

UNIVERSITÀ DEGLI STUDI DI PARMA  
FACOLTÀ DI INGEGNERIA  
Corso di Laurea Specialistica in Ingegneria Informatica

PRESA DI OGGETTI CON UN BRACCIO ROBOTICO  
MEDIANTE CONTROLLO VISIVO

Relatore:

Chiar.mo Prof. CORRADO GUARINO LO BIANCO

Correlatori:

Ing. JACOPO ALEOTTI

Ing. OSCAR GERELLI

Tesi di laurea di:

FRANCESCA FANFONI

Anno Accademico 2006/2007

# Indice

<b>Introduzione</b>	<b>2</b>
<b>1 Architettura del manipolatore Manus</b>	<b>7</b>
1.1 Il manipolatore Manus . . . . .	7
1.2 Le caratteristiche meccaniche e la cinematica . . . . .	8
1.2.1 Cinematica diretta . . . . .	11
1.2.2 Cinematica inversa . . . . .	14
1.3 Il movimento del robot . . . . .	15
1.3.1 La pianificazione della traiettoria . . . . .	16
1.3.2 Il sistema di controllo del manipolatore realizzato attraverso il regolatore PID . . . . .	19
<b>2 Visual servoing</b>	<b>22</b>
2.1 Le tecniche di asservimento visivo nei sistemi di controllo . . . . .	22
2.2 L'architettura generale del sistema e la comunicazione con il mani- polatore . . . . .	32
2.3 Le specifiche hardware del sistema di visione . . . . .	35
<b>3 Progetto software per il visual servoing</b>	<b>38</b>
3.1 Il software per il sistema di controllo . . . . .	38
3.2 Le librerie software . . . . .	47
3.2.1 ARToolkit . . . . .	47
3.2.2 L'integrazione di un'applicazione ARToolkit nel sistema YA- RA . . . . .	53
3.3 L'algoritmo di detection del marker . . . . .	55

---

3.4	L'aggiornamento della traiettoria in real-time . . . . .	58
<b>4</b>	<b>Risultati sperimentali</b>	<b>60</b>
4.1	Valutazione della stima della posizione dell'oggetto . . . . .	60
4.2	Valutazione dell'esecuzione della traiettoria per la presa dell'oggetto	63
4.2.1	Una singola traiettoria . . . . .	63
4.2.2	Aggiornamento della traiettoria in real-time . . . . .	67
	<b>Conclusioni</b>	<b>68</b>
	<b>Appendici</b>	<b>70</b>
<b>A</b>	<b>Cinematica del robot</b>	<b>70</b>
A.1	La convenzione di Denavit-Hartenberg . . . . .	70
	Versione standard della convenzione di Denavit-Hartenberg . . . . .	70
	Versione modificata della convenzione di Denavit-Hartenberg . . . . .	74
A.2	Il calcolo della cinematica diretta . . . . .	76
A.3	Il calcolo della cinematica inversa . . . . .	77
<b>B</b>	<b>YARA</b>	<b>81</b>
	<b>Bibliografia</b>	<b>83</b>

# Elenco delle figure

1	Manus su una sedia a rotelle . . . . .	5
1.1	Robot Manus con input device (joystick, keypad), box per invio dei comandi, e LED matrix display per visualizzare lo stato . . . . .	8
1.2	Robot Manus disponibile nel Dipartimento di Ingegneria dell'Informazione . . . . .	8
1.3	Struttura meccanica . . . . .	10
1.4	Posizionamento terne di riferimento secondo Denavit-Hartenberg .	13
1.5	Schema generico di un regolatore PID. . . . .	20
2.1	A sinistra un sistema eye-in-hand, a destra una configurazione eye-to-hand . . . . .	24
2.2	Position based visual servoing . . . . .	25
2.3	Image based visual servoing . . . . .	25
2.4	Modello della telecamera . . . . .	27
2.5	Sistema di riferimento per il calcolo di $T_C^B$ . . . . .	31
2.6	Sistema di riferimento per il calcolo di $T_M^B$ . . . . .	31
2.7	Schema dell'elettronica di controllo . . . . .	34
2.8	Scheda CPC-PCI . . . . .	35
2.9	Robot Manus e telecamera CCD . . . . .	35
3.1	Architettura del sistema di controllo del Manus mediante YARA . .	39
3.2	Architettura del sistema di controllo del Manus basato sulla visione mediante YARA . . . . .	40
3.3	Sequenza di attivazione di <i>trajectory</i> . . . . .	42

3.4	Sequenza di attivazione di <i>gripper</i> . . . . .	42
3.5	Sequenza di comando della traiettoria attraverso l'Activity <i>look</i> . . . . .	45
3.6	Pattern per la calibrazione . . . . .	50
3.7	Modalità di determinazione della matrice di proiezione prospettica . . . . .	50
3.8	Pattern Hiro e Pattern Kanji . . . . .	51
3.9	Multi-pattern . . . . .	51
3.10	Schema di funzionamento di un'applicazione di ARToolkit . . . . .	52
3.11	Diagramma UML dell'interfaccia Vision per la classe DrawTool . . . . .	55
3.12	Sequenza di comando della traiettoria con riaggiornamento della pianificazione . . . . .	59
4.1	Marker utilizzato per il riconoscimento dell'oggetto . . . . .	60
4.2	Immagine acquisita dalla telecamera . . . . .	61
4.3	Immagine binarizzata con threshold fissa a 100 . . . . .	61
4.4	Istogramma dei livelli di grigio . . . . .	62
4.5	Immagine binarizzata con threshold ottimizzata . . . . .	62
4.6	Percorso eseguito dal braccio robotico durante la presa dell'oggetto . . . . .	65
4.7	Traiettoria pianificata per il sesto giunto vs traiettoria eseguita . . . . .	65
4.8	Velocità per il sesto giunto . . . . .	66
4.9	Errore di inseguimento commesso dal sesto giunto . . . . .	66
A.1	Terne secondo la convenzione di Denavit-Hartenberg . . . . .	71
A.2	Terne secondo la convenzione di Denavit-Hartenberg modificata . . . . .	74

# Elenco delle tabelle

1.1	Caratteristiche meccaniche . . . . .	9
1.2	Parametri cinematici di DH . . . . .	12
1.3	Lunghezze Manus espresse in mm . . . . .	12
1.4	Formato Via Point nello spazio operativo . . . . .	17
1.5	Formato Via Point nello spazio dei giunti . . . . .	17
2.1	Distanze tra i sistema di riferimento definiti da camera frame e tool frame, espresse in [m] . . . . .	29
2.2	Caratteristiche tecniche della scheda CPC-PCI. . . . .	34
2.3	Caratteristiche telecamera Sharp CCD . . . . .	36
3.1	Requisiti di installazione . . . . .	48
3.2	Campo di visibilità di ARToolkit . . . . .	53
4.1	Distanza misurata $d(t_i)$ all'istante iniziale vs distanza percorsa $d(t_f)$ in metri . . . . .	64
A.1	Matrici di trasformazione dei giunti (Denavit-Hartenberg) . . . . .	76
A.2	Soluzioni della cinematica diretta . . . . .	77
A.3	Coefficienti della matrice di rotazione relativa ai primi tre giunti. . .	79

*“Non dir sempre quello che sai,  
ma fa di saper bene quello che dici.”*  
*San Giovanni Bosco*

*Ai miei genitori,*

# Introduzione

Il numero di centri di ricerca e di imprese che operano a livello mondiale nel settore della robotica e dell'automazione è in costante crescita. Il mercato dei robot non si rivolge solo all'ambito della produzione industriale, ma negli ultimi anni hanno trovato diffusione anche robot domestici, i robot per la chirurgia e la riabilitazione, i microrobot che esplorano il corpo umano. Nel campo dell'Assistive Technology (AT), ad esempio, la robotica viene utilizzata per creare delle tecnologie e dei prodotti che compensano le limitazioni funzionali, migliorano la qualità della vita e aiutano le persone con esigenze specifiche (disabili e anziani con o senza deficit motori, sensoriali o cognitivi).

La sicurezza è un requisito fondamentale nel progetto di robot destinati a svolgere dei compiti che prevedano un'interazione con operatori umani. I classici manipolatori industriali devono soddisfare dei requisiti che riguardano fondamentalmente la rapidità e l'accuratezza nell'inseguimento di traiettorie prefissate all'interno di ambienti noti. In queste applicazioni la rigidità dei robot consente di avere ottime performance. Nel campo dell'Assistive Technology i robot sono spesso chiamati ad operare in ambienti non conosciuti, e ad interagire con operatori umani. In tutti questi casi il requisito fondamentale del robot è quello di non costituire un pericolo per coloro che gli stanno vicino. Le specifiche di progetto più importanti diventano allora la sicurezza e l'affidabilità. Nell'interazione uomo-robot, un manipolatore industriale, caratterizzato da un'elevata rigidità, difficilmente riesce ad assicurare un livello di sicurezza accettabile. Una strategia per proteggere l'incolumità delle persone è quella di dotare il robot di sensori, che possono produrre una riduzione della velocità in prossimità di ostacoli imprevisti. La presenza di un numero elevato di sensori può comportare però l'introduzione di forti ritardi di attuazione e compromette anche la rapidità di intervento.

Il controllo visuale del braccio robotico permette di avere sostanziali vantaggi in tutte le applicazioni in cui gli oggetti con il quali il robot deve interagire occupano una posizione imprecisate e non nota a priori. Per questa ragione, ove sia possibile, si impiega un sistema di visione come unico sensore di misura. L'asservimento visivo è infatti in grado di ricostruire la posizione di un oggetto utilizzando le immagini riprese da una o più telecamere, e con queste informazioni consente di riuscire ad impartire al robot i comandi per raggiungere l'oggetto prefissato. Lo schema di controllo considerato in questa tesi, denominato *position based visual servoing*, è stato implementato attraverso la realizzazione di un sistema *eye-in-hand*, caratterizzato dal posizionamento della telecamera direttamente sull'organo di presa del braccio robotico. Nelle tecniche tradizionali di asservimento visivo viene invece spesso effettuata la misura diretta della posizione del manipolatore fornita dalla telecamera, per migliorare l'accuratezza del posizionamento della pinza. I dati derivanti dalla telecamera, in questo caso, incrementano la precisione delle informazioni fornite dall'apparato meccanico del robot, in quanto non risultano affetti dai tipici errori dovuti ad un'imprecisa conoscenza dei parametri fisici del manipolatore, che risultano particolarmente evidenti nel caso in cui si debba considerare anche la flessibilità della struttura.

Nella progettazione del *position based visual servoing* è stato necessario utilizzare algoritmi di elaborazione dell'immagine, in modo da ottenere la posizione dell'oggetto in coordinate cartesiane. L'utilizzo di un sistema di visione consente, in generale, di aumentare l'autonomia del sistema. Ad esempio, il controllo efficiente di pezzi meccanici in ambito industriale può richiedere l'osservazione del prodotto da posizioni diverse con un controllo continuo dell'illuminazione e dell'orientamento del pezzo rispetto alla telecamera. In questo caso un manipolatore può essere utilizzato per spostare il pezzo intorno ad uno o più sistemi di visione, oppure la telecamera ed il sistema di illuminazione possono essere movimentati mediante un braccio robotico.

Uno degli aspetti cruciali della visione robotica è legato al dover affrontare problemi di interpretazione delle scene del mondo reale: le immagini prese in considerazione sono tratte dal mondo circostante e, quindi, presentano tutte le anomalie e le irregolarità degli oggetti reali, diversamente da quello che avviene con immagini sintetiche o acquisite in condizioni completamente controllate. Inoltre, un'appli-

cazione robotica necessita dell'utilizzo di tecniche non troppo costose in termini computazionali, che consentano di eseguire delle elaborazioni in tempo reale.

L'ambiente, in cui si muove il robot, possiede alcune caratteristiche che inevitabilmente influenzano il processo di comprensione della scena. Nei sistemi robotici dotati di sensori di visione vengono spesso applicate le seguenti ipotesi:

- il robot, oggetto di studio, deve essere un agente autonomo, per cui non è possibile alcun intervento esterno per controllarlo;
- gli oggetti da riconoscere presenti nell'immagine devono essere fortemente caratterizzati;
- le condizioni di illuminazione della scena sono in genere fortemente variabili, questo influisce molto sul riconoscimento degli oggetti;
- le occlusioni, rendono impossibile, tranne in rari casi, il riconoscimento degli oggetti basandosi sull'analisi della loro forma.

Tutti questi aspetti hanno come effetto la richiesta di una necessaria, efficiente e scientifica fase di sperimentazione delle soluzioni adottate: teoricamente alcune tecniche potrebbero produrre risultati accettabili, ma, una volta testate e valutate, potrebbero rivelarsi inefficaci.

Questa tesi si inquadra proprio nel filone della nella robotica di servizio considerando lo studio di un possibile controllo per il manipolatore Manus. Il Manus è un manipolatore posizionabile su sedia a rotelle (vedi figura 1), che consente di aiutare utenti con deficit motori nello svolgimento degli All-Day-Living task. Il Manus, realizzato dall'azienda olandese Exact Dynamics, è un manipolatore a bassa impedenza meccanica, con sei gradi di libertà, appositamente progettato per compiti di assistenza, ed impiegato presso il Dipartimento di Ingegneria dell'Informazione con finalità di ricerca. In particolare, lo studio proposto è incentrato sull'introduzione di un meccanismo di visione all'interno del sistema di controllo, in modo da rendere più versatile l'utilizzo della piattaforma anche in ambienti non strutturati. Il sistema di controllo, sviluppato in lavori precedenti, permette di far compiere al manipolatore dei movimenti programmati, tramite un algoritmo di pianificazione della traiettoria con punti definibili sia nello spazio operativo che nello spazio dei

giunti e mediante l'uso di un sistema di attuazione retroazionato. Tale architettura ha rappresentato la base di partenza per lo sviluppo della tecnica di controllo tramite asservimento visivo.



**Figura 1:** Manus su una sedia a rotelle

Il progetto sviluppato ha permesso di integrare delle tecniche di visione artificiale applicate all'ambiente reale nel controllo del movimento del manipolatore. L'aumento delle capacità sensoriali ha permesso di incrementare sia l'autonomia delle funzionalità del robot, sia il livello di sicurezza nell'interazione con il braccio robotico attraverso l'unione di due campi di ricerca distinti, come la robotica e la visione. Ai fini del controllo visivo la telecamera acquisisce delle immagini per l'identificazione della posizione di un target assegnato nello spazio di lavoro del sistema robotico. La gestione del controllo è implementato nell'ambiente real-time YARA, framework che consente lo sviluppo di applicazioni per la robotica secondo uno schema behaviour-based. Durante la fase sperimentale sono stati valutati i vincoli temporali necessari per la gestione dei processi di controllo, sia periodici che aperiodici, da parte dell'algoritmo di scheduling. È necessaria infatti un'adeguata sincronizzazione tra i task di controllo per l'esecuzione di un compito in tempo reale, come la presa di un'oggetto. Ogni task è caratterizzato da un istante di tempo

entro il quale deve essere eseguito (deadline), e il rispetto di questo vincolo consente la correttezza dei calcoli per la determinazione della traiettoria di moto. Un'ultima analisi che è stata affrontata riguarda la possibilità di aggiornare la traiettoria seguita dal braccio robotico in tempo reale durante il moto, anche in questo caso il rispetto delle condizioni temporali ha assunto la massima importanza.

La stesura della tesi è suddivisa in una prima parte in cui vengono introdotte delle nozioni teoriche riguardanti la meccanica del braccio robotico e le tecniche di controllo, ed in seguito sono illustrate gli aspetti più tecnici.

Nel primo capitolo della tesi viene presentata l'architettura del manipolatore Manus, attraverso la descrizione delle caratteristiche meccaniche del robot. Lo studio della cinematica diretta ed inversa del manipolatore antropomorfo è realizzato tramite l'applicazione della convenzione di Denavit-Hartenberg, che permette di assegnare le terne di riferimento sui bracci. In questa parte viene illustrata anche la teoria su cui si basa il moto del braccio robotico. In particolare viene descritta la pianificazione della traiettoria e il controllo robot, realizzato mediante regolatori PID.

Nel secondo capitolo vengono descritte le diverse tecniche di utilizzo della visione nei sistemi di controllo dei robot. I differenti approcci noti nella letteratura robotica vengono classificati in base agli aspetti più caratterizzanti. In questa sezione viene proposta una panoramica sull'architettura hardware e software preesistente, su cui ci si è basati per sviluppare il sistema di asservimento visivo.

Nel terzo capitolo viene presentato in dettaglio il software necessario l'esecuzione del visual servoing. La struttura di base del software, realizzata in modo modulare attraverso l'uso del framework Yara, è stata modificata per consentire la gestione del sensore per la visione. In questo capitolo viene descritta la libreria di visione ARToolkit, utilizzata per l'elaborazione delle immagini e le modifiche ad essa introdotte per adattarlo alle finalità di controllo.

Nel quarto capitolo vengono riassunti i risultati sperimentali ottenuti .

