del manipolatore utilizzato nella tesi, assieme ad alcune informazioni sulla ditta produttrice e la ditta presso la quale è stato effettuato questo lavoro di tesi.

Il capitolo 3 spiega come è stato realizzato il primo modello che simula il funzionamento del movimento PTP introdotto nel capitolo 2. Vengono elencate poi le problematiche dovute all'utilizzo di questo tipo di approccio.

Il capitolo 4 espone un metodo alternativo di approcciarsi al problema, spiegandone l'idea di fondo e il funzionamento.

Il capitolo 5 contiene i risultati dei test effettuati, prima esposti singolarmente, poi confrontando i due algoritmi.

Il capitolo 6 è dedicato alle conclusioni a cui si è giunti nella realizzazione di questa tesi. Verranno inoltre introdotti possibili sviluppi futuri per le idee proposte.

# Capitolo 2

# Robotica e manipolatore utilizzato

In questo capitolo verrà introdotto brevemente il robot utilizzato nel corso di questa tesi, assieme ad alcune informazioni della ditta produttrice.

Infine si descriverà brevemente l'ambiente di lavoro utilizzato per la realizzazione di questa tesi.

## 2.1 Storia della robotica

I primi robot ad essere utilizzati in ambito industriale sono comparsi negli anni settanta nel mercato automobilistico americano. Questi primi robot avevano una struttura in acciaio e dei motori idraulici, molto lenti e imprecisi.

I gradi di libertà e le capacità di carico di questi robot sono molto limitate.

La prima ditta italiana ad adottare i robot in fabbrica fu la FIAT, la quale ha iniziato ad utilizzare i robot per la saldatura delle carrozzerie con il Robogate, un'invenzione completamente italiana che è stata in seguito copiata da tutte le maggiori ditte automobilistiche del mondo.

Il Robogate, introdotto nel 1978, consisteva da robot adibiti alla saldatura, assieme a dei telai a forma di portale (gate in inglese) per posizionare i vari organi della scocca da saldare.

A guidare il Robogate è adibito un computer che regola il traffico dei carrelli (Robocarrier), sui quali vengono trasportate le scocche, sia il ciclo di saldatura.

La sequenza produttiva inizia quando il carrello, prelevati nel magazzino il tetto e le fiancate da saldare, si sposta nella prima stazione dove avviene il bloccaggio della struttura tramite morsetti.

Inizialmente i robot effettuano delle saldature in punti strategici, in seguito vengono completate tutte le saldature necessarie per realizzare la scocca.

Terminata la saldatura, il Robocarrier scarica la scocca e inizia un nuovo ciclo di produzione.

La prerogativa del Robogate, oltre ad un efficienza elevata, era la flessibilità. Era infatti possibile cambiare tipo di saldatura in maniera molto veloce, permettendo di seguire velocemente i cambiamenti del mercato, il tutto con un investimento di appena il dieci percento di quello iniziale.

L'evoluzione dei robot industriali ha portato alla creazione di robot antropomorfi, con sempre più gradi di liberà, precisione, velocità e capacità di carico, permettendo così di effettuare operazioni di foratura, smerigliatura, fresatura, verniciatura e tagli con il laser.

I robot sono particolarmente utili anche in caso di stoccaggio, specialmente in aree di lavoro difficili.

In Italia è presente una delle ditte più grandi nel mondo della robotica industriale, la Comau.

## 2.2 KUKA

KUKA (**K**eller **u**nd **K**nappich **A**ugsburg) è un produttore tedesco a livello mondiale di robot industriali e per soluzioni per l'automazione industriale.

Fondata nel 1898 ad Amburgo, Germania, fondata da Johann Josef Keller e Jacob Knappich. Ha iniziato a produrre luci per case e strade, ma molto presto si è estesa a produrre altri tipi di prodotti (equipaggiamenti e soluzioni per saldature, grandi container), fino a diventare leader del mercato in veicoli commerciali nel 1966.

Nel 1973 ha creato il suo primo robot FAMULUS.

Le aree di applicazione dei robot KUKA sono le seguenti:

1. Industria del trasporto: trasporto di carichi pesanti, dove vengono usati per le loro capacità di spostare liberamente carichi pesanti.

2. Industria alimentare: caricare e scaricare di macchine per imballaggi, pallettizzare, tagliare e controllo qualità.

3. Industria delle costruzioni: per assicurare un flusso continuo di materiali.

4. Industria del vetro: utilizzati nel trattamento a caldo di vetro o quarzo.

5. Fonderie: la resistenza alle alte temperature permette al robot di lavorare senza problemi vicino a fonti di calore.

6. Industria del legno: utilizzati ad esempio per tagliare, perforare e pallettizzare.

7. Processi metallurgici: quali fresatura, piegatura, saldatura, assemblamento, carico e scarico.

## 2.3 KUKA Agilus 1100 Sixx

Robot industriale di piccole dimensioni, diventato famoso per il video che lo ritrae protagonista nella sfida contro il campione tedesco Timo Boll in una partita di table tennis [16].

Progettato per raggiungere velocità elevate, a discapito di un payload ridotto. Questo robot a differenza del suo predecessore può operare anche all'esterno, grazie ad un rivestimento a prova d'acqua.

Si è utilizzato questo robot nello svolgimento di questa tesina per via delle sue dimensioni ridotte, che hanno permesso uno studio approfondito del robot stesso.

### 2.3.1 Caratteristiche

Il manipolatore è un braccio a 6 assi fatto in lega pressofusa, con rivestimento sigillato per proteggere i componenti interni dall'infiltrazione di acqua. Inoltre il manipolatore è pressurizzato per prevenire la penetrazione di umidità e polvere. Ognuno dei sei assi è munito di freno.

Il robot è composto dai seguenti componenti, visibili in fig 2.1:

1. Polso in linea
2. Braccio
3. Link del braccio
4. Colonna rotante
5. Componenti elettriche
6. Base

### 2.3.2 Dati tecnici

I dati tecnici del KUKA Agilus 1100 Sixx sono elencati in tab 2.3.2.

L'area di lavoro invece è definita dalla grandezza del robot e dai valori che possono assumere i vari giunti. L'area di lavoro del KUKA Agilus è rappresentata in figura 2.2.

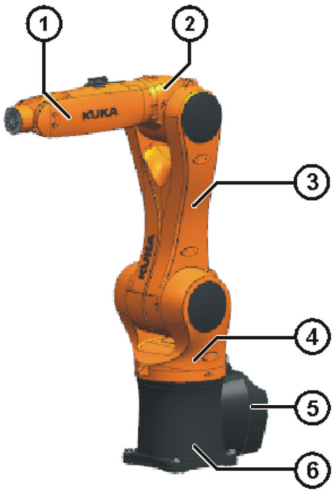


Figura 2.1: Componenti principali

Il robot è vincolato fisicamente (come pure via software) dal raggiungere determinati valori di angoli. I valori massimi sono indicati in tabella 2.3, assieme alle velocità massime che ogni singolo può raggiungere.

## 2.4 Simulatore Euclid Labs

Il simulatore fornito dalla ditta Euclid Labs è composto da un interfaccia grafica composta da una finestra che visualizza il rendering del manipolatore (realizzato dalle informazioni CAD), assieme ad alcuni gadget per cambiare la posizione attuale del robot, specificando i valori degli angoli dei singoli giunti.

Il simulatore è scritto in C# e usa delle librerie proprietarie di Euclid Labs per generare le traiettorie.

Per la tesi si è lavorato su questo framework, espandendo il simulatore con nuove funzionalità per permettere di confrontare il modello realizzato con il robot reale.

