UNIVERSITÀ DEGLI STUDI DI PADOVA

—

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE
DIPARTIMENTO DI TECNICA E GESTIONE DEI SISTEMI INDUSTRIALI

—

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCANICA

ANALISI DELLE COLLISIONI TRA MANIPOLATORI INDUSTRIALI: STIMA DELLE TRAIETTORIE

RELATORE: CH.MO PROF. ING. GIULIO ROSATI

LAUREANDO: STEFANO CONCI

ANNO ACCADEMICO 2014-2015

ai miei genitori

“ L’ingegneria non è l’arte di costruire. È invece l’arte di non costruire: cioè, è l’arte di fare bene con un dollaro quello che un qualunque pasticcione potrebbe fare con due. ”

Arthur Mellen Wellington

Indice

Sommario

La presente tesi si pone l'obiettivo di stimare la traiettoria che un manipolatore esegue durante un certo tipo di movimento. Nel mondo attuale, l'automazione delle catene produttive è un aspetto essenziale per aumentarne la produttività. L'importanza di avere una stima della traiettoria, risiede nel poter prevedere in anticipo la presenza di collisioni con ostacoli presenti nella cella di lavoro e di modificare di conseguenza i punti intermedi del movimento. Utilizzando il robot presente nel dipartimento di Robotica sono stati eseguiti diversi movimenti, ritenuti possibili in ambito industriale, per confrontarli con i risultati teorici.

Il principale obiettivo raggiunto consiste nell'aver ottenuto una traiettoria prossima a quella reale, mediante la sola conoscenza dei punti intermedi, della velocità e del valore dell'arco.

Introduzione

La tesi che viene descritta nelle pagine che seguono si inserisce nel contesto dell'analisi delle collisioni tra manipolatori industriali. Nello specifico si concentra sulla determinazione della traiettoria percorsa dal robot nell'eseguire un certo movimento. Disporre di un metodo per stimare il tragitto eseguito, permette di aggirare un ostacolo in maniera sicura e di poter lavorare con due manipolatori nella stessa cella di lavoro.

Il presente studio analizza un determinato numero di movimentazioni ed elabora un modello analitico per ricavare un percorso stimato, utilizzando come informazioni i punti di passaggio, la velocità e il valore del parametro arco.

Nel capitolo 1 vengono descritte le pianificazioni del movimento impiegate principalmente dai manipolatori industriali per eseguire i movimenti, illustrando vantaggi e svantaggi. Introducendo le problematiche relative alla stima del passaggio per i punti intermedi.

Il capitolo 2 analizza la strumentazione utilizzata per la raccolta dei dati sperimentali, sui quali è stato sviluppato il modello analitico. Segue un'introduzione del software Epson, la sua interfaccia e i principali comandi impiegati per l'esecuzione delle prove in laboratorio, focalizzandosi sui comandi di movimento.

Nel capitolo 3 si descrive come stati ricavati i dati sperimentali e sulla base di essi sia stato implementato un modello analitico che li potesse rappresentare. Illustrando i problemi riscontrati nel proseguire con l'elaborazione del modello e le soluzioni applicate per le diverse situazioni.

Il capitolo 4, infine, si occupa di esporre i risultati teorici ottenuti, confrontandoli poi con quelli sperimentali, mostrando graficamente i vantaggi ottenuti nell'applicazione del metodo descritto nella seguente tesi.

Capitolo 1

Pianificazione del movimento

La pianificazione del movimento dei robot [1] è un argomento di vaste dimensioni; in questa sezione si intenderà coglierne alcuni aspetti peculiari e di base. Esistono casi in cui le leggi di moto dell'end-effector sono imposte in maniera rigida dal particolare compito imposto al robot; in questo caso anche le leggi di moto degli attuatori risultano necessariamente definite in maniera univoca e non rimangono spazi per un'ottimizzazione della traiettoria. Tuttavia esistono numerosi altri casi in cui la legge di moto dell'end-effector è assegnata solo parzialmente; in questa situazione si possono avere molti gradi di libertà ed è opportuno avere un criterio per scegliere la legge di moto più conveniente. Nella scelta delle legge ottima occorre tenere in considerazione diversi aspetti legati alla realtà quali:

- i limiti strutturali del robot (valori di fine corsa delle articolazioni);
- presenza di eventuali ostacoli;
- limiti meccanici ed elettrici degli attuatori (velocità, coppie, tensioni, correnti massime...);
- limiti delle trasmissioni meccaniche (velocità, coppia, potenza massima);
- limiti degli azionamenti (massimi valori di tensione, corrente, frequenza che limitano la massima velocità e coppia del corrispondente attuatore elettrico).

L'individuazione di questi limiti permetterà una prima ottimizzazione del comportamento del robot ottimizzando il movimento di ciascun grado di libertà considerato separatamente; ad essa può seguire una seconda fase nella quale il robot viene esaminato globalmente, ma questa analisi e molto più articolata e complicata.

1.1 Considerazioni sulle leggi di moto di un singolo grado di libertà

1.1.1 Generalità

I criteri di buon progetto di una legge di moto indirizzano la scelta verso leggi che rispettino almeno la continuità dello spostamento e della velocità. La legge di moto più semplice che soddisfa i requisiti minimi è quella di tipo trapezoidale, dove il nome si riferisce al grafico dell'andamento della velocità nel tempo [2].

1.1.2 Criteri di scelta elementari: minimizzazione della velocità, dell'accelerazione e della potenza massima

Si supponga di avere una legge ad accelerazione costante simmetrica[1], ovvero a velocità trapezoidale, nella quale il tempo Δt_1 di accelerazione (e decelerazione) sia preso come variabile da utilizzare per l'ottimizzazione della legge. Sia noto lo spostamento Δs da effettuare e il tempo T di azionamento. Come noto, l'area sottesa dal semidiagramma delle accelerazioni rappresenta il valore della velocità massima; quindi:

$$v_{max} = a_{max} * \Delta t_1$$

[1] ove cioè l'accelerazione e la decelerazione hanno stesso modulo e stessa durata

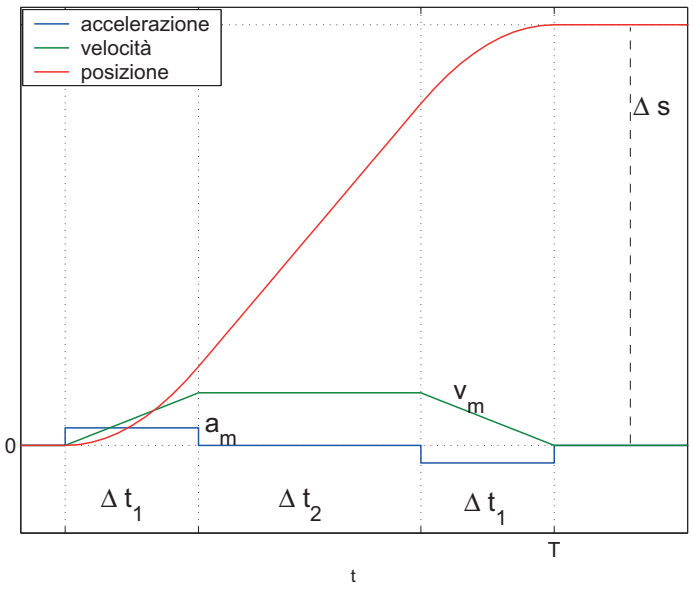


Figura 1.1: Legge ad accelerazione costante simmetrica

Allo stesso modo si ha che l'area sottesa dal diagramma delle velocità rappresenta lo spazio Δs da percorrere:

$$\Delta s = v_{max}\Delta t_2 + 2\frac{v_{max}}{2}\Delta t_1 = a_{max}\Delta t_1\Delta t_2 + a_{max}\Delta t_1^2 \tag{1.1}$$

$$\Delta s = a_{max}\Delta t_1(\Delta t_2 + \Delta t_1) \tag{1.2}$$

Poiché $\Delta t_2 = T - 2\Delta t_1$ si può scrivere:

$$\frac{\Delta s}{T^2} = a_{max}\frac{\Delta t_1}{T}(1 - \frac{\Delta t_1}{T}) \tag{1.3}$$

Indicando con λ il rapporto $\Delta t_1/T$ (rappresentativo del parametro che si vuole ottimizzare) [2], l'ultima relazione scritta assume la forma:

$$\frac{\Delta s}{T^2} = a_{max}\lambda(1-\lambda) \tag{1.4}$$

Come conseguenza di questo, la velocità e l'accelerazione massime risultano:

$$a_{max} = \frac{\Delta s}{T^2}\frac{1}{\lambda(1-\lambda)} = \frac{\Delta s}{T^2}c_a \qquad c_a = \frac{1}{\lambda(1-\lambda)} \tag{1.5}$$

$$v_{max} = \frac{\Delta s}{T}\frac{1}{(1-\lambda)} = \frac{\Delta s}{T}c_v \qquad c_v = \frac{1}{(1-\lambda)} \tag{1.6}$$

[2] ovviamente $0 \leq \lambda \leq \frac{1}{2}$

ove c_a e c_v, le cui espressioni per leggi di moto ad accelerazione costante simmetrica sono qui riportate in funzione di λ, sono detti coefficienti di accelerazione e di velocità e in generale sono così definiti:

$$c_a = \frac{|a_{max}|}{\Delta s/T^2} \qquad c_v = \frac{|v_{max}|}{\Delta s/T} \tag{1.7}$$

con $|a_{max}|$ e $|v_{max}|$ valori massimi in modulo di accelerazione e velocità. Questi coefficienti indicano quanto le velocità o accelerazioni massime superano quelle medie.

Nel caso in esame si osserva che tanto la velocità quanto l'accelerazione sono massime contemporaneamente. Pertanto supponendo che il motore debba vincere solo forze di inerzia ($f = ma$), la massima potenza specifica $P = fv/m = av$ (cioè per unità di massa) richiesta al motore è pari a:

$$P_{max} = a_{max} v_{max} = \frac{\Delta s^2}{T^3} \frac{1}{\lambda(1-\lambda)^2} = \frac{\Delta s^2}{T^3} c_p \qquad c_p = \frac{1}{\lambda(1-\lambda)^2} \tag{1.8}$$

ove c_p è assimilabile a un coefficiente di potenza e nel caso particolare di questa legge vale $c_p = c_a c_v$. L'andamento di c_v, c_a, c_p in funzione di λ è riportato in figura.

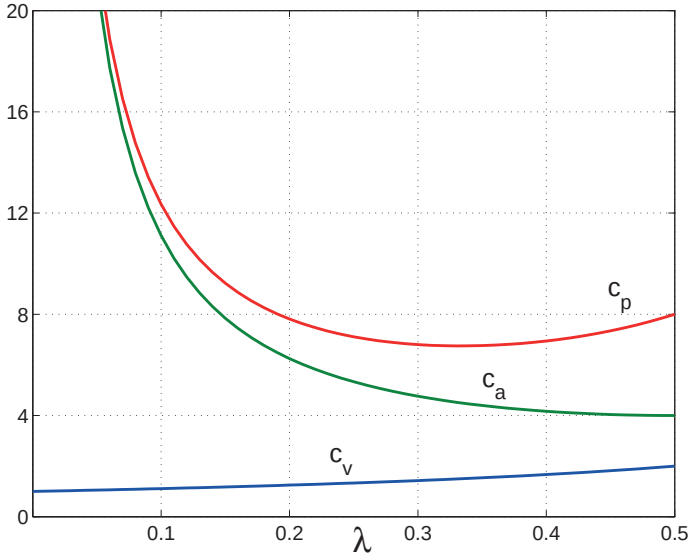


Figura 1.2: Andamento di c_v, c_a, c_p in funzione di λ.

λ	0	$\frac{1}{10}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{2}$
c_v	1	1.1	1.25	1.5	2
c_a	∞	11.1	6.25	4.5	4
c_p	∞	12.3	7.81	6.75	8

Figura 1.3: Valori di c_v, c_a, c_p corrispondenti ad alcuni λ.

L'accelerazione massima è minima per $\lambda = \frac{1}{2}$, la velocità massima è minima per $\lambda = 0$ ove però l'accelerazione è infinita. La potenza massima è minima per $\lambda = \frac{1}{3}$, cioè la legge corrispondente dedica un terzo del periodo T all'accelerazione, uno alla velocità e uno alla decelerazione. Dunque un valore del parametro λ prossimo a $\frac{1}{3}$ è ragionevolmente conveniente.

1.1.3 Scalatura delle leggi di moto con dilatazione della scala dei tempi

Si supponga di avere una legge di moto eseguita in un intervallo di tempo T_1 e di esaminarne i diagrammi di spostamento, velocità e accelerazione. Si immagini ora di voler ottenere lo stesso movimento $\bar{s}$ in un tempo T_2 maggiore di T_1 ($T_2 = kT_1$). Questo comporta che i diagrammi di spazio, velocità e accelerazione si modificano dal punto di vista dei valori massimi di velocità e accelerazione, mantenendo però la stessa forma. Infatti vale la relazione $s_2(t) = s_1(t/k)$ che derivata n volte rispetto al tempo da:

$$\frac{d^n}{dt^n} s_2(t) = \frac{d^n}{dt^n} s_1(t/k) \frac{1}{k^n} \tag{1.9}$$

In particolare si vede che per $n = 1$ e $n = 2$ risulta:

$$\frac{d}{dt} s_2(t) = v_2(t) = v_1(t/k) \frac{1}{k} \qquad \frac{d^2}{dt^2} s_2(t) = a_2(t) = a_1(t/k) \frac{1}{k^2} \tag{1.10}$$

cioè i massimi valori di velocità sono ridotti di k volte, mentre quelli di accelerazione di k^2 volte. Queste stesse considerazioni si possono ottenere anche ricordando il significato di c_a e c_v (paragrafo precedente).

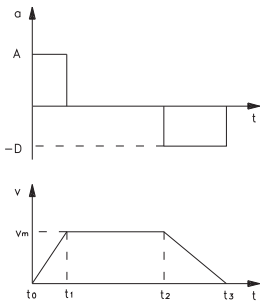
1.1.4 Tempo minimo di azionamento

Spesso risulta utile poter calcolare quale sia il tempo minimo necessario ad effettuare un certo movimento da parte di un motore con carico puramente inerziale soggetto a limiti di velocità e accelerazione massimi. Il motore è inizialmente fermo e deve arrestarsi una volta effettuata la movimentazione richiesta (cioè la velocità iniziale e quella finale sono nulle).

Si indicano con A e D i moduli dei valori di accelerazione e decelerazione massimi e con v_m il modulo del valore massimo della velocità. I vincoli con cui trovare la soluzione saranno dunque:

$$-D \leq a \leq A \qquad |v| \leq v_m \tag{1.11}$$

La legge di moto che consegue questo obiettivo è quella ad accelerazione costante con le seguenti caratteristiche (figura 1.4):



t	a	v	s
$t_0 \div t_1$	A	$A(t-t_0)$	$\frac{A(t-t_0)^2}{2}$
$t_1 \div t_2$	0	v_m	$\frac{A(t-t_0)^2}{2} + v_m(t-t_1)$
$t_2 \div t_3$	$-D$	$D(t3-t)$	$(\ldots) - \frac{D(t_3-t)^2}{2}$

Figura 1.4: Legge di moto ad accelerazione costante, con accelerazione positiva $A = a^+$ diversa da quella negativa $D = a^-$; $v_m = A(t_1 - t_0) = D(t_3 - t_2)$; $\Delta s = v_m[\frac{(t_1-t_0)}{2} + (t_2 - t_1) + \frac{(t_3-t_2)}{2}]$.

a) tratto iniziale ad accelerazione costante positiva al valore massimo consentito ($a = A$) sino al raggiungimento della velocità massima;

b) tratto a velocità costante massima ($v = v_m$) il più lungo possibile;

c) tratto ad accelerazione massima negativa fino all'arresto del motore ($a = -D$).

Ogni altra legge di moto richiederà un tempo superiore.

Poiché lo spazio percorso è pari all'area sottesa dal diagramma della velocità si può scrivere:

$$\Delta s = \frac{1}{2}A(t_1 - t_0)^2 + v_m(t_2 - t_1) + \frac{1}{2}D(t_3 - t_2)^2 \tag{1.12}$$

ma poiché l'area sottesa dal semidiagramma delle accelerazioni rappresenta la velocità massima, si può scrivere:

$$t_1 - t_0 = \frac{v_m}{A} \qquad t_3 - t_2 = \frac{v_m}{D} \tag{1.13}$$

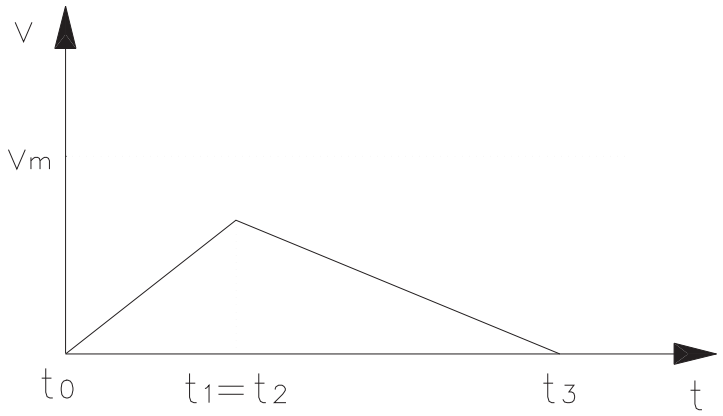
e sostituendo nell'espressione di Δs si ottiene:

$$t_2 - t_1 = \frac{\Delta s - \frac{1}{2}v_m^2\left(\frac{1}{A} + \frac{1}{D}\right)}{v_m} = \frac{\Delta s}{v_m} - \frac{1}{2}\frac{v_m(A+D)}{AD} \tag{1.14}$$

Il tempo totale d'azionamento risulta perciò:

$$\Delta t = t_3 - t_0 = \frac{\Delta s}{v_m} + \frac{1}{2}\frac{v_m(A+D)}{AD} \tag{1.15}$$

Si consideri ora una una legge di moto in cui non sia possibile raggiungere la v_{max} in quanto lo spazio Δs da percorrere è molto piccolo e quindi la fase di frenatura deve cominciare prima di raggiungere tale limite.



Procedendo analogamente a quanto fatto in precedenza, si ottengono, con semplici passaggi, le seguenti relazioni:

$$t_1 - t_0 = \sqrt{2\Delta s\frac{D}{A}\frac{1}{A+D}} \tag{1.16}$$

$$t_3 - t_2 = \sqrt{2\Delta s\frac{A}{D}\frac{1}{A+D}} \tag{1.17}$$

e il tempo di azionamento diventa:

$$\Delta t = (t_1 - t_0) + (t_3 - t_2) = \left[\sqrt{\frac{A}{D}} + \sqrt{\frac{D}{A}}\right]\sqrt{2\Delta s\frac{1}{A+D}} = \sqrt{2\Delta s\frac{A+D}{AD}} \tag{1.18}$$

1.2 Movimento punto-punto in traiettoria libera

È questo il più semplice modo di movimentare un robot quando sia richiesto di muovere la pinza da una posa X_1 a una seconda posa X_2 senza dover eseguire una traiettoria predefinita[3]. Il più semplice approccio da utilizzare in questi casi può essere scomposto nei seguenti passi:

a) indicando con $X_1 = [x_1\ y_1\ z_1\ \alpha_1\ \beta_1\ \gamma_1]^T$ e $X_2 = [x_2\ y_2\ z_2\ \alpha_2\ \beta_2\ \gamma_2]^T$ la posa iniziale e finale, si risolve, in corrispondenza di esse, il problema cinematico inverso calcolando le coordinate ai giunti espresse dai vettori Q_1 e Q_2;

b) si calcola il vettore $\Delta Q = Q_2 - Q_1 = [\Delta q_1 \dots \Delta q_m]^T$ che esprime la variazione subita dalle singole coordinate ai giunti nel passaggio dalla posa iniziale a quella finale;

c) si costruisce una legge di moto per ogni giunto, ad esempio secondo il criterio di minimizzare l'accelerazione massima oppure secondo quello di rendere minimo il tempo di azionamento; in ogni caso ogni giunto i sarà caratterizzato dal suo tempo di azionamento Δt_i;

d) si calcola T_a e cioè il tempo minimo di azionamento del sistema, il quale è il valore massimo tra tutti i tempi di azionamento:

$$T_a = \max_{i=1}^{m}(\Delta t_1)$$

e) si dilata la scala dei tempi di ogni giunto mediante l'uso di un coefficiente k_i così definito per ogni giunto i:

$$k_i = \frac{T_a}{\Delta t_i}$$

[3]Verrà indicato nel seguito con il termine posa l'insieme della posizione e dell'orientamento della pinza del robot. La posa può essere rappresentata da un vettore X contenente le coordinate cartesiane e le variabili angolari della pinza:

$$X = [x\ y\ z\ \alpha\ \beta\ \gamma]^T$$

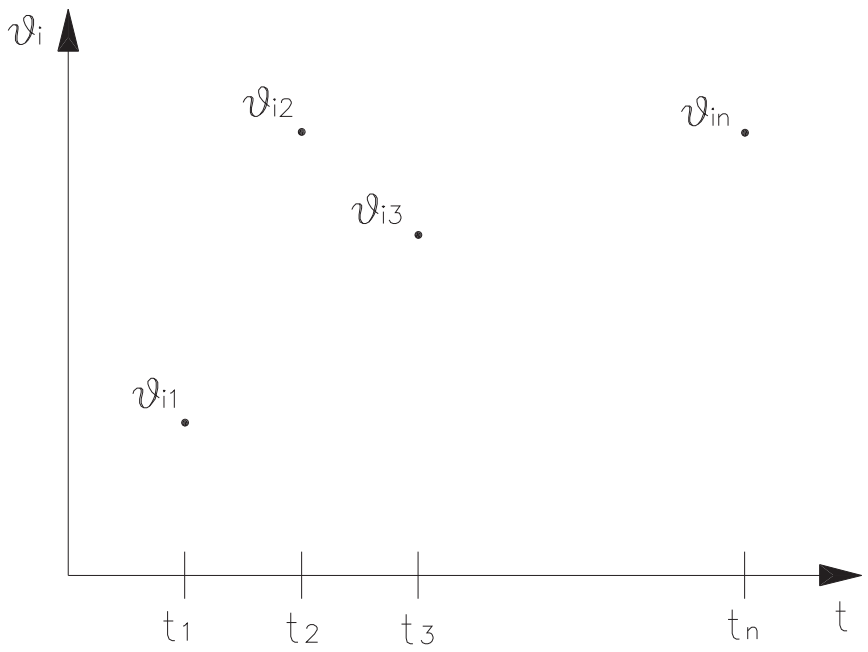
Questo vettore più in generale ha un numero di componenti pari al numero di g.d.l. della pinza: usualmente 6, 3 nel caso di robot planari.

1.3 Movimento punto-punto con i punti intermedi

1.3.1 Generalità

Si supponga di dover movimentare il robot da una posa iniziale X_1 a una posa finale X_n, imponendo però che la traiettoria passi per alcune pose intermedie X_j; questo modo di procedere è necessario ad esempio quando si devono evitare ostacoli presenti sul percorso $X_1 \to X_n$.
Per quanto concerne le pose intermedie, spesso è sufficiente che il robot passi vicino ad esse senza doverle necessariamente raggiungere in maniera esatta. In questo caso si può adottare la seguente strategia:

a) si risolve il problema cinematico inverso per ogni posa $X_1, X_2, \ldots, X_n$ ottenendo i valori degli angoli ai giunti (se essi sono rotatori) per ogni posa X_j; questo comporta che per ogni giunto i si ottiene un grafico del tipo indicato in figura, dove ogni ϑ_{ij} è detto nodo ed è l'angolo di rotazione del motore i quando la pinza assume la posa X_j;



b) si fissa il tempo per passare da X_i a X_{j+1} per ogni j, ad esempio adottando il criterio del tempo minimo di azionamento;

c) si sceglie un metodo per connettere i vari nodi in maniera esatta o approssimata (sotto-paragrafi successivi);

d) si calcolano per ogni giunto le equazioni che descrivono il moto di ciascun tratto della curva $\vartheta_i(t)$.

Si può notare come i punti a) e b) siano globali rispetto all'intero robot, mentre i punti c) e d) siano locali, cioè si opera separatamente per ogni grado di libertà. Con ϑ_{ij} si è indicato il valore di q_i al tempo t_j, nel seguito il pedice i verrà spesso sottinteso. Nei paragrafi successivi si esamineranno alcuni metodi per tracciare la curva $\vartheta_i(t)$ ($i = 1, 2, \ldots, m$ con m numero dei giunti).

1.3.2 Metodo con tratti di Rette e Parabole

Si supponga di aver scelto di collegare i vari nodi con tratti di retta raccordati da parabole con valori di curvatura (accelerazione) predefiniti che passino in prossimità dei nodi stessi (vedi figura 1.5).