# Capitolo 1

## Architettura del manipolatore Manus

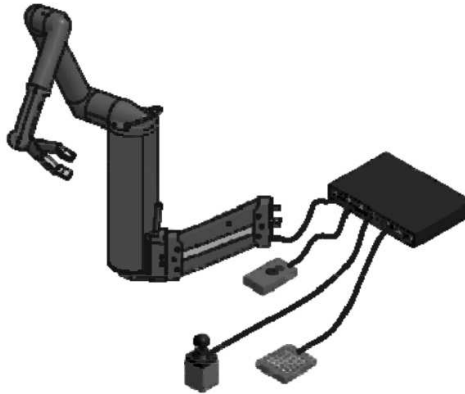
In questo capitolo viene illustrata una descrizione delle caratteristiche meccaniche del robot e la modalità secondo cui sono fissate le terne sui giunti secondo la convenzione di Denavit-Hartenberg. E viene proposta una breve descrizione sulla pianificazione delle traiettorie e sul controllo realizzato attraverso il regolatore PID.

### 1.1 Il manipolatore Manus

Il Manus, chiamato anche ARM (Assistive Robotic Manipulator), prodotto dalla azienda olandese Exact Dynamics [1], è un robot in grado di assistere le persone disabili con handicap agli arti superiori. La possibilità di installazione su una base mobile o su una sedia a rotelle consente al Manus di sostituire l'utilizzo di braccio e mano. La sicurezza dell'utente disabile durante l'utilizzo del robot deve essere garantita, e per questo motivo nella progettazione della piattaforma robotica è stato studiato, oltre che l'andamento delle forze esercitate dal robot, anche un sistema di sicurezza che permetta di fermare il robot al verificarsi di qualsiasi tipo di problema.

Il robot Manus può essere utilizzato in modo operativo e non automatico. Sono disponibili infatti un tastierino e un joystick che consentono l'azionamento del robot in modo manuale, come si può vedere in figura 1.1. Il keypad dispone anche di un display che consente di visualizzare lo stato in cui si trova il robot. Nel caso in cui si verifichi una configurazione non sicura viene visualizzato un simbolo di errore ed emesso un segnale acustico. In questa modalità di utilizzo l'utente disabile è in

grado di controllare direttamente il sistema, decidendo la posizione e l'orientamento da raggiungere. Nella progettazione del robot sono quindi stati perseguiti oltre che gli obiettivi di efficienza ed efficacia, anche la sicurezza dell'utente.



**Figura 1.1:** Robot Manus con input device (joystick, keypad), box per invio dei comandi, e LED matrix display per visualizzare lo stato



**Figura 1.2:** Robot Manus disponibile nel Dipartimento di Ingegneria dell'Informazione

## **1.2 Le caratteristiche meccaniche e la cinematica**

Il manipolatore antropomorfo Manus possiede sei gradi di libertà, dovuti ai sei giunti rotoidali, ai quali può essere aggiunto un grado di libertà dovuto al gripper, l'organo di presa posto all'estremità robot, che può essere aperto e chiuso. È possibile

considerare anche un'ulteriore grado di libertà introdotto dal supporto mobile in grado di variare l'altezza della base del manipolatore. La possibilità di poter posizionare ed orientare il Manus in modo arbitrario all'interno del suo spazio di lavoro, rappresentato da una sfera di raggio pari a 80 cm, è consentita dai sei gradi di libertà. La geometria del sistema meccanico è mostrata dalla figura 1.4 e nella tabella 1.1 vengono elencati le principali caratteristiche dimensionali e di funzionamento.

La piattaforma robotica è stata progettata in modo da potere essere installata su una sedia a rotelle predisposta o in alternativa su una base mobile. In entrambi i casi è stato necessario garantire un basso ingombro ed un peso limitato. A questo scopo è stata scelta come collocazione per i motori l'interno della base principale, e sono state usate delle cinghie collegate direttamente ai motori per il trasferimento del moto sui giunti. Se da un lato ciò è consentito di non appesantire la struttura del braccio robotico, portando il peso totale a circa 13 Kg, dall'altro l'utilizzo di cinghie ha introdotto dei giochi e delle non linearità nella catena cinematica. Inoltre il massimo carico trasportabile attraverso l'organo di presa è limitato a 1.5 Kg.

DOF:	6 + 1 (gripper) + 1 (lift, optional)
Dimensions:	Reach: 80 cm Weight: 14.3 kg Payload: 1.5 kg
Max. velocity	9.9 cm/s
Max. rot. vel	30°/s (0.52 rad/s)
Power supply	24V DC, 1.5A (nominal), 3A (peak)
Grasp force	20N
Max. opening	9 cm
Operating conditions	Temperature: -5°C to 50°C Humidity <90%

**Tabella 1.1:** Caratteristiche meccaniche

L'impiego di cinghie evita la presenza di cablaggi all'interno dei bracci, e consente di far compiere ai giunti più di un giro completo attorno al proprio asse di rotazione. Un'altro vantaggio è l'incremento di sicurezza, infatti grazie allo slittamento delle cinghie è possibile controllare la massima forza di rotazione applicata su ciascun giunto. Le imprecisioni di movimento sono proprio dovute alle cinghie e non sono da sottovalutare perchè non consentono misure precise sugli angoli di rotazione dei giunti, creando problemi al sistema di controllo. Dal punto di vista della potenza elettrica richiesta, la riduzione del peso ha permesso l'uso di motori

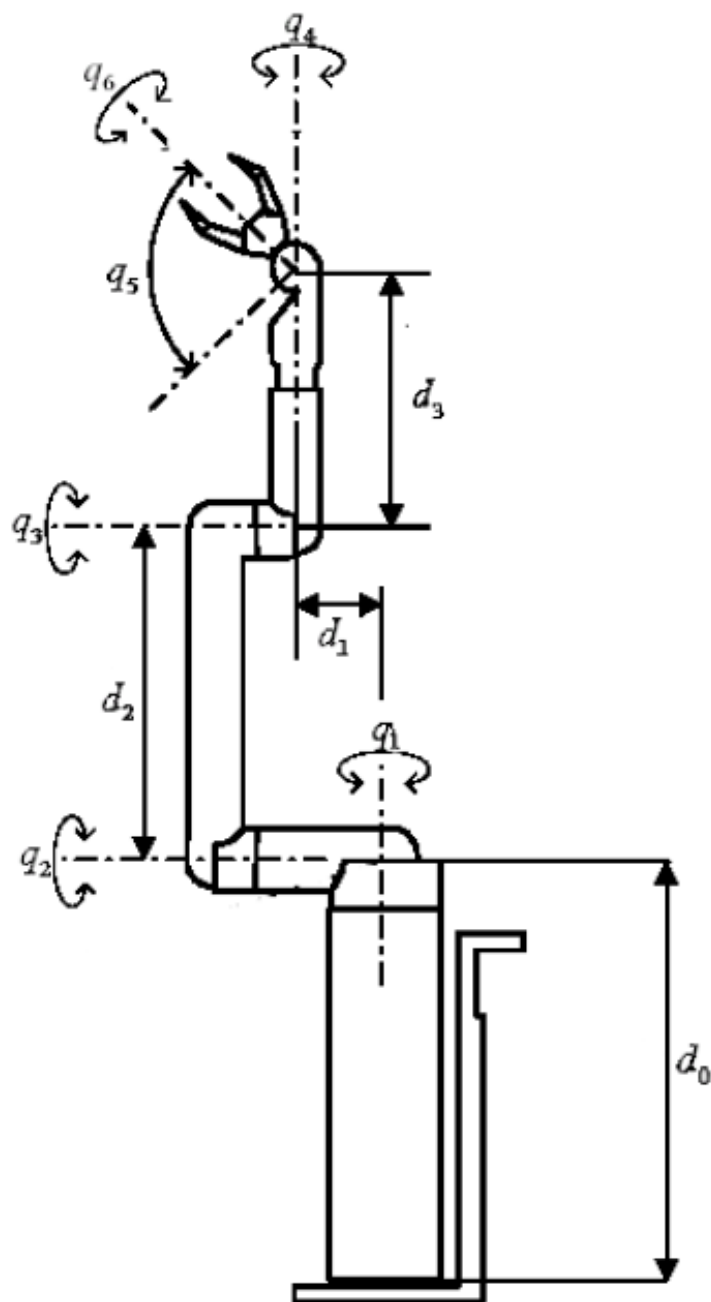


Figura 1.3: Struttura meccanica

elettrici più piccoli, leggeri e a basso consumo energetico, consentendo una maggior durata delle batterie che alimentano il sistema.

All'estremità del manipolatore, si trova il gripper che consente di afferrare oggetti ed interagire attivamente con l'ambiente circostante. Le due estremità sono state progettate in modo da muoversi in parallelo fra loro, così da garantire una presa costante. L'apertura e la chiusura del gripper sono realizzate da un attuatore collocato come tutti gli altri nella base del manipolatore, e sempre collegato tramite una cinghia. Per lo svolgimento di questa tesi, sulla pinza del gripper è stata installata una piccola telecamera, per la realizzazione di un sistema *eye in hand*, descritto nel capitolo 2.

### 1.2.1 Cinematica diretta

Il problema cinematico diretto consiste nel determinare la posizione e l'orientamento dell'end effector del manipolatore, a partire dalle coordinate di giunto. A questo scopo vengono definiti le relazioni tra gli assi di giunto e di braccio attraverso un sistema di riferimento solidale con i bracci del robot usando delle matrici di trasformazione omogenea. La definizione del sistema di riferimento per ogni giunto viene specificata attraverso l'applicazione della convenzione di Denavit-Hartenberg (per maggiori dettagli in merito vedasi l'appendice A). Nella tabella 1.2 vengono elencati i parametri della convenzione di Denavit-Hartenberg per il robot Manus, utilizzabili per il calcolo delle matrici di trasformazione, riportate in appendice A.

Il Manus è caratterizzato da sei gradi di libertà conferiti da sei giunti rotoidali. Tali giunti non sono perfettamente indipendenti in quanto esiste una relazione fra il moto del secondo e quello del terzo giunto: ad una rotazione di  $\Delta\theta_2$  gradi da parte del secondo giunto, il sistema fa compiere una rotazione di  $-\Delta\theta_2$  gradi al terzo giunto in modo da mantenere costante la direzione dell'asse del terzo braccio. Per poter considerare effettivamente indipendenti i sei giunti del manipolatore viene compensata questa dipendenza sommando al valore restituito dall'encoder del terzo giunto il valore dell'angolo del secondo giunto:

$$\theta_3 = \theta_3 + \theta_2 \quad (1.1)$$

$i$	$a_i$	$\alpha_i$	$d_i$	$\theta_i$
1	0	-90	0	$\theta_1$
2	$L_2$	0	$L_1$	$\theta_2$
3	0	90	0	$\theta_3$
4	0	-90	$L_3$	$\theta_4$
5	0	90	0	$\theta_5$
6	0	0	$L_4$	$\theta_6$

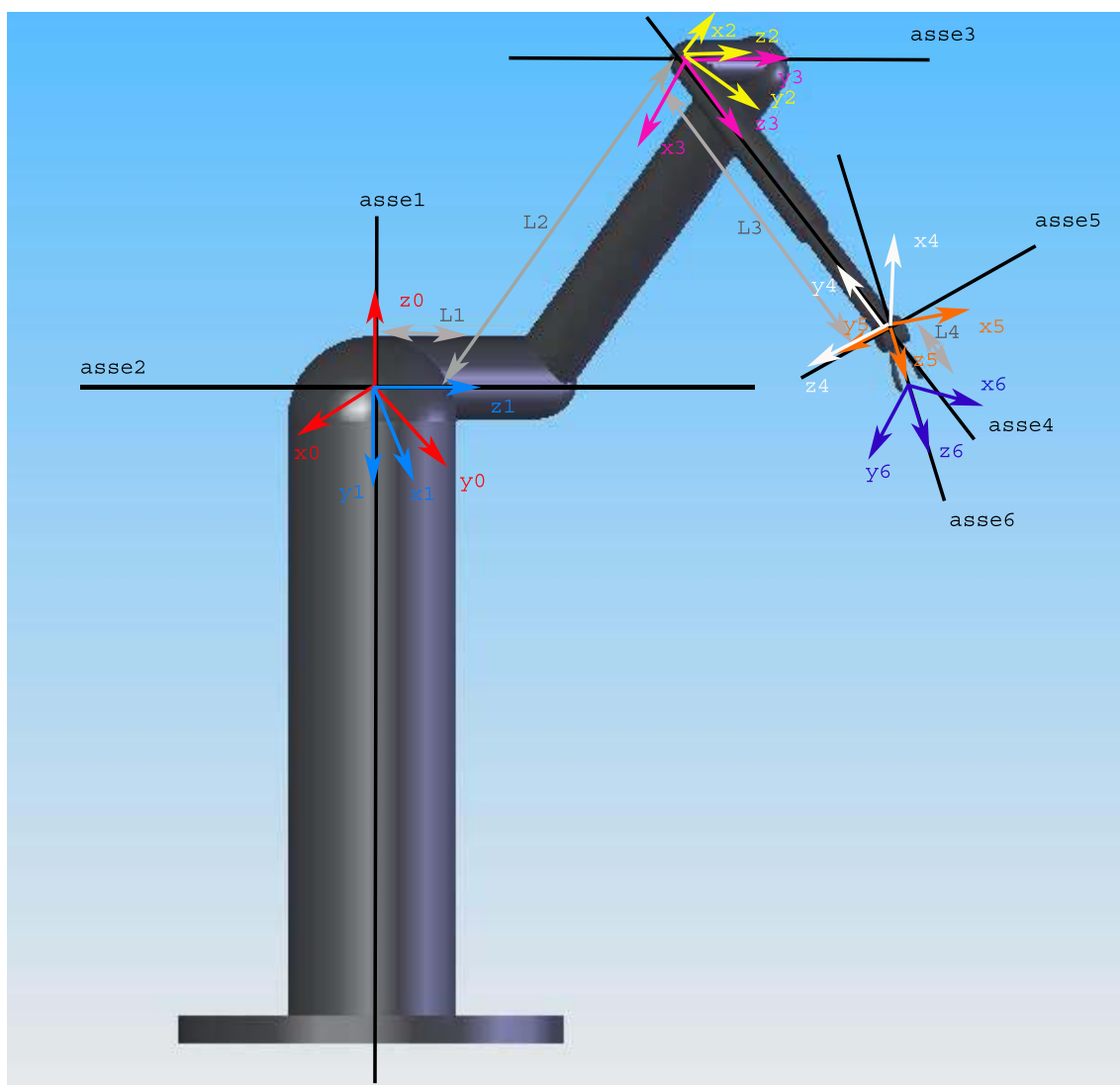
**Tabella 1.2:** Parametri cinematici di DH

$L_1$	105
$L_2$	400
$L_3$	320
$L_4$	160

**Tabella 1.3:** Lunghezze Manus espresse in mm

L'indipendenza dei giunti è una caratteristica importante del manipolatore in quanto permette la risoluzione del problema cinematico inverso in forma chiusa tramite disaccoppiamento. Le terne di riferimento e i parametri cinematici consentono di ricavare facilmente, per ciascuna coppia di bracci adiacenti, la trasformazione geometrica che lega la terna  $i$  alla terna  $i + 1$ , applicando la teoria esposta nel paragrafo A.1. Si ottengono sei matrici di trasformazione che vengono usate per il calcolo della soluzione del problema cinematico diretto. Nel paragrafo A.2 dell'appendice sono riportati in dettaglio tutti i passaggi per l'ottenimento della matrice di trasformazione complessiva, la cui forma è mostrata in tabella 1.2.

$$T_6^0 = A_1^0 A_2^1 A_3^2 A_4^3 A_5^4 A_6^5 = \left( \begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \quad (1.2)$$



**Figura 1.4:** Posizionamento terne di riferimento secondo Denavit-Hartemberg

## 1.2.2 Cinematica inversa

Il problema cinematico inverso consiste nel determinare le coordinate di giunto corrispondenti ad una data posizione ed orientamento dell'end effector del manipolatore che si vuole raggiungere. In generale, non esiste una soluzione univoca a questo problema. La cinematica inversa utilizza come dati di partenza i valori degli elementi della matrice, mostrata nella 1.2, da cui vengono ricavati i valori delle variabili di giunto corrispondenti. Il problema cinematico inverso, a differenza di quello diretto, può per particolari posizioni dell'estremità, non ammettere soluzione. L'insieme di tutti i punti in cui il problema cinematico inverso ammette soluzione coincide con la porzione di piano raggiungibile dall'estremità del sistema e definisce la sua area di lavoro.

La struttura del robot Manus, avente sei gradi di libertà e tre assi di rotazione di tre giunti rotoidali consecutivi che si incontrano in uno stesso punto, consente di risolvere il problema di cinematica inversa. In generale tutte le strutture che utilizzano un polso sferico si trovano in questa condizione e ciò spiega la loro ampia diffusione in ambito industriale.