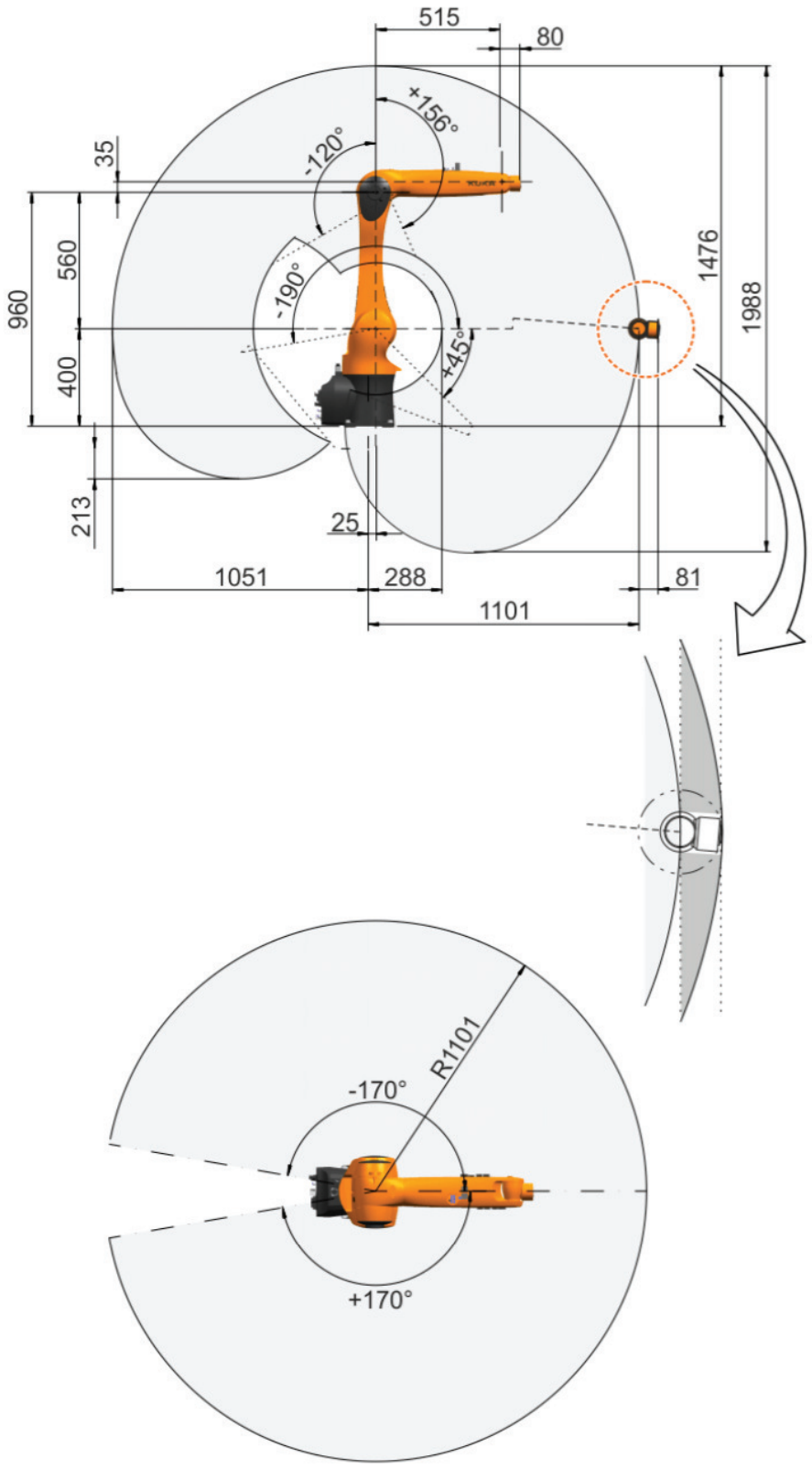


Figura 2.2: Spazio di lavoro

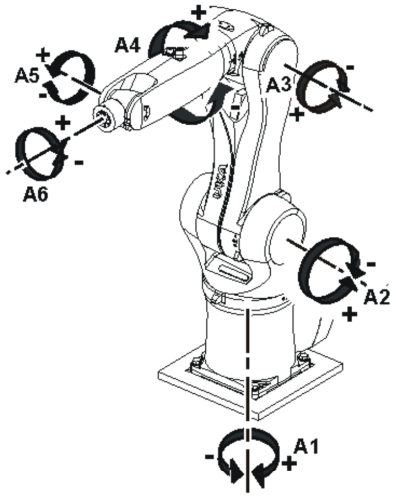


Figura 2.3: Assi di rotazione

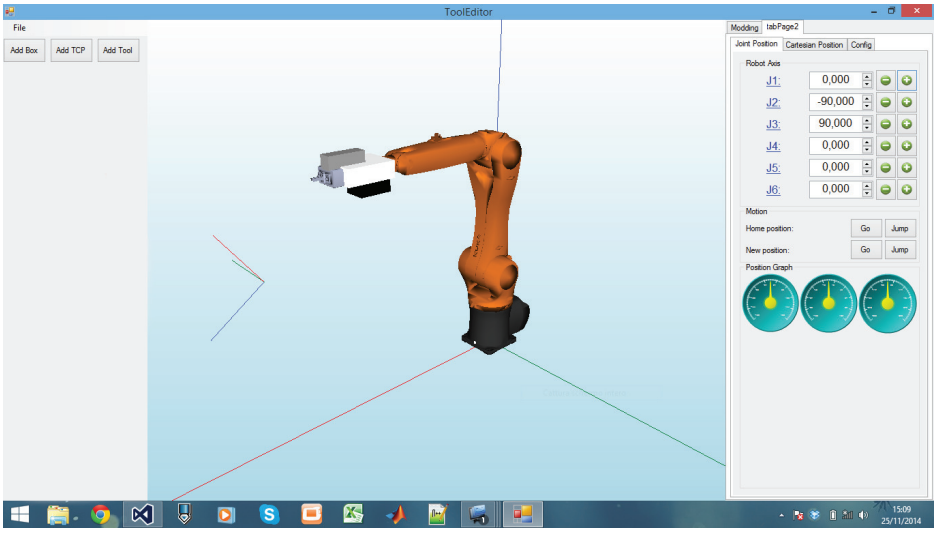


Figura 2.4: Schermata del simulatore

| | |
|---|---|
| **Polso in linea** | Il robot è composto da un polso in linea da tre assi. |
| **A4, A5, A6** | Comprende gli assi 4, 5 e 6 |
| **Braccio A3** | Rappresenta il link tra il polso in linea e il braccio. Il braccio è mosso dal motore dell'asse 3. |
| **Braccio A2** | Collega il braccio alla colonna rotante. Incorpora il motore e il riduttore dell'asse 2. |
| **Colonna rotante A1** | La colonna rotante incorpora i motori degli assi 1 e 2. |
| **Base** | E' la base del robot. Comprende le componenti elettroniche e l'interfacciamento con il cabinet. |

Tabella 2.1: Componenti manipolatore

| | |
|---|---|
| **Distanza massima** | 1101 mm |
| **Ripetibilità** | $\leq \pm 0,03mm$ |
| **Peso** | 54 kg |
| **Montaggio** | Pavimento, soffitto, muro |
| **Classe di protezione** | IP 67 |
| **Payload nominale** | 5 kg |
| **Payload massimo** | 10 kg |
| **Distanza del centro di gravità del carico $L_{XY}$** | 100 mm |
| **Distanza del centro di gravità del carico $L_Z$** | 80 mm |
| **Carico massimo totale** | 10 kg |
| **Carico supplementare** | La somma di tutti i carichi del robot non può eccedere il valore di carico massimo totale |

Tabella 2.2: Dati tecnici dal manuale

| | **Angolo min (°)** | **Angolo max (°)** | **Vel. max (°/s)** |
|---|---|---|---|
| **A1** | -170 | 170 | 300 |
| **A2** | -190 | 45 | 225 |
| **A3** | -120 | 156 | 225 |
| **A4** | -185 | 185 | 381 |
| **A5** | -120 | 120 | 311 |
| **A6** | -350 | 350 | 492 |

Tabella 2.3: Vincoli software robot

# Capitolo 3

# Modello PTP sincrono completo

In questo capitolo verrà introdotto il primo simulatore proposto. Il suo funzionamento si basa sul comportamento del robot nel caso di movimento PTP, la descrizione di questo tipo di movimento è stata introdotta nella sezione 1.2.2. Questo simulatore ha il vantaggio di essere molto semplice e di facile implementazione per qualsiasi tipo di robot. Necessita solamente delle velocità e accelerazioni massime di ogni giunto, i quali valori sono di semplice calcolo.

Questo algoritmo non tiene conto degli aspetti fisici e dinamici del robot, i quali influenzeranno negativamente le prestazioni, come era lecito aspettarsi.

Nella sezione 3.5 invece verranno presentate le problematiche di questo algoritmo, sia quelle dinamiche e inerziali, sia quelle relative al funzionamento del robot stesso.

## 3.1 Realizzazione

L'idea di base del funzionamento di questo algoritmo si basa sul funzionamento teorico del PTP completamente sincrono, ovvero la sincronizzazione di velocità e accelerazioni tra i singoli assi della quale era stato parlato nell'introduzione (1.2.2).

Da questa base teorica, conoscendo quale sarà il *leading axis* è quindi possibile poter stimare il tempo richiesto dal robot per effettuare lo spostamento.

Calcolato lo spostamento in gradi di ogni asse, è quindi possibile calcolare il *leading axis* semplicemente valutando quanto tempo è necessario per un singolo asse per cambiare la sua configurazione del numero di gradi richiesto. L'asse che impiegherà più tempo sarà il *leading axis*.

| | **Punto A** | **Punto B** | **Differenza** |
|---|---|---|---|
| **Asse 1 (°)** | 0 | 40 | 40 |
| **Asse 2 (°)** | -90 | -30 | 60 |
| **Asse 3 (°)** | 90 | 60 | 30 |
| **Asse 4 (°)** | 0 | 0 | 0 |
| **Asse 5 (°)** | 0 | 30 | 30 |
| **Asse 6 (°)** | 0 | 0 | 0 |

Tabella 3.1: Differenza di gradi tra due punti (in gradi)

## 3.2 Calcolo dei tempi

Il funzionamento del PTP sincrono completo impone che i tempi di rotazione di tutti gli assi siano uguali, ovvero:

$$t_{e,1} = t_{e,2} = t_{e,3} = t_{e,4} = t_{e,5} = t_{e,6} = t_{e,max} \tag{3.1}$$

Partendo dalle velocità e accelerazioni di ogni singolo asse, viene calcolata le velocità massima che può essere raggiunta nell'intervallo di gradi che è stato richiesto, ovvero:

$$\tilde{v}_{m,i} = min(\hat{v}_{m,i}, \sqrt{\hat{x}_{e,i}\hat{a}_{m,i}}) \tag{3.2}$$

Da questo valore, per ogni asse vengono calcolate, il tempo provvisorio di accelerazione per ogni asse $\tilde{t}_{a,i}$, il tempo provvisorio di fine movimento $\tilde{t}_{e,i}$ e il tempo iniziale provvisorio di decelerazione $\tilde{t}_{d,i}$:

$$\tilde{t}_{a,i} = \frac{\tilde{v}_{m,i}}{\tilde{a}_{m,i}} \tag{3.3}$$

$$\tilde{t}_{e,i} = \frac{\hat{x}_{e,i}}{\tilde{v}_{m,i}} + \tilde{t}_{a,i} \tag{3.4}$$

$$\tilde{t}_{d,i} = \tilde{t}_{e,i} - \tilde{t}_{a,i} \tag{3.5}$$

Data la necessità di sincronizzare i movimenti tra gli assi, si cercano i valori massimi, ovvero:

$$t_{e,max} = max(\tilde{t}_{e,1}, \tilde{t}_{e,2}, \tilde{t}_{e,3}, \tilde{t}_{e,4}, \tilde{t}_{e,5}, \tilde{t}_{e,6}) \tag{3.6}$$

$$t_{a,max} = max(\tilde{t}_{a,1}, \tilde{t}_{a,2}, \tilde{t}_{a,3}, \tilde{t}_{a,4}, \tilde{t}_{a,5}, \tilde{t}_{a,6}) \tag{3.7}$$

$$t_{d,max} = max(\tilde{t}_{d,1}, \tilde{t}_{d,2}, \tilde{t}_{d,3}, \tilde{t}_{d,4}, \tilde{t}_{d,5}, \tilde{t}_{d,6}) \tag{3.8}$$

Dato che i tempi di accelerazione $t_{a,i}$, tempo di finale di decelerazione $t_{d,i}$ e tempo totale di movimento $t_{e,i}$ devono essere sincronizzati, si impone che per tutti gli assi, questi tempi corrispondano ai tempi dell'asse più lento. Ovvero:

$$t_{e,1} = t_{e,2} = t_{e,3} = t_{e,4} = t_{e,5} = t_{e,6} = t_{e,max} \tag{3.9}$$

$$t_{a,1} = t_{a,2} = t_{a,3} = t_{a,4} = t_{a,5} = t_{a,6} = t_{a,max} \tag{3.10}$$

$$t_{d,1} = t_{d,2} = t_{d,3} = t_{d,4} = t_{d,5} = t_{d,6} = t_{d,max} \tag{3.11}$$

Dalle quali è poi possibile calcolare le velocità e accelerazioni massime per ogni asse:

$$v_{m,i} = \frac{\hat{x}_{e,i}}{t_{d,max}} \tag{3.12}$$

$$a_{m,i} = \frac{v_{m,i}}{t_{a,max}} \tag{3.13}$$

In seguito viene riportata la tabella di tutti i simboli utilizzati.

## 3.3 Calcolo dei valori di velocità e accelerazione massima

Le velocità massime sono presenti nel datasheet del KUKA Agilus [14], mentre le accelerazioni sono state calcolate singolarmente per ogni singolo asse.

Da queste informazioni è quindi possibile calcolare quanto tempo è richiesto ad un singolo giunto per eseguire una certa rivoluzione.

Per semplicità si è deciso di calcolare i tempi necessari a compiere una rotazione compresa tra 1 e 100 gradi.

In figura 3.1 è presente il grafico che visualizza i tempi necessari per ogni singolo giunto per effettuare la rotazione.

| | | | |
|---|---|---|---|
| $t_a$ | tempo di accelerazione | $t_{e,max}$ | tempo massimo di fine movimento tra gli assi |
| $\tilde{t}_a$ | tempo provvisorio di accelerazione | $a_m$ | accelerazione massima |
| $t_{a,max}$ | tempo di acc. max tra gli assi | $\hat{a}_m$ | accelerazione massima predeterminata |
| $t_d$ | tempo iniziale di decelerazione | $v_m$ | rampa di velocità massima |
| $\tilde{t}_d$ | tempo iniziale provvisorio di decelerazione | $\hat{v}_m$ | rampa di velocità massima predeterminata |
| $t_{d,max}$ | tempo massimo di inizio decelerazione tra gli assi | $\tilde{v}_m$ | rampa di velocità provvisoria massima |
| $t_e$ | tempo di fine movimento | $\hat{x}_e$ | distanza predeterminata di movimento |
| $\tilde{t}_e$ | tempo provvisorio di fine movimento | | |

Tabella 3.2: Legenda

## 3.4 Funzionamento

Il funzionamento dell'algoritmo risulta molto semplice e si può riassumere nei passaggi che seguono.