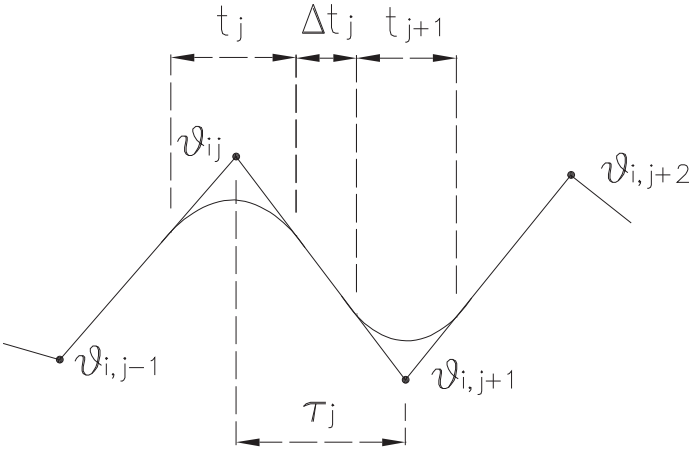


Figura 1.5: Legge di moto del tipo rette-parabole

Durante i tratti rettilinei ogni motore sarà mosso a velocità costante:

$$\dot{\vartheta}_j = \frac{\vartheta_{j+1} - \vartheta_j}{\tau_j} \tag{1.19}$$

Si fissi ora un tempo t_j incognito entro il quale raccordare i tratti rettilinei con una parabola che permette di avere accelerazioni non infinite nel punto ϑ_{ij}.

Si indichi con $|\ddot{\vartheta}_j|$ il modulo dell'accelerazione scelta per i tratti di raccordo. Un modo per determinare il tempo di raccordo t_j può essere il seguente:

$$t_j = \frac{\dot{\vartheta}_j - \dot{\vartheta}_{j-1}}{\ddot{\vartheta}_j} \tag{1.20}$$

dove:

$$\dot{\vartheta}_j = \frac{\vartheta_{j+1} - \vartheta_j}{\tau_j} \qquad \ddot{\vartheta}_j = sgn(\dot{\vartheta}_j - \dot{\vartheta}_{j-1})|\ddot{\vartheta}_j| \tag{1.21}$$

l'indice i è stato omesso per comodità di scrittura. Si dimostra che il tempo di raccordo viene automaticamente ripartito metà prima e metà dopo il nodo. Noti t_j e t_{j+1} è noto anche Δt_j da dedicare al tratto rettilineo:

$$\Delta t_j = \tau_j - \frac{t_{j+1}}{2} - \frac{t_j}{2} \tag{1.22}$$

Per i punti iniziali ϑ_{i1} e finali ϑ_{in} per i quali bisogna passare in maniera precisa e con velocità nulla, ci si comporta invece nel modo seguente:

a) punto ϑ_{i1}; per ottenere il tratto di parabola si considera un intervallo e un nuovo punto ϑ^*_{i1} che segue ϑ_{i1} di $\frac{t_1}{2}$ (figura 1.6). Come valore di ordinata si assume $\vartheta^*_{i1} = \vartheta_{i1}$. L'intervallo t da dedicare alla parabola si può determinare allora con un procedimento simile a quello visto per i punti interni della legge di moto. In questo caso si pone:

$$\dot{\vartheta}_1 = \frac{\vartheta_2 - \vartheta_1}{\tau_1 - \left(\frac{t_1}{2}\right)} = \ddot{\vartheta}_1 t_1 \qquad \Delta t_1 = \tau_1 - t_1 - \frac{t_2}{2} \tag{1.23}$$

da cui, risolvendo la prima equazione (di secondo grado), si ottiene:

$$t_1 = \tau_1 - \sqrt{\tau_1^2 - 2\frac{\vartheta_2 - \vartheta_1}{\ddot{\vartheta}_1}} \tag{1.24}$$

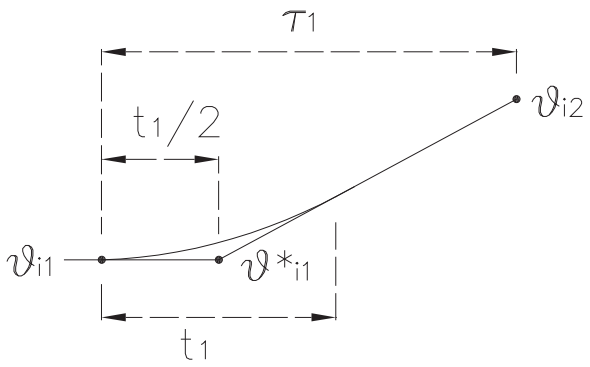


Figura 1.6: Costruzione del primo tratto di una legge del tipo rette-parabole

b) punto ϑ_{in}; allo stesso modo di ϑ_{i1} si può aggiungere un nodo ϑ^*_{in} che precede ϑ_{in} scrivendo poi (figura 1.7):

$$\dot{\vartheta}_{n-1} = \frac{\vartheta_n - \vartheta_{n-1}}{\tau_{n-1} - \left(\frac{t_n}{2}\right)} = -\ddot{\vartheta}_n t_n \qquad \Delta t_{n-1} = \tau n - 1 - \frac{tn-1}{2} - t_n \quad (1.25)$$

(infatti $0 - \dot{\vartheta}_{n-1} = t_n \ddot{\vartheta}_n$), da cui:

$$t_n = \tau_{n-1} - \sqrt{\tau n - 1^2 + 2\frac{\vartheta_n - \vartheta_{n-1}}{\ddot{\vartheta}_n}} \quad (1.26)$$

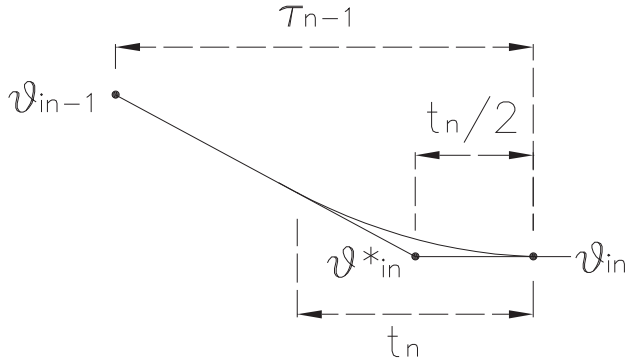


Figura 1.7: Costruzione dell'ultimo tratto di una legge del tipo rette-parabole

Nota: *Si è detto che molto spesso non è necessario passare esattamente nei punti intermedi; quando invece è necessario farlo si può procedere aggiungendo due punti α e β alla legge di moto in modo da realizzare una parabola con il vertice*

nel punto in cui si vuole il passaggio preciso. I nodi α e β devono essere assegnati in maniera tale che il punto per cui passare si trovi sulla loro congiungente, i gradi di libertà che rimangono possono essere sfruttati per assegnare la tangente cioè la velocità (figura 1.8).

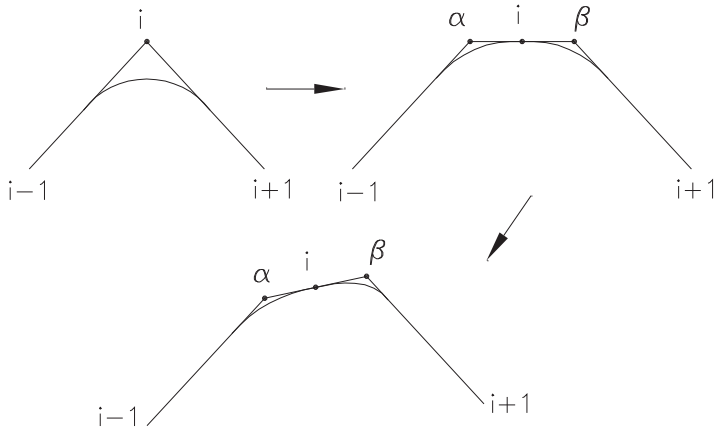


Figura 1.8: Aggiunta di nodi fittizi per passare esattamente per un punto.

Pregi e difetti dell'algoritmo

Tra i **pregi** si considerano:

- le funzioni $\vartheta(t)$ sono molto semplici in quanto sono rette o parabole (limitata complessità computazionale);
- durante il moto tra i punti i e $i+1$ è sufficiente conoscere informazioni solo sui nodi adiacenti al punto considerato (tabella seguente).

Nodi	Tratto
$i-1, i, i+1$	intorno del nodo i
$i, i+1$	retta tra i e $i+1$
$i, i+1, i+2$	intorno del nodo $i+1$

Tra i **difetti** si ha:

- discontinuità nel diagramma delle accelerazioni;
- per poter passare esattamente per un nodo occorre aggiungere due nodi fittizi.

1.3.3 Metodo delle splines cubiche

Un metodo alternativo per collegare i vari nodi è quello di utilizzare delle splines (cioè opportuni tratti di polinomi). Tale metodo consiste nel tracciare per ogni intervallo $\vartheta_{i,j}$, $\vartheta_{i,j+1}$ un polinomio di grado opportuno e nel dare condizioni di continuità al polinomio stesso (eventualmente anche sulla derivata) nei punti comuni a due intervalli contigui (figura 1.9). Per quanto riguarda il grado del polinomio vengono spesso utilizzate splines cubiche in quanto sono quelle che realizzano la curvatura minima e quindi riducono l'accelerazione, inoltre le splines cubiche garantiscono la continuità di spostamento, velocità e accelerazione; infine un aumento del grado del polinomio potrebbe portare a oscillazioni eccessive e non naturali nella funzione interpolante. Per la coordinata i-esima, in ogni intervallo $\tau_i = (t_{j+1} - t_j)$ si dovrà scrivere un polinomio del tipo:

$$S_j(t) = a_j t^3 + b_j t^2 + c_j t + d_j \tag{1.27}$$

il che comporta (poiché gli intervalli sono $n-1$) avere un numero di coefficienti incogniti (a_j, b_j, c_j, d_j) pari a $4(n-1)$. In questa equazione e nelle successive, per semplicità, l'indice i è sottinteso. Le condizioni da imporre sono:

- *condizioni al contorno:*

$$S_1(t_1) = \vartheta_1 \qquad S_{n-1}(t_n) = \vartheta_n$$

$$\dot{S}_1(t_1) = 0 \qquad \dot{S}_{n-1}(t_n) = 0$$

  cioè ogni motore dovrà rispettare precisi valori di posizionamento agli estremi oltre ad aver velocità iniziale e finale nulla. Ossia, per il primo [ultimo] tratto, vengono imposte le condizioni di passaggio per i due punti di contorno, più quella di velocità iniziale [finale] nulla e di una certa velocità nell'altro punto.

- *condizioni interne:* per ogni punto intermedio occorre imporre i valori di posizione nonché la continuità dei diagrammi della velocità e della accelerazione. Ossia, per i tratti intermedi, oltre alle due condizioni di passaggio per i punti di contorno, vengono imposte quelle di continuità di velocità e

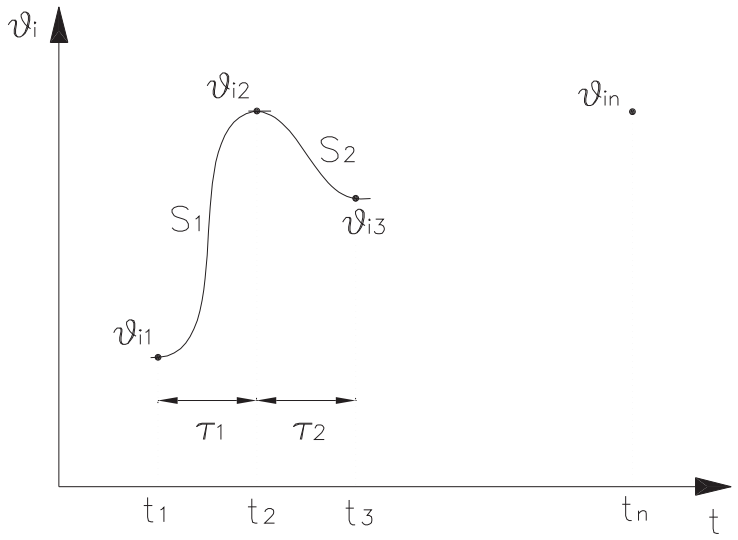


Figura 1.9: Interpolazione mediante tratti di splines.

accelerazione rispetto al tratto precedente:

$$S_{j+1}(t_{j+1}) = \vartheta_{j+1} = S_j(t_{j+1}) \qquad (doppia)$$

$$\dot{S}_{j+1}(t_{j+1}) = \dot{S}_j(t_{j+1})$$

$$\ddot{S}_{j+1}(t_{j+1}) = \ddot{S}_j(t_{j+1})$$

Il tutto equivale a un numero di condizioni pari a: $4 + 4(n-2) = 4(n-1)$. A questo punto il sistema risolvente è completo (numero di incognite pari al numero di equazioni) e si possono ottenere i coefficienti (a_j, b_j, c_j, d_j) per $j = 1, 2, dots, n-1$ risolvendo un sistema lineare. Il problema dell'algoritmo è che se il numero dei punti per cui passare è elevato, è necessaria una grossa capacità di calcolo.

Da un punto di vista computazionale è utile scrivere le splines nel modo seguente:

$$S_j(t) = \bar{a}_j(t-t_j)^3 + \bar{b}_j(t-t_j)^2 + \bar{c}_j(t-t_j) + \bar{d}_j \tag{1.28}$$

da cui derivando, si ottiene:

$$\dot{S}_j(t) = 3\bar{a}_j(t-t_j)^2 + 2\bar{b}_j(t-t_j) + \bar{c}_j \qquad \bar{c}_j = \dot{\vartheta}_j \tag{1.29}$$

$$\ddot{S}_j(t) = 6\bar{a}_j(t-t_j) + 2\bar{b}_j \tag{1.30}$$

Si nota che, imponendo le condizioni $S_j(t_j) = \vartheta_j$, si ottiene che $\bar{d}_j = \vartheta_j$; inoltre $\bar{c}_1 = 0$; cioè si ottengono immediatamente n coefficienti ed è possibile così ricondursi, per il calcolo degli altri coefficienti, a un sistema di dimensioni minori. Per ogni attuatore, le equazioni da scrivere per la determinazione dei coefficienti sono dunque:

$$
\begin{array}{ll}
\textit{nodo} & \bar{c}_1 = 0 \\
\textit{iniziale} & \bar{d}_1 = \vartheta_1 \\
 & \vdots \\
\textit{nodi} & \bar{a}_{j-1}(t_j - t_{j-1})^3 + \bar{b}_{j-1}(t_j - t_{j-1})^2 + \bar{c}_{j-1}(t_j - t_{j-1}) + \bar{d}_{j-1} = \vartheta_j \\
\textit{intermedi} & \bar{d}_j = \vartheta_j \\
 & 3\bar{a}_{j-1}(t_j - t_{j-1})^2 + 2\bar{b}_{j-1}(t_j - t_{j-1}) + \bar{c}_{j-1} = \bar{c}_j \\
 & 6\bar{a}_{j-1}(t_j - t_{j-1}) + 2\bar{b}_{j-1} = 2\bar{b}_j \\
 & \vdots \\
\textit{nodo} & \bar{a}_{n-1}(t_n - t_{n-1})^3 + \bar{b}_{n-1}(t_n - t_{n-1})^2 + \bar{c}_{n-1}(t_n - t_{n-1}) + \bar{d}_{n-1} = \vartheta_n \\
\textit{finale} & 3\bar{a}_{n-1}(t_n - t_{n-1})^2 + 2\bar{b}_{n-1}(t_n - t_{n-1}) + \bar{c}_{n-1} = 0
\end{array}
$$

Le $3n - 4$ incognite sono dunque:

$$x = [\bar{a}_1, \bar{b}_1, \bar{c}_1, \bar{a}_2, \bar{b}_2, \bar{c}_2, \ldots, \bar{a}_j, \bar{b}_j, \bar{c}_j, \ldots, \bar{a}_{n-1}, \bar{b}_{n-1}, \bar{c}_{n-1}]$$

1.3.4 Verifica dei vincoli di velocità e accelerazione

La verifica del rispetto dei vincoli di velocità e accelerazione è abbastanza onerosa indipendentemente dalla scelta del tipo di algoritmo di interpolazione (rette-parabole o splines). Per ottenere il rispetto dei vincoli occorre spaziare adeguatamente i vari nodi. In generale non è possibile trovare in maniera diretta quale sia la spaziatura che soddisfi i vincoli e che sia di durata il più breve possibile.

Nel caso di rette-parabole è evidente che il rispetto delle velocità per ogni tratto si ottiene imponendo

$$\tau_j \geq \frac{|\vartheta_{j+1} - \vartheta_j|}{|\dot{\vartheta}_{max}|} \tag{1.31}$$

e il rispetto delle accelerazioni massime può essere effettuato semplicemente imponendo l'accelerazione desiderata nei tratti di commutazione verificando che

due raccordi successivi non si sovrappongano[4] e aumentando opportunamente τ_j qualora ciò non avvenisse.

Nel caso di movimentazioni a più gradi di libertà si può inizialmente spaziare singolarmente ogni legge di ogni attuatore con il criterio appena citato, poi confrontare le leggi dei diversi attuatori e per ogni tratto scegliere la durata dell'attuatore più lento. Infine è necessario verificare che i vincoli siano soddisfatti per tutti i motori e in caso contrario dilatare l'intera legge o almeno i tratti vicini a quelli in cui i vincoli non sono rispettati. Si osservi che talvolta una modifica che impone il rispetto dei vincoli per un motore, può compromettere quelli di un altro e pertanto questo controllo ed eventuale scalatura va ripetuto iterativamente finché i vincoli sono rispettati per tutti gli attuatori. Se la scalatura è fatta correttamente, l'algoritmo converge.

Anche nel caso di interpolazione con splines occorre effettuare questa verifica iterativa sui vari tratti e sui vari giunti. Si osservi che l'accelerazione ha andamento lineare a tratti e pertanto può essere massima solo nei nodi, ed è in questi punti che occorre effettuare le verifiche. La velocità può essere massima nei nodi o nei punti in cui l'accelerazione cambia segno; la posizione può essere massima nei nodi o nei punti in cui la velocità cambia segno.

1.3.5 Confronto tra rette-parabole e splines

In base al contenuto dei sotto-paragrafi precedenti è possibile trarre le seguenti conclusioni:

- il metodo delle splines richiede maggiori quantità di calcoli perché è sempre necessario analizzare globalmente tutti i nodi da connettere mentre il metodo rette-parabole richiede solo l'uso di polinomi del primo e del secondo ordine i cui coefficienti possono essere determinati considerando al massimo due o tre nodi adiacenti;

- il metodo delle splines garantisce la continuità delle accelerazioni e quindi più dolcezza di movimento;

[4] $\frac{(t_j+t_{j+1})}{2} \leq \tau_j$

- dato che le leggi di moto create con splines hanno curvatura più dolce, a parità di tempo di esecuzione i valori massimi di velocità e accelerazioni ottenuti sono spesso maggiori di quelli relativi all'altro metodo; a parità di velocità e accelerazioni le splines richiedono un tempo di azionamento più elevato;

- la verifica del rispetto dei limiti di spostamento nelle rette-parabole è più facile: è sempre compreso tra i valori massimi e minimi di ϑ, mentre nel caso delle splines può sbordare;

- le splines passano esattamente per i punti imposti, per ottenere ciò con le rette-parabole è necessario collocare dei nodi aggiuntivi.

Capitolo 2

Il Robot e l'ambiente Epson

Un robot è un dispositivo che può ricevere istruzioni tramite comandi [3]. Nella forma in cui viene consegnato dai distributori esso non è programmato per compiere alcuna operazione. Vi sono aziende, note anche come *integratori di sistemi*, il cui lavoro consiste nel ricevere commesse dal cliente, acquistare robot, programmarli e gestire l'ambiente attorno ad essi in modo da consegnare un sistema pronto all'utilizzo. La programmazione del manipolatore avviene tramite software e linguaggi proprietari, sviluppati e distribuiti dall'azienda che li produce. Non esiste un linguaggio generico per la programmazione, ciascun produttore di manipolatori industriali implementa un codice proprietario necessario per impartire i comandi. Il manipolatore viene consegnato completo di software e manuali per la programmazione ed è frequente che il cliente abbia occasione di assistere ad una lezione introduttiva alla relativa programmazione.

Vi sono notevoli differenze fra codici sviluppati da aziende concorrenti, nonostante questi consentano di fatto di realizzare le medesime operazioni. Considerandoli da un punto di vista prettamente generale, differenti codici di programmazione sono simili tra loro: sono linguaggi che consentono di movimentare il robot, gestire segnali di input e output, comunicare con altri dispositivi, gestire processi paralleli. Nonostante ciò, ciascun codice proprietario è caratterizzato da una sintassi propria, differenti nomi di comandi ma, soprattutto, possibilità e limiti di utilizzo notevolmente diversi. La stesura di un codice per realizzare un desiderato movimento può risultare assai più semplificata utilizzando un codice

piuttosto che un altro. Il codice che ne risulta penalizzato potrebbe invece semplificare la questione in altri campi di utilizzo, quali la gestione di segnali I/O o la comunicazione con altri dispositivi.

Il linguaggio di programmazione può essere uno dei criteri di scelta fra marche differenti di robot. Poiché è stato utilizzato, presso il dipartimento di robotica dell'università di Padova, un manipolatore prodotto dalla Epson Seiko, la programmazione avviene utilizzando il linguaggio $SPEL+$. Nei prossimi paragrafi viene descritto l'apparato sperimentale impiegato per effettuare tali prove e il relativo software *Epson RC+ 7.0* per la gestione.

2.1 Il Robot e il controller

Il robot utilizzato è il modello C4 prodotto dalla Epson Seiko. E' il più recente manipolatore a 6 assi prodotto dall'azienda ed è distribuito in due versioni differenti: C4 e C4L. Il secondo è caratterizzato da uno sbraccio maggiore, 900 mm anziché 600 mm, a scapito della ripetibilità, $\pm 0,03$ mm contro i $\pm 0,02$ mm del modello C4 [4]. Una panoramica delle principali specifiche della serie C4 è riportata in figura 2.2. Il modello utilizzato è quello con sbraccio minore. Per la precisione è stato utilizzato un robot Epson C4-A601S come il modello riportato in figura 2.1.

Figura 2.1: Modello Epson C4 e controller Epson RC700 [4].

EPSON ROBOT MODEL		C4-A60***	C4L-A90***
Configuration		Articulated 6 Axis	
Mounting Configurations		Table Top, Ceiling	
Payload		Rated 1 kg / Max. 4 kg (5 kg*1)	
Repeatability		±0.02 mm	±0.03 mm
Axis Rotation	J1 (Turning)	±170 deg	
	J2 (Lower Arm)	-160 ~ +65 deg	
	J3 (Upper Arm)	+225 ~ -51 deg	
	J4 (Wrist Roll)	±200 deg	
	J5 (Wrist Bed)	±135 deg	
	J6 (Wrist Twist)	±360 deg	
Horizontal Reach	to mounting face	665	965
	to wrist center	600	900
Vertical Reach	to mounting face	885	1185
	to wrist center	820	1120
Speed	J1	450 deg/sec	275 deg/sec
	J2	450 deg/sec	275 deg/sec
	J3	514 deg/sec	289 deg/sec
	J4	555 deg/sec	
	J5	555 deg/sec	
	J6	720 deg/sec	
Cycle Time	1 kg workload	0.37 sec	0.47 sec
Allowable Moment	J4	4.41 N*m	
	J5	4.41 N*m	
	J6	2.94 N*m	
Moment of Inertia	J4	0.15 kg*m^2	
	J5	0.15 kg*m^2	
	J6	0.1 kg*m^2	
Motor Ratings	Axis 1	400 watts	
	Axis 2	400 watts	
	Axis 3	150 watts	
	Axis 4	50 watts	
	Axis 5	50 watts	
	Axis 6	50 watts	
Environment	Temperature	0 ~ 40 deg C	
	Humidity	20 ~ 80% RH (no condensation)	
	Other:	Keep away from: *Flammable/corrosive gases *Flammable/corrosive solvents *Water or oil and powder dust *Sources of electric noise	
Weight		27 kg	29 kg

Figura 2.2: Principali specifiche dei manipolatori industriali della serie C4 Epson Seiko [4].

Il controller associato al robot è l'Epson RC700, modello più recente della ditta e unico dispositivo che supporta il robot C4. Altri prodotti a 6 assi della stessa azienda consentirebbero invece il supporto da parte di diversi modelli di controller permettendo all'utilizzatore la relativa scelta. Ad esempio, sempre considerando i robot a 6 assi, sia la serie C3 sia la serie S5 sono supportate da entrambe le versioni RC180 e RC620.

La pinza montata sul manipolatore è progettata e realizzata da un'azienda per la quale il dipartimento sta realizzando un progetto commerciale. La terna utensile associata alla pinza è ottenuta tramite la trasformazione $(0, 0, 90, 0, 0, 0)$ applicata alla terna della flangia robot. Tale affermazione indica che la pinza ha un'estensione di 90 mm lungo l'asse z della flangia robot. Disponendo della terna utensile il controller è in grado di calcolare quale posizione della flangia consente di raggiungere la locazione desiderata e di conseguenza i valori in coordinate di giunto. Per questioni commerciali non viene riportata una foto dettagliata dell'end effector in questione.

2.1.1 Spazio di lavoro

Un manipolatore industriale garantisce le migliori prestazioni in corrispondenza di locazioni non in prossimità dei limiti dell'area raggiungibile. In figura 2.3 è riportato un estratto dello spazio di lavoro del robot Epson C4.

Dovendo verificare l'accuratezza del modello teorico con quello sperimentale, la scelta dei punti e dei movimenti da eseguire sono stati selezionati in maniera casuale. Nella cella in cui è installato il robot però non abbiamo completa libertà di movimento, in quanto sono presenti alcuni ostacoli. Pertanto si è dovuto prestare attenzione alla scelta dei punti, variando le quote e l'orientazione di specifici giunti, se necessario, per non danneggiare il robot.