Il problema cinematico inverso per il robot Manus viene scomposto in due parti. Nella prima fase si trovano i valori delle variabili di giunto relative ai primi tre giunti, scegliendo tra le quattro soluzioni possibili quella che minimizza la norma della differenza tra la posizione corrente e quella desiderata, consentendo quindi di effettuare uno spostamento minimo. Queste soluzioni sono riportate nel seguito (per maggiori dettagli riguardo al calcolo vengono riportati in appendice nel paragrafo A.3).

$$\tilde{\theta}_1 = 2 \arctan \left( \frac{-x_w \pm \sqrt{x_w^2 + y_w^2 - L_1^2}}{y_w + L_1} \right)$$

$$\tilde{\theta}_2 = \text{Atan2}(s_2, c_2)$$

$$\tilde{\theta}_3 = \pi - \arcsin \left( \frac{x_w^2 + y_w^2 + z_w^2 - L_1^2 - L_2^2 - L_3^2}{2L_2L_3} \right)$$

Anche per le ultime tre variabili di giunto esistono due possibili configurazioni rappresentata dalle equazioni e per ognuna viene calcolata la soluzione riportata nell'equazione A.12 dell'appendice, nel caso in cui  $\hat{\theta}_5 = 0$  o  $\hat{\theta}_5 = \pi$ . Viene scelta

anche in questo caso la soluzione che minimizza la norma della differenza della posizione corrente con la posizione desiderata.

$$\begin{aligned} \hat{\theta}_5 &= 0 & \hat{\theta}_5 &= \pi \\ \hat{\theta}_6(t) &= \theta_5(\hat{t} - 1) & \hat{\theta}_6(t) &= \theta_5(\hat{t} - 1) \\ \hat{\theta}_4 &= \text{Atan2}(r_{21}, r_{11}) - \hat{\theta}_6 & \hat{\theta}_4 &= \text{Atan2}(-r_{21}, -r_{11}) + \hat{\theta}_6 \end{aligned}$$

La soluzione delle cinematica inversa consite nel vettore descritto dalla seguente espressione:

$$\theta = \begin{pmatrix} \tilde{\theta} \\ \hat{\theta} \end{pmatrix} = \left( \theta_1 \quad \theta_2 \quad \theta_3 \mid \theta_4 \quad \theta_5 \quad \theta_6 \right)^T$$

### 1.3 Il movimento del robot

I sistemi robotici sono generalmente soggetti a numerose sorgenti di errore di tipo ripetitivo e/o causale che non consentono di posizionare il robot in un determinato modo accurato. L'approssimazione nella generazione del movimento richiesto può essere causata da imperfezioni costruttive, che determinano errori geometrici o giochi, deformazioni termiche ed elastiche o infine imprecisioni del sistema di controllo. La posizione e l'orientamento del robot può essere definita in modo fisso rispetto all'ambiente in cui è installato il robot, o può essere mobile e solidale con gli elementi del robot. In generale esistono due modalità differenti per il controllo del robot. La prima detta *posa a posa* è caratterizzata dalla specifica della posa da raggiungere ed eventualmente della velocità di moto. Questo metodo può causare che il movimento dei diversi assi del robot avvenga in modo simultaneo o consecutivo e non coordinato tra loro. L'altro schema di controllo è definito *continuo in traiettoria* ed implica che i diversi assi del robot vengano comandati in modo coordinato affinché il dispositivo di estremità raggiunga la posa comandata seguendo la traiettoria richiesta e rispettando anche la velocità di esecuzione della traiettoria. Nello svolgimento di questa tesi è stato scelto l'approccio *continuo in traiettoria*, in quanto consentono di guidare con continuità il movimento del gripper agevolando il lavoro del sistema di visione.

### **1.3.1 La pianificazione della traiettoria**

La pianificazione della traiettoria stabilisce la modalità con cui si evolve il movimento del manipolatore, da una postura iniziale ad una postura finale. Si tratta di definire sia il percorso geometrico sia la legge di moto da realizzare (ossia la sequenza temporale di posizioni, velocità ed accelerazioni)

Per lo svolgimento di questa tesi è stato utilizzato un algoritmo di calcolo della traiettoria [2], suddivisibile in tre parti fondamentali: definizione dei punti di passaggio, analisi e calcolo della traiettoria.

#### **Specifica dei punti di passaggio**

Il percorso che il manipolatore dovrà eseguire viene specificato all'interno di un file di testo come una sequenza di "Via Point", ovvero punti di passaggio. Nella versione precedente del software non integrato con il sistema di visione, il nome del file di testo veniva definito come argomento di riga al momento dell'esecuzione del programma. Nel file i punti vengono specificati nello spazio cartesiano o in quello dei giunti, inoltre all'interno dello stesso file si possono alternare punti nello spazio dei giunti e punti specificati nello spazio cartesiano. L'unica limitazione di cui si deve tener conto è che non devono essere specificati dei via point nello spazio dei giunti relativi rispetto al punto precedente nel caso in cui quest'ultimo fosse stato specificato nello spazio cartesiano. Il formato del file di specifica dei via points è mostrato in tabella 1.4 se espressi nello spazio operativo oppure in quella mostrata in tabella 1.5 se espressi nello spazio dei giunti.

Nel caso di definizione nello spazio dei giunti i primi tre campi verranno specificati i valori delle rotazioni, espresse in gradi, che si desidera far compiere ai primi tre giunti, nei seguenti tre campi si potranno indicare le velocità, espresse in gradi/s, relative ai primi tre giunti, poi ancora si indicheranno le rotazioni da far compiere agli altri tre giunti (il quarto, il quinto e il sesto) e le relative velocità.

Nel caso in cui invece i punti vengano specificati nello spazio cartesiano, i primi tre campi indicheranno la posizione (x, y e z) dell'organo terminale espressa in metri, i seguenti tre campi le velocità relative lungo i tre assi in m/s che tuttavia anche in questo caso saranno solitamente poste a zero, poi ancora si indicherà l'orientazio-

X Y Z	$V_x V_y V_z$	Yaw Roll Pitch	$V_Y V_R V_P$	time	coord	space
-------	---------------	----------------	---------------	------	-------	-------

**Tabella 1.4:** Formato Via Point nello spazio operativo

$J_1 J_2 J_3$	$V_{J_1} V_{J_2} V_{J_3}$	$J_4 J_5 J_6$	$V_{J_4} V_{J_5} V_{J_6}$	time	coord	space
---------------	---------------------------	---------------	---------------------------	------	-------	-------

**Tabella 1.5:** Formato Via Point nello spazio dei giunti

ne (Yaw, Roll, Pitch) espressa in gradi, e le velocità relative in gradi/sec. In entrambi i formati di specifica dei punti sono presenti i campi `time`, `coord` e `space`.

Il campo `time` indica il tempo in secondi che il manipolatore deve impiegare per percorrere lo spazio che separa il via point attuale da quello precedente.

Il campo `coord` può assumere tre valori:

- 0: lo spostamento da compiere è espresso in maniera assoluta;
- 1: lo spostamento da compiere è specificato come un incremento rispetto alla posizione indicata dal primo Via Point;
- 2: lo spostamento da compiere è relativo al punto precedente.

Il campo `space` può assumere i seguenti valori:

- 0: indica che il punto è definito nello spazio dei giunti e pertanto gli altri campi verranno definiti secondo la struttura indicata nella tabella 1.5
- 1: indica che il punto è definito nello spazio operativo e gli altri campi seguiranno la struttura indicata nella tabella 1.4.

Con l'introduzione del sistema di visione il file di definizione dei via point viene creato in real-time, quando si verifica il rilevamento del marker che identifica l'oggetto, obiettivo della presa del braccio robotico, come viene descritto nel capitolo successivo.

### Analisi del percorso

Terminata la lettura dei dati indicanti i punti di passaggio che costituiscono il percorso che il manipolatore dovrà compiere, deve essere attivata la funzione che, a partire

dal percorso assegnato, generi una traiettoria. La traiettoria diventerà poi il riferimento spaziale-temporale del controllore del sistema. Inizialmente, alla ricezione della sequenza dei via point viene determinato il primo punto del percorso, rappresentato dalla configurazione attuale del manipolatore, attraverso la chiamata delle routines `setStartingJoints()` e `setStartingViaPoint()`. All'interno del file di testo, non viene infatti fissato il primo punto del percorso in quanto dipende dalla posizione attuale dei giunti del Manus, e deve essere determinata dai valori restituiti dagli encoder. Le variabili di giunto così ottenute servono per poter calcolare la cinematica diretta allo stato attuale del robot. In seguito verranno determinate le velocità per ciascun via point, e per ogni coppia di punti si calcolerà una spline cubica, si procederà in modo diverso a seconda del tipo di spazio in cui sono definiti i via point. La fase di analisi dei via point permette di calcolare per ciascun punto del percorso assegnato i valori degli angoli dei giunti, o i valori della posizione e dell'orientamento nello spazio operativo, prima di procedere con l'interpolazione della traiettoria, che verrà effettuata nello spazio dei giunti.

### **Interpolazione della traiettoria**

Questa fase consiste nel calcolo della traiettoria, attraverso la valutazione dei coefficienti di una funzione polinomiale del terzo ordine, utilizzando come dati assegnati il punto iniziale, il punto finale e il tempo di percorrenza. Il moto dei giunti viene rappresentato attraverso una spline cubica interpolante, rappresentata dall'equazione 1.3.1, con un profilo di velocità parabolico ed accelerazione lineare, come si vede dalle rispettive espressioni riportate in 1.3.1 e 1.3.1.

$$q(t) = a_3t^3 + a_2t^2 + a_1t + a_0 \quad (1.3)$$

$$\dot{q}(t) = 3a_3t^2 + 2a_2t + a_1 \quad (1.4)$$

$$\ddot{q}(t) = 6a_3t + 2a_2 \quad (1.5)$$

I due passi fondamentali eseguiti in questo stadio, sono rappresentati dalle seguenti funzioni:

1. `computeSpeed()` determina le velocità dei punti intermedi che non sono note a priori.
2. `interpolate()` calcola per ogni coppia di via point consecutivi i coefficienti del polinomio cubico interpolante.

### 1.3.2 Il sistema di controllo del manipolatore realizzato attraverso il regolatore PID

Il controllo del braccio robotico deve permettere l'inseguimento della traiettoria di riferimento. In questa tesi è stato utilizzato il sistema di controllo a giunti indipendenti, oggetto di studio in [2]. Ciascun giunto è controllato come se fosse un sistema "single input-single output", e viene quindi implementato un regolatore di tipo PID, che ne controlla il movimento, mediante il calcolo delle velocità che devono essere applicate per seguire la traiettoria assegnata. Ogni effetto associato dovuto al moto degli altri link o viene ignorato o trattato come disturbo. Il regolatore PID, rappresentato dallo schema di figura 1.5, , contiene tre azioni di controllo proporzionale, integrale e derivativa che appaiono nella legge di controllo (1.6), attraverso le costanti dove  $K_P \geq 0$  per l'azione proporzionale,  $K_I \geq 0$  per l'azione integrale, e  $K_D \geq 0$  per l'azione derivativa.

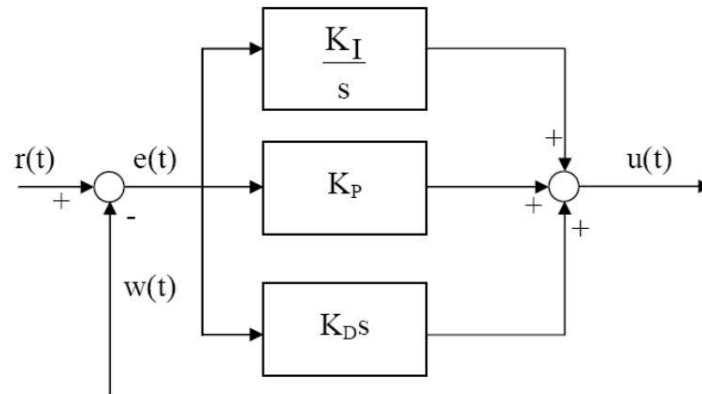
$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt} \quad (1.6)$$

La legge di controllo indicata dalla 1.6, trasformata secondo Laplace diventa la funzione di trasferimento del regolatore PID:

$$U(s) = \left( K_P + \frac{K_I}{s} + K_D s \right) E(s) \quad (1.7)$$

ovvero

$$U(s) = K_P \left( 1 + \frac{1}{T_i s} + T_d s \right) E(s) = K_P \frac{1 + T_i s + T_i T_d s^2}{T_i s} E(s) \quad (1.8)$$



**Figura 1.5:** Schema generico di un regolatore PID.

dove  $K_I = \frac{K_P}{T_i}$  e  $K_D = K_P T_d$  e le costanti  $T_i$  e  $T_d$  sono positive, e hanno le dimensioni di un tempo.

Attraverso opportuni calcoli matematici viene calcolata l'espressione della forma di velocità seguente:

$$u_n = u_{n-1} + K_P \left[ e_n - e_{n-1} + \frac{T}{2T_i}(e_n + e_{n-1}) + \frac{T_d}{T}(e_n - 2e_{n-1} + e_{n-2}) \right] \quad (1.9)$$

Il ciclo di controllo utilizza il valore dei coefficienti del polinomio interpolante della spline cubica corrispondente alla porzione di traiettoria che si vuole generare e la posizione corrente dei giunti. Viene in seguito effettuato il calcolo della posizione e della velocità, per ciascun giunto attraverso la valutazione della forma di velocità descritta dalla (1.9) in modo da avere i valori delle velocità dei giunti, che consentono al manipolatore di realizzare la traiettoria specificata. Prima del loro invio all'interfaccia che consente la comunicazione diretta con il robot Manus, però si effettuano degli aggiustamenti per renderle effettivamente attuabili da parte del manipolatore.

Le prestazioni ottenute mediante il regolatore hanno dimostrato che è in grado di migliorare la precisione e la robustezza ai disturbi per mezzo dell'introduzione di un polo nell'origine, ed anche di permettere una buona prontezza di risposta per via degli zeri che esso introduce, nonché un miglioramento del margine di fase grazie

all'antico introdotto da tali zeri.

# Capitolo 2

## Visual servoing

In questo capitolo verranno descritte le tecniche di elaborazione dell'immagine indispensabili per la creazione di un sistema di asservimento visivo su un manipolatore antropomorfo. In generale gli organi di visione artificiale sono sensori utili per la robotica poiché imitano il senso umano della vista e consentono misure dell'ambiente senza il contatto. Esistono infatti diversi controllori robotici che integrano anche dei sistemi di visione. Nel caso in cui la visione e la manipolazione vengono combinate in una tecnica ad anello aperto allora il sistema di visione guida il sistema di controllo posizionale, tale configurazione è denominata *look and move*. Alternativamente se le misure visive possono essere utilizzate direttamente in un anello di retroazione per realizzare il controllo di posizione in anello chiuso dell'organo terminale, si tratta proprio di *asservimento visivo* o *visual servoing*. Quest'ultimo tipo sarà oggetto di studio di questa tesi.

### 2.1 Le tecniche di asservimento visivo nei sistemi di controllo

L'obiettivo principale del visual servoing è utilizzare le informazioni acquisite tramite il sensore di visione, la telecamera, per chiudere l'anello del sistema di controllo del robot. I dati della visione possono essere acquisiti da una telecamera direttamente installata sul robot manipolatore o su un robot mobile, in entrambi i casi il moto del robot produce anche un moto della telecamera, oppure la teleca-

mera potrebbe essere installata nello spazio di lavoro, in modo da poter osservare il moto del robot da una postazione fissa. Altre configurazioni ibride potrebbero essere generate, utilizzando anche più telecamere contemporaneamente. La trattazione matematica di questi sistemi è pressochè simile.

L'asservimento visivo è realizzato mediante l'applicazione contemporanea di tecniche di elaborazioni delle immagini e strategie tipiche della teoria dei controlli. In un sistema di visione computazionale la complessità e l'imprecisione può essere introdotta da diversi fattori. Ad esempio, nella fase di acquisizione dell'immagine attraverso il frame grabber può essere introdotto un errore di campionamento, ma ancor prima il sensore CCD può introdurre un ulteriore errore dovuto all'approssimazione sulla trasduzione dell'intensità luminosa ed anche la lente può aggiungere approssimazione dovuta alla distorsione radiale. La teoria dei controlli, per un sistema di asservimento visivo generale, ha come obiettivo la minimizzazione della funzione errore  $e(t)$ , tipicamente definita come:

$$e(t) = s(m(t), a) - s^* \quad (2.1)$$

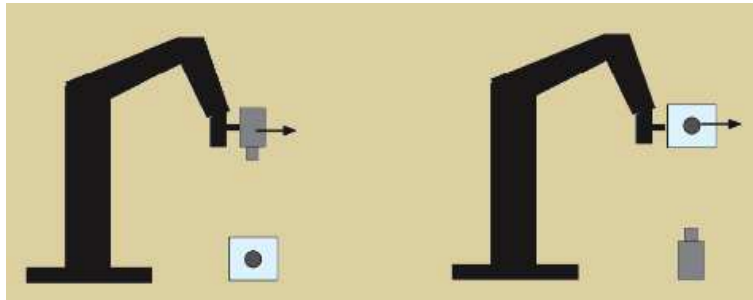
dove con  $m(t)$  si indica una misura stimata attraverso l'immagine, come ad esempio le coordinate di un punto di interesse, ed  $a$  rappresenta un insieme di parametri che caratterizzano il sistema di visione (come ad esempio i parametri intrinseci della telecamera o il modello 3D di un oggetto). Il vettore  $s^*$  contiene il valore desiderato delle features.