Dati i valori di:

- Velocità massima
- Accelerazione massima

di ogni singolo asse, l'algoritmo valuta iterativamente per tutti gli assi del robot:

1. La differenza di gradi dalla posizione di partenza a quella di arrivo
2. Calcola il tempo necessario ad eseguire la rotazione (tramite le formule di sezione 3.2)

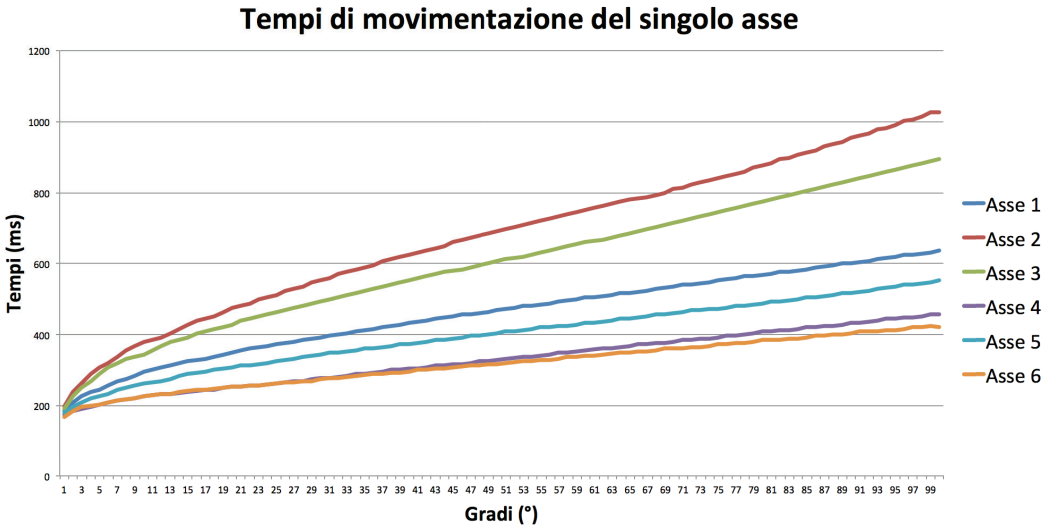


Figura 3.1: Tempi di rotazione dei singoli assi per valori di angoli crescente

Dai tempi calcolati è così possibile ricavare il tempo dell'asse più lento, tramite la formula:

$$t_{e,max} = max(\tilde{t}_{e,1}, \tilde{t}_{e,2}, \tilde{t}_{e,3}, \tilde{t}_{e,4}, \tilde{t}_{e,5}, \tilde{t}_{e,6}) \tag{3.14}$$

Il tempo stimato per percorrere una movimento PTP corrisponderà al tempo massimo richiesto dall'asse più lento.

## 3.5 Problematiche

Questo modello, per quanto valido matematicamente, non si presta bene a simulare il comportamento di un robot reale, dato che non vengono considerati due fattori importanti:

- Limitazione della velocità del tool
- Controllo inerziale

Il robot esegue il movimento tenendo sempre in considerazione queste due problematiche al fine di ottenere un movimento quanto più possibile fluido e privo di sobbalzi.

Queste problematiche verranno discusse nelle sottosezioni seguenti.

### 3.5.1 Tool

Come tool si definisce il meccanismo connesso alla flangia del robot, ovvero la parte terminale del manipolatore.

Questo varia in base al tipo di lavoro che il robot deve eseguire. Può consistere in una pinza, in una torcia per saldatura, una telecamera, un laser o qualsiasi cosa sia necessaria per eseguire il lavoro richiesto.

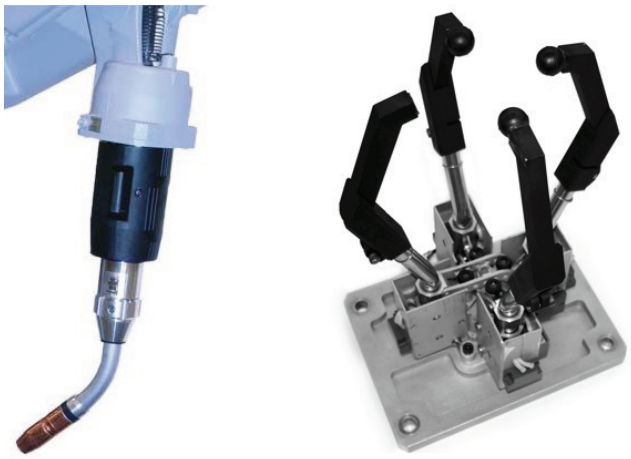

Figura 3.2: Tool, a sinistra una torcia saldatrice, a destra una presa

### 3.5.2 Limitazione alla velocità massima del tool

Il movimento del manipolatore è funzione della velocità finale del tool. Questo per evitare di sottoporre il tool stesso a dei movimenti troppo rapidi, nel caso i giunti, muovendosi in maniera concorde, aumentino in maniera troppo elevata la velocità di rotazione e traslazione del tool stesso.

Questo vincolo viene posto per evitare rotture, dovute a forze troppo elevate nel caso si voglia spostare il tool in modo molto rapido.

Questa limitazione è facilmente visibile muovendo gli assi 4 e 6 in maniera concorde o meno. I due assi infatti sono posizionati nello stesso modo, quindi una rotazione degli stessi in un verso concorde aumenterebbe la velocità di rotazione finale del tool.

In figura 3.3 sono visibili i versi di rotazione degli assi 4 e 6.

Com'è visibile dal grafico di figura 3.4, la rotazione degli assi 4 e 6 varia in base all'angolo di rotazione dei singoli.

In questo caso si vede chiaramente che le velocità, di conseguenza i tempi per effettuare la rotazione, differiscono in base alla rotazione dei due assi.

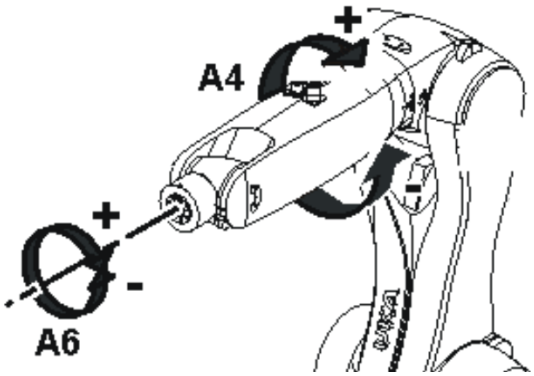


Figura 3.3: Rotazioni asse 4 e asse 6

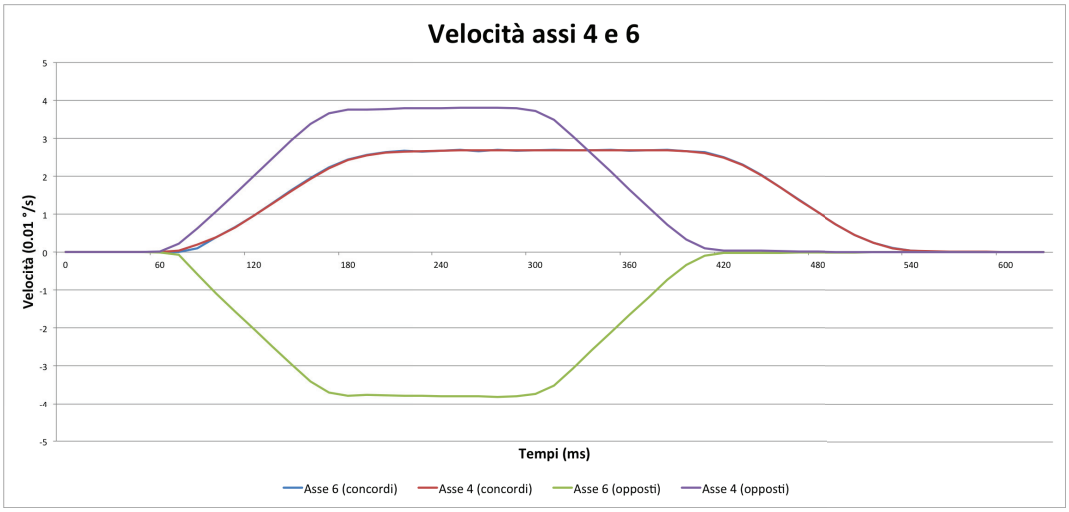


Figura 3.4: Velocità rotazione asse 4 e 6

Nel caso la rotazione sia concorde, la velocità finale di entrambi gli assi viene rallentata, al fine di non superare la velocità massima di rotazione del tool.

La velocità di rotazione massima del tool non è specificata nel manuale dell'Agilus, ma è stato comunque possibile stimarla dai dati. Risulta infatti di circa 540°/s.

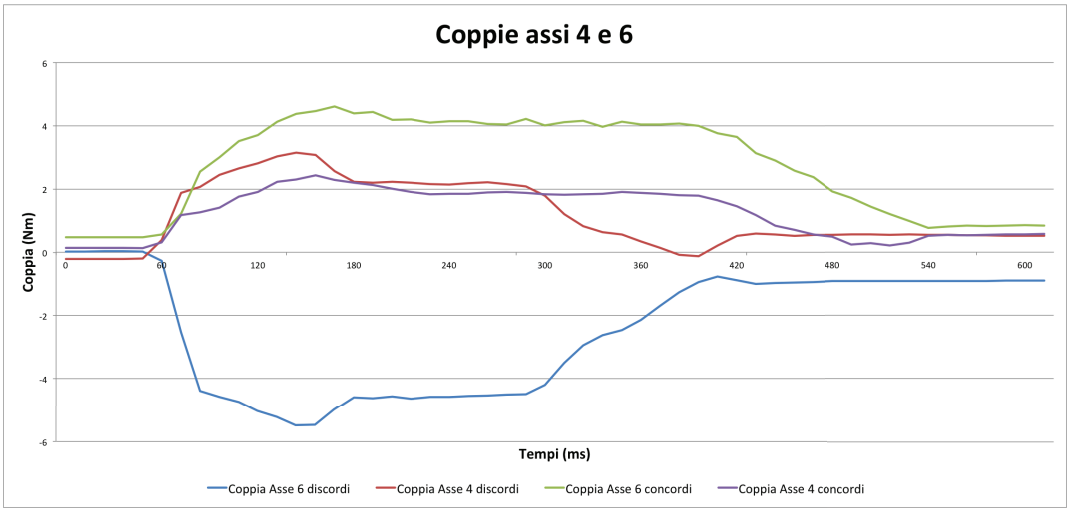


Figura 3.5: Coppie rotazione asse 4 e 6

Come si può vedere nel grafico di figura 3.5, le coppie necessarie per movimentare gli assi differiscono nei due movimenti. Nel caso la rotazione avvenga nello stesso verso, i valori delle coppie (soprattutto per l'asse 6), sono inferiori al caso in cui la rotazione avvenga in versi opposti.

Per quanto riguarda la velocità massima di traslazione del tool, non è stato possibile trovare un valore massimo, in quanto nonostante le varie prove effettuate non si è notato un troncamento della velocità degli assi, che non dipendesse dall'asse più lento.

Si può comunque presumere che gli sviluppatori abbiano inserito una velocità di traslazione massima, anche se con questo robot, nonostante la sua velocità, non viene mai raggiunta.

### 3.5.3 Rallentamento dovuto al controllo inerziale

Un'altra limitazione del modello PTP base risiede nel fatto che nel modello, l'inerzia del robot non viene tenuta in considerazione. Il robot ha una sua

inerzia non indifferente, nonostante le dimensioni ridotte, ha comunque un peso di 55 chilogrammi.

Questo fa si che alcuni movimenti siano parecchio influenzati dalla posizione nella quale si trova il robot. Più in generale, la velocità del robot viene pesantemente condizionata dalla posizione del centro di massa.

Nel caso questi sia raccolto su se stesso, può raggiungere velocità più elevate al caso in cui risulti completamente allungato e parallelo al terreno.

Un'altra casistica fondamentale sta nel tipo di movimento. Nel caso gli venga richiesto di estendersi fino a raggiungere una posizione completamente estesa, il centro di massa si sposterà di molto dalla base del robot, richiedendo uno sforzo maggiore ai motori per contrastare la forza di gravità.

Il controllo numerico del robot genera una traiettoria quanto più pulita e priva di sbalzi. Per ottenere questo, gestisce il valore di coppie degli assi, adattandoli in base alla posizione del robot e velocità richiesta.

Questo si può chiaramente vedere nel grafico di figura 3.6, la velocità aumenta in modo molto più lento rispetto ai grafici visti precedentemente, mentre la decelerazione risulta molto più brusca dell'accelerazione.