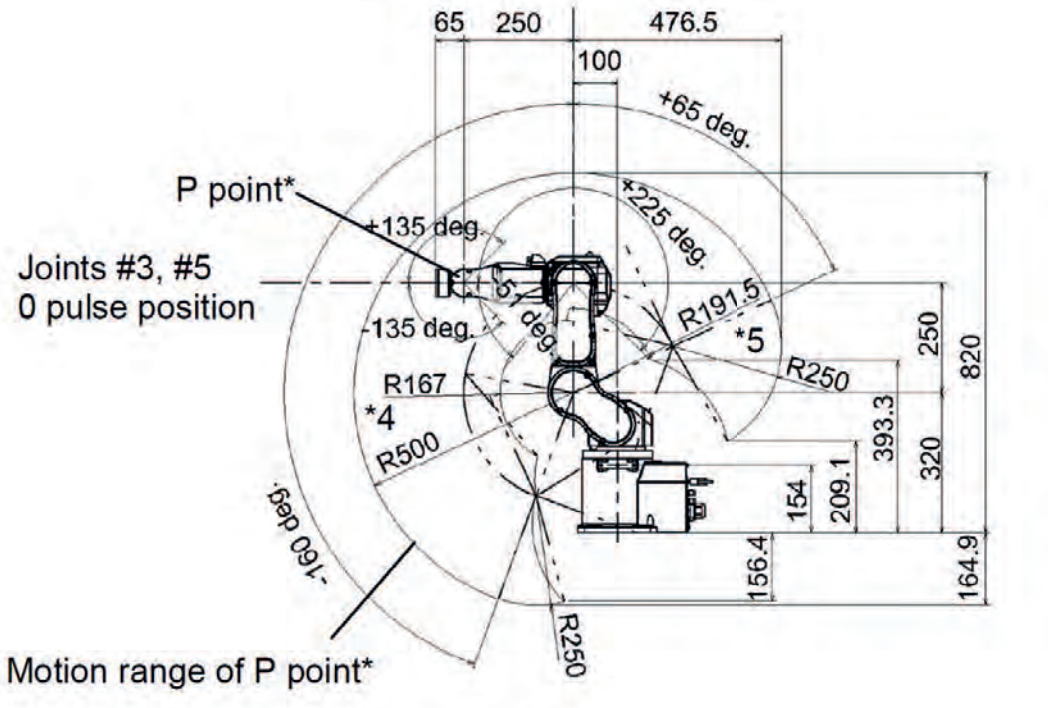


Figura 2.3: Schema dello spazio di lavoro raggiungibile dal manipolatore. Nel manuale sono riportate le restanti viste [4].

2.2 Ambiente Epson

Il software Epson RC+, interprete del linguaggio $SPEL+$ e interfaccia fra utente e controller, ci permette di configurare alcune impostazioni del robot, definire l'ambiente di lavoro e scrivere il programma di esecuzione. Di seguito verrà illustrato il software con una breve descrizione delle sue funzioni principali e alcuni dei codici principalmente usati per l'esecuzione dei movimenti, si rimanda ai manuali ufficiali rilasciati dai produttori per la descrizione dettagliata dei linguaggi di programmazione e la sintassi richiesta per ottenere un codice compilabile e correttamente eseguibile.

2.2.1 Software Epson RC+

Una volta lanciato, il programma risulta spoglio, come mostra la figura sottostante. Come prima operazione bisogna selezionare, attraverso l'apposito menù, quale tipo di collegamento stiamo utilizzando per comunicare col robot. Nel laboratorio di robotica per effettuare le prove è stata impiegata una connessione Ethernet.

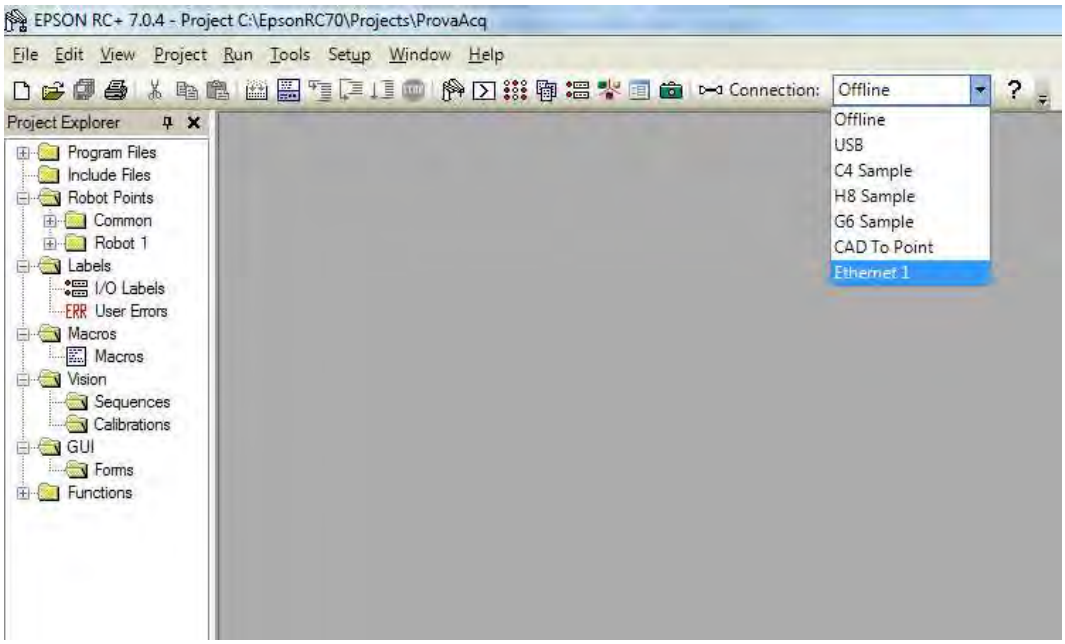


Figura 2.4: Scelta della connessione PC - controller.

Come si può notare dell'immagine precedente, di default il software crea anche delle macchine virtuali (C4 Sample, H8 Sample, G6 Sample) che permettono di

creare virtualmente il robot e di simulare l'esecuzione dei movimenti, dando così la possibilità all'utente di scrivere i codici dei comandi e verificarli anche senza la presenza fisica del robot. Nella figura 2.5 è mostrato in ambiente simulato il robot realmente installato nel laboratorio.

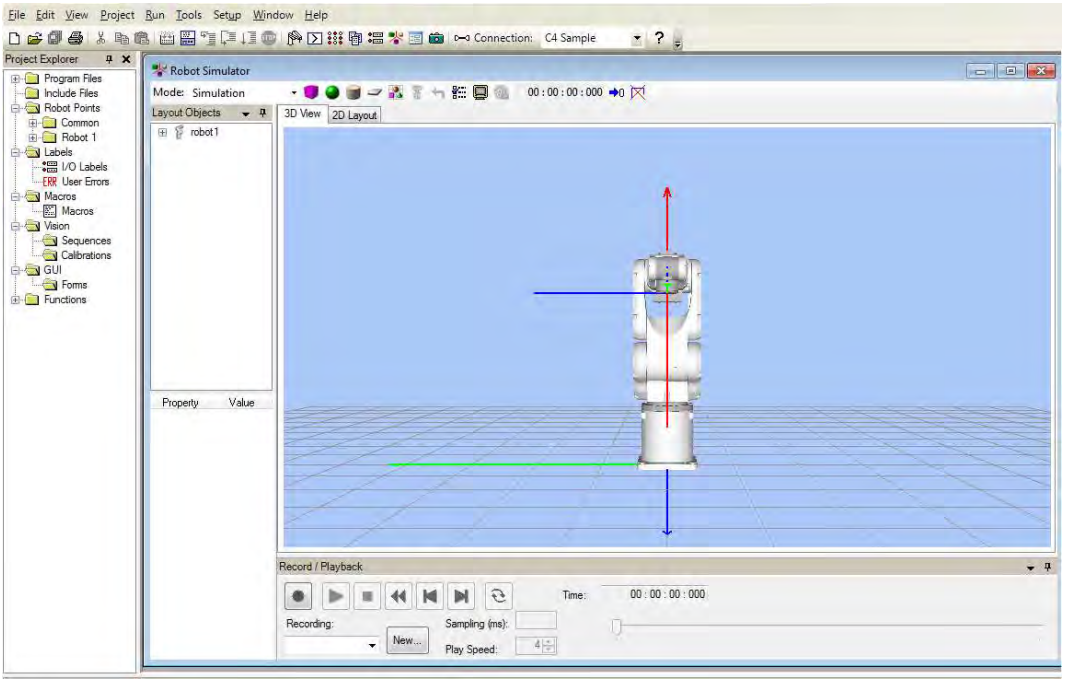


Figura 2.5: Robot C4 Sample simulato.

Definita la connessione si apre tramite il task rapido del pannello *Robot Manager*, che permette di impostare i parametri della terna utensile, del peso e dell'inerzia. Un altro task importante è quello *Jog & Teach*, raffigurato nell'immagine sottostante, il quale consente di muovere l'end effector nello spazio cartesiano o i singoli giunti in maniera precisa. Nella realtà le quote dei punti determinate in partenza non sempre sono quelle finali, a causa delle incertezze o degli errori, risulta perciò utile questa possibilità di muovere il manipolatore e salvare i nuovi punti una volta installato nell'ambiente di lavoro.

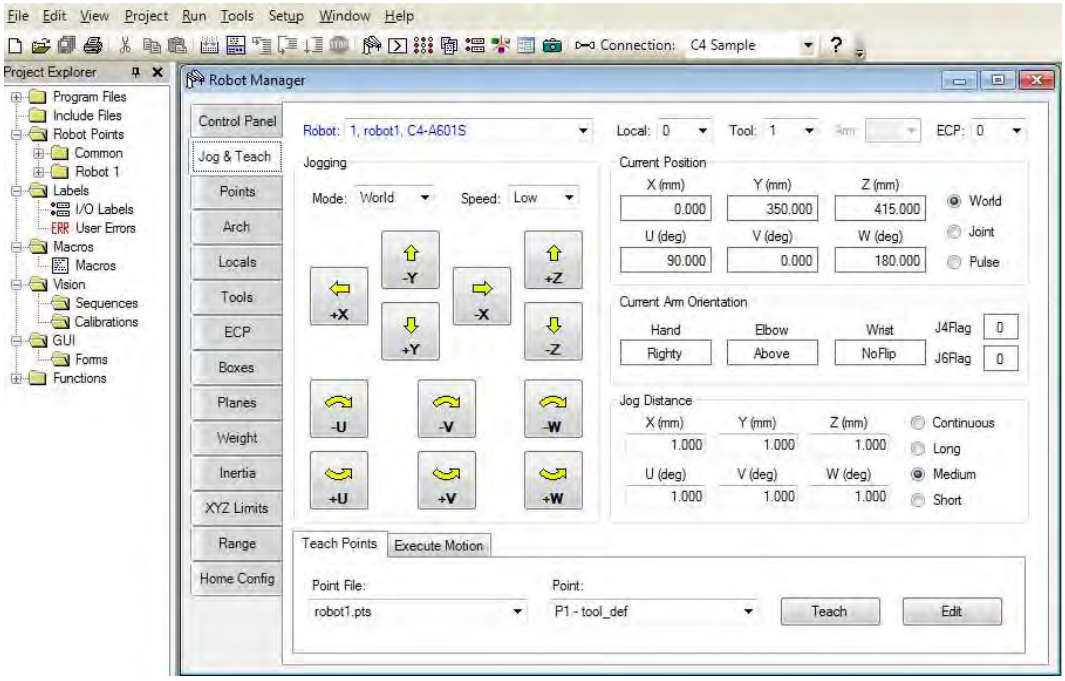


Figura 2.6: Task *Jog & Teach*.

2.2.2 Definizione delle locazioni

Con il termine *locazione* si intende una variabile adibita a rappresentare una posa del manipolatore. Si tratta di variabili tipicamente utilizzate nei comandi di movimento per indicare le destinazioni richieste. Nei linguaggi di programmazione robot le locazioni possono essere istruite tramite coordinate cartesiane o di giunto.

Una locazione fornita in coordinate di giunto consiste nell'insieme dei valori assunti da ciascun motore rispetto al proprio riferimento, tipicamente in gradi, e identifica un'univoca configurazione del manipolatore. Una locazione fornita in coordinate cartesiane, invece, consiste nell'indicazione delle coordinate dell'end effector in una terna di riferimento, non necessariamente quella utensile o quella relativa alla base robot.

Annesse alle coordinate cartesiane, indicate in mm, vi sono le coordinate dell'orientazione desiderata per l'end effector riportate secondo la convenzione scelta dal produttore.

Considerando manipolatori a 6 gradi di libertà con gli ultimi tre assi perpendicolari, rotoidali e concorrenti in un punto, caso tipico dei robot industriali, una

posa indicata in coordinate cartesiane può essere raggiunta con 8 configurazioni differenti, identificabili in forma chiusa [5].

In particolare è possibile raggiungere una determinata posizione con due differenti configurazioni della mano o, secondo alcuni autori spalla, indicate con i termini *righty* e $lefty$. Per ciascuna di esse vi sono inoltre due possibili configurazioni del gomito, *above* o *below*, e del polso, $flip$ o *no flip*. Una variabile di locazione definita in coordinate cartesiane contiene tipicamente anche informazioni riguardo alla configurazione desiderata.

Il software Epson RC+, rende disponibili all'utilizzatore le variabili P0 - P999 già dichiarate come locazioni. Indipendentemente dalla modalità di assegnazione dei valori della posa, ciascun punto P(i) contiene le coordinate dell'end effector nello spazio cartesiano, le indicazioni sulla configurazione assunta dal robot, un'etichetta e un'eventuale descrizione. Se l'assegnazione è stata effettuata in coordinate cartesiane e la configurazione non è specificata il software assegna automaticamente la posa Righty - Above - NoFlip: la configurazione salvata in una locazione può essere modificata in un momento successivo.

Tabella 2.1: Struttura delle variabili di locazione in $SPEL+$ [4].

Num	Lab	X	Y	Z	U	V	W	Loc	Hand	Elb	Wri	J4F	J6F	Desc

É possibile assegnare un'etichetta ad una locazione, ovvero associare un nome alternativo alla numerazione imposta da $SPEL+$. L'assegnazione di etichette non impone variazioni alla numerazione P(i), di conseguenza una stessa locazione è richiamabile sia tramite l'eventuale relativa etichetta sia tramite la numerazione già esistente.

Ciascun progetto genera un file con estensione *.pts* contenente le informazioni relative ai punti utilizzati. É consentito effettuare qualsiasi comune operazione tra file come creare una copia di salvataggio o importare un differente file *.pts*, operazioni eseguibili anche tramite comandi diretti. La GUI *Points manager* disponibile nel software EPSON RC+ consente inoltre di visualizzare e modificare il file in esame.

Il codice $SPEL+$ impone l'utilizzo di uno dei punti P0 - P999 per definire una locazione. Nella stesura di codici con un utilizzo massivo di differenti locazioni

la scelta del numero del punto a cui associare la nuova locazione può creare confusione, in quanto, anche lavorando con etichette, è necessario associare tale nome ad un punto specifico dell'elenco a disposizione.

I parametri J4Flag e J6Flag indicano le convenzioni di angolo assunte per le rotazioni dei giunti 4 e 6. Durante l'utilizzo di un robot a 6 assi è infatti possibile ottenere una medesima posa e configurazione anche se uno di questi giunti è ruotato di 360 gradi. Per distinguere le due situazioni in $SPEL+$ si sfruttano gli attributi JFlag. Un valore nullo di tale attributo indica che l'angolo del giunto in esame è compreso fra $-180°$ e $+180°$. Il valore unitario indica invece $-360° < \alpha < -180°$ o $180° < \alpha < 360°$. Il parametro è settato di default per ciascuna locazione pari a 0.

L'assegnazione delle coordinate nello spazio cartesiano avviene tramite il comando XY. La convenzione utilizzata per la terna (u, v, z) di rotazione corrisponde ad una prima rotazione attorno all'asse z locale ($roll$) seguita da una rotazione sull'asse y ($pitch$) e sull'asse x (yaw). JA impone invece l'assegnazione di coordinate espresse nello spazio dei giunti. Alcuni esempi sono riportati in figura 2.7.

```
P1 = XY(x,y,z,u,v,w) /L /B /F
P2 = JA(j1,j2,j3,j4,j5,j6)
PLABEL 3, "esempio"
HAND esempio, LEFTY
```

Figura 2.7: Assegnazione della locazione P1 in coordinate cartesiane con configurazione lefty - below - flip. Assegnazione della locazione P2 in coordinate di giunto e imposizione di una configurazione lefty alla locazione con etichetta esempio [4].

Sebbene in memoria ciascuna locazione sia gestita come un array di parametri in un preciso ordine la locazione in $SPEL+$ non è direttamente manipolabile come un vettore.

Non è possibile, ad esempio, inserire la definizione della locazione in un costrutto for specificando ad ogni ciclo una differente coordinata della posizione. In altre parole la sintassi $P1(i)$ non viene interpretata dal compilatore.

Per trasformare una locazione in vettore è necessario utilizzare più comandi distinti come riportato in figura 2.8. L'elenco di punti tuttavia è interpretabile come array, ovvero la sintassi P(i) è ammessa.

```
vettore = [CX(P1), CY(P1), CZ(P1), CU(P1), CV(P1), CW(P1)]
Go P1 +X(20) :Y(150) -TLZ(50)
```

Figura 2.8: Esempio di codice per trasformare una locazione in vettore.
Movimento verso un punto posizionato +20 mm rispetto al punto P1 in coordinate world (x), -50mm rispetto alla terna utensile (z) e ad una quota pari a 150mm rispetto alla terna world (y) [4].

Le operazioni consentite fra locazioni in $SPEL+$ si limitano alla semplice somma. L'espressione $P3 = P1 + P2$ effettua automaticamente la somma delle coordinate cartesiane e degli angoli di rotazione. Anche se le locazioni sono state definite in coordinate di giunto la somma così espressa viene effettuata in coordinate cartesiane, in accordo con quanto esposto in merito alla modalità di salvataggio in memoria delle locazioni.

É prevista inoltre la possibilità di usare offsets relativi esprimibili con i comandi $\pm[X, Y, Z, U, V, W]$ per gli spostamenti rispetto alla terna base robot e $\pm[TLX, TLY, TLZ, TLU, TLV, TLW]$ per quelli rispetto alla terna utensile. Per operare su una locazione modificandone le coordinate assolute invece si utilizza la sintassi: $[X, Y, Z, U, V, W]$.

In $SPEL+$ il calcolo di locazioni relative è indipendente dal modello di robot in uso. La nuova locazione mantiene le caratteristiche di quella che è stata elaborata per ottenerla.

2.2.3 Tipologie di movimento

Il movimento che può compiere un manipolatore per raggiungere una locazione, a parità di posizione di partenza, non è unico. Non vi è un solo motore in movimento, il moto dell'end effector è determinato dal moto di distinti motori che lavorano su assi differenti. Il moto lineare nello spazio cartesiano dell'end effector non è normalmente quello che consente le tempistiche minori, nonostante il percorso lineare fra due punti sia quello di lunghezza minore. Per mantenere tale traiettoria ciascun motore è costretto infatti ad effettuare rotazioni superiori a quelle strettamente indispensabili. Questa tipologia di moto è comunque contemplata dai differenti produttori in quanto utile per operazioni che richiedono linearità cartesiana, come la saldatura di un componente. A livello di controllo un moto di questo tipo risulta però notevolmente complesso e richiede elevate tempistiche computazionali.

Il movimento più rapido da effettuare per un manipolatore è invece quello che minimizza le rotazioni di ciascun motore. Il sistema di controllo, tramite analisi cinematica inversa, calcola le coordinate dei giunti nella posizione finale e associa una legge primitiva del moto per ciascuno di essi. Tali leggi vengono in seguito scalate temporalmente rispetto al motore che richiede il più elevato *tempo minimo* di azionamento. Il tempo minimo di azionamento dipende dalla rotazione complessiva richiesta al motore e dalle sue caratteristiche. Le tempistiche richieste sono le minime possibili, tuttavia il moto dell'end effector risultante è di difficile previsione. Tale tipologia di moto è nota come *point-to-point motion*.

In ambito robotico industriale è ampiamente utilizzato un moto combinato delle due modalità descritte. Tale soluzione è imposta dalla necessità di allontanarsi dalla posizione di partenza con moto lineare lungo l'asse z utensile per evitare collisioni. Un discorso analogo vale per il raggiungimento della posizione di arrivo desiderata. Il moto risultante è composto da un allontanamento lineare iniziale (depart), un moto point-to-point veloce intermedio e un avvicinamento lineare (approach) fino al raggiungimento della destinazione imposta. Esso è noto come *gate motion*.

Tale moto è ottenibile tramite combinazione dei movimenti base definiti precedentemente tuttavia, vista la frequenza di utilizzo in ambito industriale, disporre

del relativo comando già implementato nel linguaggio di programmazione semplifica la stesura del codice al programmatore. É questo il caso del linguaggio $SPEL+$ come si avrà cura di esporre nei prossimi paragrafi.

Un'ulteriore tipologia di moto ampiamente utilizzata nel lavoro con i manipolatori è il passaggio attraverso punti di via. Vi possono essere dei movimenti imposti al manipolatore dettati dalla sola necessità di evitare la collisione con un ingombro interno alla cella. Tali movimenti non richiedono il passaggio esatto per la destinazione indicata ma è richiesto il solo avvicinamento ad un punto, scelto spesso con ampia approssimazione. A livello pratico tale scopo viene raggiunto raccordando le traiettorie del movimento verso il punto di via e da lì verso quello successivo. Il moto risultante è indicato con il termine *continuous path*.

Le tipologie di movimentazione implementate dai produttori di robot sono quelle precedentemente esposte e rappresentate graficamente in figura 2.9. Variano tuttavia le opzioni che possono essere specificate nei comandi di moto e la flessibilità di utilizzo dei relativi comandi.

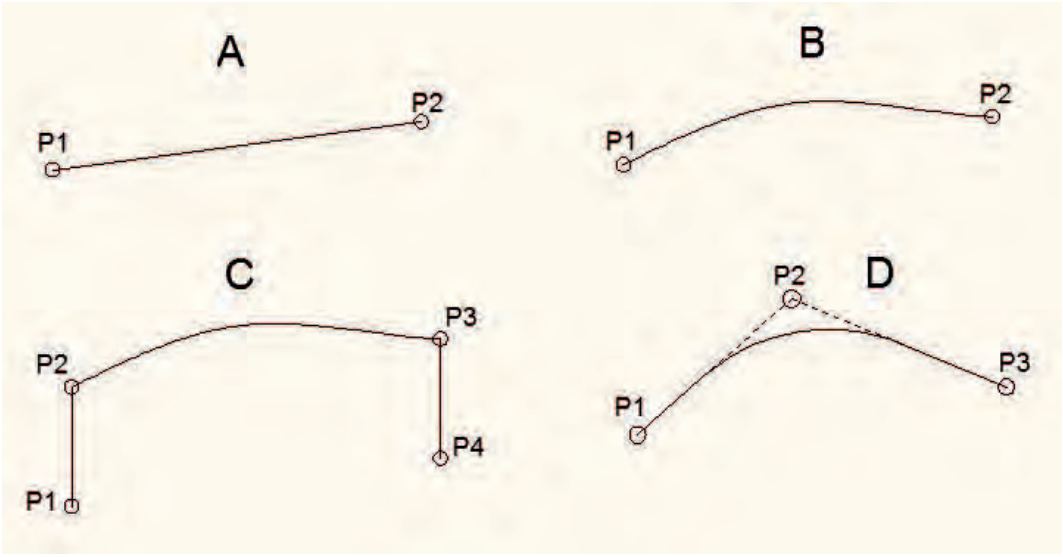


Figura 2.9: Rappresentazione grafica delle tipologie di moti implementate nei linguaggi di programmazione robot. A: linear, B: point-to-point, C: gate motion, D: punti di via.

Point-to-Point motion

Il movimento point-to-point è ottenuto tramite il comando `GO`, quello con interpolazione lineare nello spazio cartesiano invece con il comando `MOVE` Per ottenere un comando di tipo `GO` o `MOVE` è sufficiente inserire il punto target di destinazione che si vuole raggiungere. Il robot muoverà tutti i giunti simultaneamente, fino ad arrestarsi nel punto designato. Come si può notare dalle figure sottostanti, in tali comandi il *continuous path* è opzionale.

Figura 2.10: Sintassi per la scrittura dei comandi Go e Move [4].

Nel caso di due movimenti consecutivi, l'inserimento dell'opzione `CP` consente al manipolatore di non fermarsi una volta terminato il primo comando, ma di avanzare direttamente verso il secondo punto target raccordando le due traiettorie, rendendo così il primo punto target un punto di via.

Gate motion

L'opportunità più vantaggiosa offerta da $SPEL+$ è la presenza di tre distinti comandi che consentono di realizzare il gate motion. Sono comandi implementati nel solo linguaggio $SPEL+$ e consentono di semplificare notevolmente la stesura del codice per la movimentazione. Jump è il nome utilizzato per indicare il moto in esame.