Gli schemi di asservimento visivo presenti nella letteratura robotica, si distinguono per:

- Posizione della telecamera

Una prima categoria fa riferimento al modo in cui viene usato il sistema di visione. Se la telecamera inquadra direttamente l'oggetto dal punto di vista del manipolatore, si parla di sistemi eye-in-hand. In questo caso la posizione e l'orientamento della telecamera sono solidali con l'end-effector del manipolatore. Un'altro tipo di configurazione consente di avere la telecamera che mantiene una posizione fissa nell'ambiente e al più con orientamento variabile, ed inquadra sia il manipolatore che l'oggetto target. Quest'ultima modalità viene denominata eye-to-hand. È possibile notare le differenze nelle capacità

visive del manipolatore in questa due modalità di configurazione dalla figura 2.1. Esistono anche strutture miste in cui si usano sia telecamere fissate sul manipolatore e sia telecamere posizionate nell'ambiente di lavoro.

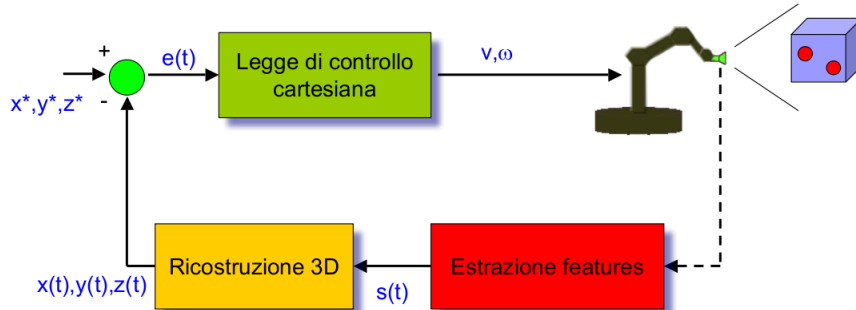


**Figura 2.1:** A sinistra un sistema eye-in-hand, a destra una configurazione eye-to-hand

- Tipo di dato fornito al controllore
  - Asservimento visivo basato sulla posizione o PBVS (Position Based Visual Servoing)

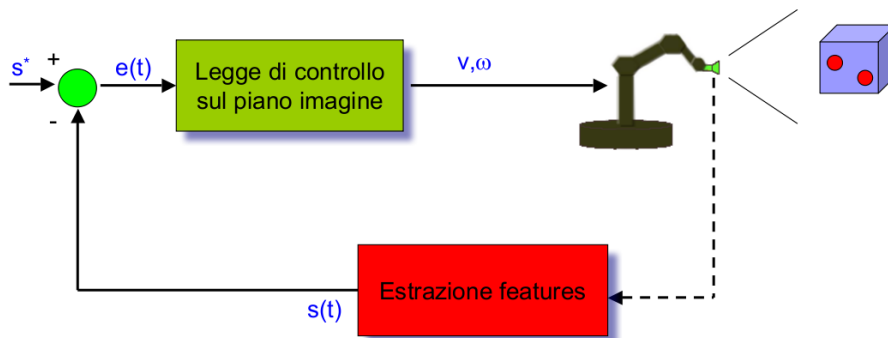
Questa tecnica è basata sulla stima della posizione dell'oggetto obiettivo della presa del manipolatore. Le features estratte dall'immagine sono usate, insieme ad un modello geometrico del target e al modello della telecamera, per stimare la posizione e l'orientamento di un oggetto rispetto alla telecamera. Lo schema è mostrato in figura 2.2. Il vettore  $s$  presente nella funzione di errore, di questo sistema di controllo basato sulla posizione e descritta dall'equazione 2.1, rappresenta proprio l'insieme di parametri  $3D$ .
  - Asservimento visivo basato sull'immagine o IBVS (Image Based Visual Servoing)

In questa modalità il controllo viene sintetizzato nel piano immagine e viene calcolato l'errore tra l'apparenza desiderata di alcune feature immagine e l'apparenza reale delle medesime features (punti, rette, etc.). L'obiettivo è generare un moto relativo telecamera-oggetto di interesse che riduca progressivamente l'errore. Nell'espressione della funzione



**Figura 2.2:** Position based visual servoing

errore che governa il controllo il vettore  $s$  indica il valore delle features che possono essere immediatamente estratte dall'immagine. In questa tecnica può inoltre essere effettuata una pianificazione nel piano immagine: programmando un'evoluzione graduale delle features desiderate verso la destinazione finale. Lo schema è mostrato in figura 2.3.



**Figura 2.3:** Image based visual servoing

- La struttura di controllo può essere gerarchica, ovvero il sistema di visione definisce i setpoint per gli anelli di controllo di posizione a basso livello, oppure il controllore servovisivo può calcolare direttamente i comandi per gli attuatori. Si distinguono due casi:

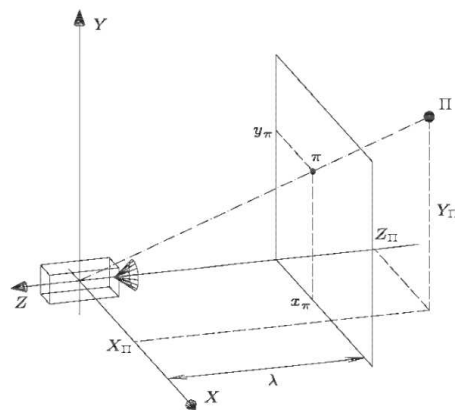
- Dynamic look and move: la visione fornisce i punti nel quale il manipolatore si deve portare, ed il controllore aziona i giunti in modo da assumere la configurazione necessaria per il raggiungimento di tali punti. Il robot viene visto dal sistema di controllo visivo come un posizionatore ideale nello spazio operativo. Questa tecnica è caratterizzata da bassi tempi di campionamento del segnale visivo, che non compromettono le prestazioni del controllo posizionale.
- Direct visual servo: il controllore è eliminato poiché il sistema visivo controlla direttamente i giunti.

Ai fini del controllo è necessario vi sia un legame tra le grandezze misurate nel piano immagine e quelle nello spazio operativo (postura dell'oggetto rispetto alla telecamera, o viceversa). Utilizzando come sistema di visione una telecamera, si ottiene una proiezione  $2D$  della scena inquadrata. Questa proiezione causa una perdita delle informazioni di profondità: ciascun punto nel piano immagine corrisponde ad un raggio nello spazio  $3D$ . La ricostruzione delle coordinate  $3D$  di un punto corrispondente ad un punto  $2D$  nel piano immagine, è un'operazione denominata "trasformazione prospettica inversa", che ammette più soluzioni essendo una funzione non biunivoca. Per ottenere un'unica soluzione occorrono opportune informazioni addizionali, che possono essere di tre tipi:

- viste multiple con una singola telecamera;
- telecamere multiple;
- conoscenza di relazioni geometriche tra punti caratteristici di un oggetto.

La stima di una posizione nel mondo  $3D$  è un problema strettamente legato alla calibrazione della telecamera e ha differenti soluzioni a causa del tipo di informazioni che vengono usate ed anche rispetto al metodo matematico di risoluzione impiegato. Infatti può essere determinata una soluzione analitica oppure ai minimi quadrati. Si può ricavare la stima della posizione con il metodo analitico se l'oggetto è identificato da almeno tre punti nell'immagine; mentre il secondo metodo si basa sulla definizione di una funzione di errore che deve essere minimizzata agendo sui parametri che caratterizzano la locazione dell'insieme dei punti rispetto alla telecamera.

In particolare un punto di coordinate  $(x, y, z)$  nel sistema di riferimento della telecamera viene proiettato nel punto del piano immagine. È possibile osservare, come viene anche mostrato nella figura 2.4, che il centro della lente della telecamera si trova a una distanza  $\lambda$  dall'immagine, e quando la camera è a fuoco  $\lambda$  coincide con la distanza focale della lente.



**Figura 2.4:** Modello della telecamera

Il braccio robotico, può essere rappresentato da una catena cinematica composta da corpi rigidi connessi tramite giunti. Un estremo della catena è vincolato ad una base, mentre all'estremo opposto è collocato l'organo di presa, su cui è fissata la telecamera. Il moto complessivo del manipolatore si realizza tramite, quindi, la composizione dei moti elementari di ciascun singolo braccio, e deve essere guidato dal sistema di visione. Se si definiscono alcune terne di riferimento per descrivere la posizione e l'orientamento del braccio robotico è possibile studiare quali relazioni intercorrono tra il sistema di riferimento del manipolatore e quello fissato dal sistema di visione. Considerando quindi le seguenti terne principali:

- $[B]$  base frame: la terna di riferimento del manipolatore (definita anche terna  $[0]$ );
- $[G]$  gripper frame: la terna fissata sulla pinza del manipolatore nel calcolo della cinematica diretta;
- $[M]$  marker frame: la terna obiettivo, posta sull'oggetto da prendere;

- $[C]$  camera frame: la terna definita dal sistema di visione associata alla microcamera.

E a questi riferimenti aggiungiamo una rototraslazione fissa  $[T]$  tool frame, che rappresenta una terna di riferimento orientata nello stesso modo del marker frame, ma posta sulla pinza del manipolatore.

La posizione di un punto di un corpo rigido rispetto ad una terna di riferimento è espressa dalla relazione:

$$p = p_x x + p_y y + p_z z \quad (2.2)$$

dove  $p_x, p_y, p_z$  rappresentano le componenti del vettore  $p$  lungo gli assi della terna  $A$ , descritto da:

$$P^A = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \quad (2.3)$$

L'orientazione del punto  $P$  rispetto alla terna si esprime attraverso una matrice di rotazione. Inoltre la matrice di rotazione consente di poter esprimere l'orientazione di un sistema di coordinate rispetto ad una terna di riferimento. Se quindi si conosce la descrizione di un punto  $P$  rispetto ad una terna di riferimento  $A$  e la rotazione tra le due terne  $R_A^B$  è possibile determinare  $P$  rispetto ad un'altra terna  $B$ , attraverso la seguente espressione:

$$P^B = R_A^B P^A + P_A^B \quad (2.4)$$

dove  $P_A^B$  è la descrizione dell'origine della terna  $A$  rispetto alla terna  $B$ . In forma compatta la trasformazione di coordinate fra le due terne, trasformazione intesa come operazione di traslazione e rotazione ed utilizzando per i vettori la loro rappresentazione omogenea ottenuta aggiungendo una quarta componente unitaria, si ricava la matrice di trasformazione:

$$T_B^A = \left( \begin{array}{ccc|c} & & & \\ & R_A^B & & P_B^A \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \quad (2.5)$$

In base alla descrizione di un punto rispetto a diversi sistemi di coordinate, è stato possibile determinare quali relazioni intercorrono tra le terne definite. La terna di tool frame è stata posizionata in modo da avere lo stesso orientamento del marker frame, ed è quindi ruotato di 180 gradi rispetto all'asse z del gripper frame. Il legame esistente tra la terna di gripper frame e quella di tool frame viene descritto attraverso la matrice  $T_T^G$ , rappresentata dall'equazione 2.6. Questa matrice ha permesso di definire il punto di presa dell'oggetto rispetto alla pinza. Un'altra roto-

$$T_T^G = \left( \begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos\pi & -\sin\pi & 0 \\ 0 & \sin\pi & \cos\pi & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \quad (2.6)$$

traslazione costante introdotta è rappresentata dalla matrice di trasformazione  $T_C^G$ . La relazione della posizione della telecamera rispetto alla pinza del manipolatore è infatti fissa. In una prima fase la stima della traslazione tra il camera frame e il tool frame è stata realizzata attraverso la misurazione delle distanze reali presenti, considerando che la telecamera è stata fissata direttamente sulla pinza, come descritto nel paragrafo 2.3. La rotazione tra il camera frame e il gripper frame è stata teoricamente valutata come uno scostamento di 180 gradi rispetto all'asse x. Il risultato ottenuto è mostrato nell'espressione (2.7).

$$T_C^G = \left( \begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos\pi & -\sin\pi & L_5 \\ 0 & \sin\pi & \cos\pi & L_6 \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \quad (2.7)$$

$L_5$	0.042
$L_6$	0.120

**Tabella 2.1:** Distanze tra i sistema di riferimento definiti da camera frame e tool frame, espresse in [m]

La determinazione della relazione tra il camera frame e il gripper frame effettuata in questo modo si è dimostrata, durante la fase sperimentale, non accurata, in quanto non rendeva possibile l'eliminazione di eventuali errori dovuti all'allineamento imperfetto delle terne per imprecisioni fisiche. Per ovviare a questi problemi

la valutazione di  $T_C^G$  è stata realizzata nel seguente modo:

$$T_C^G = T_M^C (T_M^G)^{-1} \quad (2.8)$$

dove  $T_M^C$  rappresenta la posizione ed orientamento dell'oggetto rispetto alla telecamera, ed è espresso attraverso l'espressione (2.9),  $T_M^G$  è stata determinata attraverso delle misure. In particolare la stima è avvenuta posizionando il gripper ad un'altezza nota dal centro del marker e considerando una rotazione di 180 gradi rispetto all'asse y della terna di marker.

$$T_M^C = \left( \begin{array}{ccc|c} & & & \\ & R_C^M & & P_C^M \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \quad (2.9)$$

Queste trasformazioni permettono di ottenere la posizione e l'orientamento della telecamera rispetto alla base del manipolatore, utilizzando anche la matrice  $T_G^B$ , che coincide con  $T_6^0$ , ottenuta dal calcolo della cinematica diretta descritta nel paragrafo 1.2.1 e rappresentata dalla matrice in (1.2). La matrice di trasferimento  $T_C^B$  descrive la telecamera rispetto al frame di base, come è anche mostrato nella figura 2.5.

$$T_C^B = T_G^B T_T^G T_C^T \quad (2.10)$$

La conoscenza di queste relazioni matriciali tra i diversi sistemi di riferimento, consente di valutare la posizione dell'oggetto nello spazio di lavoro del manipolatore, rispetto alla terna di base,  $T_M^B$ . I passaggi matematici per l'ottenimento di  $T_M^B$  sono visualizzabili grazie alla figura 2.6.

$$T_M^B = T_C^B T_M^C \quad (2.11)$$

Per generare la traiettoria che consente al manipolatore di portare il gripper nella posizione dell'oggetto e orientarlo in modo da effettuare la presa, devono essere

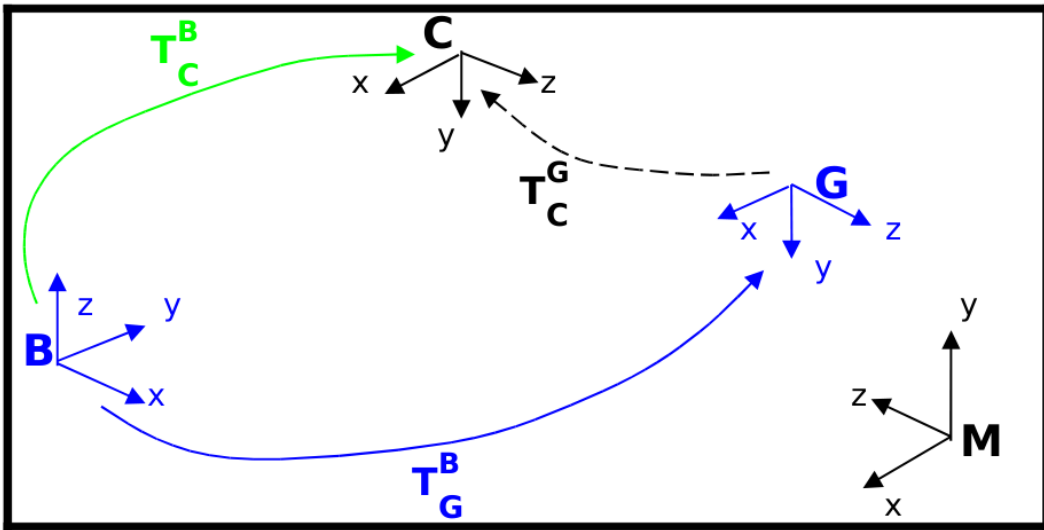


Figura 2.5: Sistema di riferimento per il calcolo di  $T_C^B$

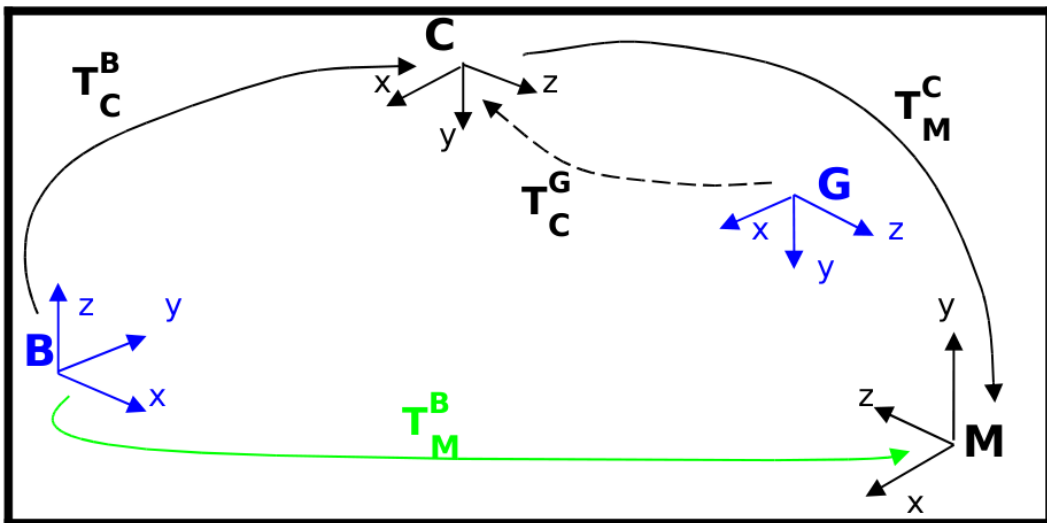


Figura 2.6: Sistema di riferimento per il calcolo di  $T_M^B$

calcolati dei punti di passaggio, che soddisfino la seguente relazione:

$$T_T^B(q_f) = T_M^B(q_i) \quad (2.12)$$

dove  $T_T^B(q_f)$  rappresenta la trasformazione che descrive la posizione e orientazione del tool frame, punto nel quale avverrà la presa, e  $q_f$  è il vettore delle variabili di giunto all'istante finale. Mentre  $T_M^B(q_i)$  viene calcolato attraverso la 2.1, e rappresenta la descrizione dell'oggetto rispetto alla terna di riferimento della base, valutato attraverso la cinematica diretta all'istante attuale  $T_G^B(q_i)$ , in cui  $q_i$  rappresenta il vettore delle variabili di giunto all'istante di tempo iniziale.