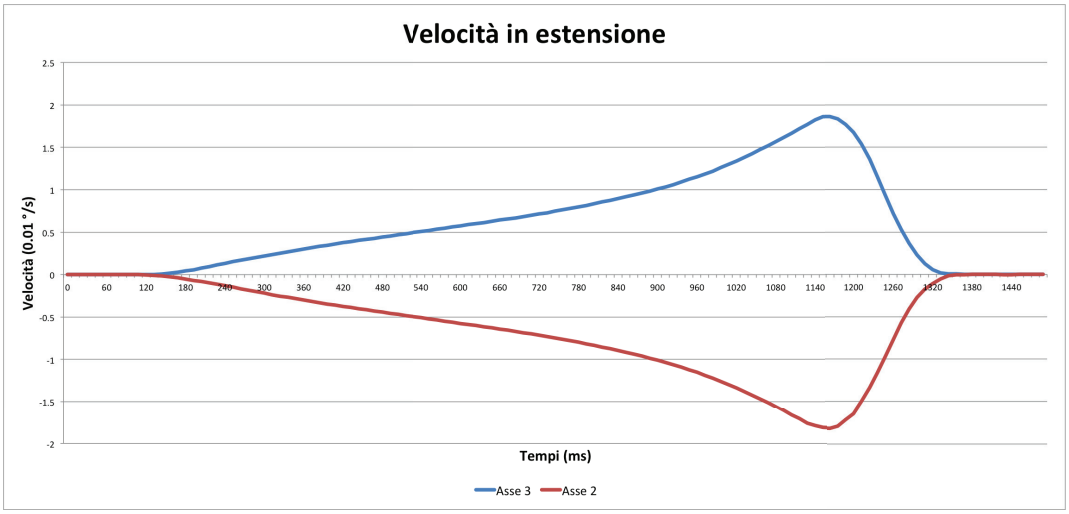


Figura 3.6: Velocità in estensione

Lo stesso si verifica nel caso si voglia ruotare il robot attorno l'asse 1. I tempi, come è lecito aspettarsi, cambiano nel caso il robot sia allungato (figura 3.8) o raccolto in se stesso (figura 3.7).

Il grafico in figura 3.9 mostra l'andamento della velocità dell'asse 1 nel caso il robot sia raccolto, oppure completamente steso.

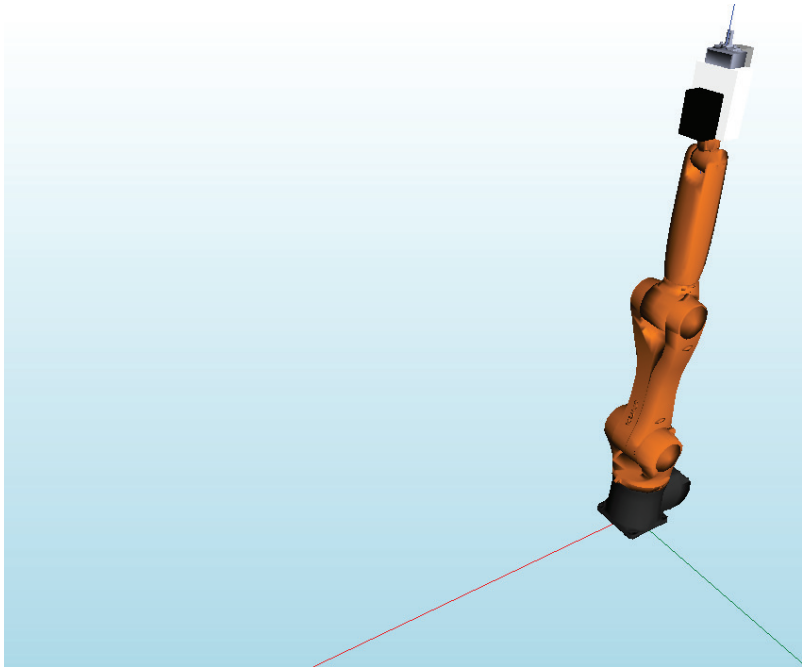

Figura 3.7: Rotazione asse 1 con manipolatore posto nella condizione di minore inerzia

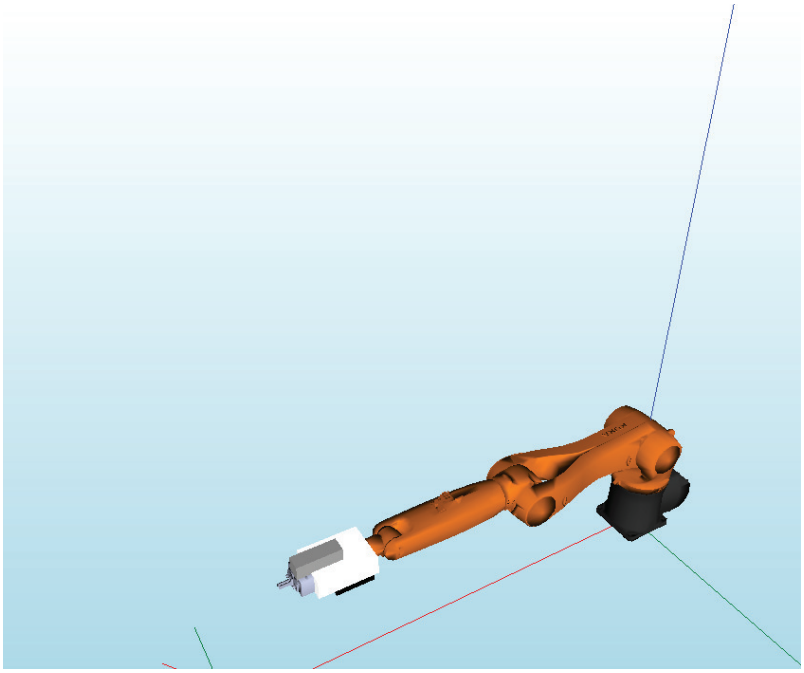

Figura 3.8: Rotazione asse 1 con manipolatore posto nella condizione di maggiore inerzia

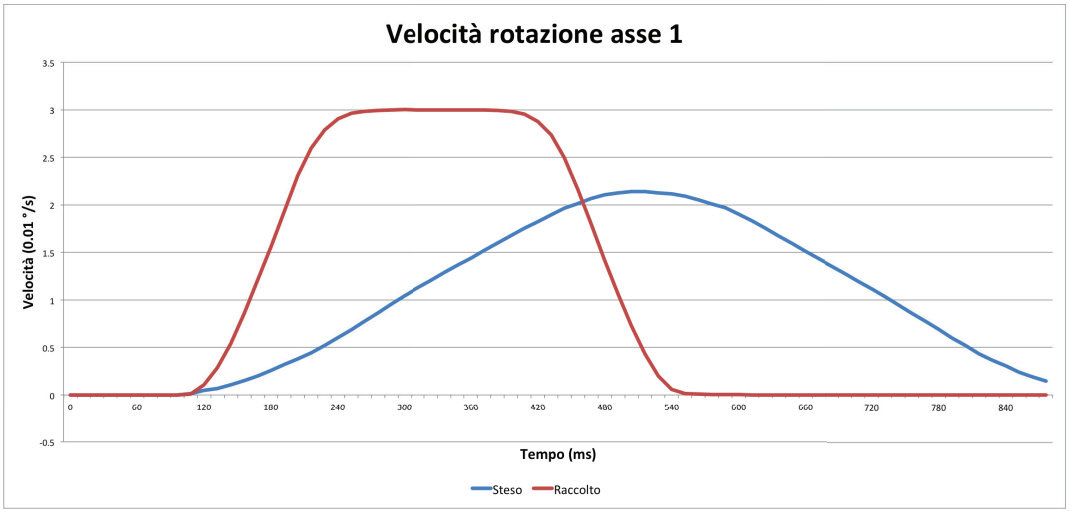


Figura 3.9: Velocità asse 1

Come si può notare, nel caso il robot si trovi in una posizione in cui il braccio risulta completamente steso, la velocità massima che riesce a raggiungere risulta minore di quella raggiunta nel caso il robot sia raccolto.

Dal grafico delle coppie è possibile vedere meglio cosa succede nel robot.

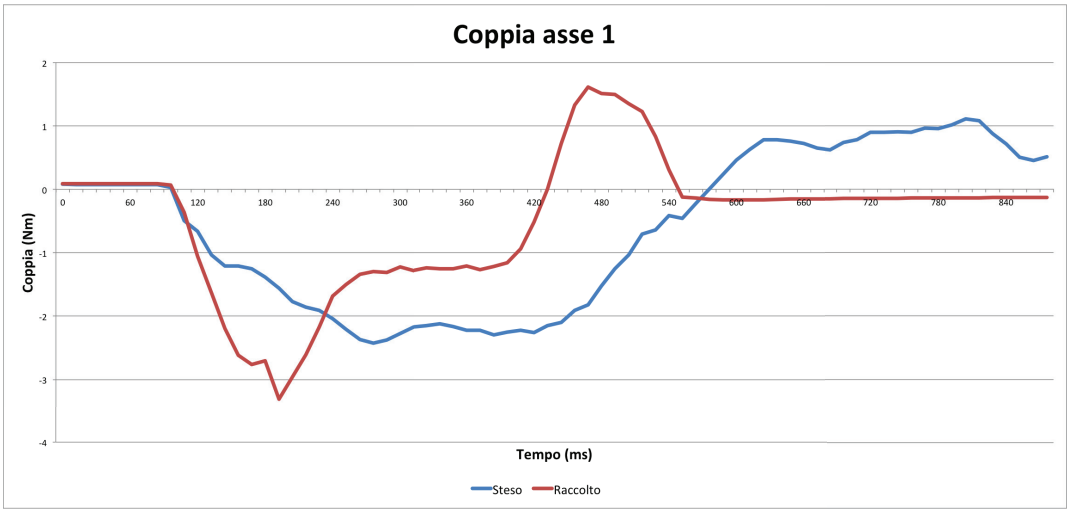


Figura 3.10: Coppia asse 1

In figura 3.10, è possibile vedere l'andamento della coppia applicata al motore dell'asse 1 durante i due movimenti.

Nel caso questi sia raccolto, la coppia applicata risulta maggiore rispetto l'altro caso. Questo perché il controllo di inerzia stima che è possibile eseguire il movimento richiesto senza generare dei sobbalzi durante la traiettoria.

Nel caso in cui il braccio risulta sbracciato, un'applicazione eccessiva di coppia comporterebbe innanzitutto un usura superiore degli assi, oltre ad un sobbalzo del braccio durante l'accelerazione o la decelerazione.

Data la necessità di mantenere un movimento quanto più possibile fluido, il controllo di inerzia gestisce questi movimenti riducendo la potenza applicata ai motori.

### 3.5.4 Discrepanze rispetto al modello PTP ideale

Un caso particolare, del quale è difficile predirre il comportamento è visualizzato in figura 3.11. In questi due casi infatti, vengono fatti ruotare l'asse 1 e l'asse 4 di 90°. L'asse 1 ruota da 0° a 90°, mentre l'asse 4 ruota da 0° a 90° in una prova, nell'altra ruota in senso opposto, ovvero da 0° a -90°.

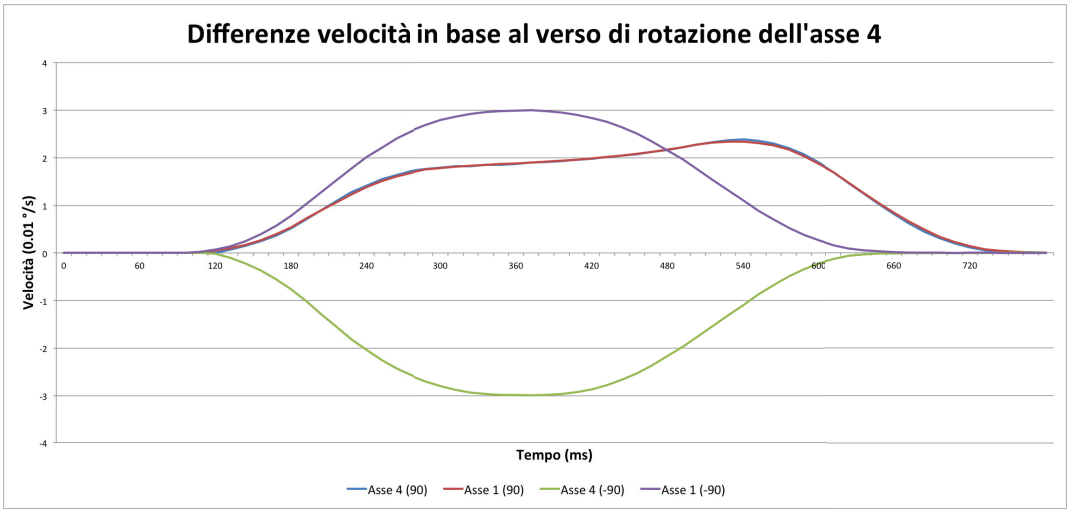


Figura 3.11: Velocità rotazione asse 1 e 4 in base al verso di rotazione dell'asse 4

Dalle formule matematiche di questo capitolo ci si aspettava che il tempo di movimento dipendesse solamente dalla velocità di rotazione dei singoli assi, quindi in questo caso ci si aspettava il *leading axis* fosse l'asse 1.