Il comando `JUMP` è previsto solo per i manipolatori a 4 assi. Essendo questi robot di tipo SCARA il movimento lungo z è gestito da un solo asse e il controllo risulta semplificato: i moti di approach e depart vengono effettuati controllando un solo motore e trascurando i restanti, consentendo tempi di esecuzione ridotti. Il movimento intermedio avviene ad una quota z specificata precedentemente ed è di tipo PTP. É sufficiente indicare la quota z a cui effettuare il moto intermedio poiché il controller gestisce automaticamente il calcolo dei punti intermedi. Nel caso di manipolatori a 6 gradi di libertà i comandi analoghi sono invece `JUMP3`

e `JUMP3CP`. I movimenti di approach e depart possono avvenire in qualsiasi direzione nello spazio e il movimento intermedio è di tipo PTP nel caso di Jump3 e interpolato nel caso del Jump3CP. É necessario indicare entrambi i punti intermedi in questi casi vista l'assenza di limitazioni sulla destinazione dei moti lineari.

Alquanto pratica la possibilità di specificare il parametro arch durante un qualsiasi moto di tipo Jump. Tale approccio consente di raccordare i tratti lineari con quello intermedio specificando l'entità del raccordo: quello per il tratto di approach può essere differente da quello di depart. L'utilizzo dell'opzione arch è analogo a considerare i punti intermedi del Jump come punti di via, la sua utilità consiste nel poter effettuare il movimento intermedio ad una quota senza imporre che i movimenti lineari, lenti, avvengano fino al raggiungimento di essa.

I singoli comandi di movimento in $SPEL+$ consentono vaste opportunità di controllo del moto grazie ai numerosi argomenti facoltativi in coda allo stesso. É possibile ad esempio imporre operazioni di I/O parallele al movimento o, prima di iniziare il depart, valutare se completare il moto o meno a seconda dell'esito di una condizione specificata. La gestione dell'accuratezza con cui la posizione finale è considerata raggiunta non può tuttavia essere espressa all'interno del movimento, ma richiede un comando apposito (`FINE`).

La gestione di un numero limitato di punti di via può essere effettuata aggiungendo la dichiarazione `CP` ai singoli comandi di moto o con il comando `PASS`. Se il numero di punti è elevato è consigliato utilizzare la funzione di moto curvilinea. Per eseguire il comando `Jump3` è necessario definire il punto di depart, quello di appro ed il punto target finale. In opzione si può definire l'arco e il comando `CP`.

Per rendere più chiaro il movimento, si riportano le due immagini successive, la prima mostra un percorso passante per i punti di depart e approach, movimento senza la definizione del parametro *arch*. Nella seconda viene invece mostrato il percorso qualora si definisca il parametro.

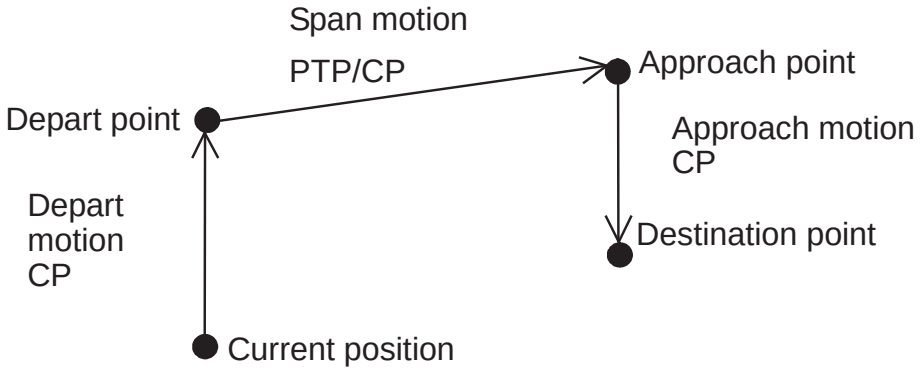


Figura 2.11: Rappresentazione grafica di un movimento Jump3 senza l'opzione arch [4].

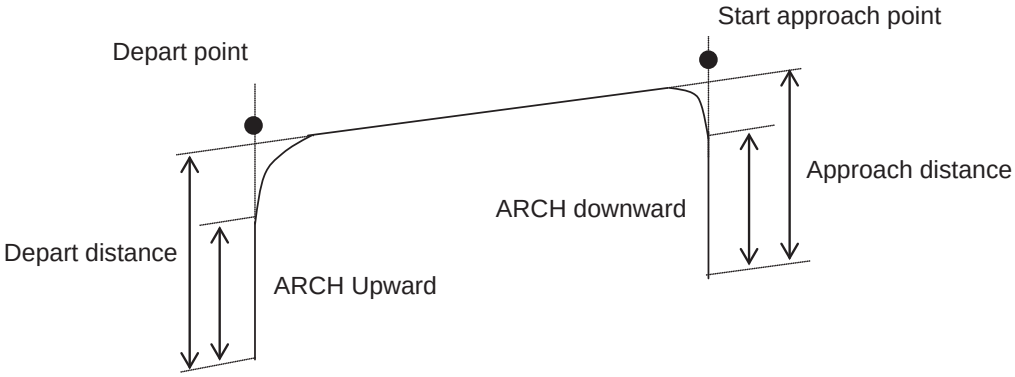


Figura 2.12: Rappresentazione grafica di un movimento Jump3 con l'opzione arch [4].

Nella figura sottostante viene riportato un esempio dell'utilizzo degli archi nel comando `Jump3`. Per prima cosa si definisce l'arco 0, con una quota di 20 mm di depart e 20 mm di approach. Con il comando `GO` il robot viene mandato in locazione P1 e successivamente viene definita la posizione P2.

L'esecuzione del comando `Jump3` si traduce in: avanzamento lineare verso il punto P2 (punto di depart), dopo aver percorso una quota di 20 mm inizierà a staccarsi dalla traiettoria lineare e si muoverà verso il punto P3 - 100 mm riferito alla terna utensile (punto di approach), ma prima di raggiungerlo raccorderà nuovamente la sua traiettoria e si dirigerà verso il punto P3 (target).

```
Arch 0,20,20
Tool 1
Go P1

P2 = P1 -TLZ(100)
Tool 2
Jump3 P2, P3-TLZ(100), P3 C0
```

Figura 2.13: Esempio di comando Jump3 [4].

Continuous Path

Si prenda in esempio due movimenti point-to-point consecutivi, la cui legge temporale è raffigurata in figura 2.14, il robot eseguirà i due spostamenti in maniera separata, terminato il primo comando passerà ad eseguire il secondo. Attivando il continuous path si permette al controller di eliminare la fase di decelerazione e di eseguire l'istruzione successiva. Così facendo il robot non passerà per il punto intermedio, che diventa così un punto di via, il movimento ne risulterà più fluido e si riduce il tempo per raggiungere la posizione finale. Il continuous path è un opzione utilizzabile nelle istruzioni di movimento `Arc,Go,Jump,Jump3` e `Move`. Può essere impostato sempre attivo anche all'inizio del programma, altrimenti di default è spento.

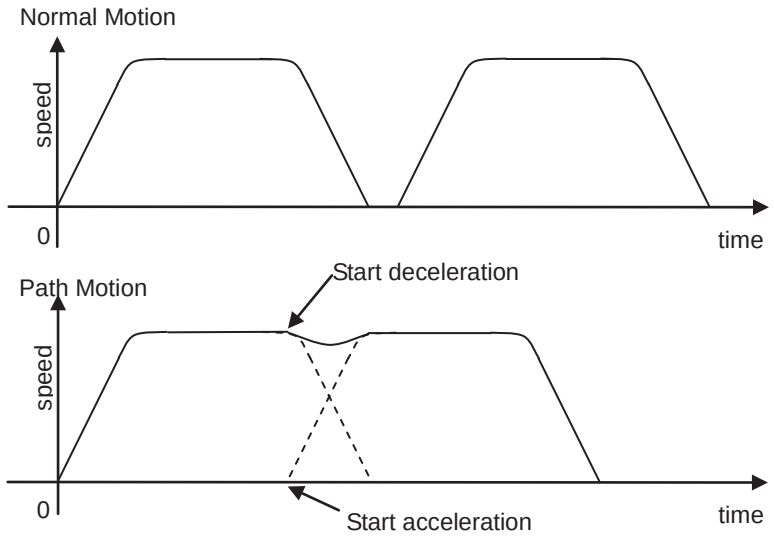


Figura 2.14: Esecuzione di due movimenti point-to-point con CP disattivato (sopra) e attivato (sotto) [4].

2.2.4 Velocità ed accelerazione

Prima di poter eseguire un qualsiasi movimento è necessario definire al robot quali siano la velocità e l'accelerazione massime che può utilizzare.

Attraverso il comando `SPEED` si specifica la velocità che il robot potrà disporre, essa è specificata come percentuale della velocità massima, ed il suo valore deve essere intero e compreso tra 1 e 100. `SPEED` 100 rappresenta la velocità massima possibile erogabile dal motore. Le velocità massime raggiungibili dai diversi giunti del nostro robot sono elencate nella figura successiva:

Item		Specification	
Model Number		C4-A601**	C4-A901**
Model Name		C4	C4L
Mounting type		Table Top mounting (Ceiling mounting) *1	
Weight (not include the weight of cables or shipping jigs)		27 kg (59.5 lb.)	29 kg (63.9 lb.)
Driving method	All joints	AC servo motor	
Max. operating speed *2	Joint #1	450 deg/s	275 deg/s
	Joint #2	450 deg/s	275 deg/s
	Joint #3	514 deg/s	289 deg/s
	Joint #4	555 deg/s	
	Joint #5	555 deg/s	
	Joint #6	720 deg/s	

Figura 2.15: Velocità massime raggiungibili per ogni giunto [4].

Attraverso il comando `ACCEL` invece si definisce la percentuale dell'accelerazione massima e decelerazione massima utilizzata per eseguire i movimenti. Il caso è analogo alla definizione della velocità, ma questa volta dobbiamo definire due valori interi compresi tra 1 e 100. Rispettivamente uno per la fase di accelerazione e uno per la fase di decelerazione. Nei manuali non vengono fornite indicazioni sui valori massimi di accelerazione e decelerazione. L'immagine successiva mostra un semplice programma dove l'accelerazione (Accel) e velocità (Speed) sono impostate attraverso variabili predefinite.

```
Function acctest
   Integer slow, accslow, decslow, fast, accfast, decfast

   slow = 20      'set slow speed variable
   fast = 100     'set high speed variable
   accslow = 20   'set slow acceleration variable
   decslow = 20   'set slow deceleration variable
   accfast = 100  'set fast acceleration variable
   decfast = 100  'set fast deceleration variable

   Accel accslow, decslow
   Speed slow
   Jump pick
   On gripper
   Accel accfast, decfast
   Speed fast
   Jump place
   .
   .
   .
Fend
```

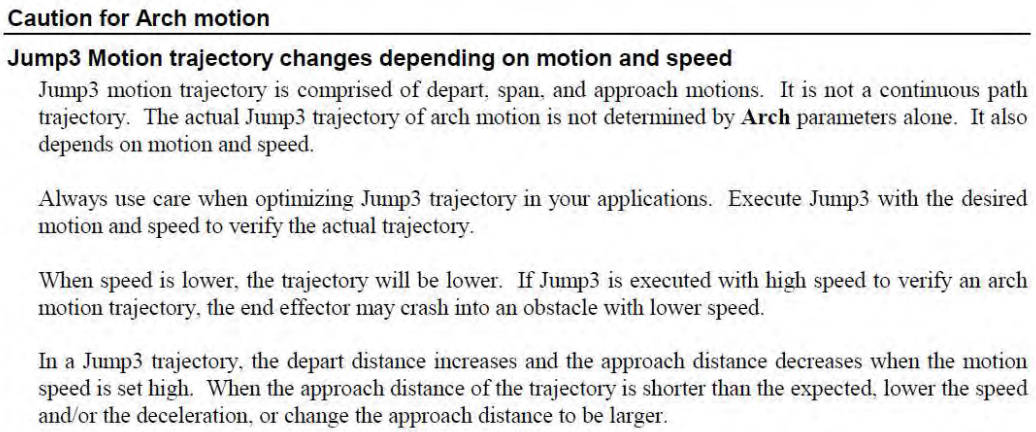
Figura 2.16: Esempio di utilizzo del comando Speed e Accel [4].

Capitolo 3

Analisi dei movimenti

Nel capitolo precedente sono stati illustrati alcuni movimenti semplici dei comandi `JUMP` e `GO`. Come si è potuto notare, il robot nell'eseguire tali movimenti non raggiunge tutte le locazioni che vengono definite, alcune sono punti di passaggio, per la precisione punti di via, nei quali il robot si limita a passarci vicino. Il problema che si riscontra nello studio delle traiettorie sta proprio nel capire come esso si comporta in prossimità di questi punti.

L'immagine successiva mostra parte delle avvertenze riportate nel manuale, riferite al movimento `JUMP`, ma valide anche per il `GO` qualora sia attivo il continuous path. Tali avvertenze richiamano all'attenzione del lettore che le traiettorie eseguite dal robot dipendono sia dal movimento che dalla velocità di esecuzione. Se un movimento viene eseguito inizialmente in slow power per accertarsi che non urti con gli ostacoli presenti, non è assicurato che salendo con la velocità il robot riesca ugualmente ad evitarli. Per completare i raccordi, essendo in velocità elevata, il controller anticipa il movimento.



Caution for Arch motion

Jump3 Motion trajectory changes depending on motion and speed

Jump3 motion trajectory is comprised of depart, span, and approach motions. It is not a continuous path trajectory. The actual Jump3 trajectory of arch motion is not determined by **Arch** parameters alone. It also depends on motion and speed.

Always use care when optimizing Jump3 trajectory in your applications. Execute Jump3 with the desired motion and speed to verify the actual trajectory.

When speed is lower, the trajectory will be lower. If Jump3 is executed with high speed to verify an arch motion trajectory, the end effector may crash into an obstacle with lower speed.

In a Jump3 trajectory, the depart distance increases and the approach distance decreases when the motion speed is set high. When the approach distance of the trajectory is shorter than the expected, lower the speed and/or the deceleration, or change the approach distance to be larger.

Figura 3.1: Avvertenze riportate nel manuale nell'utilizzo del continuous path [4].

Ignari di come il robot gestisca il passaggio per questi punti, si sono dovute eseguire delle movimentazioni con il manipolatore installato nel laboratorio di Robotica dell'università e sulla base di tali dati sperimentali definire un metodo che li potesse rappresentare restando entro un margine di tolleranza definita. Eseguendo le movimentazioni in ambiente simulato si sono riscontrati degli errori nell'acquisizione dei punti durante il movimento, la traiettoria risultava non continua in alcuni tratti, specialmente alle alte velocità, dovuta alla perdita di informazioni. Pertanto il metodo di stima della traiettoria, è stato realizzato basandosi solamente su dati ottenuti dal robot presente in laboratorio, ovvero attraverso movimentazioni realmente eseguite e non simulate, per una migliore affidabilità del risultato.

3.1 Dati sperimentali

Si è ipotizzato di studiare delle effettive possibili movimentazioni utilizzate in ambito industriale, immaginando di dover prelevare un oggetto da una locazione iniziale P1, percorrere un tratto lineare (non necessariamente verticale) fino a P2, successivamente passare per dei punti di via per evitare qualche ostacolo (P3,... P98) e terminare con un ulteriore tratto lineare per depositare l'oggetto nella locazione finale P99. Tale movimentazione in linguaggio Epson si traduce attraverso un comando `JUMP` iniziale, comandi `GO` tra i punti di via e un comando `JUMP` finale.

La figura successiva mostra parte del codice utilizzato per eseguire i movimenti col robot. Per semplicità sono raffigurati solamente i principali comandi utilizzati per ottenere le informazioni necessarie alla rappresentazione della traiettoria.

```
Function traiettorie
    Motor On
    Power High
    gain_speed(1) = 30
    gain_speed(2) = 60
    gain_speed(3) = 100
    racc = 0
    P0 = JA(0, 0, 0, 0, -90, 0)
    TLSet 1, XY(0, 0, 90, 0, 0, 0)
    Tool 1
    n_via_point = 3
        For i = 0 To UBound(gain_speed) - 1
            SetSpeed(gain_speed(i))
            For q = 1 To n_via_point - 1
                time1 = PTPTime(P(q), 0, 1, P(q + 1), 0, 1)
            Next
            Go P1
            Xqt printData
            Jump3 P2, P3, P3 C(racc) CP
            Go P3 CP
            Go P4 CP
            Go P5 CP
            Jump3 P5, P6, P7 C(racc) CP
            WaitPos
            Quit printData
        Next
Fend
Function printData
    i_t = 0
    Do While True
        Wait 0.01
            i_t = i_t + 0.01
            P99 = RealPos
            Print #data, Agl(1), Agl(2), Agl(3), Agl(4), Agl(5), Agl(6)
            Print #data, CX(P99), CY(P99), CZ(P99), CU(P99), CV(P99), CW(P99)
    Loop
Fend
```

Figura 3.2: Parte del codice utilizzato per eseguire i movimenti sperimentali.

Il nome della funzione è *traiettorie*, le prime righe sono sempre richieste in quanto servono per accendere i motori del robot e indicare le diverse velocità (nell'esempio: 30, 60, 100). Senza l'accensione dei motori e se non viene indicata la velocità, sempre intesa in percentuale rispetto a quella massima, il programma non viene eseguito, ma genera un messaggio di errore. Successivamente viene definito il valore del raccordo che vogliamo nel comando `JUMP`, parametro necessario affinché il robot raccordi la traiettoria in tale comando (nell'esempio è impostato C0). Di seguito viene definita la locazione P0, dove poter far stazionare il robot una volta terminato un ciclo di lavoro, la terna utensile per definire l'utensile montato sul robot ed infine il numero di via point che vogliamo.

Il ciclo *for* serve per eseguire il movimento per ogni velocità da noi impostata, affinché il programma esegua in automatico tutte le velocità senza dover ogni volta cambiare il parametro e rilanciare l'esecuzione. All'interno di tale ciclo viene eseguito il comando *PTPTime*, il quale restituisce il tempo che il robot impiegherebbe ad eseguire un point-to-point motion (`GO`). La necessità di tale dato serve a realizzare la pianificazione del movimento, ma nella pratica si vuole evitare di dover interrogare il robot, pertanto tale dato è stato estratto solamente per verificarlo con uno stimato da noi.

Il robot è dunque pronto per eseguire il movimento, esso inizia dirigendosi in P0, poi in P1, da qui esegue un comando `JUMP` senza la fase di approach, terminando in P3, si muoverà poi verso P4,P5 e termina con un comando `JUMP` senza la fase di depart, arrestandosi in P7. Raggiunta la posizione P1, si avvia in background la funzione *printData* la quale ha lo scopo di stampare in un file di testo specifico la posizione dell'end effector (prima in coordinate di giunto e poi in quelle cartesiane) con una frequenza di 0,01 secondi. Questo ci permette di avere la traiettoria che realmente il manipolatore ha eseguito.

Tale traiettoria deve poi essere confrontata con quella senza il raccordo, che corrisponde alla traiettoria passante fisicamente per sette punti da noi definiti. Inizialmente quest'ultima veniva ottenuta facendo eseguire al robot tutto il movimento, ma significava ripetere ulteriormente il programma. Solo dal grafico rappresentante la legge temporale dei giunti si è notato che tale traiettoria può essere ottenuta unendo i valori dei giunti di ogni punto tramite tratti lineari.

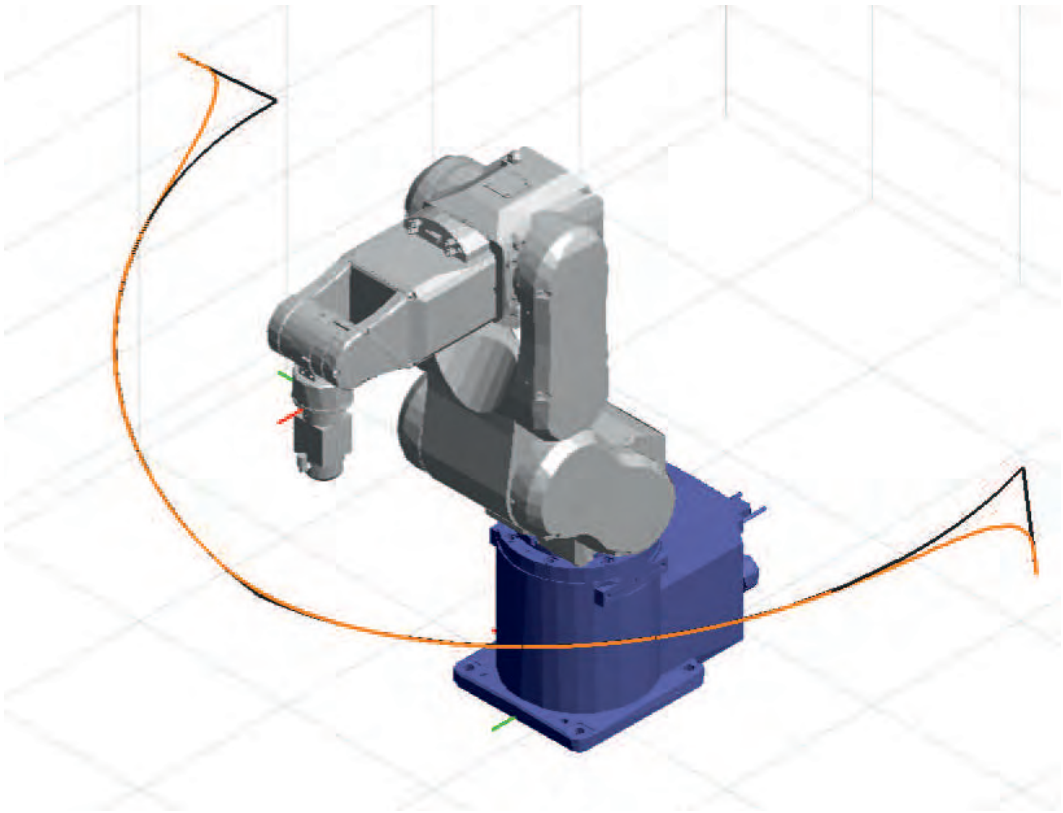


Figura 3.3: Raffigurazione in Matlab di un movimento a velocità massima con un punto di via. In arancione quella eseguita dal robot, in nero quella primitiva.

Per visualizzare e analizzare la traiettoria è stato necessario utilizzare il software di calcolo Matlab, il quale grazie alla sua potenza di calcolo ci permette di elaborare una grande quantità di dati e plottare le traiettorie in questione in un grafico 3D, un esempio di tale grafico è mostrato in figura 3.3.

Oltre a tale traiettoria disponiamo anche dell'andamento temporale di ogni giunto, pertanto possiamo raffigurare tali posizioni nel tempo, che essendo diverso per ogni velocità è stato normalizzato al valore unitario. Un esempio complessivo di tutti i giunti è raffigurato in figura sottostante. Per maggiore chiarezza si osservi la figura 3.5, la quale mostra a titolo di esempio l'andamento del singolo motore 4. Per ogni motore è riportato col colore celeste l'angolazione reale, mentre i tratti lineari rossi servono per congiungere tra loro i punti di via.

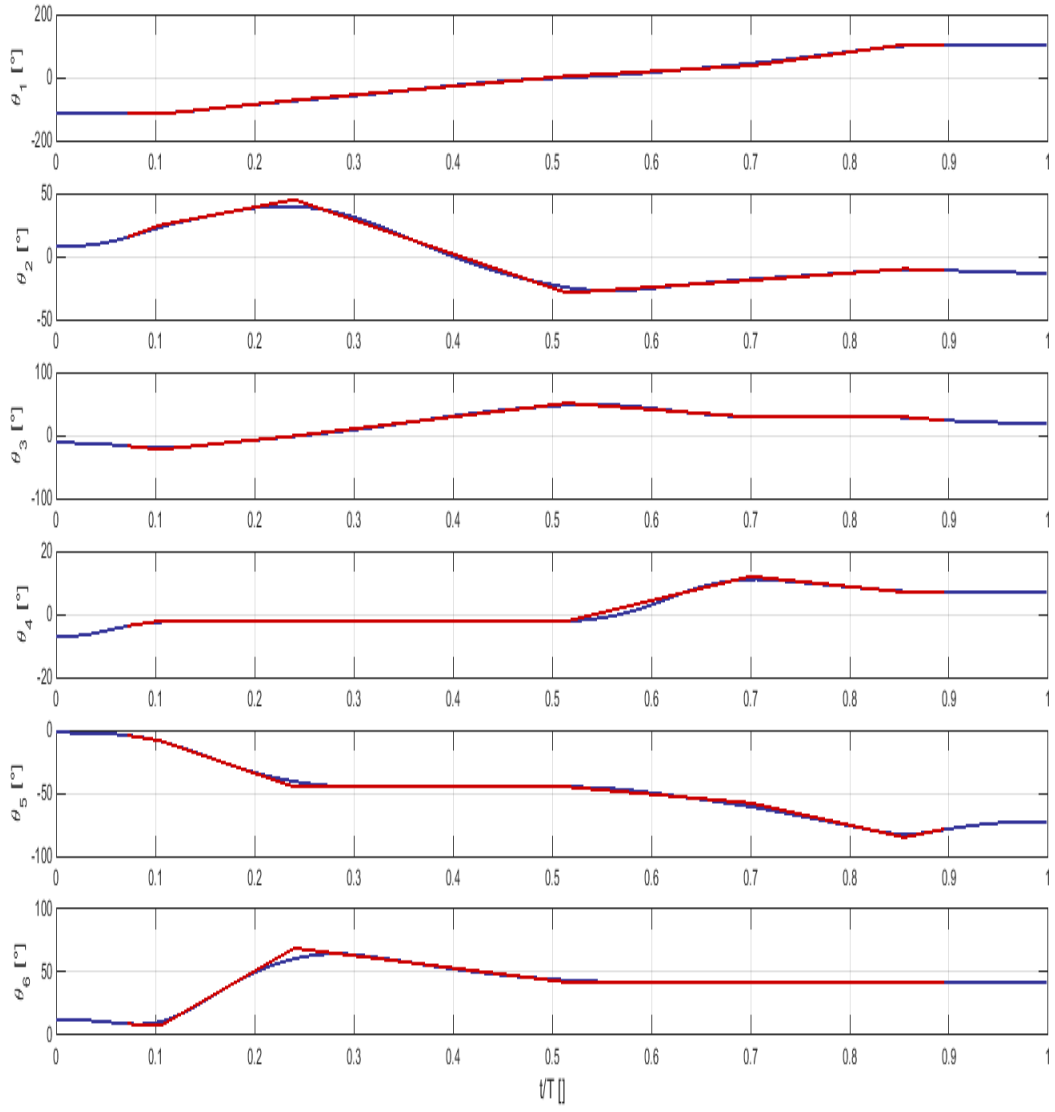


Figura 3.4: Andamento temporale di ogni giunto del movimento raffigurato in figura 3.3. Il tempo è stato normalizzato per una migliore rappresentazione.