L'imposizione di questa condizione permette di trovare la descrizione dei punti in cui deve passare la pinza espressi rispetto alla terna di base ed usati nel calcolo della traiettoria. L'equazione 2.13 esprime come viene determinata la matrice di trasformazione  $T_G^B(q_f)$ , che contiene le coordinate della posizione che deve assumere il gripper rispetto alla base  $(x_G^B, y_G^B, z_G^B)$ , e la sua orientazione espressa attraverso gli angoli di roll, pitch e yaw  $(\alpha_G^B, \beta_G^B, \gamma_G^B)$  determinati attraverso la matrice di rotazione  $3 \times 3$   $R_G^B(\alpha_G^B, \beta_G^B, \gamma_G^B)$ .

$$T_G^B(q_f) = T_M^B(q_i)(T_T^G)^{-1} = \left( \begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & x_G^B \\ r_{21} & r_{22} & r_{23} & y_G^B \\ r_{31} & r_{32} & r_{33} & z_G^B \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \quad (2.13)$$

## 2.2 L'architettura generale del sistema e la comunicazione con il manipolatore

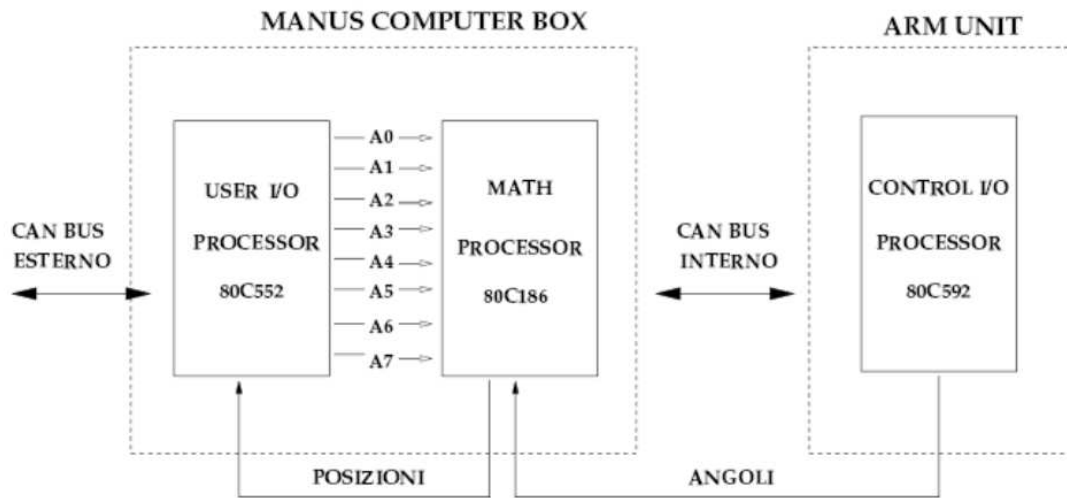
Il movimento del braccio robotico viene gestito attraverso un software con struttura modulare e flessibile, che consente di testare rapidamente algoritmi di controllo ed utilizzare il robot, ed in particolare è stato sviluppato sulla piattaforma YARA, descritto in dettaglio in appendice B. La struttura principale del software è rappresentata in figura 3.1, su cui sono state apportate delle modifiche per la realizzazione della versione di figura 3.2, in cui è stato inserito il sistema di visione. La presentazione in dettaglio del software viene effettuata nel capitolo 3.

La comunicazione tra il manipolatore antropomorfo e il PC è possibile grazie ad un'interfaccia elettronica, che consente di accettare comandi impartiti con il joystick, o quelli provenienti da un dispositivo esterno ed è in grado di generare i valori

delle velocità da inviare ai motori elettrici del manipolatore, sulla base dei valori di velocità di giunto ricevuti in ingresso. Lo stato di funzionamento del sistema e l'eventuale presenza di errori può essere segnalata all'utente tramite un display grafico. La comunicazione tra il manipolatore e un dispositivo esterno, come mostra la figura 2.7, viene gestita attraverso CAN bus, ed in particolare dal processore 80C522. I dati che questo processore riceve contengono le informazioni per muovere il manipolatore e i comandi per eseguire le procedure automatiche di apertura e chiusura, dette di Fold In e Fold Out. La lettura dei dati sul CAN bus avviene periodicamente ad intervalli di  $20ms$ . Le modalità di comando possono essere di due tipi:

- Joint Mode: vengono controllate le singole velocità di ogni giunto, e devono essere espresse in gradi al secondo. In modo da determinare di quanto debba essere ruotato ciascun giunto nei successivi  $20ms$ .
- Cartesian Mode: vengono controllate la posizione e l'orientazione del gripper lungo i tre assi cartesiani. I dati specificati devono indicare di quanti millimetri debba essere mosso il gripper lungo i tre assi cartesiani nei successivi  $20ms$  e di quanti gradi debba essere ruotato il polso attorno ai tre assi di rotazione. In questo caso il processore matematico calcola la cinematica inversa dei valori ricevuti, e controlla il verificarsi di possibili autocollisioni.

In entrambe le modalità di funzionamento lo spostamento da compiere nei successivi  $20ms$  viene realizzato a velocità costante da parte del controllore interno. I valori ricevuti vengono usati, insieme a quelli sugli angoli dei giunti provenienti dal manipolatore, per calcolare ogni  $10ms$  tramite un controllo di tipo proporzionale-integrativo (PI) il valore delle coppie da applicare ai motori. Quest'ultimi vengono infine inviati tramite un secondo CAN bus (*Internal Can Bus*) al processore 80C592, che si occupa della loro trasmissione agli attuatori. Nella modalità trasparente di utilizzo del robot l'interfaccia tra la Control Box e il computer viene realizzata attraverso una scheda per rete CAN basata su slot PCI, mostrata in figura 2.8. Una caratteristica importante di questa scheda è la possibilità di funzionamento con il sistema operativo Linux mediante i driver forniti dal produttore. In particolare viene usato l'integrato SJA1000[3] prodotto da Philips Semiconductors largamente utilizzato in tutti i dispositivi di questo tipo, mentre l'interfacciamento con il

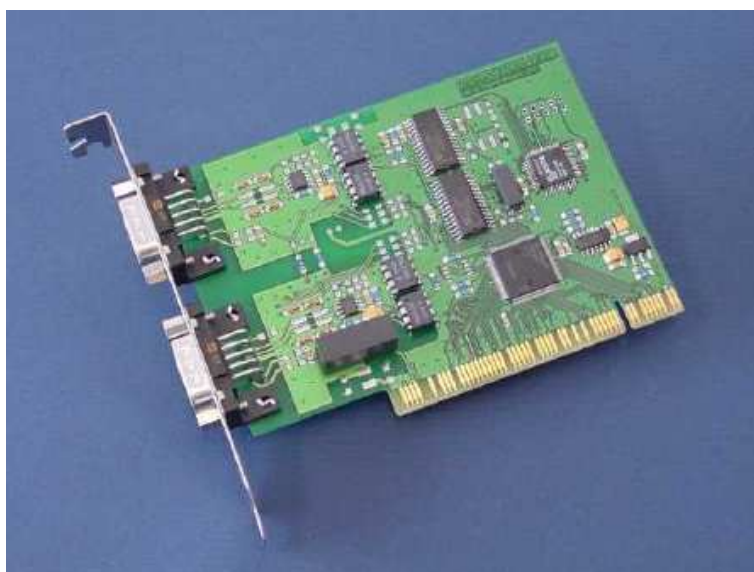


**Figura 2.7:** Schema dell'elettronica di controllo

bus PCI viene invece realizzato tramite il PCIController PCI9052 prodotto da PLX Technology[4]. La comunicazione CAN con la CPC-PCI può avvenire in due modi: “interrupt controlled mode” o “polled mode”. Inoltre come opzione la CPC-PCI consente di avere una separazione galvanica tra PC e CAN bus. Le caratteristiche tecniche sono elencate nella tabella 2.2.

Pin assignment	ConnectorDSub9 (CiA Ds-102)
Type of Phy. connection	ISO 11898, Transceiver PCA82C251
Maximum voltage on bus pins	$\pm 30V$ referring to bus ground
Isolation voltage with galvanic separation	$\pm 1000V$

**Tabella 2.2:** Caratteristiche tecniche della scheda CPC-PCI.



**Figura 2.8:** Scheda CPC-PCI

## 2.3 Le specifiche hardware del sistema di visione

La configurazione eye-in-hand, presentata nel paragrafo 2.1, è stata realizzata attraverso l'installazione, direttamente sulla superficie piana del gripper del manipolatore, della telecamera CCD a colori, mostrata in figura 2.9. La telecamera ha un sensore Sharp 1/4' dotata delle caratteristiche elencate in tabella 2.3. Il sensore CCD, charge coupled device, è rappresentato da pixel di materiale semiconduttore che accumulano carica libera se colpiti da fotoni (effetto fotoelettrico), e vengono letti in un processo sequenziale di read-out, che trasforma carica in tensione.



**Figura 2.9:** Robot Manus e telecamera CCD

La telecamera, in una prima fase era stata collegata ad un supporto metallico a

Risoluzione	420 linee (NTSC 510(H)x492(V) pixel PAL:(500(H)x582(V) pixel)
Uscita video	1.0Vp-p 75Ω
Temperatura di lavoro	-10° C + 50° C RH 95%
Sensibilità	1,5 Lux (F:2.0)
Alimentazione	12V DC
Dimensioni	32x32mm
Bilanciamento automatico	

**Tabella 2.3:** Caratteristiche telecamera Sharp CCD

“L“ con del velcro, che è installato sulla pinza del Manus con del velcro. In questo modo era possibile rimuoverla con estrema semplicità, per utilizzarla solo quando necessario, e lasciare libero il manipolatore per altre applicazioni. Il collegamento con il velcro comportava però oltre che la non rigidità dell’ancoraggio del supporto, anche l’introduzione di errori di orientazione della telecamera. Questo tipo di installazione è stata ritenuta non conveniente. La telecamera è stata successivamente fissata sulla pinza attraverso un nastro biadesivo.

L’interfacciamento della telecamera con il PC viene effettuata mediante la scheda frame grabber Terratec Cinergy 250 PCI [5], che consente l’acquisizione audio-video. È infatti dotato di quattro porte di ingresso che gestiscono i seguenti tipi di segnale:

1. Immagine televisiva tramite antenna
2. Immagini televideo ad alta velocità
3. Immagini provenienti da sorgenti e/o apparecchi esterni
4. Segnali audio esterni
5. Segnali radio FM

In particolare questo tipo di frame grabber è basato sul chipset Philips Semiconductor della famiglia saa713x ( in questo caso saa7133), che dell’acquisizione e dell’elaborazione dei vari tipi di segnali, e per questo modello di chipset esiste un

## *Visual servoing*

---

driver implementato direttamente nel kernel di Linux. Per reperire le informazioni necessarie al sistema di asservimento visivo è stato usato il solo ingresso video composito, che fornisce segnale di tipo S-Video.

## Capitolo 3

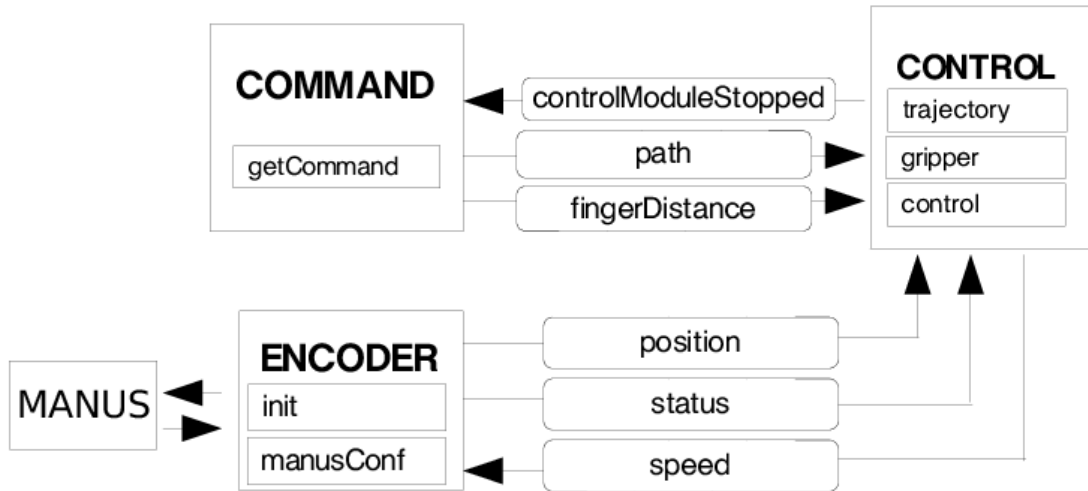
# Progetto software per il visual servoing

In questa sezione viene descritta la realizzazione delle modifiche apportate al software esistente, oltre che una presentazione dettagliata della libreria di visione utilizzata.

### 3.1 Il software per il sistema di controllo

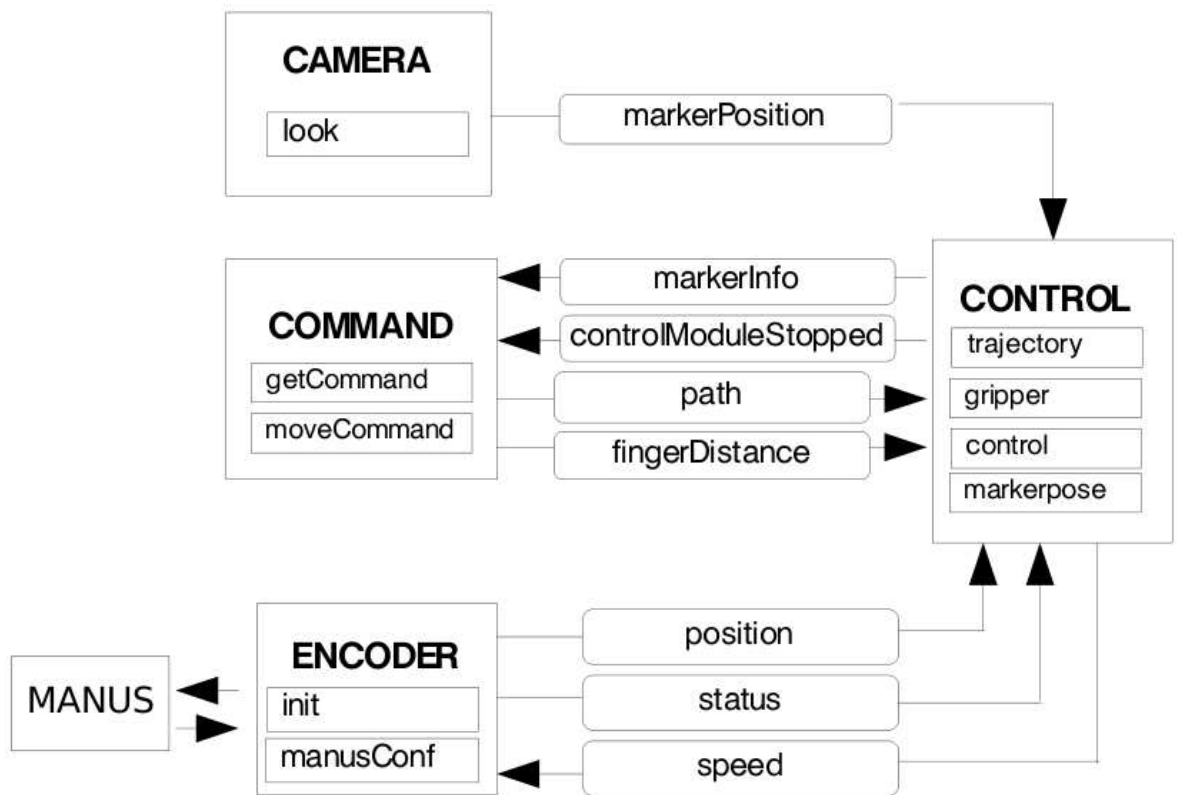
L'architettura dell'intero sistema di controllo è composta da diversi moduli che sono definiti mediante la piattaforma Yara. Il framework Yara permette di definire degli elementi attivi ed autonomi, denominati moduli, che possono svolgere delle attività indipendenti e cooperare tra loro scambiando informazioni e comandi. Le funzionalità che realizza ogni modulo sono indicate come Activity del modulo, e possono essere risvegliate periodicamente o scatenate da un evento esterno. Nella versione originale del software, mostrata dallo schema in figura 3.1, vi erano tre moduli *Encoder*, *Command* e *Control*, che complessivamente utilizzano sei Activity per la realizzazione del controllo e ad altrettanti pattern *Information* per le comunicazioni.