Il sistema si comporta in maniera adeguata nel caso l'asse 4 venga fatto ruotare da 0° a -90°. Mentre nel caso opposto, la velocità di rotazione risulta inferiore, con un andamento decisamente diverso dal caso precedente.

Questo è riconducibile alla composizione interna del robot. La causa va ricercata nella costruzione del robot, ovvero motori e riduttori.

Una condizione simile si è verificata anche in un'altra configurazione. In questo caso asse 1 e asse 4 venivano ruotati dello stesso numero gradi e verso di rotazione. Nelle tre configurazioni cambiava la posizione dell'asse 5. Nelle tre prove l'asse 5 è stato posto fisso a -90°, 0° e 90°.

In questa configurazione ci si attendeva un comportamento molto simile in tutte tre le combinazioni, dato che l'asse 5 non influisce nel PTP classico (non esegue rotazione).

I risultati ottenuti però sono stati diversi da quanto ci si aspettava. In figura 3.12 è visibile l'output delle velocità degli assi 1 e 4 durante l'esecuzione del movimento.

Si nota chiaramente che la sola posizione dell'asse 5 influisce il risultato finale. Nel caso l'asse 5 abbia un angolo di 0° o 90° i tempi (come anche le velocità) risultano mediamente allineate, anche se dall'andamento delle velocità si può notare chiaramente che il controllo inerziale rallenta la rampa di

accelerazione, come pure la velocità massima che i giunti possono raggiungere.

Nel caso invece l'asse 5 sia posizionato a -90°, il tempo risulta decisamente inferiore agli altri due casi, e come si può notare dal grafico delle velocità, il controllo inerziale non interviene per modificare le accelerazioni e velocità dei giunti.

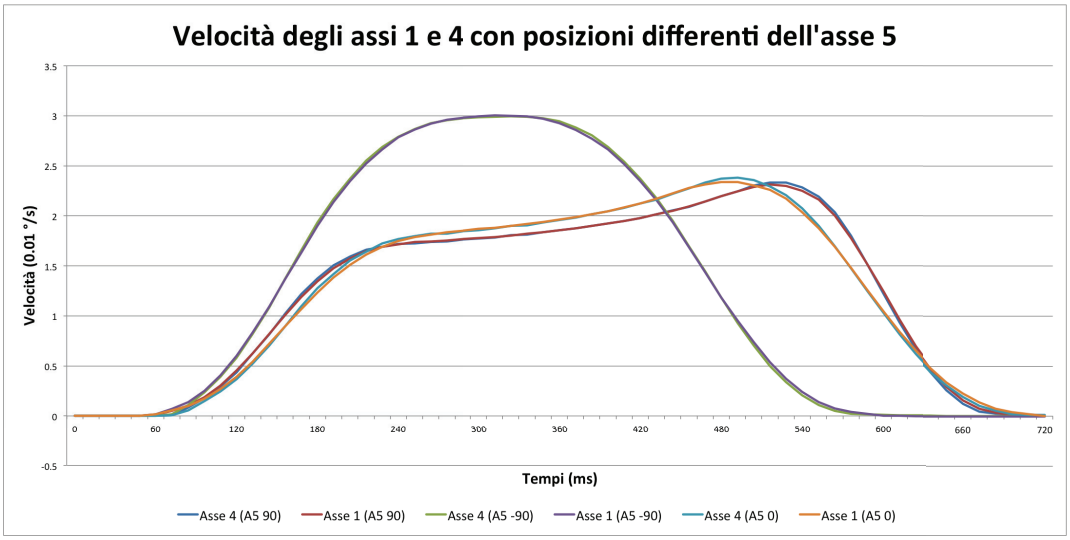


Figura 3.12: Velocità rotazione asse 1 e 4 in base all'angolo dell'asse 5

## 3.6 Conclusioni

Dai risultati ottenuti si può notare che simulare il comportamento PTP di un robot attraverso delle formule matematiche non risulta particolarmente accurato.

Questo perché non è facile conoscere a priori tutti i movimenti che vengono influenzati dal controllo d'inerzia o dal controllo di velocità del tool.

Alcuni casi sono di semplice individuazione, ad esempio i casi in cui il robot risulta steso parallelamente al pavimento.

Ancora più difficile risulta individuare i casi come quelli citati nella sottosezione 3.5.4. In quei movimenti non ci si attendeva una limitazione così marcata del controllo di inerzia, dato che la dinamica del robot non era particolarmente diversa nei vari casi (si veda figura 3.12).

Includere nel modello la componente inerziale non risulta di facile implementazione. I valori delle velocità e accelerazioni dei giunti variano di molto in base alla posizione e configurazione del robot. Per includerli nel sistema

è necessario isolare tutti i possibili casi e registrare le velocità e accelerazioni relative.

La presenza di questi casi particolari ci ha spinto a realizzare un modello che non faccia solamente affidamento al modello PTP teorico, ma tenga in considerazione solamente il comportamento reale del robot.

# Capitolo 4

# Modello PTP avanzato

In questo capitolo verrà introdotto un ulteriore algoritmo per la previsione del tempo richiesto per una traiettoria PTP, cercando sempre di realizzare un simulatore che non implementi un modello dinamico del robot, per i motivi citati precedentemente.

Viste le problematiche presentate nella sezione 3.5, si è deciso di optare per un approccio diverso al problema.

Dopo un'ulteriore analisi del robot, si è deciso di realizzare un modello che implementasse un comportamento quanto più reale.

La nuova idea sta nel creare una mappa di tempi e posizioni. Tale mappa conterrà tutti i possibili movimenti del robot (seppur in un ambito limitato di utilizzo) assieme al tempo registrato per effettuare la traiettoria.

Da questi mappa si vuole poi estrapolare il movimento più simile a quello che il robot deve effettuare, al fine di avere un movimento stimato quanto più simile a quello reale che il robot esegue.

## 4.1 Realizzazione

Per realizzare questo tipo di sistema si è deciso di registrare i tempi necessari al robot per compiere un'ampia gamma di movimenti al fine di creare una mappa che associa movimenti e tempi.

Non è consigliabile registrare tutte le possibili combinazioni di movimenti eseguibili dal robot. Basta un semplice calcolo matematico per rendersi conto dell'enormità di possibili combinazioni di movimenti possibili.

Il tempo necessario per effettuare tutte le posizioni renderebbe lo studio del robot molto dispendioso in termini di tempo.

E' quindi necessario trovare un sottoinsieme di posizioni che combini un numero di registrazioni ridotte, assieme ad un livello precisione buono.

### 4.1.1 Riduzione dell'ambiente di lavoro

Registrare tutte le possibili combinazioni ottenibili dal robot risulta un compito particolarmente oneroso.

Per gli scopi di questa tesi si è deciso di limitare lo studio di questo modello ad un sottoinsieme limitato di casi. Si è scelto per semplicità di restringere il campo dei possibili movimenti solamente agli assi 1, 3, 4 e 5, vincolando la posizione degli assi 2 e 6.

| **Gradi (°)** | **Minimo** | **Massimo** |
|---|---|---|
| **Asse 1** | 0 | 90 |
| **Asse 2** | -90 | -90 |
| **Asse 3** | 0 | 90 |
| **Asse 4** | -90 | 90 |
| **Asse 5** | 0 | 90 |
| **Asse 6** | 0 | 0 |

Tabella 4.1: Range di lavoro

### 4.1.2 Intervalli di registrazione

Discretizzare il campo di lavoro permette di generare un sottoinsieme di tutte le possibili configurazioni generabili dal KUKA Agilus. Questo ha permesso di risparmiare tempo, mantenendo comunque un certo grado di accuratezza.

#### Discretizzazione rada

In questo tipo di discretizzazione, la variazione di 20°, serve a considerare maggiormente i movimenti nei quali i tempi di accelerazione e decelerazione sono predominanti, mentre le variazioni di 90° considerano maggiormente i movimenti nei quali la velocià limite è stata raggiunta e viene mantenuta per un intervallo di tempo superiore.

Utilizzando questa discretizzazione degli assi si ottengono 135 possibili combinazioni di angoli per ogni asse, come visibile nella formula:

$$3 \cdot 3 \cdot 3 \cdot 5 = 135 \tag{4.1}$$

Questa discretizzazione ha il vantaggio di essere molto veloce da realizzare. Le registrazioni del robot impiegano poco meno di 5 minuti.

| **Gradi (°)** | | | | | |
|---|---|---|---|---|---|
| **Asse 1** | 0 | 20 | 90 | nd | nd |
| **Asse 3** | 0 | 20 | 90 | nd | nd |
| **Asse 4** | -90 | -20 | 0 | 20 | 90 |
| **Asse 5** | 0 | 20 | 90 | nd | nd |

Tabella 4.2: Discretizzazione leggera

### Discretizzazione fitta

In questa discretizzazione si è deciso di non differenziare i movimenti come nel caso precedente, ma si è scelto di suddividere lo spazio di lavoro in maniera uniforme.

In questa registrazione sono inoltre stati aumentate le possibile combinazioni, rendendo il modello più completo rispetto al caso precedente.

In tabella 4.3 è presente la suddivisione dello spazio degli angoli per ogni asse.

| **Gradi (°)** | | | | | |
|---|---|---|---|---|---|
| **Asse 1** | 0 | 30 | 60 | 90 | nd |
| **Asse 3** | 0 | 30 | 60 | 90 | nd |
| **Asse 4** | -90 | -45 | 0 | 45 | 90 |
| **Asse 5** | 0 | 30 | 60 | 90 | nd |

Tabella 4.3: Discretizzazione fitta

Utilizzando una discretizzazione più fitta si ottengono invece:

$$4 \cdot 4 \cdot 4 \cdot 5 = 320 \tag{4.2}$$

possibili combinazioni dei vari angoli.

## 4.2 Registrazione configurazioni e tempi

Per registrare le combinazioni di configurazioni e tempi si è realizzato un semplice codice macchina KUKA, il quale iterativamente esegue i seguenti passi fino al completamento di tutti i movimenti.

- Azzera il timer interno
- Porta il robot nella posizione di home predefinita
- Avvia il timer
- Sposta il robot nella successiva configurazione
- Ferma il timer e salva il tempo impiegato

```
DEF SaveTimes( )
INI
counter = 0
numvalues = 320

WHILE counter <= numvalues
XEND.A1 = stop_position[counter].A1
XEND.A2 = stop_position[counter].A2
XEND.A3 = stop_position[counter].A3
XEND.A4 = stop_position[counter].A4
XEND.A5 = stop_position[counter].A5
XEND.A6 = stop_position[counter].A6

PTP START Vel=100 % START Tool[0] Base[0]
WAIT SEC 0
$TIMER[1] = 0
$TIMER_STOP[1] = FALSE

PTP END Vel=100 % END Tool[0] Base[0]
WAIT SEC 0
$TIMER_STOP[1] = TRUE
times[counter] = $TIMER[1]
counter = counter + 1

ENDWHILE

END
```

Come si può notare dallo snippet 4.2, il codice KUKA non differisce di molto da altri linguaggi di programmazione.

Per riuscire ad avere dei tempi accurati è stato necessario però inserire alcuni accorgimenti.

I programmi presenti all'interno del manipolatore vengono precompilati prima dell'esecuzione. Per migliorare l'efficienza dei movimenti, il compilatore analizza tre istruzioni alla volta. Questo però ha portato inizialmente a delle registrazioni temporali sballate.