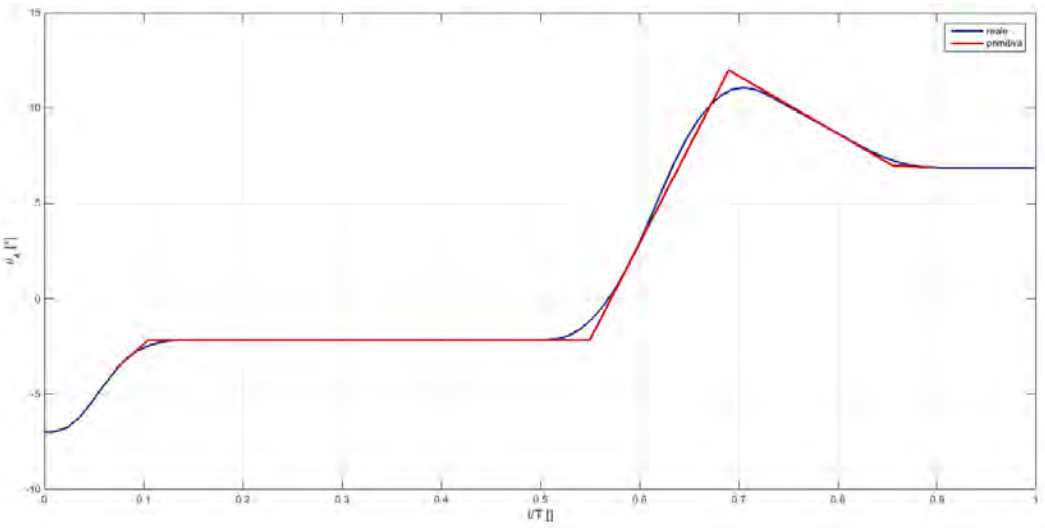


Figura 3.5: Ingrandimento dell'andamento temporale del giunto 4 precedentemente raffigurato.

3.2 Analisi di ripetibilità

Nelle caratteristiche del robot viene dichiarata una ripetibilità di $\pm$ 0,02 mm, ma quest'ultima è riferita al raggiungimento dei punti target. Per assicurarci che il manipolatore esegua sempre la stessa traiettoria, si è eseguito un test di ripetibilità, nel quale un movimento è stato ripetuto dieci volte per ogni velocità da noi decisa a inizialmente. Per coprire in maniera globale tutto il range della velocità sono stati decisi i seguenti valori in percentuale: 5, 10, 15, 25, 35, 50, 60, 70, 80, 90, 100.

Successivamente per ogni velocità si sono confrontate le dieci traiettorie, per ogni punto i-esimo si è calcolato il valore medio e ricavato lo scarto quadratico medio attraverso le formule seguenti:

$$\bar{x_i} = \frac{1}{N} \sum_{l=1}^{N} x_l$$

$$\sigma_i = \sqrt{\frac{\sum_{l=1}^{N} (x_l - \bar{x_i})^2}{N-1}}$$

Nella tabella successiva vengono riportati esclusivamente i valori massimi dello scarto quadratico medio, risultanti tra tutti i punti delle traiettorie alle diverse

velocità analizzate.

Velocità	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
%	[°]	[°]	[°]	[°]	[°]	[°]
5%	0.188	0.158	0.132	0.068	0.173	0.181
10%	0.317	0.237	0.240	0.131	0.284	0.230
15%	0.302	0.219	0.199	0.092	0.232	0.187
25%	0.413	0.278	0.195	0.120	0.241	0.289
35%	0.430	0.386	0.271	0.092	0.245	0.413
50%	0.564	0.323	0.368	0.239	0.273	0.457
60%	0.877	0.854	0.600	0.290	0.381	0.641
70%	0.740	0.530	0.383	0.247	0.463	0.778
80%	1.211	1.001	0.807	0.519	0.474	0.650
90%	1.066	0.928	0.652	0.328	0.699	1.177
100%	1.441	1.045	0.734	0.426	0.931	1.565

Tabella 3.1: Valori massimi dello scarto quadratico medio riscontrati nelle traiettorie per le diverse velocità.

Verificato così la ripetibilità della traiettoria è stato possibile limitare, ad uno soltanto, il numero di movimenti da eseguire per ogni velocità. Consentendo un risparmio in termini di tempo e di elaborazione dei risultati, evitando anche di sforzare eccessivamente il robot.

3.3 Modello analitico

Dopo aver eseguito un numero sufficiente di movimentazioni si è cominciato analizzando i grafici ottenuti per ricavare più informazioni possibili da essi.

Dal grafico tridimensionale si nota chiaramente che al variare della velocità la traiettoria si modifica, precisamente i punti di inizio e fine raccordo vengono anticipati, in accordo con le avvertenze dichiarate nel manuale e citato in precedenza. Un'illustrazione di tale variazione è mostrata nelle figure successive, nelle quali è raffigurato il passaggio per il medesimo punto di via a velocità differenti. Col colore arancione si è rappresentato il movimento del robot, mentre in nero si ha il

passaggio preciso per il via point. In ambito industriale il robot viene impiegato spingendolo verso le velocità più elevate, per ridurre i tempi di spostamento e aumentare il rendimento complessivo. Ricercare dunque traiettorie per velocità inferiori al 50% può sembrare improduttivo, ma lo scopo di questa tesi è quello di poter rappresentare al meglio una generica movimentazione senza limitare la scelta delle capacità del manipolatore. Tutte le prove eseguite in laboratorio sono state realizzate utilizzando i valori di velocità utilizzati per il test di ripetibilità.

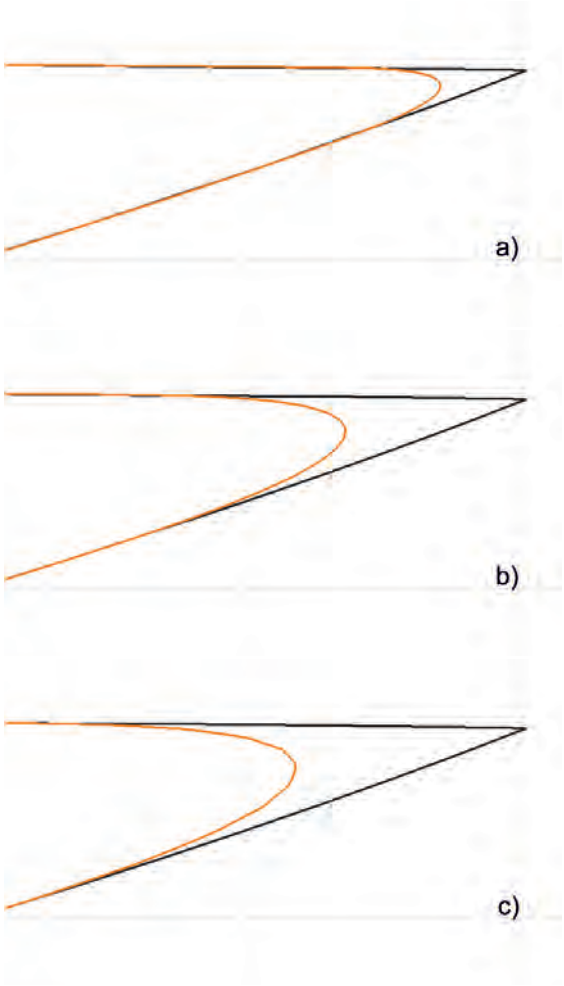


Figura 3.6: Passaggio in prossimità di un via point a differenti velocità.
a) 5%, b) 50% e c) 100%.

Un ulteriore accorgimento ricavato sempre dalla figura 3.4 è che nel comando `Jump` il raccordo non è della stessa entità di quello nei via punt, inoltre in tale movimento uno dei due punti in cui la traiettoria cambia è a noi noto perché fissato dal parametro arco. Si è preferito quindi studiare separatamente la pianificazione dei movimenti nei due comandi per poi unire le soluzioni ottenute.

Osservando il grafico dei giunti, in accordo con quanto detto nel primo capitolo, la legge utilizzata dal robot per la pianificazione di essi ricorda il metodo con tratti di rette e parabole. In accordo con tale ipotesi si è proceduto cercando di verificarla, andando a stimare i punti di inizio e fine raccordo.

I risultati riportati come esempi in questo capitolo fanno riferimento al movimento riportato nell'immagine 3.7, il quale prevede la partenza dalla locazione P1, P2 è il punto di depart, P5, P6 e P7 sono i via point, P3 è il punto di approach e P4 è il target finale da raggiungere. Nella figura si è voluto rappresentare il robot nella configurazione iniziale e quella finale per una migliore rappresentazione del movimento.

Di seguito sono mostrate, a titolo di esempio, le immagini relative al percorso eseguito dalle ulteriori movimentazioni che sono state realizzate. Nella scelta dei punti non si è fatta particolare attenzione, se non quella di verificare il corretto raggiungimento di essi senza urtare elementi nella cella di lavoro. Si è cercato di variare il più possibile la tipologia di movimento, variando i punti e le lunghezze di depart e approach. Sono stati ripetuti anche più volte gli stessi movimenti, ma cambiando il parametro dell'arco per valutarne l'influenza.

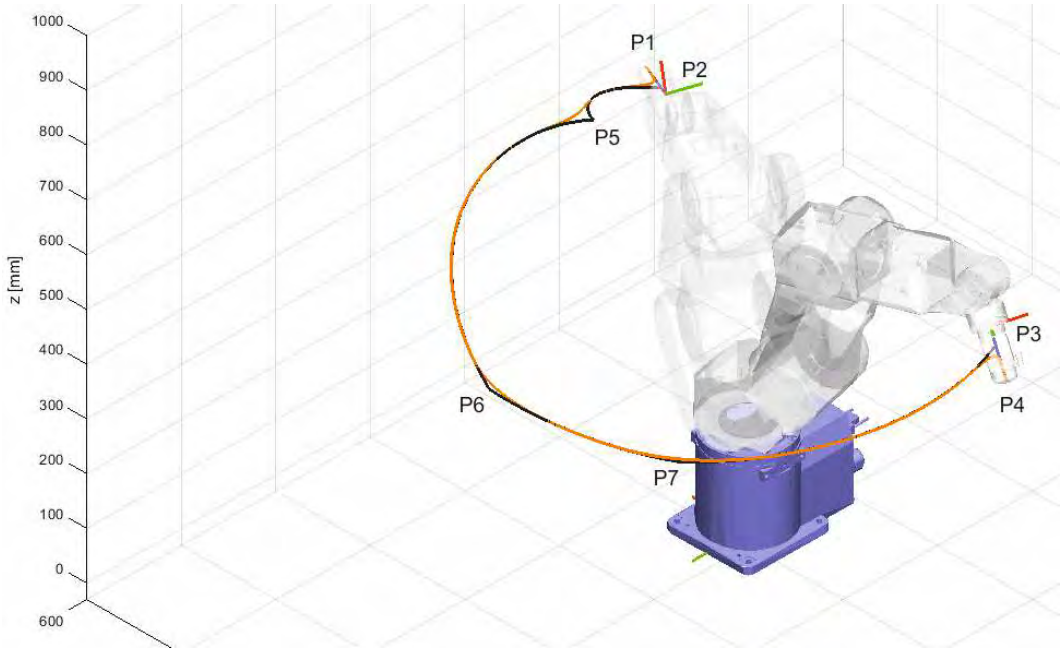


Figura 3.7: Traiettoria presa come riferimento per l'esposizione dei risultati ottenuti.

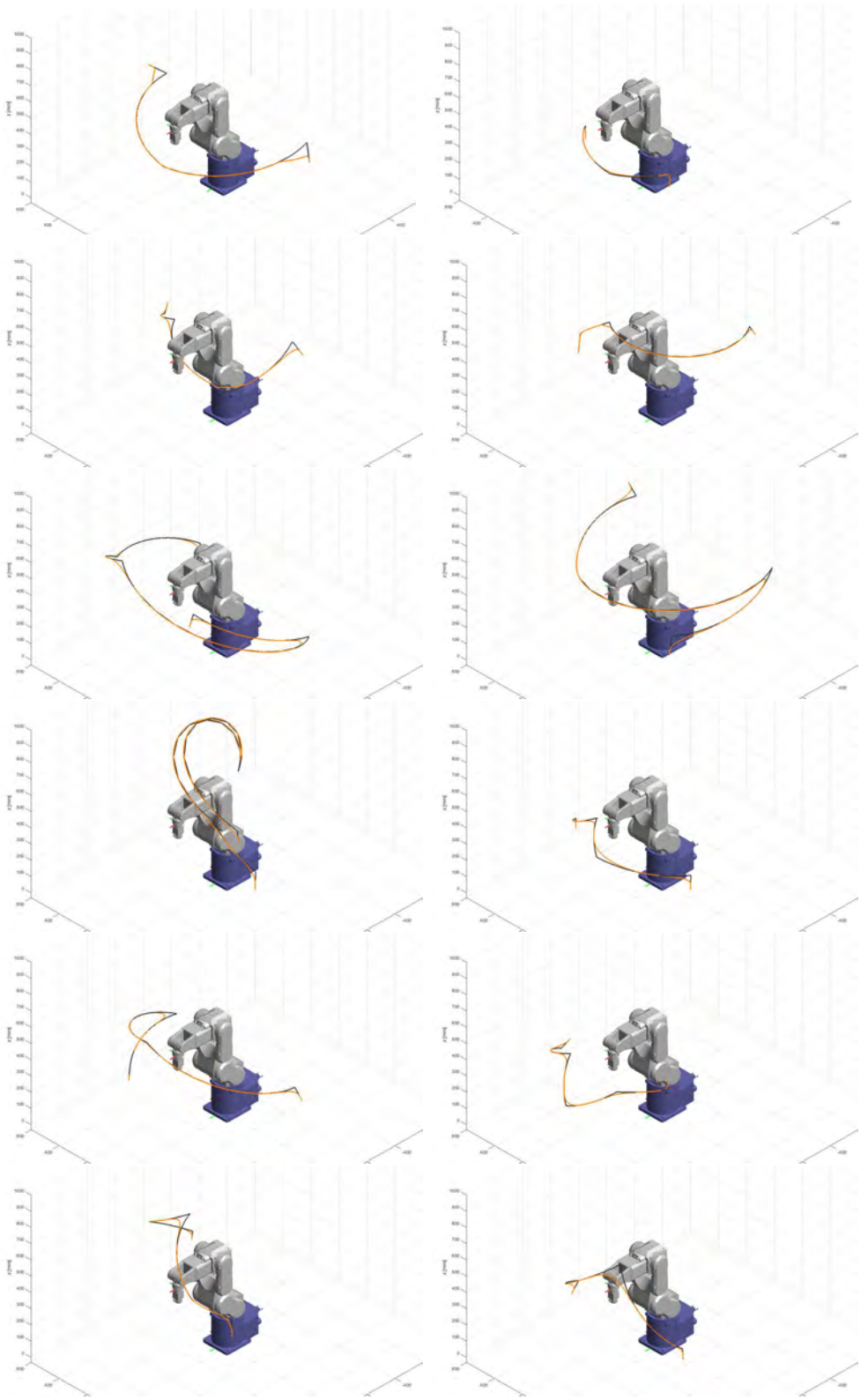


Figura 3.8: Ulteriori movimenti eseguiti in laboratorio.

3.3.1 Passaggio punti di via

Focalizzandoci esclusivamente sul tratto intermedio del movimento complessivo, come prima approssimazione dei punti da individuare, è stata avanzata l'ipotesi che essi fossero quelli di inizio e fine tratto a velocità costante. Per ricavarli si è dovuto calcolare la pianificazione, ma non avendo l'accelerazione di cui il robot dispone, è necessario avere il tempo totale di esecuzione del movimento. L'unico modo per ottenere il valore reale, è utilizzare il comando $PTPTime$ in Epson, il quale restituisce il tempo impiegato per un singolo movimento `GO` tra due locazioni. É necessario disporre di tali tempi per tutti i movimenti intermedi, solo così poi, utilizzando le formule per la legge a velocità trapezoidale viste nel paragrafo 1.1.2, riusciamo a calcolare la pianificazione dei giunti. Con essa siamo in grado di ottenere le posizioni, le velocità, l'accelerazione e i tempi di accelerazione e decelerazione.

Questo problema limita la possibilità di avere una risposta rapida sulla stima della traiettoria senza eseguire il programma Epson. É possibile ricavare attraverso formule analitiche una stima del tempo globale di esecuzione, questo ci consente di aggirare il problema precedente. Per poter eliminare l'incognita sul tempo si deve definire la durate del tratti ad accelerazione e decelerazione costanti e quello a velocità costante. Tale durata è stata ipotizzata uguale per tutti e pari a un terzo del tempo totale.

$$\Delta s = v_{max}\Delta t_2 + 2\frac{v_{max}}{2}\Delta t_1$$

$$\Delta t_1 = \Delta t_2 = \frac{T}{3} \qquad v_{max} = v_{100} * speed$$

$$T = \frac{3\Delta s}{2v_{max}} \tag{3.1}$$

I tempi così stimati sono stati confrontati con quelli dati dal controller in precedenza per valutarne l'affidabilità e verificare l'ipotesi precedente. La tabella successiva mostra i risultati ottenuti e quelli reali di un movimento contenente tre punti di via (P5 - P6 - P7).

	Velocità	P2 - P5	P5 - P6	P6 - P7	P7 - P3
	%	[s]	[s]	[s]	[s]
Reale	5%	2.303	3.836	1.712	3.167
Teorico	5%	2.418	3.894	1.862	3.282
Reale	10%	1.223	2.013	0.906	1.622
Teorico	10%	1.322	2.214	1.005	1.681
Reale	15%	0.851	1.390	0.630	1.101
Teorico	15%	0.873	1.395	0.636	1.108
Reale	25%	0.545	0.887	0.403	0.680
Teorico	25%	0.551	0.894	0.415	0.68
Reale	35%	0.410	0.655	0.302	0.496
Teorico	35%	0.419	0.683	0.314	0.537
Reale	50%	0.304	0.482	0.224	0.357
Teorico	50%	0.293	0.506	0.220	0.372
Reale	60%	0.262	0.413	0.193	0.302
Teorico	60%	0.244	0.422	0.183	0.328
Reale	70%	0.232	0.363	0.170	0.263
Teorico	70%	0.219	0.361	0.157	0.276
Reale	80%	0.208	0.325	0.153	0.238
Teorico	80%	0.183	0.316	0.137	0.249
Reale	90%	0.190	0.295	0.143	0.224
Teorico	90%	0.183	0.281	0.129	0.238
Reale	100%	0.174	0.271	0.138	0.214
Teorico	100%	0.156	0.253	0.120	0.223

Tabella 3.2: Confronto tra i tempi reali impiegati dal robot con quelli determinati teoricamente ipotizzando una pianificazione ad accelerazione costante.

L'errore riscontrato non incide sulla traiettoria primitiva nello spazio cartesiano, ciò che comporta è una diversa posizione del robot in un determinato istante. Pertanto utilizzando il tempo stimato possiamo calcolare l'andamento della posizione, velocità e accelerazione di ogni giunto e ricavare i punti di nostro interesse.

I punti reali in cui il robot modifica il suo percorso sono stati ricavati utilizzando le capacità di Matlab. Avendo constatato che parte della traiettoria combacia con la primitiva è stata realizzata una funzione la quale ricava la distanza tra due percorsi. I due percorsi contenenti il medesimo numero di punti vengono elaborati con Matlab, il quale calcola la distanza per ogni punto i-esimo di entrambe le traiettorie, facendone il modulo. Quando il distaccamento supera una determinata soglia da noi definita, significa che il robot ha iniziato il passaggio per il via point. Analogamente per il punto di ricongiungimento alla traiettoria è stato fatto il procedimento inverso, ovvero quando la soglia era prossima al valore nullo si interrompeva il processo. A causa delle oscillazioni di posizione, non è possibile utilizzare una soglia troppo bassa, si riscontra che essa venga superata prima di arrivare al punto reale. Pertanto si verificava graficamente che il punto risultante fosse quello reale.

Preso come riferimento di tempo il valore $\alpha_0 = \frac{1}{3}$ e definiti T_1 il tempo del movimento precedente al punto di via interessato e T_2 il tempo del movimento successivo, i punti di inizio e fine del raccordo sono stati ricavati in modo seguente:

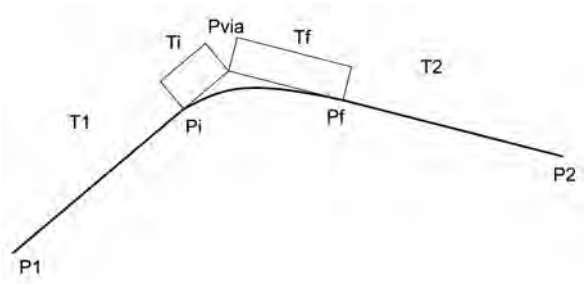


Figura 3.9: Illustrazione dei punti di inizio e fine arco.

$$\alpha = \alpha_0 = \frac{1}{3}$$

$$T_i = \alpha_0 T_1 \qquad T_f = \alpha_0 T_2$$

$$P_i = P(T_1 - T_i) \qquad P_f = P(T_f)$$

I punti teorici ottenuti dal metodo precedente, confrontati con quelli reali, mostrano un eccessivo errore per i movimenti a velocità elevate. Alle basse velocità il robot non ha bisogno di anticipare i movimenti, bisogna dunque modificare il parametro α affinché sia dipendente dalla velocità. Si è deciso di mantenere costante il valore di α fino al raggiungimento di una determinata velocità v_0, oltre tale velocità il parametro varia linearmente da α_0 ad α_1. I valori dei parametri α_1 e v_0 sono stati ricavati sperimentalmente, facendo la media dei loro valori ricavati dai diversi movimenti in nostro possesso.

Sopra il grafico mostra l'andamento del parametro α al variare della velocità.

$$\alpha_0 = \frac{1}{3} \qquad \alpha_1 = 0.4 \qquad v_0 = 50\%$$

$$\begin{cases} \alpha(v) = \alpha_0 & (v < v_0) \\ \alpha(v) = \alpha_1 + (\alpha_0 - \alpha_1)\frac{v}{100\%} & (v \leq v_0) \end{cases}$$

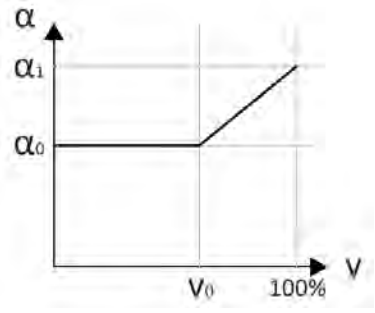


Figura 3.10: Andamento del parametro α in funzione della velocità.

	distanza [mm] $\alpha = \alpha_0$		distanza [mm] $\alpha = \alpha(v)$		differenza [mm]	
Velocità [%]	Pi	Pf	Pi	Pf	Pi	Pf
5 %	15.237	8.214	15.237	8.214	0	0
10 %	8,145	11,394	8,145	11,394	0	0
15 %	14,168	18,347	14,168	18,347	0	0
25 %	13,351	8,230	13,351	8,230	0	0
35 %	15,167	14,344	15,167	14,344	0	0
50 %	11,037	7,306	11,037	7,306	0	0
60 %	21,341	18,328	10,286	8,270	-11.055	-10.058
70 %	29,194	18,067	9,318	8,047	-19.876	-10.020
80 %	30,119	24,779	8,648	11,273	-21.471	-13.506
90 %	41,818	29,142	11,634	12,812	-30.184	-16.330
100 %	32,531	25,608	12,399	12,891	-20.132	-12.717

Tabella 3.3: Distanza dei punti reali di inizio e fine raccordo, confrontati con i rispettivi punti teorici calcolati mediante la pianificazione, utilizzando il parametro α costante o variabile.

Nella tabella soprastante sono riportate le distanze tra i punti reali di inizio e fine raccordo, con quelli teorici, sia quelli ricavati utilizzando il parametro α costante sia quelli ottenuti con α variabile. Tale tabella ci permette di quantificare l'errore, in termini di posizione, del punto di inizio raccordo Pi e di quello finale Pf. Le ultime due colonne mostrano il guadagno ottenuto, facendo la differenza tra la distanza con α variabile e quella con α costante.