Il modulo *Encoder* è il componente del sistema preposto alla comunicazione tra l'architettura software ed il manipolatore, tramite l'interfaccia hardware costituita dall'elettronica di controllo del Manus e dal controller CAN posto sul calcolatore. In



**Figura 3.1:** Architettura del sistema di controllo del Manus mediante YARA

questo modulo ci sono due Activity: *init* e *manusConf*. La prima, denominata *init*, viene attivata come Activity periodica all'avvio del sistema e il suo ruolo è quello di svolgere un'azione di inizializzazione sui giunti del manipolatore. L'inizializzazione è stata introdotta per ovviare ad una inconsistenza nei valori degli ultimi tre giunti che si manifesta durante i primi movimenti immediatamente successivi all'accensione del manipolatore; pertanto viene fatto compiere un movimento impercettibile ai tre giunti interessati prima dell'inizio del ciclo di controllo vero e proprio. L'Activity *init*, rispettando le tempistiche di comunicazione del manipolatore, invia per quattro volte consecutive a distanza di 60 ms un comando di movimento ai tre giunti, invertendo ogni volta il verso del moto di rotazione in modo da annullarne l'effetto complessivo. L'attivazione periodica di *init* viene successivamente sospesa, dopo aver attivato l'Activity *manusConf*. Quest'ultima è un'Activity periodica con periodo di 10 ms, che viene avviata al termine della fase di inizializzazione del manipolatore. Il compito di *manusConf* è quello di leggere dal CAN bus le informazioni sulla configurazione attuale del manipolatore (stato del sistema e valori delle variabili di giunto), rendere disponibili queste informazioni al modulo che si



**Figura 3.2:** Architettura del sistema di controllo del Manus basato sulla visione mediante YARA

occupa del controllo, ed inviare al manipolatore sempre tramite Can bus le velocità dei giunti che si vogliono impostare.

Il secondo modulo, denominato *Command*, ha il compito di inoltrare al modulo *Control* le indicazioni sul task da eseguire, fornendogli contemporaneamente tutti i dati necessari per la realizzazione del movimento richiesto. La prima versione del sistema, proposta nello schema 3.1, prevedeva l'esecuzione di uno dei seguenti task:

- il controllo del moto del manipolatore lungo un percorso assegnato,
- la possibilità di aprire e chiudere il gripper.

Nel software di questa prima versione, era previsto che il costruttore del modulo *Command* memorizzava in due vettori distinti le sequenze dei via point dei percorsi da interpolare e le misure di apertura del gripper che dovevano essere tenute

durante le diverse fasi del task. Queste informazioni erano reperite dagli argomenti di riga di comando specificati al momento dell'esecuzione del programma. In particolare per i via point era indicato il nome dei file, in cui vengono specificate le coordinate dei punti attraverso i quali deve passare il manipolatore per compiere la traiettoria desiderata, mentre le informazioni relative all'apertura della pinza erano passate invece direttamente da riga di comando tramite valori numerici, che rappresentano la distanza in mm tra le due dita che formano il gripper. I due tipi di argomenti in ingresso dovevano essere alternati fra di loro, ed essere disposti secondo l'esatta sequenza temporale richiesta. Il modulo *Command* era dotato di una sola Activity aperiodica: *getCommand*. Il suo funzionamento era quindi riconducibile a quello di una macchina a stati, con due soli stati che commutano ad ogni attivazione. Ad esempio, se si considera la realizzazione di un task di esecuzione di movimenti del manipolatore alternata ad azioni di controllo del gripper, il primo stato di funzionamento *getCommand* si occupava di inviare al modulo *Control* i punti del percorso da interpolare. Il passaggio dei via points fra i due moduli veniva effettuato tramite l'aggiornamento (*update()*) dell'oggetto Sender *path* da parte di *getCommand*. A tale operazione seguiva l'attivazione dell'Activity *trajectory*, e la successiva sospensione di *getCommand*. Una nuova attivazione di *getCommand* era prevista in relazione ad un possibile aggiornamento dell'oggetto Receiver *controlModuleStopped*, che segnalava la conclusione dell'attività di controllo in corso e la possibilità da parte del controllore di iniziare un nuovo task di controllo. Alla sua riattivazione *getCommand* era nel secondo stato di funzionamento, in cui era prevista la procedura di apertura e chiusura della pinza. Questa modalità di funzionamento schematizzata anche attraverso le figure 3.3 e 3.4, è stata mantenuta, in quanto consente di comandare il braccio robotico attraverso una traiettoria stabilita dall'utente.

Per soddisfare le richieste di un task di asservimento visivo, al modulo *Command* sono state apportate delle modifiche per ampliare le sue funzionalità, e consentirgli di utilizzare, ad esempio, dei via point definiti in base alla posizione dell'oggetto, valutata in real-time dalle routines del sottosistema di visione. In particolare è stata aggiunta una modalità di funzionamento alternativa. Attraverso la definizione di un oggetto *maker* di tipo Receiver a cui è associata un'Activity, denominata *move*, viene gestita la ricezione dei via point aggiornati dal modulo *Control* e

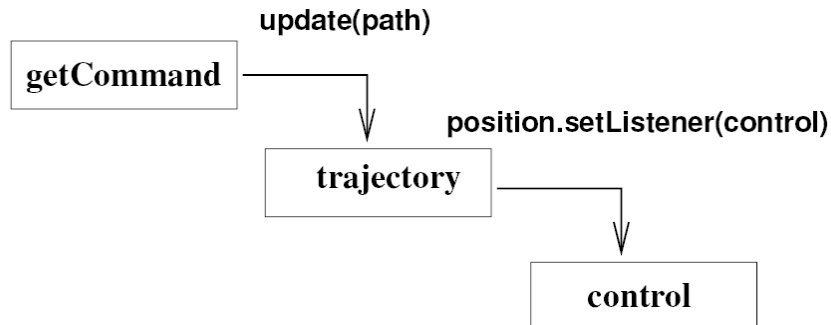


Figura 3.3: Sequenza di attivazione di *trajectory*

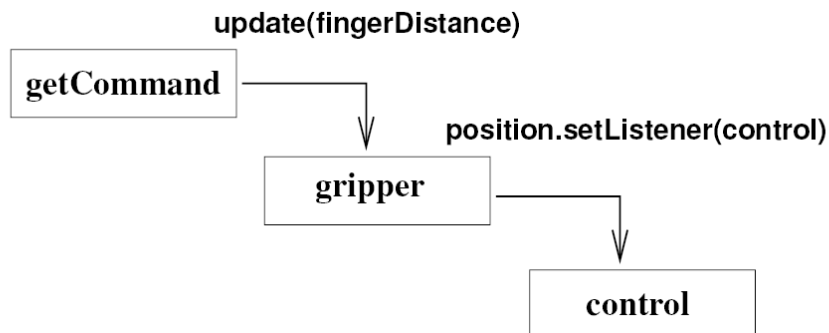


Figura 3.4: Sequenza di attivazione di *gripper*

memorizzati in un'opportuno file condiviso tra i moduli. Il modulo *Command* si occupa di richiedere i comandi da eseguire, in ordine temporale, al modulo *Control* per effettuare la presa dell'oggetto. In principio si avrà l'apertura del gripper, seguito dall'esecuzione del moto che porterà il Manus alla posizione finale con una postura adeguata per la chiusura del gripper e quindi la presa. Il controllo del moto avverrà come nella modalità di funzionamento precedente, ossia i via point saranno memorizzati nell'oggetto *Sender path* e l'Activity *trajectory* del modulo *Control* si occuperà della generazione della traiettoria. Nello schema attuale riportato in figura 3.2, si possono vedere le modifiche effettuate, e l'introduzione di uno scambio di informazioni tra i moduli *Command* e *Control*, necessario per il calcolo dei valori dei via point nello spazio operativo, espressi attraverso la definizione indicata nella tabella 1.4.

Il modulo *Control* rappresenta il componente centrale dell'architettura di gestione del moto del Manus. La prima versione del sistema, proposta nello schema 3.1 prevedeva tre *Activity*: *trajectory*, *control* e *gripper*. La prima implementava le funzioni per la generazione delle traiettorie che l'organo terminale deve eseguire, la seconda gestiva il controllo dell'esecuzione e la terza si occupava dell'apertura e chiusura del gripper. Era prevista una comunicazione con i moduli *Command* ed *Encoder*, attraverso i sei pattern *Information*: quattro di tipo *Receiver* (*position*, *status*, *path*, *fingerDistance*) e due di tipo *Sender* (*speed* e *controlModuleStopped*). Questi oggetti oltre a garantire uno scambio continuo di dati, consentivano anche una sincronizzazione tra i moduli, dal momento che le attivazioni e le sospensioni di alcune *Activity* sono legate alle operazioni di *update()* sui pattern *Information* stessi.

L'*Activity trajectory* genera una traiettoria a partire dai punti di passaggio specificati nel file di via points; tale traiettoria generata costituirà successivamente il riferimento spazio-temporale del controllore del moto del manipolatore. Al termine dell'interpolazione, *trajectory* prima di essere sospesa dallo scheduler del framework in attesa di una nuova attivazione, aggancia l'esecuzione dell'*Activity control* alla ricezione dei dati contenuti nell'oggetto *position* (fig. 3.3). In questo modo, al successivo aggiornamento di *position* da parte dell'*Activity manusConf* il controllore viene attivato, ed ha inizio la generazione del moto.

L'*Activity gripper* ha il compito di inoltrare al controllore un comando di apertura o chiusura della pinza del gripper. In particolare la sua attivazione è legata all'operazione di *update()* sull'oggetto *fingerDistance*, realizzato dall'*Activity getCommand*. L'*Information* pattern *fingerDistance* contiene deve indicare il valore in mm della distanza a cui devono essere portate le estremità delle due dita che formano il gripper. L'*Activity* recupera il valore memorizzato in *fingerDistance*, rendendolo disponibile al controllore, e poi aggancia l'attivazione dell'attività di controllo all'evento di aggiornamento dell'oggetto *Receiver position*, in modo da garantire l'inizio della procedura di azionamento del gripper nei successivi 60 ms.

L'*Activity* principale di questo module è *control*. L'attivazione di *control* è legata alla ricezione di un nuovo set di valori delle variabili di giunto nel pattern *Information position*. Il vincolo tra aggiornamento di *position* ed attivazione di *control* non è permanente, si instaura al termine dell'attività delle *Activity trajectory* o

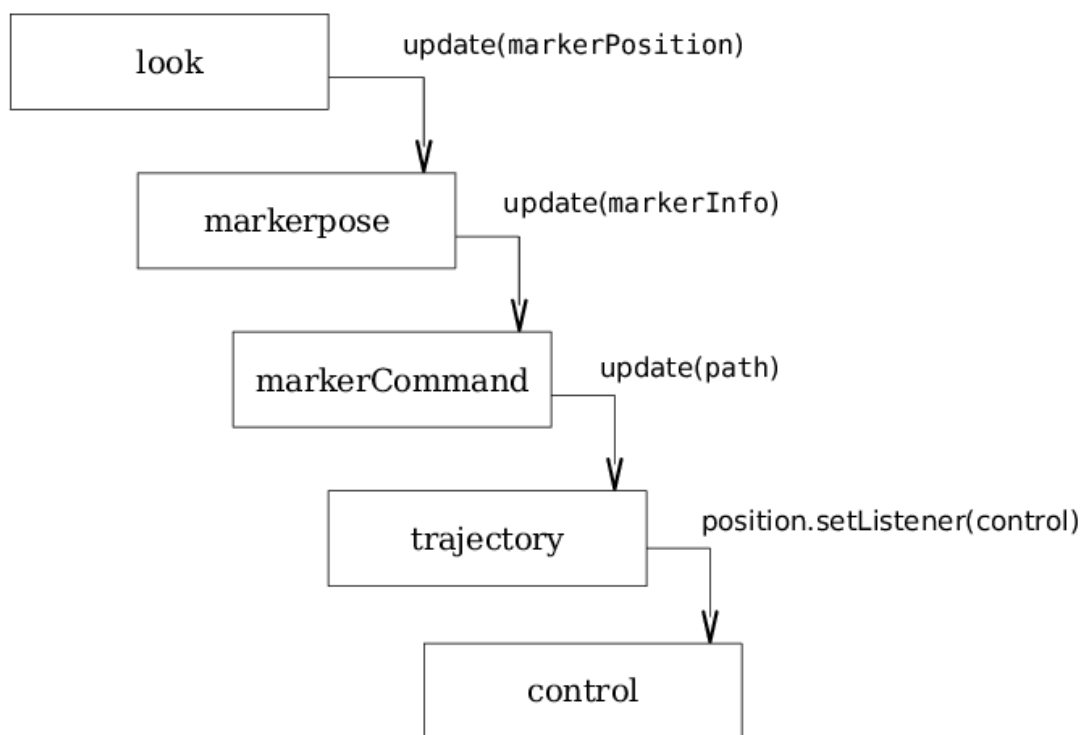
*gripper* che provvedono entrambe a registrare *control* come oggetto Listener di *position*, e termina con la fine dell'azione di controllo. Il ciclo di controllo inizia con la lettura dei due campi memorizzati nell'oggetto *status*, che contengono i codici indicanti lo stato di funzionamento del Manus. Il contenuto di *status* viene analizzato e, in caso di errore, il manipolatore viene fermato altrimenti *control* procede scegliendo una fra le due possibili azioni di controllo:

- Controllo del moto del manipolatore nello spazio dei giunti.
- Apertura e chiusura del gripper.

Il metodo *controlLoop()* implementa le funzionalità dell'Activity *control*, ed in base al valore di un flag, viene impostato il controllo del moto o del gripper. La presenza di una sola Activity di controllo consente di sovrapporre temporalmente il movimento del manipolatore con la chiusura/apertura del gripper, semplicemente eseguendo entrambi i cicli di controllo in modo sequenziale ad ogni attivazione e poi inviando simultaneamente al manipolatore i sei byte con le velocità delle variabili di giunto ed il byte con l'informazione sul gripper.

Per realizzare l'architettura per il visual servoing in questo modulo è stato inserito un pattern Receiver *markerPosition* e *markerCommand* di tipo Sender. Il primo serve per stabilire un canale di comunicazione tra il modulo *Control* e il modulo *Camera*, in modo che alla ricezione delle coordinate *3D* dell'oggetto venga calcolato attraverso la cinematica diretta la posizione dell'oggetto rispetto alla base del robot, utilizzando le espressioni descritte nel capitolo 2. L'Activity *markerpose* viene eseguita proprio alla ricezione di un'aggiornamento del pattern *markerPosition*, e consente quindi di generare il file di via point. Le interazioni tra le Activity dei moduli *Camera*, *Control*, e *Command*, necessarie per la generazione della traiettoria in base alla lettura real-time dei dati, sono riassunte nello schema di figura 3.5.

Il modulo *Camera* rappresenta l'interfaccia tra l'architettura del sistema di controllo e la telecamera, ed è definito nel listato 3.1. La telecamera è gestita attraverso la libreria ARTToolkit [6], e per questo è stata creata una classe *Vision*, che ne richiama le funzionalità. In particolare il costruttore del modulo *Camera* riceve dal programma principale un puntatore alla classe *Vision*, in questo modo all'interno del modulo vengono richiamate le funzioni di ARTToolkit utilizzando tale puntatore.