Inserendo tra le righe di codice, delle istruzioni di blocco, ovvero *WAIT SEC 0*, il compilatore è costretto ad arrestare il movimento del robot quando queste istruzioni vengono richiamate, spezzando così il flusso continuo.

Dopo un'attenta analisi del manuale KUKA fornita, è stato possibile rintracciare il parametro che consente di modificare il numero di istruzioni analizzate dal compilatore.

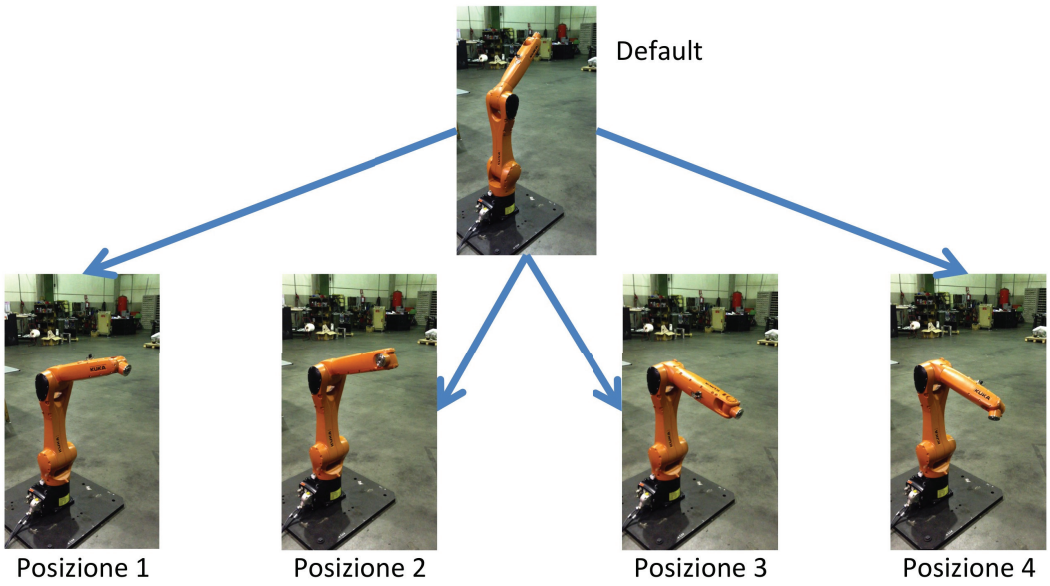


Figura 4.1: Sequenza registrazione configurazione e tempi

In figura 4.1, vengono mostrati alcuni movimenti eseguiti dal robot durante la registrazione delle combinazioni. Mentre in tabella 4.4 sono elencati i valori degli angoli dei singoli assi che corrispondono alla posizione di home utilizzata nelle registrazioni.

| **Posizione Home** | **Asse 1** | **Asse 2** | **Asse 3** | **Asse 4** | **Asse 5** | **Asse 6** |
|---|---|---|---|---|---|---|
| **Gradi (°)** | 0 | 90 | 45 | 0 | 0 | 0 |

Tabella 4.4: Posizione di home nelle registrazioni

## 4.3 Funzionamento

Il funzionamento dell'algoritmo è molto semplice e si compone dei seguenti passaggi.

1. Calcolo differenza di gradi degli assi, tra posizione di partenza e posizione di arrivo
2. Calcolo del rank per ogni configurazione presente nel database
3. Ordinazione delle configurazioni in base al rank ottenuto
4. Ricerca dei tempi nel database, del best fit e second best fit
5. Interpolazione del risultato, al fine di ottenere una stima quanto più realistica del tempo finale

In figura 4.2 è presente il flowchart che raffigura il funzionamento complessivo dell'algoritmo implementato.

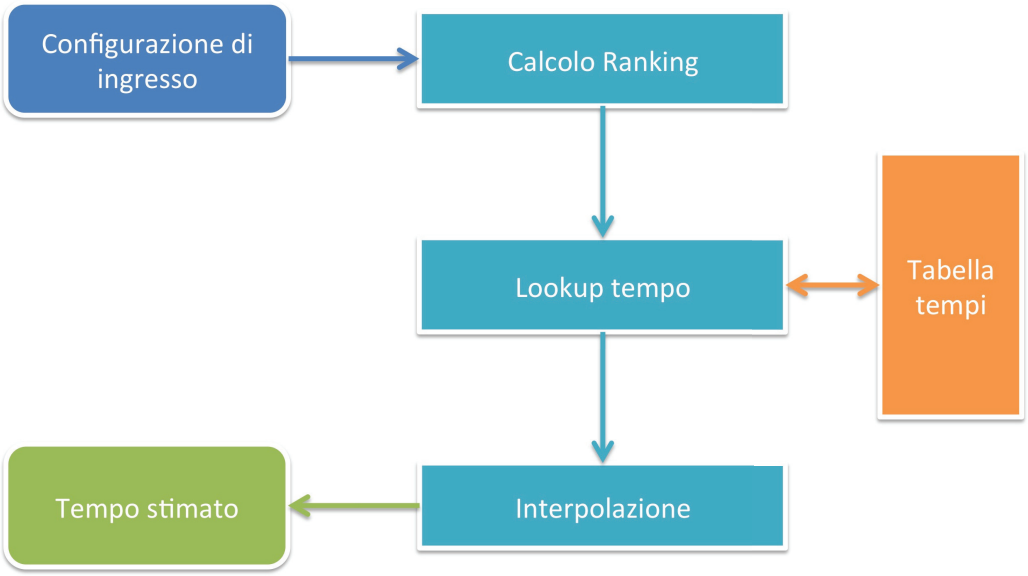


Figura 4.2: Flowchart funzionamento

### 4.3.1 Funzione di ranking

Per stimare quanto simile una configurazione risulta a quelle presenti nel database, si il modulo della differenza tra gli angoli della configurazione richiesta e quelle presente nel database. Ovvero:

$$rank_i = |A1 - A1_i| + |A3 - A3_i| + |A4 - A4_i| + |A5 - A5_i| \tag{4.3}$$

Questo valore rappresenta lo scarto complessivo, in gradi, tra le due configurazioni. Dato che non è possibile effettuare un ranking in base all'asse con il maggior grado da compiere, visti i casi particolari introdotti nel capitolo precedente, si è optato per questo semplice approccio, che per quanto semplice riesce ad offrire degli ottimi risultati.

### 4.3.2 Ricerca best fit

Dalla 4-upla delle differenze di angoli tra le due posizioni, l'algoritmo iterativamente confronta questa 4-upla con le 4-uple presenti nel database, valutando, tramite la funzione di Ranking, il valore di tale 4-upla, associandolo ad essa.

Una volta calcolato il valore di ranking per tutte le combinazioni presenti in memoria, queste vengono ordinate in base al ranking ricevuto. Quella con ranking inferiore tenderà ad avere un comportamento simile a quello del robot reale.

| **Configurazione (id)** | **Ranking** |
|---|---|
| 4 | 20 |
| 87 | 48 |
| ... | ... |
| ... | ... |
| 57 | 72 |

Tabella 4.5: Tabella ranking

Nella tabella 4.5 è presente un output del programma, dove nella colonna di sinistra vi sono gli id delle varie configurazioni salvate nel sistema, nella colonna di destra invece sono presenti i rank associati a tale configurazione.

Per costruzione delle configurazioni di registrazione, data una posizione in input, è sempre possibile trovare una best fit e una second best fit.

Si è visto con alcune prove, che l'interpolazione dei dati fornisce prestazioni superiori rispetto al solo utilizzo della best fit.

### Caso 1

In questa prima configurazione, l'errore nella stima del tempo risultava di per sé basso, ma con l'utilizzo dell'interpolazione si è riusciti a ridurre ulteriormente l'errore complessivo.

| **Configurazione** | 50 | 50 | -56 | 72 | | 468 |
|---|---|---|---|---|---|---|
| | **A1** | **A3** | **A4** | **A5** | **Rank** | **Tempo** |
| **Config 1** | 60 | 45 | -45 | 60 | ***38*** | ***444*** |
| **Config 2** | 30 | 45 | -45 | 60 | 48 | 408 |
| **Config 3** | 60 | 45 | -90 | 60 | 61 | 456 |
| **Config 4** | 60 | 45 | -45 | 90 | ***44*** | ***516*** |

Tabella 4.6: Esempio di interpolazione

Dai dati di tabella 4.6, si nota che la configurazione 1 e 4 sono rispettivamente la best fit e la second best fit.

Da queste due configurazioni, si calcolano le differenze dell'asse che distingue i due valori, in questo caso l'asse 5.

Per la configurazione best fit, si ottiene

$$|60 - 72| = 12$$

Mentre per la configurazione second best fit

$$|90 - 72| = 18$$

Da questi valori si calcolano le percentuali di somiglianza di una configurazione rispetto a best e second best fit.

| | **Best fit** | **Second best fit** |
|---|---|---|
| % | 60 | 40 |

Tabella 4.7: Percentuali di somiglianza

Da queste percentuali è possibile calcolare il valore temporale finale:

$$estimated\ time = 444 \cdot 0.6 + 516 \cdot 0.4 = 472ms$$

Dalla tabella è quindi possibile calcolare l'errore percentuale ottenuto valutando solamente la best fit, e calcolando l'interpolazione tra best fit e second best fit.

| | **Best fit** | **Interpolazione** |
|---|---|---|
| **Errore %** | 5,12 | -0,85 |

Tabella 4.8: Riduzione errore percentuale

### Caso 2

In questo caso invece, l'errore nella stima iniziale era particolarmente elevato. Grazie all'interpolazione dei dati si è riusciti ad ottenere un netto miglioramento nella stima finale del tempo.

| **Configurazione** | 36 | 51 | -18 | 80 | | 396 |
|---|---|---|---|---|---|---|
| | **A1** | **A3** | **A4** | **A5** | **Rank** | **Tempo** |
| **Config 1** | 30 | 45 | 0 | 90 | ***40*** | ***516*** |
| **Config 2** | 60 | 45 | 0 | 90 | 58 | 540 |
| **Config 3** | 30 | 75 | 0 | 90 | 58 | 600 |
| **Config 4** | 30 | 45 | -45 | 90 | ***49*** | ***400*** |
| **Config 5** | 30 | 45 | 0 | 60 | 50 | 504 |

Tabella 4.9: Esempio di interpolazione

Dai dati di tabella 4.6, si nota che la configurazione 1 e 4 sono rispettivamente la best fit e la second best fit.

Da queste due configurazioni, si calcolano le differenze dell'asse che distingue i due valori, in questo caso l'asse 5.

Per la configurazione best fit, si ottiene

$$|90 - 80| = 10$$

Mentre per la configurazione second best fit

$$|60 - 80| = 20$$

Da questi valori si calcolano le percentuali di somiglianza di una configurazione rispetto a best e second best fit.

| | **Best fit** | **Second best fit** |
|---|---|---|
| % | 66 | 33 |

Tabella 4.10: Percentuali di somiglianza

Da queste percentuali è possibile calcolare il valore temporale finale:

$$estimated\ time = 516 \cdot 0.66 + 400 \cdot 0.33 = 472ms$$

Dalla tabella è quindi possibile calcolare l'errore percentuale ottenuto valutando solamente la best fit, e calcolando l'interpolazione tra best fit e second best fit.

| | **Best fit** | **Interpolazione** |
|---|---|---|
| **Errore %** | 30,6 | 19,3 |

Tabella 4.11: Riduzione errore percentuale

In questo caso, l'errore risulta ancora parecchio alto, ma la riduzione introdotta dall'interpolazione riduce considerevolmente l'errore complessivo.

# Capitolo 5

# Risultati

In questo capitolo vengono presentati i risultati ottenuti dagli algoritmi sopra citati.

Nella sezione 5.1 vengono esposti i risultati dell'algoritmo che implementa il PTP sincrono completo. Come già introdotto nel capitolo 3, l'algoritmo non risulta molto performante. Le cause sono state elencate nella sezione 3.5.