3.3.2 Polinomio cubico nei punti di via

Utilizzando i punti iniziali e finali del raccordo ricavati col metodo precedente, siamo a conoscenza delle posizioni dei giunti e le loro relative velocità grazie alla pianificazione, possiamo ricavare il raccordo che li congiunge tramite un polinomio di terzo grado, definendo l'accelerazione utilizzata o il tempo di durata. Anche in questa situazione non disponiamo con certezza di entrambi i dati, ma si è ipotizzato un tempo di durata, per ogni giunto, pari a:

$$T = 2\frac{|\Delta s_i| + |\Delta s_f|}{|v_i| + |v_f|}$$

Dove v_i e v_f sono le velocità appartenenti al punto iniziale e quello finale, mentre Δs_i è la distanza del punto P_i dal via point e rispettivamente Δs_f è la distanza tra il via point e il punto P_f.

L'equazione del polinomio di terzo grado è la seguente:

$$\theta(t) = a_4 + a_3(\frac{t}{T}) + a_2(\frac{t}{T})^2 + a_1(\frac{t}{T})^3$$

Per risolvere l'equazione è necessario disporre delle condizioni al contorno, quindi per ogni punto di via bisogna conoscere il punto di inizio e fine raccordo e le relative velocità. Attraverso la loro conoscenza si ricavano i valori a_1, a_2, a_3 e a_4 tramite la seguente equazione:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & \frac{1}{T} & 0 \\ \frac{3}{T} & \frac{2}{T} & \frac{1}{T} & 0 \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{Bmatrix} = \begin{Bmatrix} x_i \\ x_f \\ v_i \\ v_f \end{Bmatrix}$$

Utilizzando il polinomio di terzo grado si ottiene un elevato numero di punti che consentono una perfetta rappresentazione della parabola. Il percorso risultante è prossimo a quello reale, nelle immagini successive si rappresentano i percorsi primitivo, reale e cubico per un singolo punto di via. Si illustra un singolo punto per consentire un adeguato ingrandimento che consente una maggiore visibilità delle traiettorie.

In nero è rappresentata la traiettoria primitiva, ovvero quella passante per il via point, in arancione quella reale eseguita dal robot e infine quella verde è quella ottenuta dal polinomio di terzo grado.

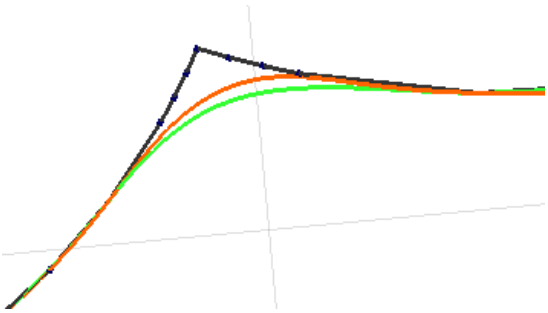


Figura 3.11: Traiettorie con velocità 5%.

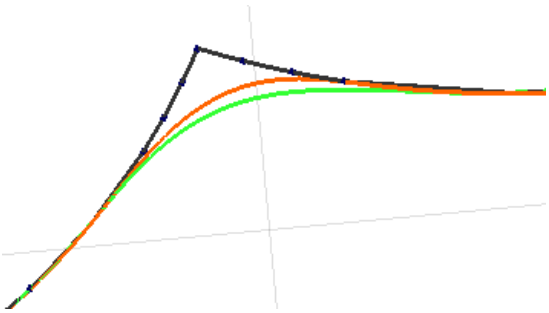


Figura 3.12: Traiettorie con velocità 10%.

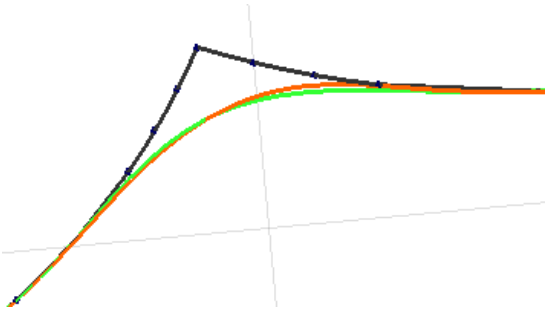


Figura 3.13: Traiettorie con velocità 15%.

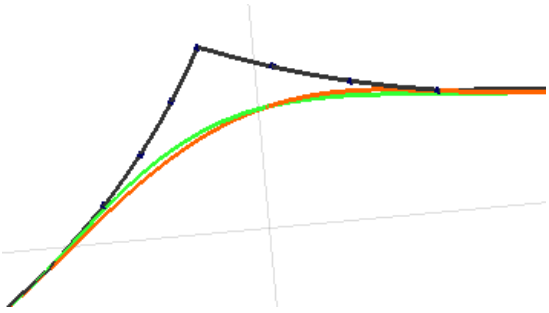


Figura 3.14: Traiettorie con velocità 25%.

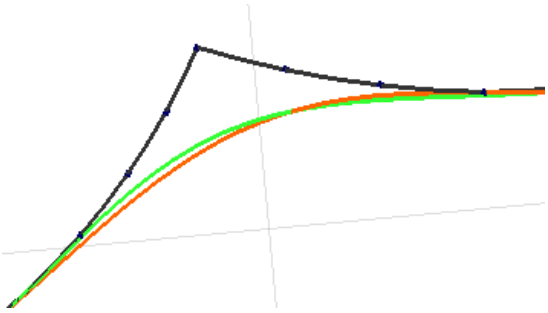


Figura 3.15: Traiettorie con velocità 35%.

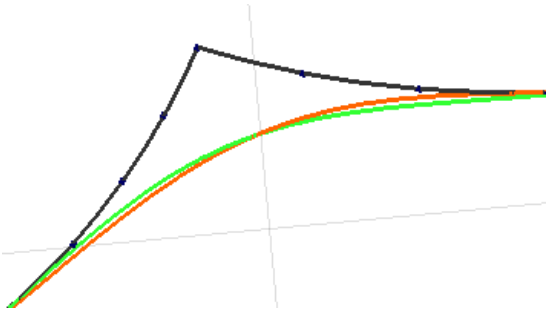


Figura 3.16: Traiettorie con velocità 50%.

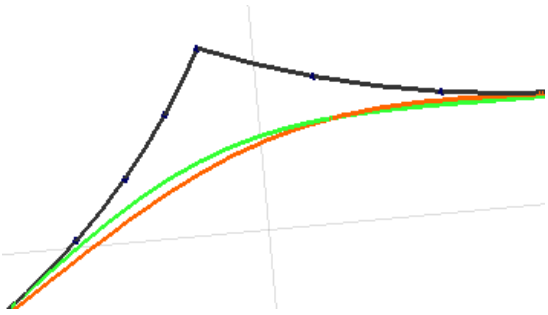


Figura 3.17: Traiettorie con velocità 60%.

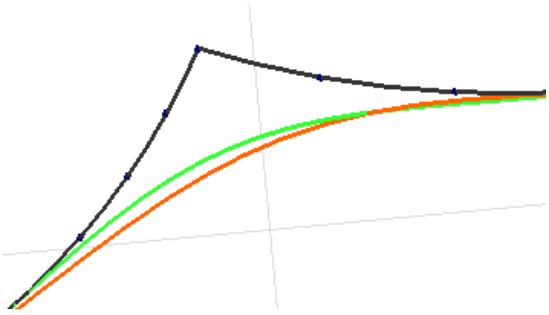


Figura 3.18: Traiettorie con velocità 70%.

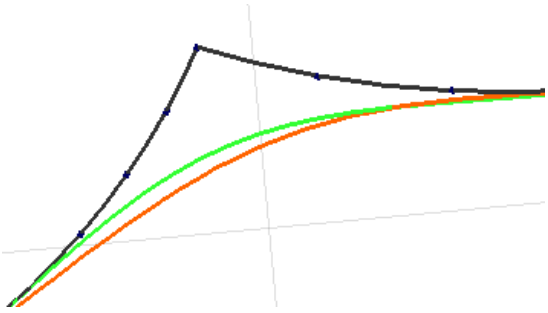


Figura 3.19: Traiettorie con velocità 80%.

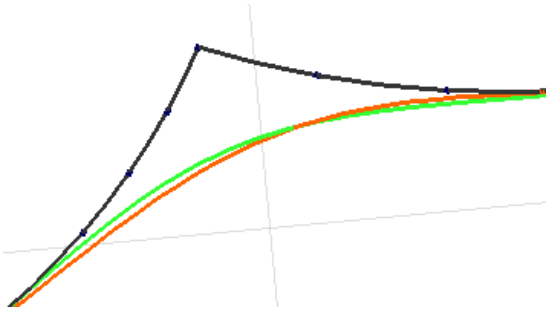


Figura 3.20: Traiettorie con velocità 90%.

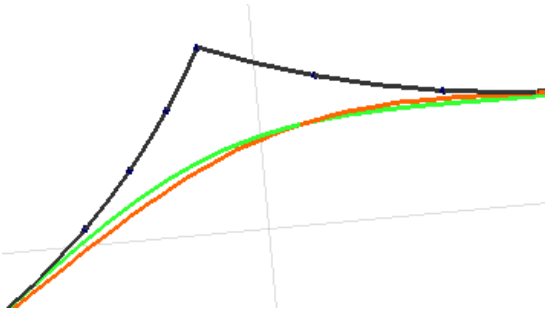


Figura 3.21: Traiettorie con velocità 100%.

3.3.3 Passaggio punti depart - approach

Come per i punti di via, il robot si limita a passare nelle vicinanze dei punti definiti come depart e approach. In questo caso, a differenza del precedente, siamo a conoscenza di uno dei due punti del raccordo, per la precisione sappiamo le coordinate del punto di distacco dalla traiettoria nell'intorno del punto di depart e quello di ricongiungimento nei pressi dell'approach. Tali locazioni sono note in quanto sono vincolate dal parametro *arch*, definito nel comando `JUMP`, come illustrato nel paragrafo 2.2.3. Durante i test eseguiti in laboratorio sono stati utilizzati diversi valori di arco e diverse lunghezze di depart e approach, per non concentrarsi esclusivamente su una serie di casistiche. Nella traiettoria utilizzata come esempio in questa tesi, l'arco impostato nel comando `JUMP` è `C0`, che nel software Epson RC+ è impostato come 30 mm. All'interno del programma è possibile definire fino a sei valori di arco, di default sono impostati come seguente:

`C0` = 30 mm
`C1` = 40 mm
`C2` = 50 mm
`C3` = 60 mm
`C4` = 70 mm
`C5` = 80 mm
`C6` = 90 mm
`C7` = in questo caso il robot non esegue alcun raccordo

La prima fase è sempre quella dedicata alla ricerca dei punti estremi in cui le traiettorie si ricongiungono. A seguire verrà illustrato come calcolare le posizioni dei punti per la fase di depart, sapendo che per la fase opposta, di approach, il metodo sarà analogo in quanto speculare.

Il parametro *arch* consente all'operatore di stabilire dopo quale distanza percorsa il robot deve cominciare a deviare dalla traiettoria primitiva, al crescere della velocità tale punto può variare.

Tramite Matlab si genera una pianificazione lineare tra il punto di partenza P1 e il punto di depart P2, composta da un numero adeguato di punti interme-

di. La pianificazione, essendo un movimento lineare nello spazio cartesiano, non richiede un tempo specifico di esecuzione. É necessario congiungere linearmente nello spazio cartesiano i punti estremi e suddividere la traiettoria in n tratti. É importante avere un numero adeguato di punti intermedi, così facendo sarà più precisa la stima del punto di inizio raccordo.

La pianificazione è stata ottenuta mediante il seguente script:

$$P1 = XY(x_1, y_1, z_1, u_1, v_1, w_1)$$
$$P2 = XY(x_2, y_2, z_2, u_2, v_2, w_2)$$
$$t = [0 : 1/n : 1 - 1/n]$$

$$x = P2(1) + (P1(1) - P2(1)) * (1 - t)$$
$$y = P2(2) + (P1(2) - P2(2)) * (1 - t)$$
$$z = P2(3) + (P1(3) - P2(3)) * (1 - t)$$
$$u = P2(4) + (P1(4) - P2(4)) * (1 - t)$$
$$v = P2(5) + (P1(5) - P2(5)) * (1 - t)$$
$$w = P2(6) + (P1(6) - P2(6)) * (1 - t)$$

Sfruttando sempre il software di calcolo è possibile ricercare il punto la cui distanza, in modulo, dal punto P_1 sia pari al valore dell'arco definito dall'utente. La funzione seguente permette di calcolare la distanza corrispondente tra il punto P_1 e il punto P_i in questione, se essa è superiore o uguale a quella richiesta allora si esce dal ciclo for evitando ulteriori calcoli.

$for\ i = 1 : n$
    $distanza = norm(P_1 - P_i)$
    $if\ distanza \geq valore\ arco$
        $break$
    end
end

Avendo un movimento lineare nello spazio operativo il calcolo del primo punto risulta semplice e non eccessivamente complesso.

	Depart distanza [mm] $\alpha = \alpha(v)$		Approach distanza [mm] $\alpha = \alpha(v)$	
Velocità [%]	Pi	Pf	Pi	Pf
5%	1.245	4.643	8.214	1.510
10%	0.972	6.137	7.339	1.341
15%	0.843	8.304	7.253	0.942
25%	1.193	7.594	6.732	1.107
35%	1.467	6.296	6.713	1.422
50%	1.133	5.941	7.206	1.698
60%	0.723	6.816	6.505	1.393
70%	0.813	7.364	6.241	1.265
80%	1.346	5.889	6.435	1.108
90%	1.210	7.061	5.892	0.735
100%	1.341	6.342	6.122	1.082

Per determinare dove le traiettorie si ricongiungono, si è mantenuta l'ipotesi fatta per i punti di via, ovvero P_f è ricavato dalla pianificazione del tratto P_2 - P_5 al tempo T_f calcolato allo stesso modo. Analogamente P_i, in corrispondenza del punto P_3, è stato ottenuto dalla pianificazione del tratto P_7 - P_3 al tempo T_i.

Nella tabella soprastante vengono confrontati i punti così calcolati, con quelli reali ricavati, sia relativi al depart sia all'approach.

In questa situazione non è possibile applicare il metodo del polinomio di terzo grado, in quanto non siamo a conoscenza della velocità dei punti giacenti nel movimento lineare. In mancanza di tale informazione non è possibile ricavare le incognite dell'equazione e di conseguenza la parabola. Per aggirare il problema si è deciso di rappresentare la parabola attraverso la sua interpolazione.

3.3.4 Interpolazione parabola depart - approach

Per ottenere una buona interpolazione è necessario disporre di una ragionevole quantità di punti, ma all'aumentare di essi è proporzionale il tempo di elaborazione della traiettoria finale. Una adeguata stima è stata ottenuta utilizzando sette punti interposti ai due precedentemente calcolati. L'interpolazione della parabola è stata eseguita nello spazio dei giunti, essendo la pianificazione basata su di essi.

Definito con P_i le posizioni dei giunti al punto di inizio arco, P_v quelle al punto di via e P_f quelle al punto di fine arco, l'interpolazione utilizzata è descritta in seguito.

Per ogni giunto si divide il tratto tra P_i e P_v e quello tra P_v e P_f in quattro parti equivalenti, in modo da ottenere nove punti totali, comprendenti anche i due estremi e il punto di via nel mezzo, figura 3.22.

Il primo step di interpolazione consiste nel congiungere linearmente il punto medio del primo tratto con quello del secondo. Il nuovo segmento ottenuto viene suddiviso in quattro parti sempre equamente distribuite, per mantenere inalterato il numero di punti iniziali. Lo step successivo prevede di modificare le coordinate degli estremi del nuovo tratto, dando loro il valore medio tra i punti adiacenti. Per migliorare la curvatura e renderla più omogenea si utilizza lo stesso concetto su entrambi i punti adiacenti a quello mediano (figura 3.25 e 3.26). In questo modo la parabola viene rappresentata adeguatamente congiungendo linearmente 9 punti, ricavabili attraverso semplici divisioni. Come si può osservare nell'ultima immagine, in cui è raffigurata la traiettoria reale e la sua interpolazione finale, il risultato ottenuto da questo metodo, anche se grossolano, ci permette di avvicinarci alla realtà.

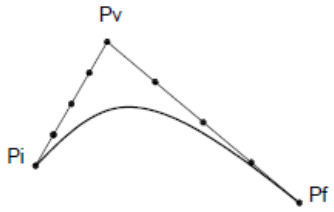


Figura 3.22: Traiettoria primitiva e parabola.

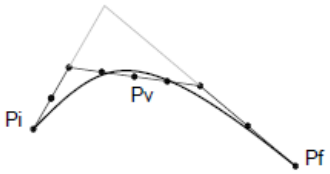


Figura 3.23: Primo step di interpolazione.

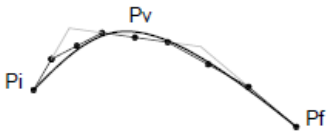


Figura 3.24: Secondo step di interpolazione.

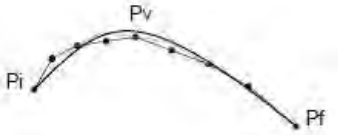


Figura 3.25: Terzo step di interpolazione.

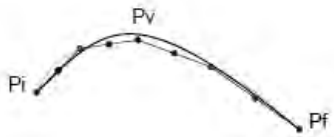


Figura 3.26: Step finale di interpolazione.

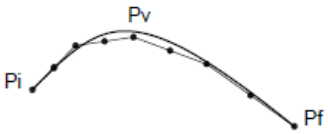


Figura 3.27: Rappresentazione finale della parabola.

Di seguito, per mostrare i risultati ottenuti da questo metodo di interpolazione, si riportano delle immagini raffiguranti le diverse traiettorie nel piano cartesiano per tutte le velocità di cui disponiamo del percorso reale. In arancione è mostrata la traiettoria reale eseguita dal robot, col colore nero è rappresentata quella primitiva passante per i via point e infine in verde quella interpolata secondo il metodo appena descritto.

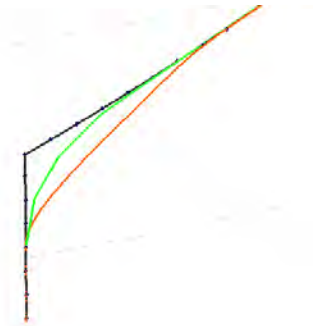


Figura 3.28: Traiettorie con velocità 5%.

Figura 3.29: Traiettorie con velocità 10%.

Figura 3.30: Traiettorie con velocità 15%.

Figura 3.31: Traiettorie con velocità 25%.

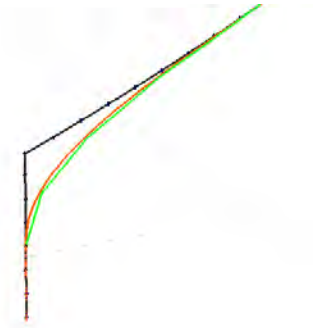


Figura 3.32: Traiettorie con velocità 35%.

Figura 3.33: Traiettorie con velocità 50%.

Figura 3.34: Traiettorie con velocità 60%.

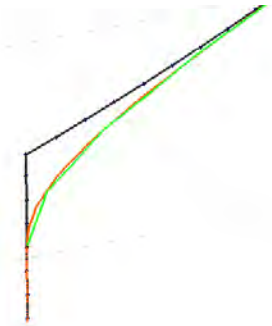


Figura 3.35: Traiettorie con velocità 70%.

Figura 3.36: Traiettorie con velocità 80%.

Figura 3.37: Traiettorie con velocità 90%.

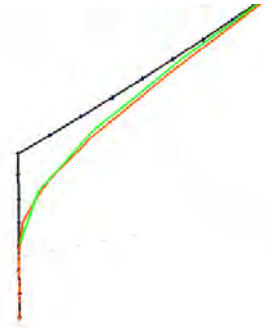


Figura 3.38: Traiettorie con velocità 100%.

3.3.5 Tempistiche di calcolo

Il software Matlab rende disponibile all'utente i comandi *tic* e *toc* grazie ai quali è possibile effettuare misure di tempo assoluto. Il comando *tic* indica l'istante di inizio del conteggio temporale e il duale comando *toc* esprime l'istante in cui si desidera effettuare la misura. Nelle versioni di Matlab precedenti alla R2006 [6] i comandi *tic* e *toc* erano connessi alla funzione *clock*, sfruttavano cioè il wall clock interno del sistema. Ogni comando *tic* o *toc* prelevava il tempo istantaneo riportato in tale metodo di conteggio effettuato dal sistema operativo. Matlab, in seguito, valutava la differenza fra i due valori rilevati per fornire una stima della misura richiesta. Analizzando il caso in Windows, la funzione *clock* chiama il servizio di sistema *GetLocalTime* usando l'API (Application Programming Interface). Il servizio in esame, tuttavia, ha scarsa accuratezza e un richiamo immediatamente successivo dello stesso può addirittura determinare un valore negativo del tempo trascorso. Dalla versione R2006b di Matlab è stato modificato il sistema di misura delle tempistiche. Il servizio di sistema attualmente usato per effettuare misure di tempo è *QueryPerformanceCounter* che permette di avviare un counter a 64 bit in unità di sistema anziché secondi. La differenza fra le due rilevazioni viene poi divisa per la frequenza restituita dal servizio di sistema *QueryPerformanceFrequency* grazie al quale si può giungere al risultato in secondi.

Dalla versione R2008b in poi la sintassi *tic-toc* inoltre è stata estesa per consentire l'avvio di contatori multipli. La nuova sintassi consente di salvare un preciso contatore in una variabile per poi richiamare tale parametro in una successiva funzione toc.

La presente funzione è stata utilizzata per avere un riferimento sui tempi di calcolo necessari per la stima della traiettoria complessiva. Nei tempi riportati nella tabella 3.4, Matlab elabora le coordinate dei punti iniziali da noi definiti nello spazio dei giunti, la velocità impostata e il valore dell'arco. Pianifica la movimentazione tra una locazione e la successiva, ricava mediante i metodi descritti in questo capitolo i punti di inizio e fine arco per ogni via point. Avendo in memoria i dati necessari, applica l'algoritmo di interpolazione per tutto il movimento, ottenendo di fatto i punti finali.

Tabella 3.4: Tempistiche per il calcolo della stima completa della traiettoria col metodo del polinomio cubico.

Velocità (%)	Tempo coordinate di giunto (s)	Tempo coordinate cartesiane (s)
5	0.0772	0.2085
10	0.0140	0.1474
15	0.0129	0.1464
25	0.0103	0.1447
35	0.0117	0.1446
50	0.0113	0.1429
60	0.0098	0.1323
70	0.0114	0.1359
80	0.0111	0.1439
90	0.0111	0.1423
100	0.0110	0.1439

Nella tabella sono riportati due valori di tempi, nel caso si vogliano i punti finali in coordinate di giunto bisogna fare riferimento al primo valore. Il secondo valore si riferisce al tempo impiegato per avere in uscita i punti in coordinate cartesiane. Di fatto il procedimento per ricavare le locazioni è il medesimo, la differenza consiste nel dover eseguire una cinematica diretta nel secondo caso, per trasformare i punti da uno spazio operativo all'altro.

Avere i punti nello spazio cartesiano risulta più pratico perché consente la verifica fisica dei punti in ambiente lavorativo.

3.3.6 Interpolazione parabola punti di via

Per completezza abbiamo voluto applicare il metodo interpolatore anche nei punti di via, per verificare come si comporta nella stima della traiettoria, qualora non si volesse utilizzare il polinomio di terzo grado. I risultati dimostrano che è possibile impiegare tale metodo anche nei via point, ma essendo il numero di punti limitato a sette, la sua rappresentazione avviene per tratti. Per la descrizione degli step di interpolazione si rimanda al paragrafo 3.2.4, mentre a seguire vengono mostrate le traiettorie risultanti dopo l'applicazione del metodo interpolatore. La linea arancione rappresenta il percorso reale, la nera quella primitiva e quella azzurra è quella interpolata.

Si è scelto di illustrare solamente un punto di via per avere un ingrandimento adeguato della traiettoria eseguita.

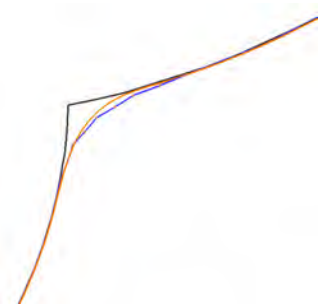


Figura 3.39: Traiettorie con velocità 5%.

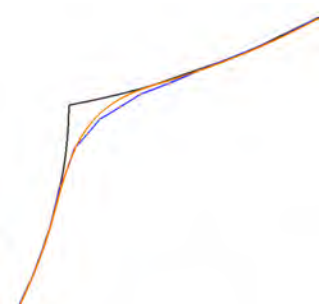


Figura 3.40: Traiettorie con velocità 10%.

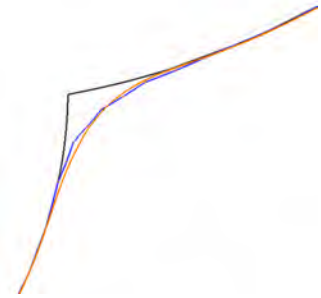


Figura 3.41: Traiettorie con velocità 15%.

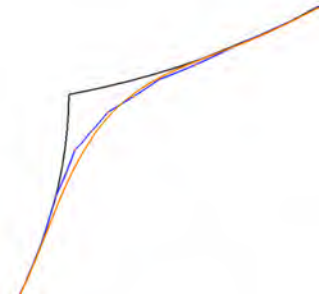


Figura 3.42: Traiettorie con velocità 25%.