**Figura 3.5:** Sequenza di comando della traiettoria attraverso l'Activity *look*

Il modulo *Camera* richiama l'inizializzazione della configurazione della telecamera, il caricamento della matrice di calibrazione e del file contenente la descrizione del marker. Questo modulo, presente nello schema di figura 3.2, ha una sola Activity denominata *look*, che consente di fare l'acquisizione dell'immagine e la detection del marker a cadenza periodica. Nel caso di identificazione del marker sull'immagine viene aggiornato il Sender pattern *markerPosition*, che risveglia un'Activity del modulo *Control*.

```
class CameraModule : public Module
{
  private :
    char          *model_name ;
    ObjectData_T  *object ;
    int          objectnum , selected_marker ;
    char          *patt_name , *cparam_name , *vconf  ;
    Vision        *appview ;
    ARParam       cparam ;
    int          xsize , ysize ;
    vector<double> point ;

    void start () ;
    void stop () ;

    static void cleanup(void) ;
    int  setupCamera(void) ;
    static int  setupMarker(void) ;
    void  initCamera () ;
    void  search(void) ;
    void  argInitLoop(void) ;

public :
    CameraModule( Unit& u, Vision *view ) ;
    Activity< void , void > look ;
    Sender < vector<double> > px ;
```

```
};
```

**Listato 3.1:** Il modulo Camera

## 3.2 Le librerie software

In questa sezione vengono introdotte le librerie utilizzate per la realizzazione del programma che governa i vari componenti del sistema. Per la realizzazione di un sistema visual servoing position based è necessario che la visione fornisca la posizione relativa tra telecamera ed oggetto. Questo compito è assolto dalla libreria ARToolkit, basata in parte su OpenGL, che mediante una conoscenza a priori dell'oggetto permette il calcolo sia della posizione che dell'orientazione. Per la creazione e la manipolazione di strutture matematiche anche complesse è stata usata la libreria Newmat, che consente di eseguire su matrici gran parte delle operazioni dell'algebra lineare ed è impiegata per l'implementazione delle funzioni di cinematica del manipolatore.

### 3.2.1 ARToolkit

ARToolKit è una libreria open source per la visione che permette lo sviluppo di una larga varietà di applicazioni di Augmented Reality. I promotori di questa libreria sono il Prof. Hirokazu Kato e Dr. Mark Billinghurst [6]. ARToolKit usa OpenGL per fare il rendering, GLUT come gestore degli eventi relativi alla configurazione della finestra per la visualizzazione delle immagini, e una libreria specifica per l'hardware video. Le specifiche API di ARToolkit sono sviluppate in C. Le caratteristiche principali di questa libreria possono essere sintetizzate nei seguenti punti:

- supporto multiplatforma (Windows, Linux, MacOS) e fornisce un tutorial di installazione per ogni sistema operativo;
- richiede dei requisiti hardware minimi, indicati nella tabella 3.1. Possono essere usati diversi tipi di telecamera, e schede di acquisizione. Inoltre può supportare Head Mounted Display, dispositivo tipico dei sistemi di realtà aumentata, che consente all'utente di visualizzare testo e grafica in sovrapposizione;

- rende semplice la calibrazione della telecamera;
- fornisce delle funzioni per definizione di marker personalizzati;
- fornisce un tutorial per lo sviluppo di applicazioni specifiche;
- consente lo sviluppo di applicazioni real-time.

#### Prerequisiti Hardware

Telecamera	USB 1.1, IEEE1394a, PCI32Bit1.0, PCI64Bit2.0, PCI32Bit2.1, PCI64Bit2.1
Video capture card	PAL/NTSC
Head Mounted Display (HMD)	SAAB Advisor 150, MicroOptical SV Viewer

#### Prerequisiti Software

OpenGL e GLUT	Versione RunTimes (opengl.so, glu.so, e glut.so), versione SDK (opengl.h, glu.h and glut.h)
Video Library	Su Linux: V4L RunTime e SDK o dv1394 e dv1394 RunTime and SDK.
OpenVRML	(opzionale) per fare VRML rendering.

**Tabella 3.1:** Requisiti di installazione

L'aspetto probabilmente più interessante di una telecamera è quello di fare delle misurazioni in seguito ad opportune valutazioni dell'oggetto inquadrato. Le numerose applicazioni di visione finalizzate all'estrazione di informazioni geometriche dagli oggetti presenti nel campo di visione della telecamera iniziano con la fase di calibrazione del sensore visivo.

La calibrazione consiste nel determinare tutti i parametri di orientamento interni alla telecamera stessa. I parametri intrinseci sono ovviamente legati al tipo di dispositivo usato e rappresentano le lunghezze focali. Il passo successivo alla calibrazione, consiste nel determinare i parametri esterni od estrinseci alla telecamera, che determinano la posizione ed l'orientamento della telecamera rispetto all'oggetto di riferimento. L'obiettivo è la valutazione della matrice di calibrazione espressa nel modo seguente:

dove  $x_c$  e  $y_c$  rappresentano le coordinate del centro dell'immagine,  $s$  è spesso zero e dipende dal metodo di calibrazione utilizzato, mentre i parametri estrinseci della

$$C = \begin{pmatrix} sf_x & s & x_c & 0 \\ 0 & sf_y & y_c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.1)$$

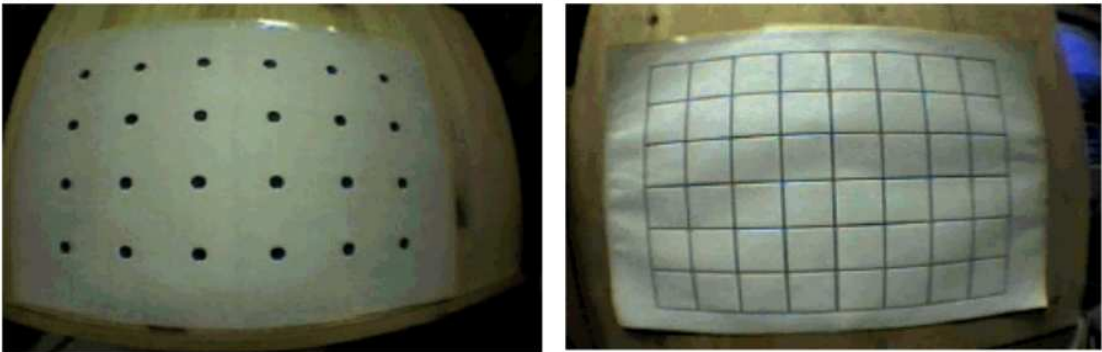
telecamera sono rappresentati da  $sf_x$  e  $sf_y$ . Conoscendo le coordinate in pixel sui relativi piani dell'immagine e la lunghezza focale  $\lambda$  della telecamera è possibile conoscere la profondità del punto nel mondo  $3D$ . ARToolkit fornisce un processo di calibrazione, che permette di generare un file di parametri personalizzato per la telecamera in uso, denominato `camera_para.dat`. È possibile scegliere tra due tipi di metodi:

1. Calibrazione in due passi. Più difficoltoso da utilizzare ma dovrebbe raggiungere una migliore accuratezza nei risultati.
2. Calibrazione in un passo. Facile da usare e consente di avere un'accuratezza accettabile ma non precisa per misure  $3D$ .

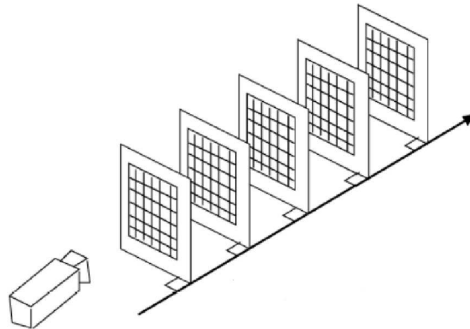
In entrambi i casi viene utilizzato pattern di  $6 \times 4$  punti, mostrato in figura 3.6, in cui ogni punto dista 40mm. Attraverso l'esecuzione del programma `calib_dist`, fornito dalla libreria, viene determinato il punto centrale dell'immagine e la distorsione delle lenti. Nel caso di calibrazione in due passi, viene eseguita anche un'altra utility `calib_param`, che stima la distanza focale della telecamera, facendo uso del pattern riportato in figura 3.6, rappresentante una griglia di righe. In questa fase la griglia viene posta perpendicolare alla telecamera ad una distanza di 100mm, e sull'immagine della griglia acquisita dalla telecamera vengono disegnate le righe orizzontali e verticali in modo che siano il più possibile allineate con quelle visualizzate. Il procedimento viene ripetuto almeno cinque volte, spostando la griglia lontano dalla telecamera di 100mm ogni volta, come mostrato dalla figura 3.7. .

Questi processi richiedono un'elevata quantità di lavoro manuale, in quanto è l'utente che deve indicare il centro dei punti del pattern attraverso il puntatore del mouse, o trovare una sovrapposizione adeguata delle righe della griglia, e ciò causa la non accuratezza dei parametri di calibrazione individuati. Il risultato ottenuto è, per questi motivi, abbastanza approssimativo.

ARToolkit permette di effettuare il riconoscimento di diversi pattern già installati, mostrati in figura 3.8, e anche di multi-pattern, come quelli di figura 3.9, ol-



**Figura 3.6:** Pattern per la calibrazione

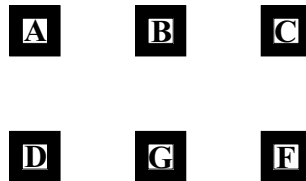


**Figura 3.7:** Modalità di determinazione della matrice di proiezione prospettica

tre che di crearne uno personalizzato. Il programma per creare un nuovo template è chiamato `mk_patt`, ed è contenuto nella libreria. Il nuovo pattern deve essere contenuto in un quadrato nero, con al centro un'altro quadrato bianco più piccolo. L'esecuzione del programma identifica il quadrato più grande attraverso una linea colorata, sia quando il pattern è posto perpendicolarmente davanti alla telecamera, sia cambiando orientamento. Una volta memorizzato il file `patt.yourpatt` dell'immagine del pattern creato, è quindi possibile utilizzarlo nelle applicazioni di ARToolkit nella fase di pattern matching. In una singola applicazione è possibile effettuare il riconoscimento contemporaneo di diversi tipi di pattern, associandoli a differenti oggetti 3D, e visualizzare forme diverse, come coni, cubi o sfere, per ogni marker riconosciuto. In questo caso deve essere creato un file che contenga



**Figura 3.8:** Patter Hiro e Pattern Kanji



**Figura 3.9:** Multi-pattern

l'elenco dei marker e specificando per ognuno: nome, nome del file di riconoscimento del pattern (`patt.yourpatt`), larghezza del marker in millimetri, e coordinate del centro del marker.

Un'applicazione basata sulla libreria ARToolkit si suddivide principalmente in tre parti:

1. Inizializzazione: fase in cui avviene il caricamento delle caratteristiche del video, dei file di riconoscimento dei marker e dei parametri telecamera.
2. MainLoop: comprende l'insieme di operazioni che sono eseguite periodicamente, come la cattura immagine, il rilevamento e l'identificazione del marker, il calcolo della posizione e orientamento del marker rispetto alla telecamera e il disegno dell'oggetto virtuale ( in caso di progetti di realtà aumentata).
3. Chiusura: fase finale di interruzione del flusso video.

Possono essere gestiti anche gli eventi associati al mouse e alla tastiera. Una descrizione più schematica dell'esecuzione di una generica applicazione è mostrata in figura 3.10, in cui si può notare che la funzione di detection dei marker nell'immagine catturata individua, se esiste, una forma corrispondente al "fiducial marker"

configurato. Se nessun marker viene rilevato il programma continua con l'acquisizione dell'immagine successiva. Se invece c'è una detection del marker, allora è possibile attraverso un'opportuna funzione, disegnare sullo schermo una forma geometrica o visualizzare un'immagine che indica l'avvenuto ritrovamento.

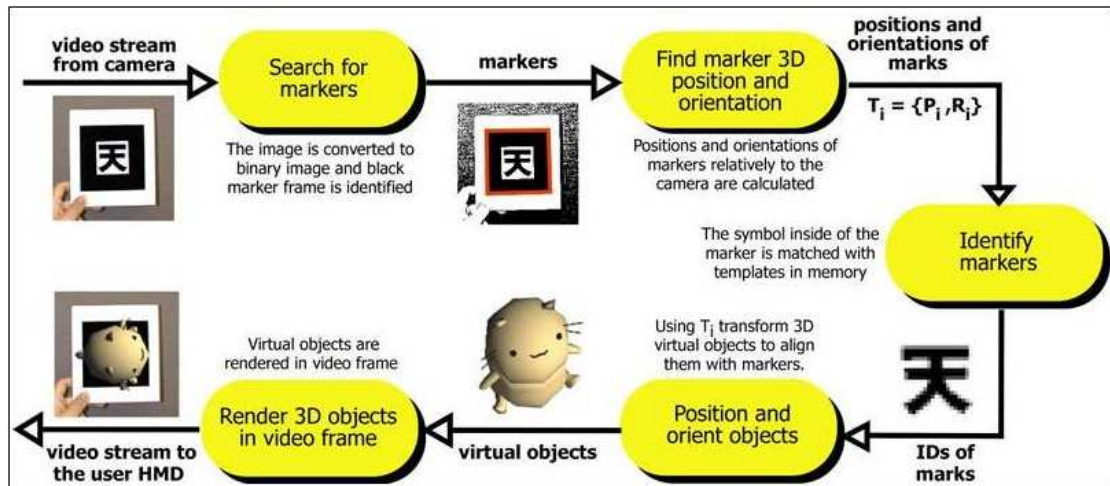


Figura 3.10: Schema di funzionamento di un'applicazione di ARToolkit

Nella libreria è presente una funzione chiamata `arGetTransMatCont()`, che permette di determinare una matrice di rototraslazione ed indica la posizione e l'orientamento del marker, rispetto al sistema di riferimento della telecamera. Rappresenta quindi la matrice  $T_M^C$  indicata dall'equazione 2.1.

Le applicazioni di visione sono soggette ad alcune limitazioni, ad esempio un'osservazione ovvia è che il rilevamento del marker può avvenire soltanto se esso appartiene al campo di vista della telecamera. Un'eventuale copertura del marker dovuta alla presenza di occlusioni, potrebbe anche causare un fallimento dell'operazione di rilevamento. Altri problemi possono essere dovuti alle dimensioni fisiche del pattern, e come mostra la tabella 3.2, le funzionalità di ARToolkit consentono di riconoscere dei pattern a una ben determinata distanza in relazione alla loro grandezza.

<i>Dimensione(cm)</i>	<i>Distanza(cm)</i>
6.98	40.60
8.89	63.50
10.79	86.36
18.72	127.00

**Tabella 3.2:** Campo di visibilità di ARToolkit

### **3.2.2 L'integrazione di un'applicazione ARToolkit nel sistema YARA**

Il tracking di oggetti 3D viene richiesto da molti tipi di applicazioni, sia infatti da progetti di asservimento visivo per bracci robotici sia da sistemi di realtà aumentata, che generano la sovrapposizione di oggetti virtuali all'ambiente reale, per fornire informazioni supplementari. La visione offre soluzioni pratiche, non invasive ed economiche. L'approccio utilizzato è in particolare basato sulla libreria ARToolkit, e ha come obiettivo la stima della posizione 3D di fiducial marker fermo. L'oggetto obiettivo della presa del manipolatore viene infatti identificato attraverso il fiducial marker. Per lo sviluppo di un task di asservimento visivo basato sulla schema del sistema mostrato in figura 3.1, è stato necessario oltre che introdurre un nuovo apparato hardware, come la telecamera, anche gestire l'interazione tra la piattaforma Yara e la libreria ARToolkit. Quest'ultima intergrazione non è stata semplice, in quanto le funzionalità ARToolkit sono per la maggior parte basate sulla libreria Glut di OpenGL. Glut in particolare è una libreria che definisce delle finestre in cui OpenGL possa disegnare degli elementi grafici e gestire l'interazione con l'utente su tali finestre.

Un programma basato su OpenGL e Glut è formato da una funzione principale `main`, che contiene il codice Glut per compiere l'inizializzazione, la creazione di almeno una finestra, la registrazione di callback per gli eventi di interesse nelle finestre (almeno la callback di ridisegnamento), l'eventuale creazione di menù con relative callback, ed innesco del main loop. Le funzioni callback, che contengono del codice personalizzabile in base all'utilizzo, sono registrate nel `main` per poter essere eseguite in risposta ad eventi nelle finestre o all'attivazione dei menù. Un'applicazione sviluppata con ARToolkit segue esattamente le stesse modalità, con la

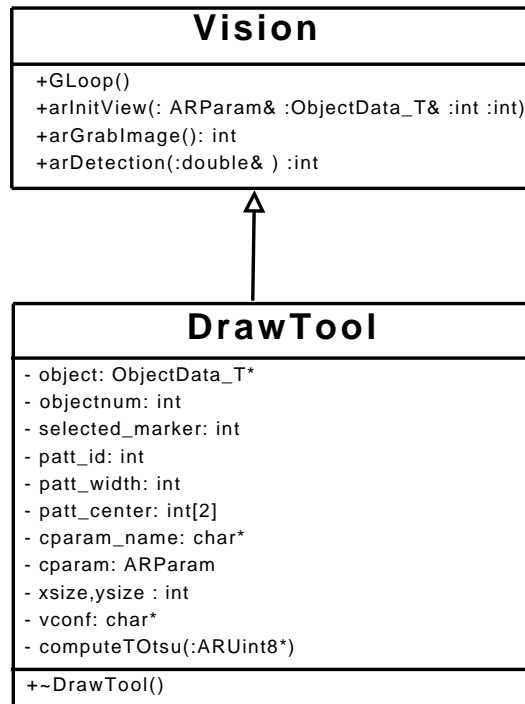
differenza che è prevista la configurazione della telecamera e il contenuto della finestra Glut è l'immagine acquisita. Vengono sfruttate le callback di disegno per creare degli elementi grafici visualizzabili sopra l'immagine reale.