Nella sezione 5.2 invece, verranno presentati i risultati utilizzando il secondo algoritmo. Le prestazioni in questo caso risultano sensibilmente migliori.

## 5.1 Modello PTP sincrono completo

Questo modello segue attentamente l'andamento teorico del movimento PTP sincrono completo del robot. Nel caso reale del robot, l'inerzia gioca una parte fondamentale nella movimentazione del robot, come la limitazione della velocità del tool.

Queste due limitazioni possono essere implementate con un certo grado di accuratezza nel sistema. Le limitazioni dovute all'accoppiamento dei motori in alcuni casi particolari però, troncano pesantemente le prestazioni di questo algoritmo.

Per evitare che questo accada sarebbe necessario trovare e registrare i vari valori di velocità e accelerazioni. Trovare tutte le possibili casistiche risulta un approccio dispendioso in termini di tempo, dato che questa operazione deve essere ripetuta per ogni robot che si vuole implementare.

### 5.1.1 Caso 1

In questa prova, il robot si sposta nelle due seguenti posizioni:

| | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** |
|---|---|---|---|---|---|---|
| **Start** | 0 | -90 | 90 | 0 | 0 | 0 |
| **Stop** | 0 | 0 | 0 | 0 | 0 | 0 |

Tabella 5.1: Combinazione assi caso 1

Il robot dalla posizione di home si porta in una posizione completamente sbracciato.

| | **Simulatore** | **Reale** | **Errore %** |
|---|---|---|---|
| **Tempo (ms)** | 960 | 1296 | 25,92 |

Tabella 5.2: Tempo caso 1

In questo caso il *leading axis* risulta il numero 2, con un tempo stimato di esecuzione della traiettoria di circa 960 ms. Il robot reale invece, impiega circa 1300 ms, 336 ms più lento del simulatore.

L'errore elevato di questa prova è dovuto alla gestione dell'inerzia nel controllore del robot. In questo caso lo spostamento del robot da una posizione nella quale il centro di massa del robot risulta molto vicina alla base, verso una posizione nella quale il centro di massa risulta molto al di fuori della base del robot, costringe il controllore a rallentare le velocità dei giunti, al fine di ottenere una traiettoria quanto più possibile priva di sobbalzi.

### 5.1.2 Caso 2

In questo movimento, il simulatore risulta particolarmente accurato nella stima del tempo. Il robot esegue il seguente movimento:

| | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** |
|---|---|---|---|---|---|---|
| **Start** | 0 | -90 | 90 | 0 | 0 | 0 |
| **Stop** | 0 | -45 | 60 | 0 | 75 | 0 |

Tabella 5.3: Combinazione assi caso 2

Le prestazioni in questo caso risultano particolarmente accurate. L'errore percentuale risulta molto basso.

| | **Simulatore** | **Reale** | **Errore %** |
|---|---|---|---|
| **Tempo (ms)** | 660 | 648 | 1,85 |

Tabella 5.4: Tempo caso 2

Ma, dal grafico delle velocità di figura 5.1, si può notare che il controllo numerico rallenta i movimenti del robot.

Questo perché il centro di massa del robot si sposta al di fuori della piattaforma di appoggio del robot stesso, quindi il controllo d'inerzia deve intervenire per evitare di far oscillare il tool finale.

Questo caso rappresenta una condizione particolare, dato che, con l'intervento del controllo d'inerzia ci si aspetta che il tempo reale sia superiore al tempo stimato dal simulatore PTP. In questo caso la situazione non si verifica e si ha un errore molto basso.

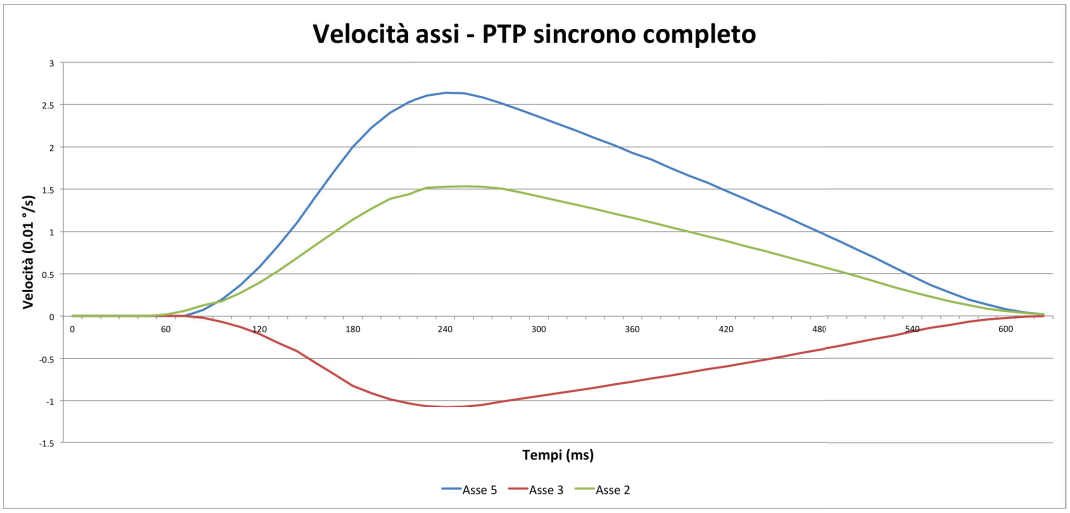


Figura 5.1: Velocità giunti in PTP sincrono completo

### 5.1.3 Prestazioni

Le prestazioni complessive dell'algoritmo sono dunque molto dipendenti dal tipo di movimento che il robot deve compiere.

In figura 5.2 è visibile il grafico dell'errore % commesso dall'algoritmo.

Dal grafico si vede chiaramente che le prestazioni sono molto scarse, con picchi di errore fino all'80 % del tempo reale.

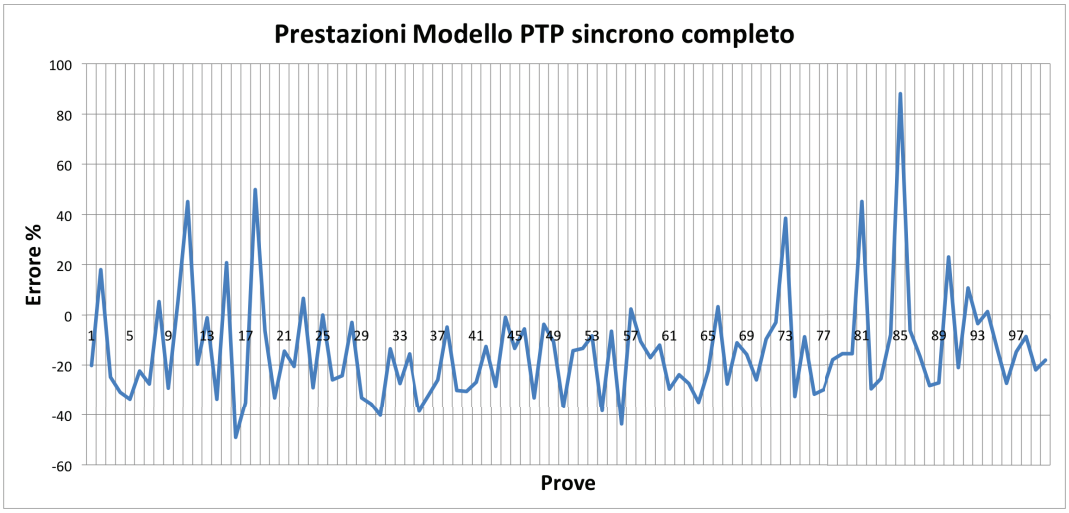


Figura 5.2: Errore % modello PTP sincrono completo

| | $\mu$ | $\sigma$ |
|---|---|---|
| **Modello PTP** | -14,34 | 21,46 |

Tabella 5.5: Prestazioni modello PTP sincrono completo

Mentre, dalla tabella 5.5, si può vedere come la media e varianza dell'errore percentuale sono elevate. Sia il valore della media, che la deviazione standard risultano particolarmente elevati.

Il modello PTP sincrono completo, per quanto teoricamente corretto, risulta invece poco accurato nelle situazioni reali

## 5.2 Modello PTP accurato

Le prestazioni dell'algoritmo sono state valutate registrando dei movimenti random del robot, registrandone i tempi e confrontando questi valori con i tempi restituiti dal simulatore.

Dai risultati ottenuti si vede chiaramente che minore sono il numero di registrazioni entro le quali cercare corrispondenze, peggiori saranno le prestazioni, dato che la discrepanza in angoli tra posizione reale e posizione inserita nel simulatore risulta elevata in molti casi.

Dalle prove effettuate sul robot reale si è visto che esso è molto sensibile alla posizione di partenza e al tipo di movimento che devono effettuare gli angoli (di questo è stato discusso nella sezione 3.5).

Le posizioni di test sono state generate da un algoritmo che, dato l'insieme di lavoro, creava diverse possibili posizioni valide per il robot.

### 5.2.1 Prestazioni 135 registrazioni

Il primo test è stato effettuato utilizzando il primo set di 135 registrazioni. Come si vede dai risultati, le prestazioni dell'algoritmo utilizzando dei campioni molto distanti tra loro sono molto basse. Questo è dovuto al fatto che nella maggior parte dei casi provati, il valore del ranking risultava particolarmente elevato, questo a causa dell'elevata differenza tra gli angoli dei vari assi.

In questo caso, il robot per effettuare un movimento partiva dalla posizione di home.

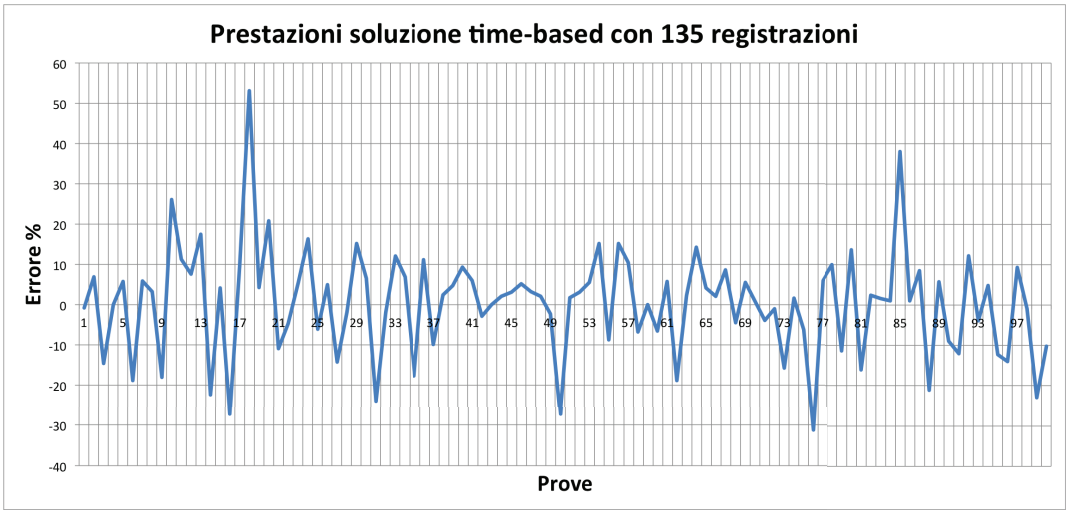


Figura 5.3: Errore % con 135 registrazioni e posizione di partenza home

Come si può vedere dalla figura 5.3, il grafico ha un andamento molto altalenante.

### 5.2.2 Prestazioni 320 registrazioni da home

Questo test verifica le prestazioni dei movimenti con posizione di partenza home, utilizzando però un database di 320 registrazioni, a differenza delle 135 della sezione precedente.

In questo caso l'errore medio è inferiore a quello precedente. Con un campionamento più fitto, le prestazioni, come era lecito aspettarsi, sono migliorate e la deviazione standard dell'errore percentuale risulta inferiore.


Figura 5.4: Errore % con 320 registrazioni e posizione di partenza home

### 5.2.3 Prestazioni 320 registrazioni non da home

In questo test, le prestazioni sono state valutate muovendo il robot in posizioni casuali, ma a differenza della sezione precedente, la posizione di partenza risulta diversa da quella di home predefinita. Questo per verificare il funzionamento dell'algoritmo in un ambiente reale, dove raramente la posizione di partenza corrisponde alla posizione di home del robot.