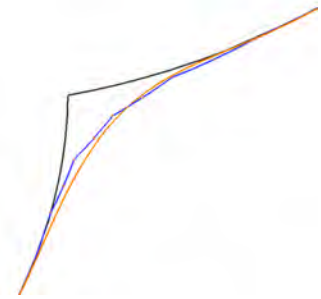


Figura 3.43: Traiettorie con velocità 35%.

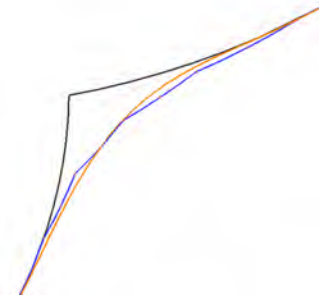


Figura 3.44: Traiettorie con velocità 50%.

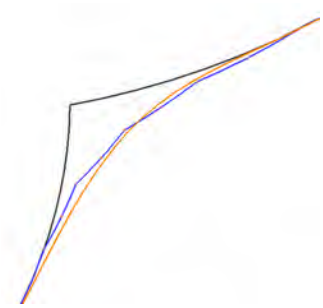


Figura 3.45: Traiettorie con velocità 60%.

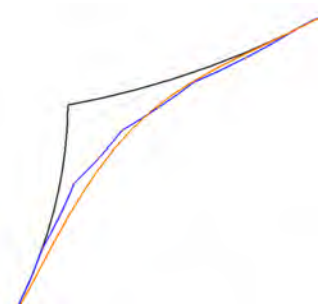


Figura 3.46: Traiettorie con velocità 70%.

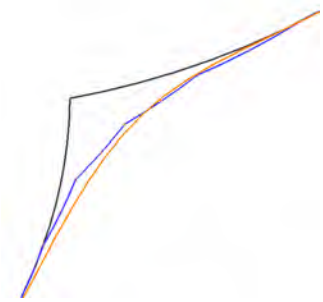


Figura 3.47: Traiettorie con velocità 80%.

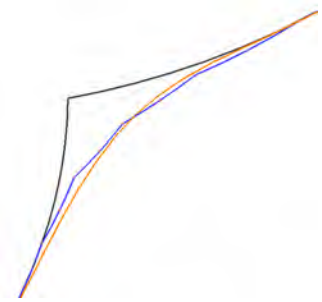


Figura 3.48: Traiettorie con velocità 90%.

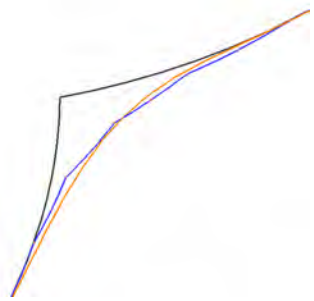


Figura 3.49: Traiettorie con velocità 100%.

Tabella 3.5: Tempistiche per il calcolo della stima completa della traiettoria mediante il metodo interpolato.

Velocità (%)	Tempo coordinate di giunto (s)	Tempo coordinate cartesiane (s)
5	0.098874	0.173461
10	0.008447	0.072803
15	0.006553	0.073287
25	0.005990	0.071563
35	0.005243	0.072120
50	0.005511	0.072330
60	0.004829	0.072337
70	0.004638	0.070833
80	0.004746	0.070602
90	0.004505	0.069722
100	0.004584	0.069172

Nel caso venga impiegato tale metodo, le tempistiche di calcolo della traiettoria sono simili a quelle richieste impiegando il polinomio cubico. Nella tabella 3.5 sono riportati i rispettivi tempi.

Capitolo 4

Analisi dei risultati

I metodi descritti per la stima dei percorsi eseguiti dal manipolatore alle diverse situazioni viste in precedenza, richiede la sola conoscenza dei punti iniziale, intermedi e finale. In aggiunta alle locazioni bisogna indicare il parametro dell'arco, qualora fosse utilizzato, e la velocità con la quale il robot viene azionato. Disponendo di queste semplici informazioni, sempre note per poter eseguire una movimentazione, siamo in grado di stimare una traiettoria prossima a quella reale.

I risultati ottenuti dall'applicazione dei due metodi, polinomio cubico ed interpolatore, nei grafici dei giunti si ottiene un andamento molto prossimo a quello reale. Non avendo le tempistiche dei reali per i metodi da noi utilizzati, i grafici sono stati normalizzati.

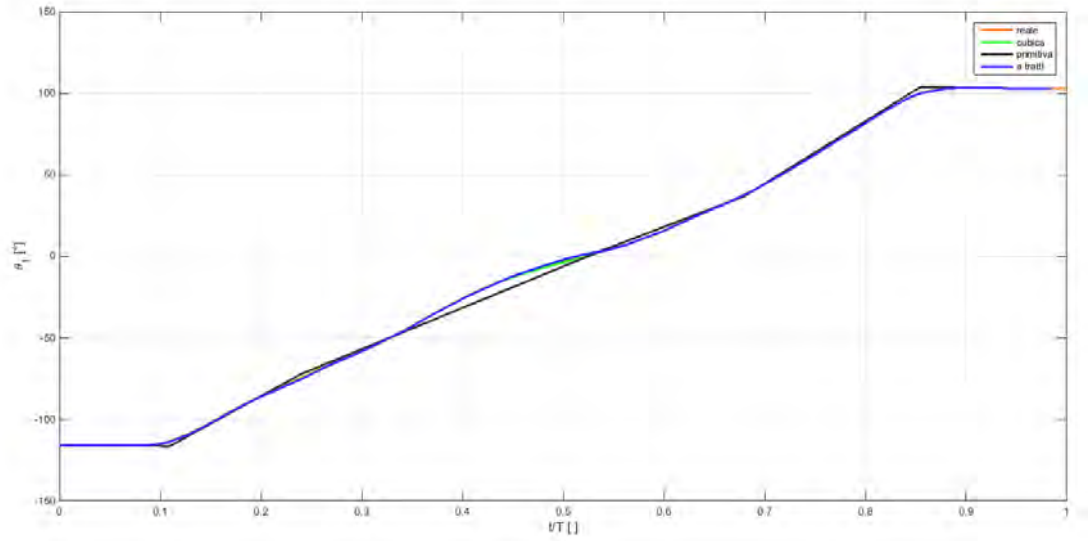


Figura 4.1: Andamento della posizione assunta dal giunto 1 durante il movimento.

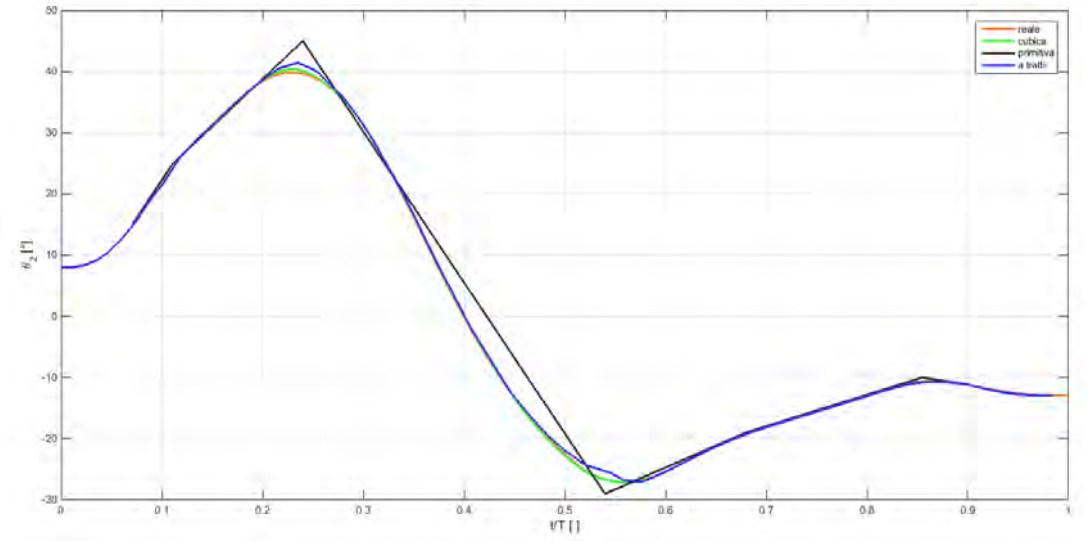


Figura 4.2: Andamento della posizione assunta dal giunto 2 durante il movimento.

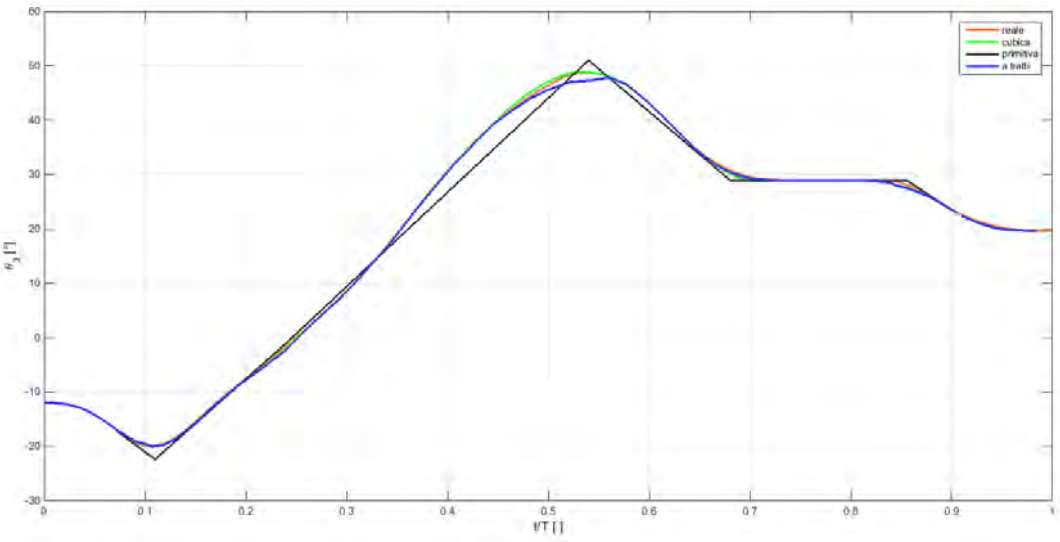


Figura 4.3: Andamento della posizione assunta dal giunto 3 durante il movimento.

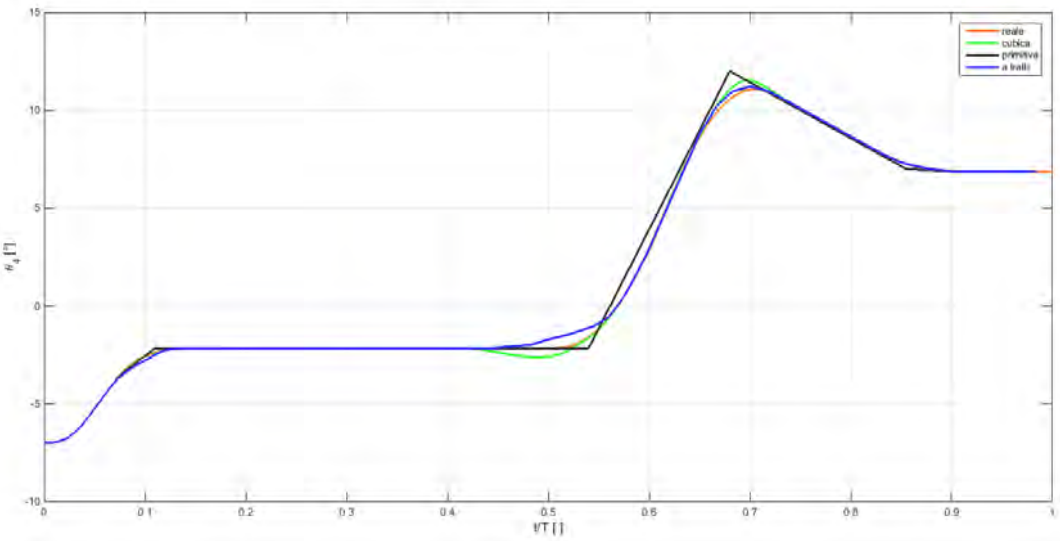


Figura 4.4: Andamento della posizione assunta dal giunto 4 durante il movimento.

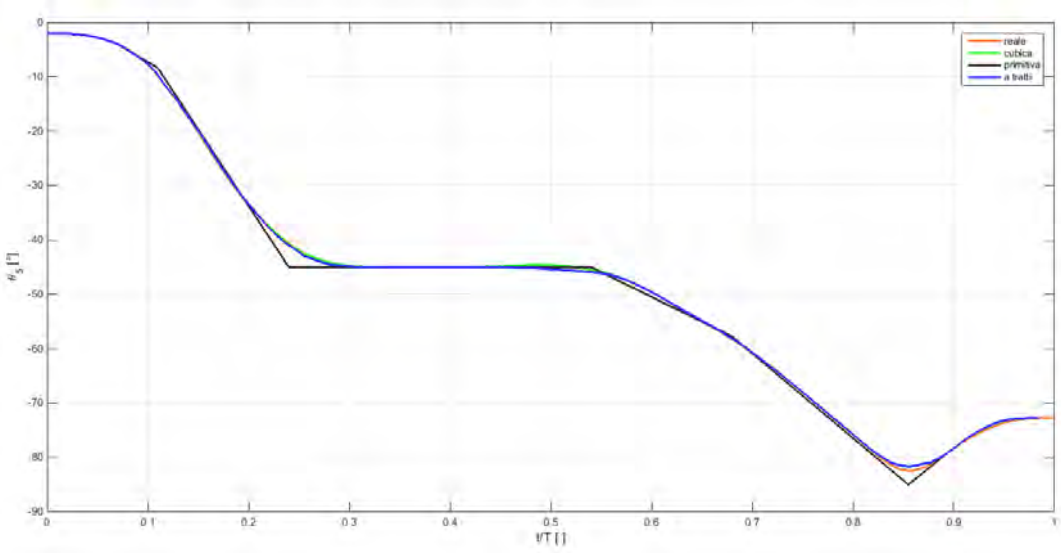


Figura 4.5: Andamento della posizione assunta dal giunto 5 durante il movimento.

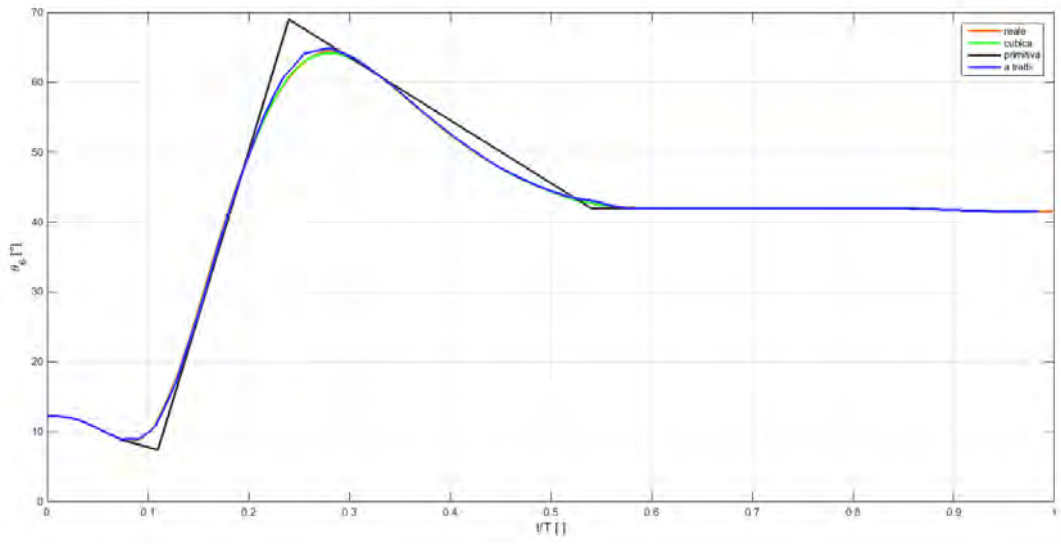


Figura 4.6: Andamento della posizione assunta dal giunto 6 durante il movimento.

Le pianificazioni dei giunti appena mostrate, sono riferite alla traiettoria di esempio finora utilizzata e riportata sottostante.

Per mettere in luce l'errore commesso in termini di distanza tra le diverse traiettorie, viene eseguita in Matlab una funzione che calcola la distanza tra due percorsi cartesiani. I grafici successivi riportano le distanze tra la traiettoria reale del robot con tutte quelle calcolate con i metodi descritti in questa tesi. Presa come riferimento la traiettoria reale, si rappresentano in nero la distanza relativa a quella primitiva, in verde quella relativa alla traiettoria cubica e in azzurro quella relativa alla traiettoria a tratti.

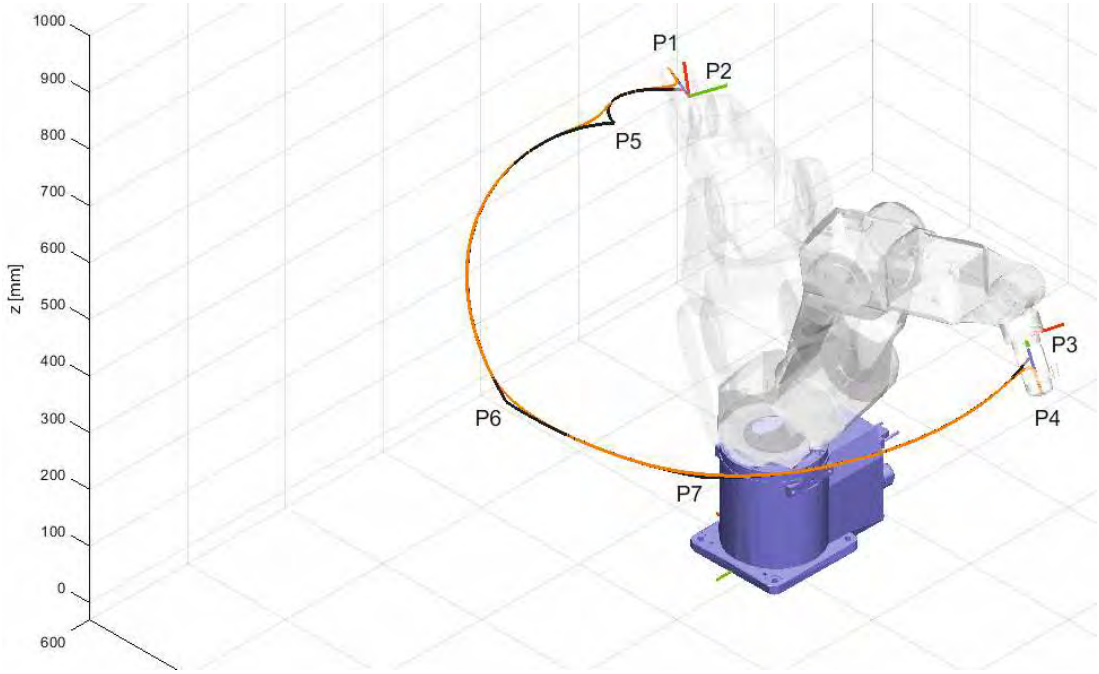


Figura 4.7: Movimento di esempio per l'esposizione dei risultati.

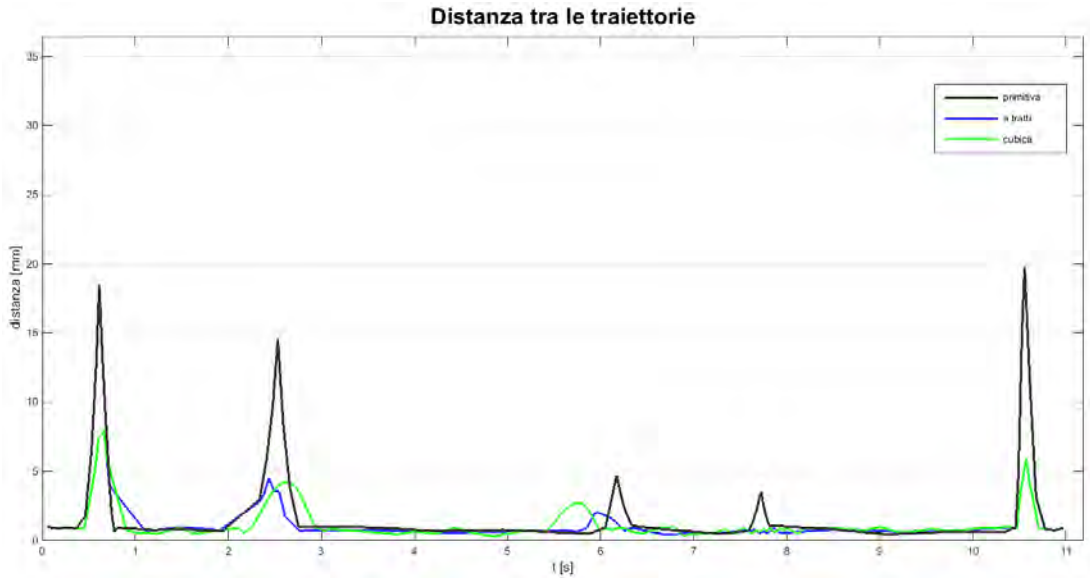


Figura 4.8: Distanze tra le diverse traiettorie sul movimento al 5% di velocità.

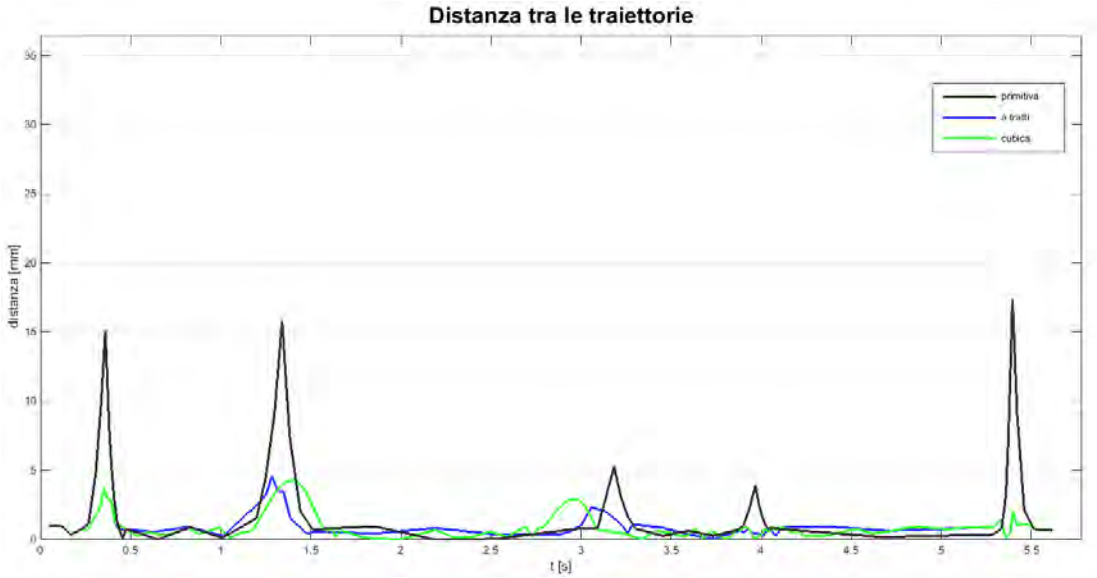


Figura 4.9: Distanze tra le diverse traiettorie sul movimento al 10% di velocità.

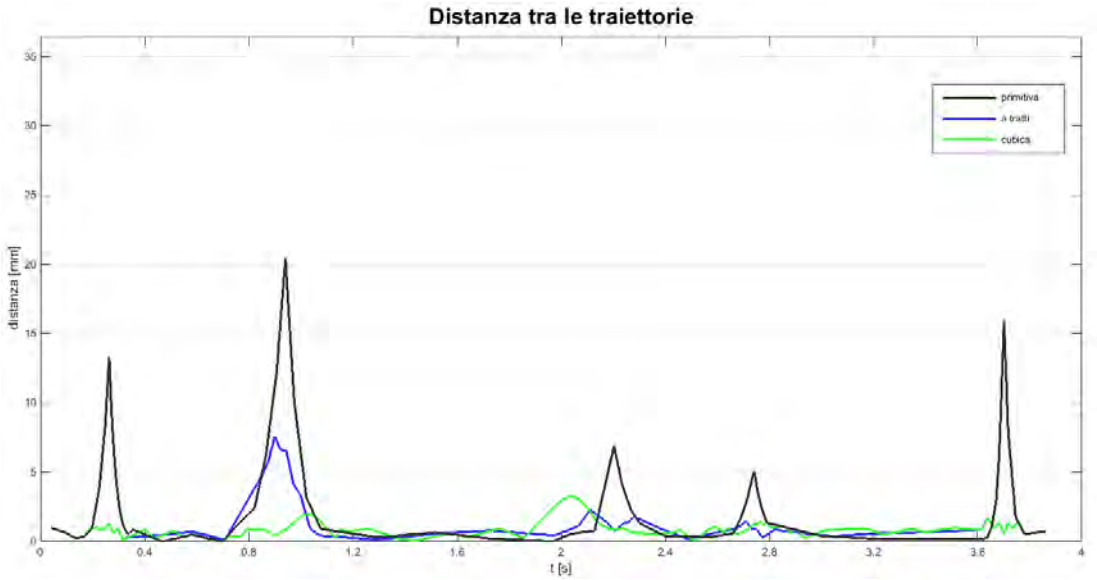


Figura 4.10: Distanze tra le diverse traiettorie sul movimento al 15% di velocità.