Come si può vedere dal grafico di figura 5.5, le prestazioni sono leggermente inferiori al caso precedente.

Come si può vedere, le prestazioni sono discrete, ottime se la posizione iniziale è quella di home, cosa che in un caso reale non sarà mai.

### 5.2.4 Differenze in base alle registrazioni utilizzate

Come visto nel capitolo precedente, e come era lecito aspettarsi, le prestazioni del modello peggiorano nel caso si utilizzi una discretizzazione meno fitta.

Questo si verifica perché l'algoritmo cerca di interpolare due posizioni che non son molto simili tra di loro. Nel caso la configurazione richiesta si trovi in

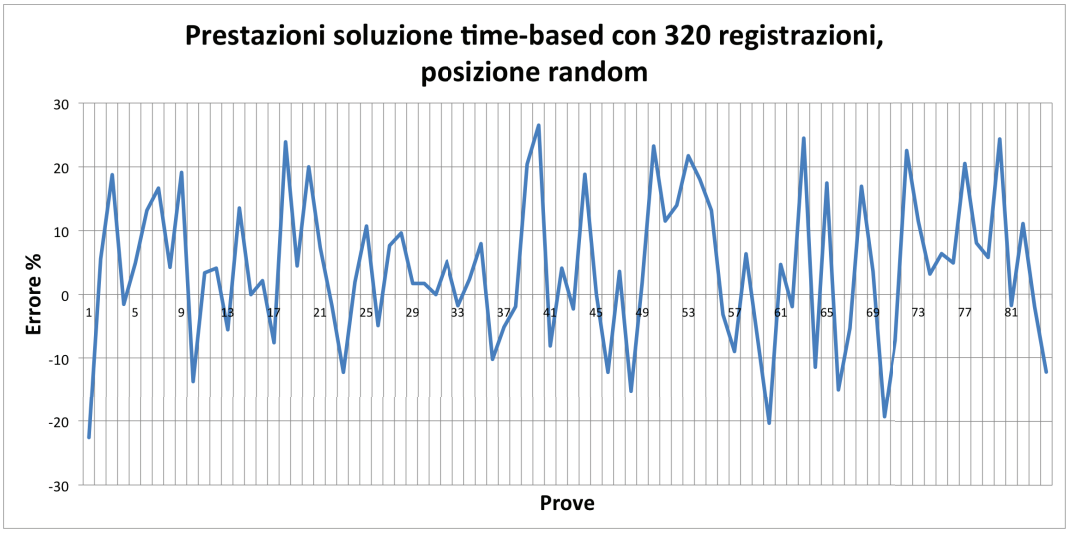


Figura 5.5: Errore % con 320 registrazioni e posizione partenza random

| Numero registrazioni | Posizione partenza | $\mu$ | $\sigma$ |
|---|---|---|---|
| 135 | home | 0,398 | 13,129 |
| 135 | random | 7,451 | 24,637 |
| 320 | home | 0,173 | 5,869 |
| 320 | random | 4,01 | 11,6 |

Tabella 5.6: Tabella confronto prestazioni

un punto intermedio tra due configurazioni registrate, il tempo finale stimato non risulterà particolarmente accurato, dato che non è presente nel sistema una configurazione simile.

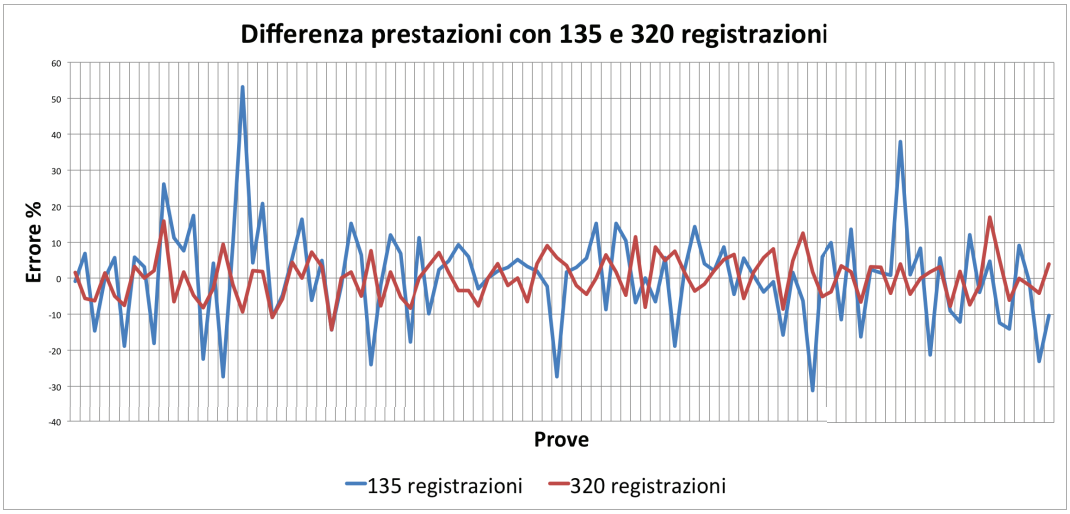


Figura 5.6: Differenza di errore % utilizzando 135 o 320 registrazioni

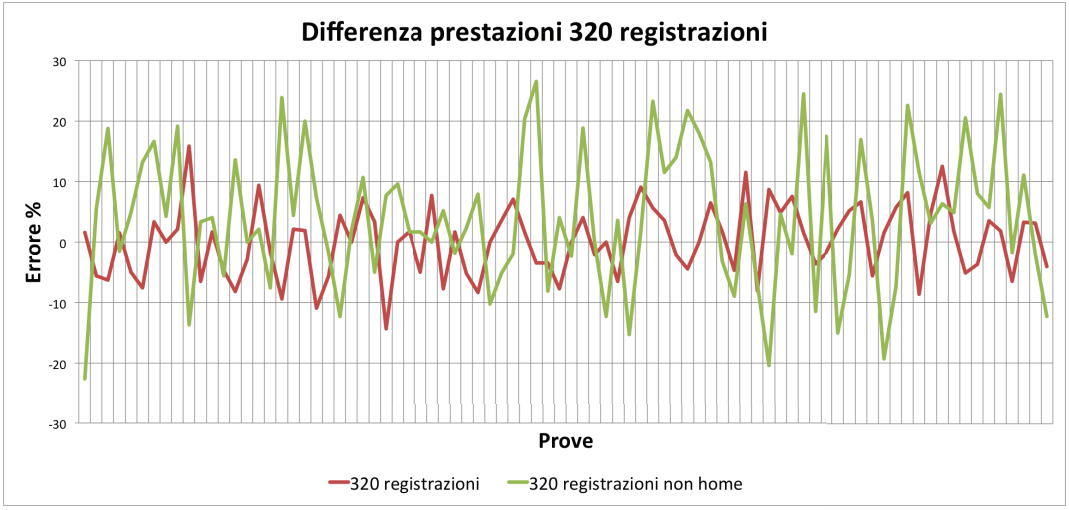


Figura 5.7: Differenza tra partenza da home e random

# Capitolo 6

# Conclusioni e sviluppi futuri

In questo capitolo finale si discuterà delle conclusioni e di alcune possibili idee per migliorare l'accuratezza del simulatore.

## 6.1 Conclusioni

Per realizzare questa tesi è stato necessario risolvere problemi di varia natura.

La comprensione del robot e del suo funzionamento hanno richiesto parecchio tempo e prove. Il manuale fornito, per quanto completo, non è subito di semplice comprensione e richiede la conoscenza di alcune caratteristiche del robot per saperlo interpretare al meglio.

Come si è visto nei capitoli precedenti, il comportamento del manipolatore non è sempre di facile previsione data la presenza di molti fattori correttivi (controllo d'inerzia, limitazione della velocità del tool) e fattori meccanici (accoppiamento dei motori).

Questi fattori rendono il sistema reale molto diverso dal modello teorico di movimento PTP, come spiegato nel capitolo 3.

Data la difficoltà di prevedere, solamente dai valori degli angoli, il tempo stimato, si è optato per un approccio differente, che seppur dispendioso in termini temporali, si è dimostrato particolarmente efficace per simulare il comportamento del robot finale.

Questa tesi ha verificato che l'idea di realizzazione del modello risulta valida in un ambito limitato dell'area di lavoro del robot.

Questo modello risulta utile alla ditta durante il lavoro di progettazione di nuovi impianti. La conoscenza della traiettoria più veloce può aiutare ad aumentare l'efficienza di un robot diminuendo il tempo/ciclo necessario per svolgere un lavoro di un robot.

## 6.2 Sviluppi futuri

Questo approccio al momento richiede la registrazione di molte combinazioni di configurazione di angoli e tempo di esecuzione del movimento.

Nel caso si dovessero usare tutti e sei gli assi non vincolando il tipo di movimento possibile, le registrazioni totali presentano un ostacolo per questo approccio.

E' necessario quindi trovare un sotto-set di registrazioni che rappresentino in maniera quanto più verosimile il comportamento del robot, soprattutto nei casi in cui il comportamento non è prevedibile come ci si aspetterebbe, come nei casi in cui il controllo inerziale entra in funzione, oppure quando la velocità del tool viene limitata.

Nei casi rimanenti infatti, il movimento PTP è prevedibile con un certo grado di accuratezza utilizzando il primo metodo proposto.

Un altro sviluppo riguarda l'implementazione del movimento LIN (lineare) del robot.

# Bibliografia


[1] Euclid Labs S.r.l., *azienda informatica, progetta e realizza soluzioni ad alta tecnologia per robotica e automazione industriale*, `www.euclidlabs.it`.

[2] *Introduction to Robotics, Mechanics and Control, Third Edition*, John J. Craig.

[3] *Dynamic Modeling of Serial Manipulator Arms*, M. Thomas e D. Tesar, 1984.

[4] *Mathematics for Dynamic Modeling, Second Edition*, Edward Beltrami, 1997.

[5] *Dynamic Modeling and Simulation of Robot Manipulators, The Newton-Euler Formulation*, Herman Hoifodt, 2011.

[6] Mathworks. *MATLAB - The Language of Technical Computing*, `http://www.mathworks.it/products/matlab/`.

[7] Maplesoft. *Maple - Tecnical Computing Software for Engineers*, `http://www.maplesoft.com/`.

[8] Parametric Technology Corporation. *Pro/ENGINEER - modellatore CAD tridimensionale parametrico*, `http://www.ptc.com/product/creo/proengineer`.

[9] Microsoft Visual Studio 2013 Express. *IDE di programmazione C#*, `http://msdn.microsoft.com/it-it/library/dd831853.aspx`.

[10] OrangeEdit. *IDE di programmazione per codice robot KUKA, sviluppato da Orange Apps*, `http://www.orangeapps.de/?lng=en&page=apps/orangeedit`.

[11] KUKA Robotics. *Produttore tedesco a livello mondiale di robot industriali e soluzioni per l'automazione industriale*, `http://www.kuka-robotics.com/italy/it/`.

[12] KUKA Agilus 1100. *Robot antropomorfo KUKA*, `http://www.kuka-robotics.com/en/products/industrial_robots/small_robots/kr10_r1100_sixx_wp/`.

[13] Manuale KR C4 compact. *Manuale del controllore del KUKA Agilus*, Spez KR C4 compact en.

[14] Manuale KUKA Agilus Sixx WP. *Manuale del KUKA Agilus Sixx WP*, Kuka Agilus R10 1100 Sixx.

[15] KUKA System Software 8.2. *Manuale comprensivo di ogni robot KUKA, ne descrive tutte le caratteristiche, dalle operazioni di sicurezza, manutenzione, configurazione, programmazione e diagnosi.*, Handling Tool.

[16] The Duel: Timo Boll vs. KUKA Robot. *Discussa sfida a table tennis tra il campione tedesco Timo Boll e il KUKA Agilus*, `https://www.youtube.com/watch?v=tIIJME8-au8`.