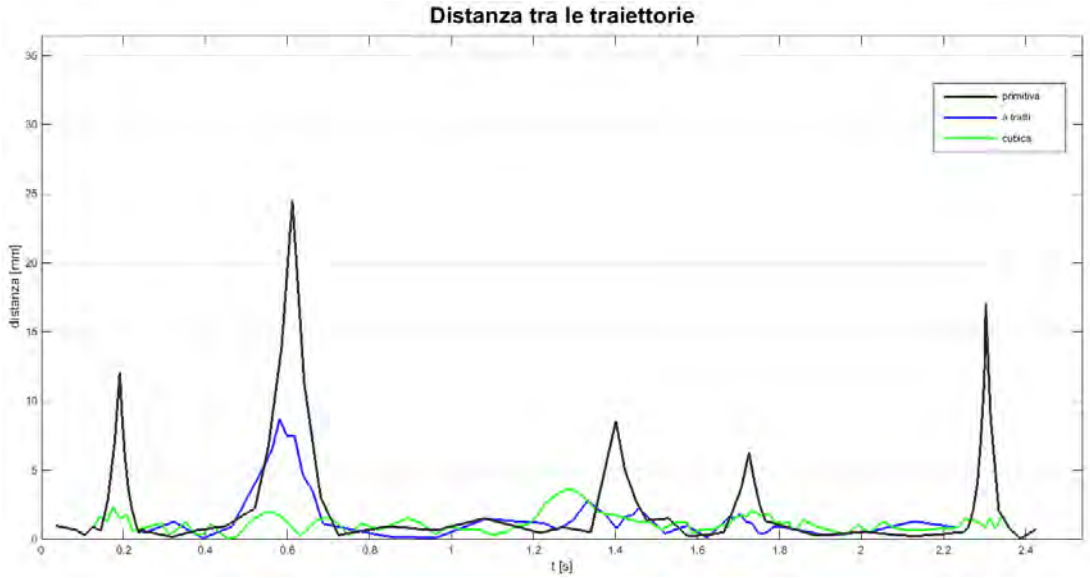


Figura 4.11: Distanze tra le diverse traiettorie sul movimento al 25% di velocità.

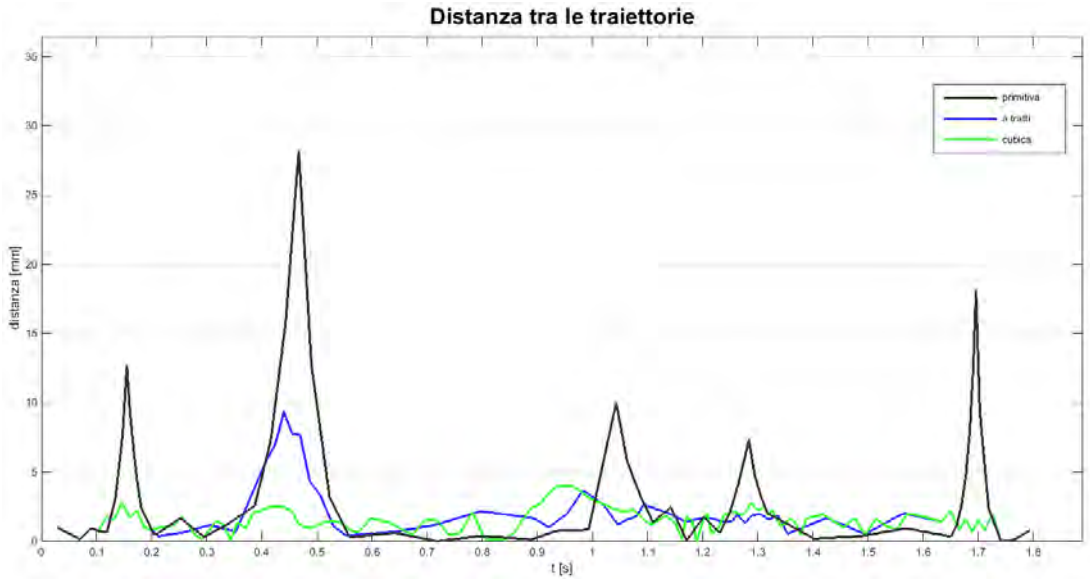


Figura 4.12: Distanze tra le diverse traiettorie sul movimento al 35% di velocità.

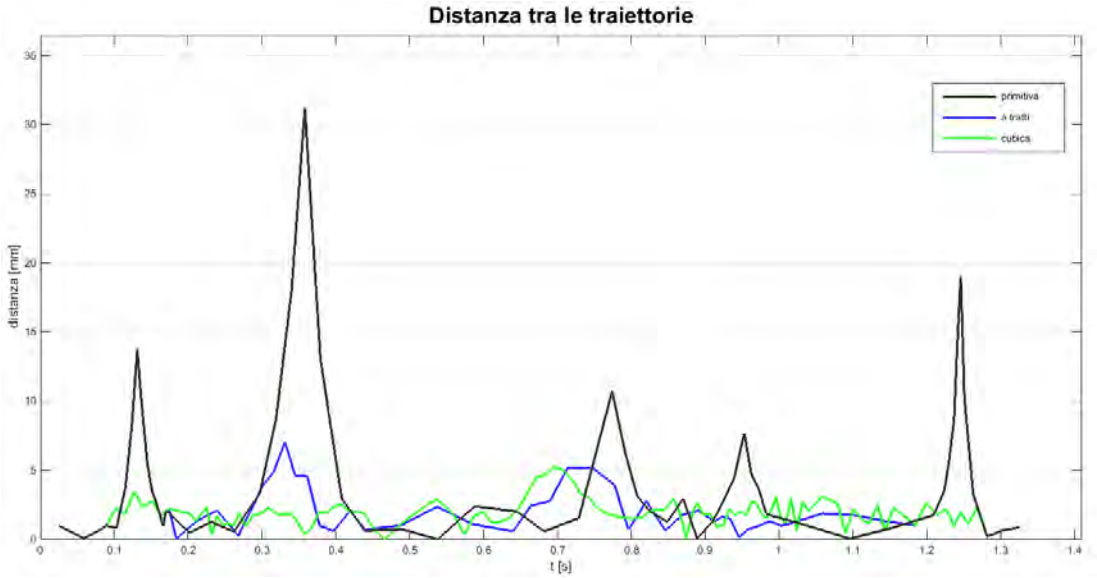


Figura 4.13: Distanze tra le diverse traiettorie sul movimento al 50% di velocità.

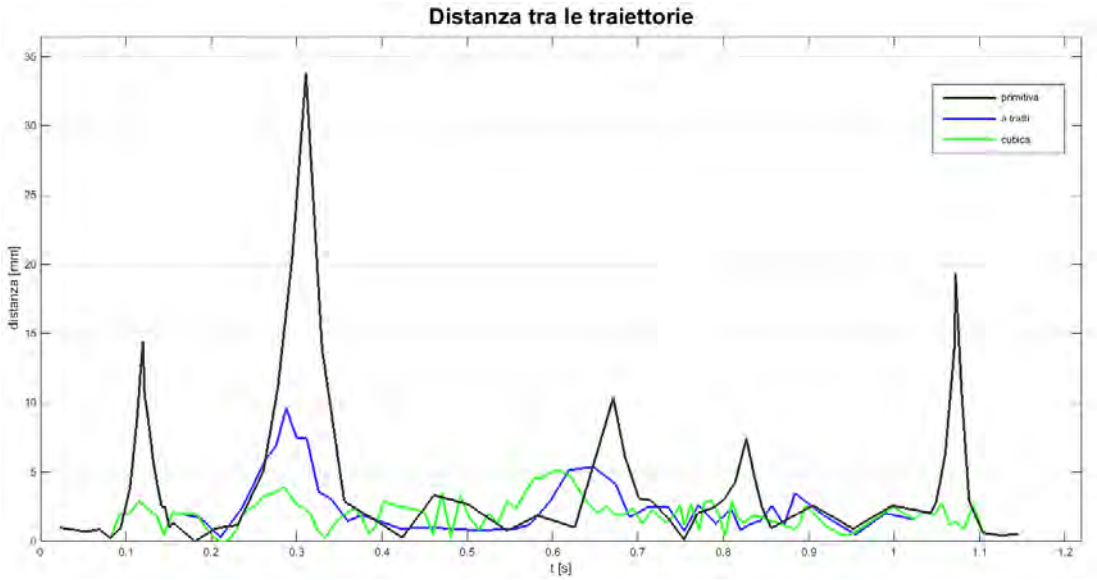


Figura 4.14: Distanze tra le diverse traiettorie sul movimento al 60% di velocità.

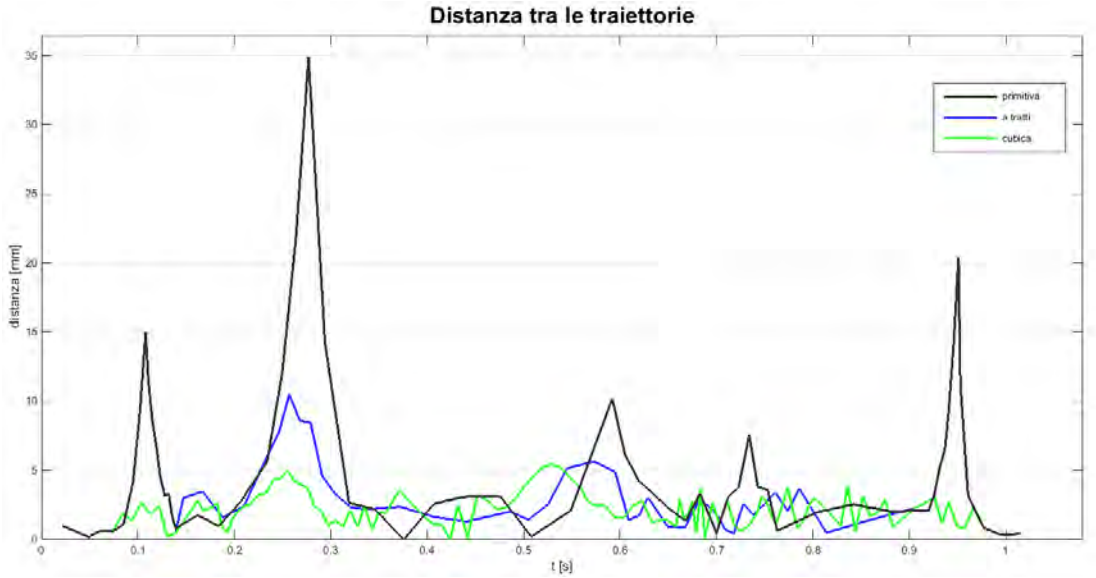


Figura 4.15: Distanze tra le diverse traiettorie sul movimento al 70% di velocità.

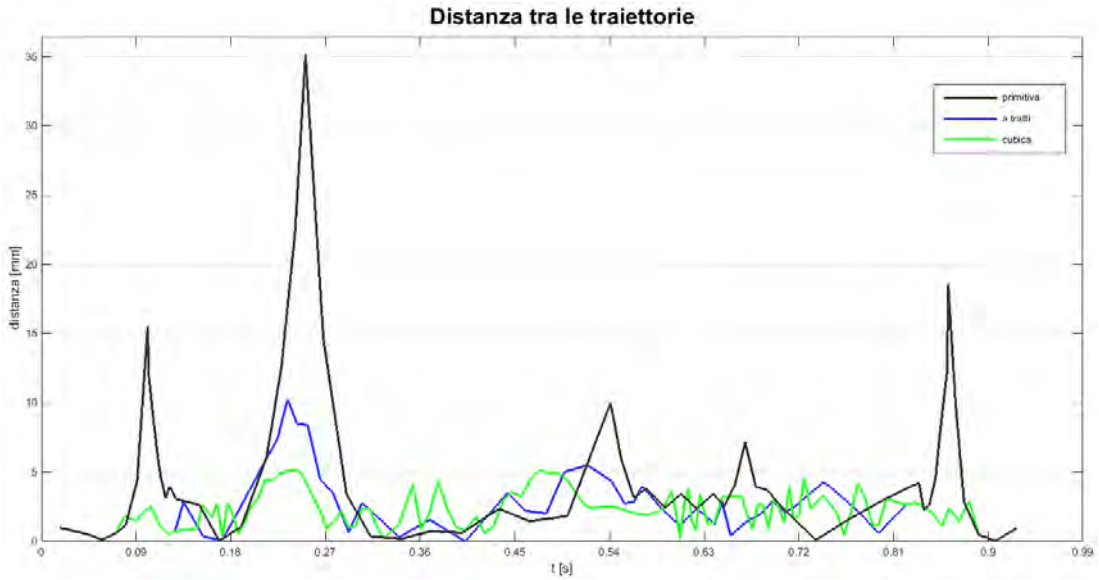


Figura 4.16: Distanze tra le diverse traiettorie sul movimento al 80% di velocità.

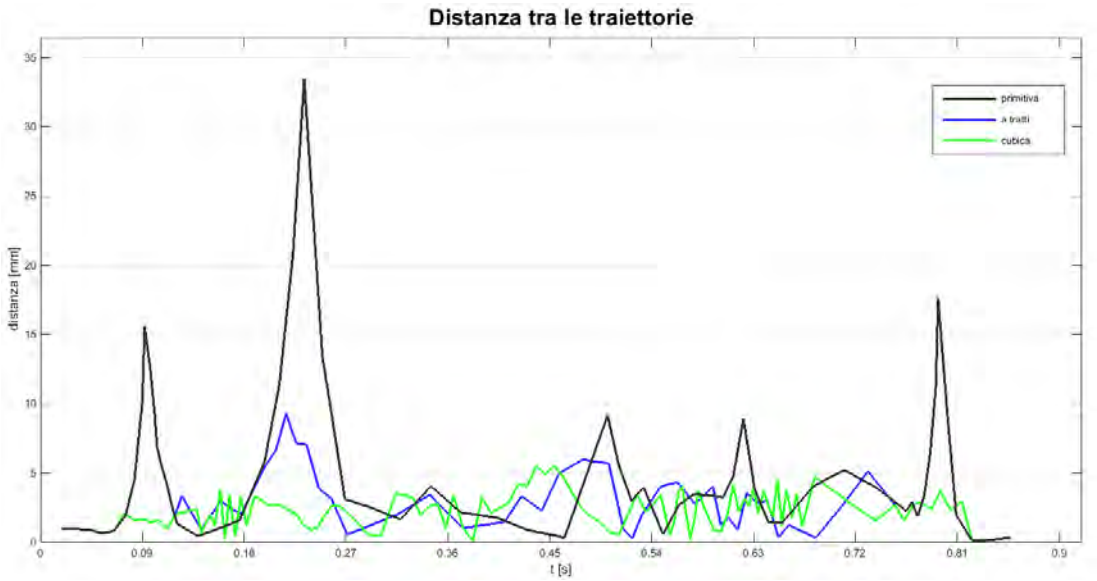


Figura 4.17: Distanze tra le diverse traiettorie sul movimento al 90% di velocità.

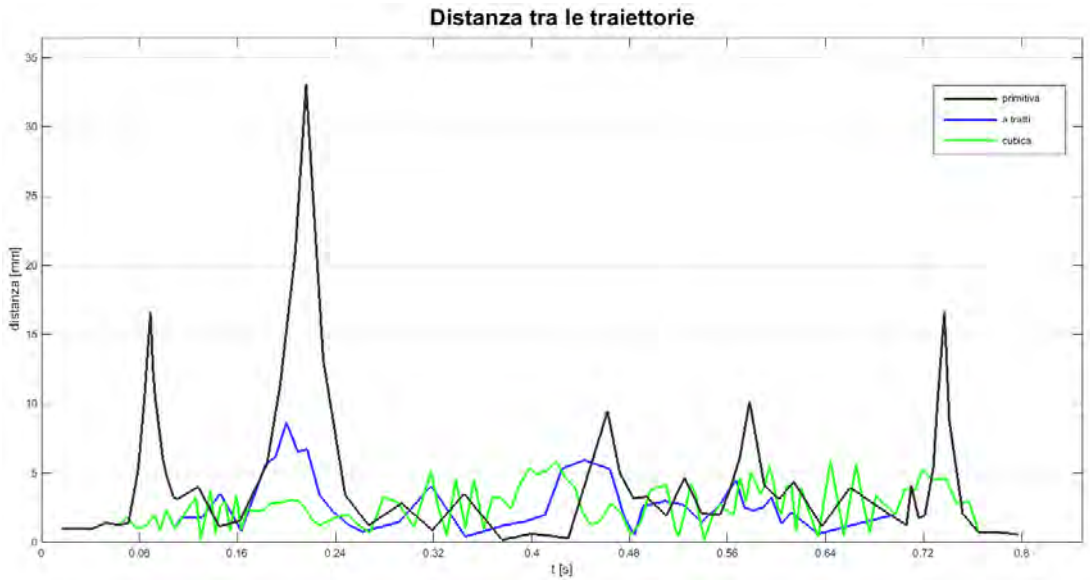


Figura 4.18: Distanze tra le diverse traiettorie sul movimento al 100% di velocità.

I picchi presenti in maniera evidente nella linea nera rappresentano di fatto i punti intermedi, in cui le traiettorie si distaccano. I loro valori non rimangono invariati al crescere della velocità, in concordanza col fatto che il percorso reale varia con essa.

Come chiaramente si nota dai grafici, applicando i diversi metodi, si ottiene una riduzione dell'errore, portando la distanza a mantenersi contenuta, in questo particolare caso al di sotto dei 10 millimetri per tutto il tratto.

I risultati ottenuti da questo percorso sono molto soddisfacenti, ma la verifica del metodo di stima è stato applicato anche alle ulteriori movimentazioni sviluppate in laboratorio e raffigurate nella figura 3.8. Nella tabella successiva si riportano gli errori massimi riscontrati anche per le altre movimentazioni.

Tabella 4.1: Movimento 1: Distanze massime riscontrate tra le diverse traiettorie rispetto a quella reale presa in riferimento.

Movimento 1			
Velocità (%)	Traiettoria primitiva (mm)	Traiettoria cubica (mm)	Traiettoria a tratti (mm)
5	3,438	7,377	3,609
10	4,402	2,776	3,562
15	4,867	3,296	6,847
25	6,320	3,580	3,443
35	7,339	2,833	3,788
50	9,694	5,703	9,944
60	10,332	4,781	9,364
70	8,609	4,888	11,786
80	8,614	4,721	5,036
90	9,026	2,869	5,369
100	32,333	7,060	13,369

Tabella 4.2: Movimento 2: Distanze massime riscontrate tra le diverse traiettorie rispetto a quella reale presa in riferimento.

Movimento 2			
Velocità (%)	Traiettoria primitiva (mm)	Traiettoria cubica (mm)	Traiettoria a tratti (mm)
5	49,817	34,071	34,071
10	46,709	28,797	28,797
15	44,505	24,479	24,479
25	43,471	20,374	20,374
35	45,131	18,883	18,883
50	46,619	18,301	18,301
60	47,822	19,383	19,383
70	48,363	20,321	20,321
80	47,653	19,246	19,246
90	45,817	19,328	16,924
100	44,066	19,033	21,879

Tabella 4.3: Movimento 3: Distanze massime riscontrate tra le diverse traiettorie rispetto a quella reale presa in riferimento.

Movimento 3			
Velocità (%)	Traiettoria primitiva (mm)	Traiettoria cubica (mm)	Traiettoria a tratti (mm)
5	83,993	61,924	60,984
10	76,496	76,950	50,666
15	71,400	91,501	42,166
25	64,395	23,910	28,269
35	59,613	31,765	20,272
50	55,481	21,307	24,515
60	56,437	17,862	16,278
70	58,097	14,263	13,083
80	62,518	11,403	14,675
90	60,440	21,056	39,764
100	61,434	17,333	17,155

Tabella 4.4: Movimento 4: Distanze massime riscontrate tra le diverse traiettorie rispetto a quella reale presa in riferimento.

Movimento 4

Velocità (%)	Traiettoria primitiva (mm)	Traiettoria cubica (mm)	Traiettoria a tratti (mm)
5	24,088	13,543	15,331
10	19,490	5,349	9,277
15	16,817	6,505	7,804
25	16,883	8,784	9,817
35	17,976	7,087	12,258
50	19,066	9,067	17,429
60	19,755	14,019	15,761
70	20,189	13,529	14,809
80	20,386	9,739	13,639
90	21,475	7,462	13,412
100	27,902	7,078	9,527

Tabella 4.5: Movimento 5: Distanze massime riscontrate tra le diverse traiettorie rispetto a quella reale presa in riferimento.

Movimento 5			
Velocità (%)	Traiettoria primitiva (mm)	Traiettoria cubica (mm)	Traiettoria a tratti (mm)
5	30,410	16,957	16,957
10	26,508	10,387	10,387
15	24,030	6,398	6,398
25	23,408	3,880	5,229
35	25,051	4,123	5,685
50	26,824	5,256	5,145
60	27,554	4,863	6,861
70	28,544	5,542	9,475
80	28,616	6,310	10,837
90	29,299	7,864	12,315
100	29,931	8,753	14,162

Tabella 4.6: Movimento 6: Distanze massime riscontrate tra le diverse traiettorie rispetto a quella reale presa in riferimento.

Movimento 6			
Velocità (%)	Traiettoria primitiva (mm)	Traiettoria cubica (mm)	Traiettoria a tratti (mm)
5	51,811	30,978	30,978
10	46,305	21,588	21,588
15	43,281	14,707	23,692
25	44,847	16,810	25,096
35	44,729	15,188	19,731
50	47,293	11,635	9,761
60	48,653	11,637	11,637
70	49,866	13,461	13,461
80	50,924	14,633	14,633
90	51,921	15,393	15,393
100	52,535	17,122	17,122

Tabella 4.7: Movimento 7: Distanze massime riscontrate tra le diverse traiettorie rispetto a quella reale presa in riferimento.

Movimento 7			
Velocità (%)	Traiettoria primitiva (mm)	Traiettoria cubica (mm)	Traiettoria a tratti (mm)
5	31,040	7,485	7,860
10	33,804	4,410	7,936
15	43,369	7,523	14,033
25	52,946	17,256	17,122
35	61,447	25,899	18,912
50	68,648	34,173	15,776
60	72,248	27,772	18,699
70	71,278	25,757	18,158
80	70,514	23,355	20,251
90	68,384	20,241	16,240
100	65,786	14,809	13,714

Tabella 4.8: Movimento 8: Distanze massime riscontrate tra le diverse traiettorie rispetto a quella reale presa in riferimento.

Movimento 8

Velocità (%)	Traiettoria primitiva (mm)	Traiettoria cubica (mm)	Traiettoria a tratti (mm)
5	34,265	22,412	34,265
10	30,603	19,453	30,603
15	27,626	21,534	27,626
25	24,559	22,621	24,559
35	25,611	18,175	25,611
50	25,549	17,610	25,549
60	26,993	14,428	26,993
70	27,868	18,355	29,608
80	25,071	19,532	29,854
90	23,325	19,675	29,072
100	25,214	17,809	26,179

Conclusioni

In questa tesi è stato sviluppato un modello analitico con lo scopo di stimare la traiettoria reale percorsa da un manipolatore durante l'esecuzione di un movimento in continuous path. Le pianificazioni dei movimenti, il calcolo dei punti aggiuntivi e le relative interpolazioni sono stati ricavati in MATLAB.

Grazie al robot allestito nel dipartimento di Robotica, mediante l'esecuzione di diversi movimenti, è stato possibile raccogliere i dati sperimentali. Confrontando la traiettoria stimata mediante interpolazione, con quella sperimentale, i risultati ottenuti dimostrano una buona approssimazione tra le due.

La distanza massima calcolata è risultata essere inferiore ai 20 millimetri, ottenuta solo in un caso particolare. Nella maggior parte dei casi l'errore è intorno ai 10 millimetri, valore ritenuto accettabile in quanto presente solo nell'intorno dei punti intermedi.

Uno dei possibili sviluppi di questo lavoro è rappresentato dalla modifica dell'algoritmo di interpolazione, ad esempio aumentandone i punti intermedi, oppure dando una distribuzione non uniforme. Un'ulteriore sviluppo può riscontrarsi nella stima dei punti di inizio e fine arco, valutando l'ipotesi che siano dipendenti dall'ampiezza del movimento eseguito.

Bibliografia

[1] G. Legnani, *Robotica Industriale*. Casa Editrice Ambrosiana, 2003.

[2] G. F. Gurini, "Modellazione cinematica e pianificazione del movimento per semplici robot planari e per il robot planare farm dell'enea di frascati," Tesi di Laurea in Ingengeria Informatica, indirizzo Automazione, Universitï¿½ degli Studi di Roma, 2003-2004.

[3] N. Giordani, "Analisi e ottimizzazione di cella robotizzata flessibile," Tesi di Laurea Magistrale in Ingegneria Meccanica, Dipartimento di Ingegneria Industriale, 2013-2014.

[4] S. EPSON CORPORATION, *SPEL+ Language Reference*, ver.7.0.

[5] D.L.Pieper, "The kinematics of manipulators under computer control," PhD thesis, Department of Mechanical Engineering,Stanford University, 1968.

[6] M. Knapp-Cordes and B. McKeeman, *Improvements to tic and toc Functions for Measuring Absolute Elapsed Time Performance in MATLAB*. MathWorks, 2011.

Ringraziamenti

Qua vanno messi i ringraziamenti (facoltativi)