La creazione di un'applicazione che permetta l'utilizzo del rilevamento di un marker da un'immagine, ed eseguita attraverso la piattaforma Yara, ha richiesto la completa modifica della struttura concettuale di un programma classico sviluppato con ARToolkit e Glut. La causa principale è stata la gestione della sincronizzazione tra l'accesso all'immagine per la visualizzazione e per la detection. Queste fasi che normalmente erano state progettate dalla libreria ARToolkit per essere effettuate in ordine sequenziale sono state divise. In particolare Yara, che gestisce la sincronizzazione tra i moduli del sistema di controllo del manipolatore, ha anche il compito di ordinare a cadenze temporali costanti l'acquisizione dell'immagine e la successiva analisi per la verifica della presenza del marker, che vengono specificate nel modulo *Camera*. È stato quindi modificata la funzione principale del programma di controllo del manipolatore in modo che, oltre a creare i moduli Yara, venga definito anche un puntatore di una classe *Vision*.

La classe *Vision* è un'interfaccia astratta che definisce i principali metodi necessari per lo svolgimento del task di asservimento visivo:

- la configurazione della telecamera,
- l'acquisizione dell'immagine,
- il rilevamento del marker.

Il grafico UML di figura 3.11, mostra come l'implementazione concreta di questi metodi venga effettuata attraverso la classe *DrawTool*, al fine di rendere possibile, in futuro, un cambiamento dell'apparato di elaborazione dell'immagine indipendentemente dalla realizzazione dello schema di asservimento visivo realizzato mediante Yara e rappresentato dallo schema di figura 3.2. È stato previsto che nel caso sia necessaria la visualizzazione in tempo reale delle immagini acquisite dalla telecamera, nel corpo principale del programma venga innescato il main loop della libreria Glut.



**Figura 3.11:** Diagramma UML dell’interfaccia Vision per la classe DrawTool

### 3.3 L’algoritmo di detection del marker

Dopo l’acquisizione di una scena da parte di un sensore (telecamera o scanner) deve essere eseguita la fase di riconoscimento del marker, al fine di ottenere le informazioni spaziali per coordinare successivamente l’azione del manipolatore. Questa operazione viene effettuata attraverso la funzione `arDetectMarker`, che determina il numero di pattern riconosciuti nell’immagine, crea una lista di strutture contenenti le informazioni riguardo alle loro coordinate e al ”confidence value” associato ad ogni pattern riconosciuto.

La funzione di detection del marker nell’immagine avviene attraverso una labelizzazione delle componenti connesse identificate, e un successivo confronto con il ”fiducial marker”. Prima di queste fasi di elaborazione, sull’immagine acquisita a colori RGB, viene applicata una binarizzazione, da cui si ottiene un’immagine costi-

tuita da soli pixel bianchi o neri. La trasformazione è ottenuta imponendo un valore di soglia, indicato dal parametro `thresh` nella funzione `arDetectMarker`. La soglia determina la luminosità al di sotto del quale tutti i pixel sono bianchi; viceversa tutti i pixel che originariamente hanno luminosità superiore alla soglia sono trasformati in neri. Questo tipo di processamento dell'immagine è denominato binarizzazione a soglia fissa. L'operazione di binarizzazione se non controllata potrebbe compromettere il contenuto rilevante dell'immagine e causarne anche un'errata elaborazione, che nel caso del task di asservimento visivo, potrebbe implicare il fallimento completo del sottosistema di visione.

Nella tecnica di binarizzazione a soglia fissa è molto importante utilizzare il valore della soglia adeguato all'immagine e questo valore è condizionato dall'illuminazione della scena. In letteratura sono stati proposti numerosi algoritmi di binarizzazione, che determinano il valore di soglia ottimo, ciò è giustificato dall'intrinseca difficoltà del problema e dalla diversità delle caratteristiche delle immagini da trattare. Le modalità principali per individuare il valore di soglia ottimale sono: la valutazione della soglia mediante l'istogramma bimodale o con l'istogramma modificato, il metodo di Otsu, il metodo della "soglia adattiva" ed il metodo della massima entropia. In particolare nella maggioranza dei casi la valutazione della soglia automaticamente necessita la stima dell'istogramma dei livelli di grigio dell'immagine originale.

Per migliorare le prestazioni dell'algoritmo di detection proposto dalla libreria ARToolkit è stato aggiunta una fase di determinazione della soglia ottima attraverso il metodo di Otsu [7], basato sul concetto di separabilità delle classi. Questa tecnica di valutazione della soglia, richiede che lo sfondo abbia un livello di grigio sufficientemente diverso da quello degli oggetti e viene implementato attraverso un procedimento iterativo, che permette di suddividere l'istogramma dei livelli di grigio dell'immagine in due gruppi, per ciascun valore possibile della soglia.

Considerando  $k \in [0, 255]$  il valore del livello di grigio e con  $h(k)$  viene definito il valore dell'istogramma in  $k$ . La probabilità di ogni livello di grigio nell'immagine viene stimata mediante la seguente espressione:

$$p(k) = \frac{h(k)}{\sum_{k=1}^{255} h(k)} \quad (3.2)$$

Per un valore fissato della soglia  $S$ , si determina una ripartizione dei livelli di grigio in due classi  $K_0$  e  $K_1$ . Le probabilità che un pixel appartenga alla prima classe è indicato dal valore di  $P_0$  e mentre  $P_1$  stima la probabilità di appartenenza alla classe  $K_1$ , e sono valutate attraverso le seguenti espressioni:

$$P_0 = \sum_{k=0}^S p(k) \quad (3.3)$$

$$P_1 = \sum_{k=S+1}^{255} p(k) = 1 - P_0 \quad (3.4)$$

Vengono poi stimati i valori di grigio medi delle due classi:

$$m_0 = \sum_{k=0}^S k \cdot p(k) \quad (3.5)$$

$$m_1 = \sum_{k=S+1}^{255} k \cdot p(k) \quad (3.6)$$

Il livello di grigio medio quindi nell'intera immagine è  $m = P_0 m_0 + P_1 m_1$ . La varianza del valore di grigio nella classe  $K_0$  viene valutata come scostamento dal valore medio, attraverso la valutazione di  $\sigma_0^2$ , allo stesso modo si calcola  $\sigma_1^2$ .

$$\sigma_0^2 = \sum_{k=0}^S (k - m_0)^2 \cdot p(k) \quad (3.7)$$

$$\sigma_1^2 = \sum_{k=S+1}^{255} (k - m_1)^2 \cdot p(k) \quad (3.8)$$

Infine è possibile calcolare la separazione tra le due classi di livelli di grigio  $\sigma_B^2$ , attraverso la varianza tra le classi, e la compattezza delle due classi viene misurata dalla varianza interna alle classi  $\sigma_W^2$ .

$$\sigma_B^2 = P_0(m_0 - m)^2 + P_1(m_1 - m)^2 \quad (3.9)$$

$$\sigma_W^2 = \sigma_0^2 P_0 + \sigma_1^2 P_1 (m_1 - m)^2 \quad (3.10)$$

Il valore della soglia  $S$  viene scelto in modo da massimizzare la separazione tra le classi e la compattezza di ciascuna, ed equivale a massimizzare il rapporto:

$$p(k) = \frac{\sigma_B^2}{\sigma_W^2} \quad (3.11)$$

Il metodo di Otsu ha permesso di selezionare il valore di threshold ottimale, che viene utilizzato per convertire l'immagine a colori in immagine binaria, durante la fase di detection del marker. L'algoritmo di stima della posizione del marker della libreria ARToolkit prosegue, dopo la fase di binarizzazione, con i seguenti passi:

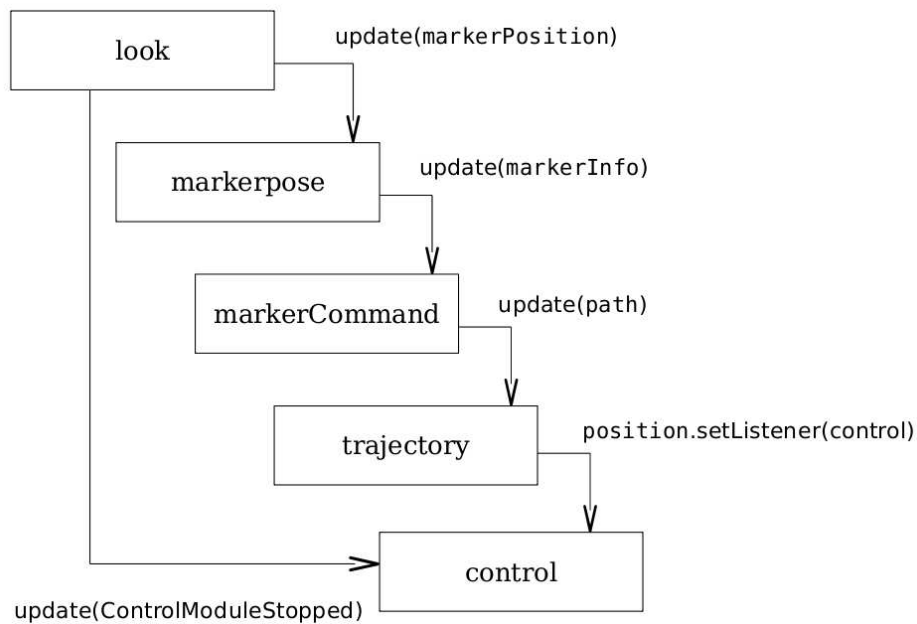
- la labellizzazione delle componenti connesse, che consente di distinguere e classificare gli oggetti presenti nella scena;
- l'estrazione del contorno degli oggetti, attraverso l'identificazione dei quattro segmenti;
- la determinazione dei quattro vertici che identificano una regione, attraverso l'intersezione dei segmenti trovati;
- la normalizzazione delle regioni;
- il confronto dell'elemento trovato nell'immagine con il pattern memorizzato nel sistema.

### **3.4 L'aggiornamento della traiettoria in real-time**

La pianificazione della traiettoria in base ai dati ottenuti da una telecamera, in questo caso, o da altri sistemi di visione in generale, incrementa l'autonomia dell'intero controllo del manipolatore antropomorfo. A causa di possibili imprecisioni di movimento dovuto alla meccanica del robot o a errori di accuratezza nella stima della posizione dell'oggetto, obiettivo della presa, si è deciso di introdurre il ricalcolo della traiettoria a cadenza temporale periodica.

Con riferimento allo schema del sistema complessivo, mostrato nella figura 3.2, ad ogni ricezione di una nuova posizione del marker da parte del modulo *Control*, viene generata una sequenza di via point per la pianificazione della traiettoria. In particolare il moto lungo la traiettoria corrente viene interrotto, per essere consentito l'esecuzione di una nuova traiettoria. La frequenza del cambiamento della traiettoria è stabilita dalla temporizzazione con cui viene schedulata l'esecuzione dell'Activity *look* del modulo *Camera*.

La modalità di funzionamento e di interazione tra le Activity è rappresentata nello schema di figura 3.12, dove si può notare che è stato necessario introdurre uno scambio di informazione tra l'Activity *look* e l'Activity *control*, per riuscire ad avere un aggiornamento del moto in real-time. Questa procedura di rinnovamento della traiettoria può essere realizzata però solo se il marker che identifica l'oggetto da prendere, rimane sempre visibile.



**Figura 3.12:** Sequenza di comando della traiettoria con riaggiornamento della pianificazione

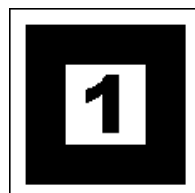
# Capitolo 4

## Risultati sperimentali

In questo capitolo verranno descritte le prove sperimentali che sono state eseguite al fine di valutare le prestazioni del sistema implementato.

### 4.1 Valutazione della stima della posizione dell'oggetto

Il funzionamento corretto dell'algoritmo di rilevamento della posizione e orientamento del marker, che identifica l'oggetto, è molto importante, in quanto determina il risultato del sistema di visione. L'elaborazione delle immagini acquisite mediante la telecamera deve essere indipendente dalle caratteristiche dell'ambiente di utilizzo. Un attributo del luogo che potrebbe condizionare il risultato dell'algoritmo di elaborazione è l'illuminazione. Il marker utilizzato nella fase sperimentale è mostrato in figura 4.1.



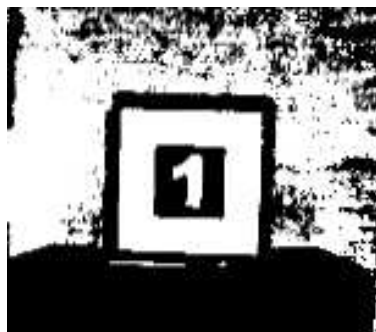
**Figura 4.1:** Marker utilizzato per il riconoscimento dell'oggetto

L'algoritmo di detection del marker utilizzato, facente parte della libreria AR-

Toolkit, applica una binarizzazione a soglia globale fissata a 100. Questo significa che ai pixel dell'immagine di valore inferiore vengono trasformati in bianco e viceversa gli altri sono trasformati in nero. Un esempio del risultato dell'elaborazione dell'immagine acquisita dalla telecamera di figura 4.2, viene mostrato dalla figura 4.3.

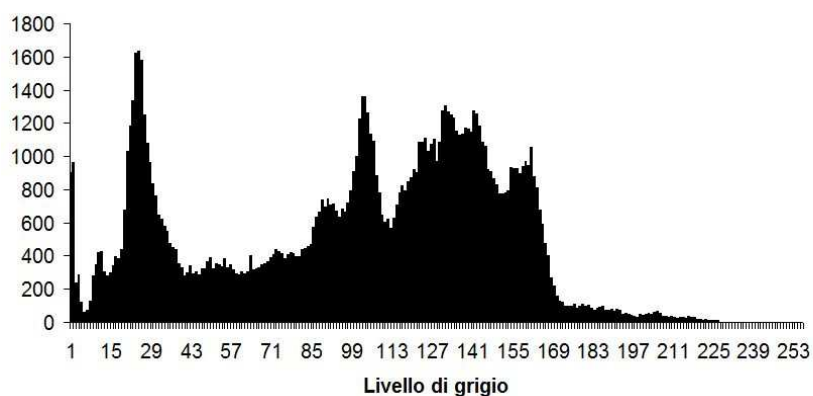


**Figura 4.2:** Immagine acquisita dalla telecamera



**Figura 4.3:** Immagine binarizzata con threshold fissa a 100

Nell'immagine 4.3 il marker viene rilevato già in modo accettabile, ma affinché la binarizzazione riesca sempre ad effettuata, indipendentemente dalla luminosità e senza che comporti una perdita di informazione, è stato implementato un algoritmo per la determinazione del valore di soglia ottimo. Il calcolo del valore dell'istogramma dei livelli di grigio ha consentito, infatti, di rappresentare la distribuzione della frequenza con cui i livelli di luminanza appaiono in un'immagine. Un esempio del valore dell'istogramma di un'immagine direttamente acquisita dalla telecamera è presentato dal grafico di figura 4.4.



**Figura 4.4:** Istogramma dei livelli di grigio

Il valore dell'istogramma dei livelli di grigio viene utilizzato nella determinazione del valore di soglia ottima, attraverso l'applicazione del metodo di Otsu, descritto nel paragrafo 3.3. In questo modo ad ogni acquisizione dell'immagine viene determinata la soglia ideale da applicare. Ad esempio, la binarizzazione effettuata sull'immagine 4.5 con soglia ottima pari a 125 mostra che i contorni del marker sono più precisi.



**Figura 4.5:** Immagine binarizzata con threshold ottimizzata

## **4.2 Valutazione dell'esecuzione della traiettoria per la presa dell'oggetto**

L'analisi della pianificazione della traiettoria, utilizzando i via point determinati in real-time, è stata scomposta in due modalità di raggiungimento della posizione finale. In un primo studio è stato verificato se attraverso una singola pianificazione il manipolatore era in grado di portarsi nella posizione di presa corretta. La determinazione dei coefficienti della traiettoria, espressa attraverso una funzione polinomiale, come descritto nel paragrafo 1.3.1, è condizionata dalla stima della posizione e dell'orientamento del marker, che identifica l'oggetto. L'accuratezza del risultato della fase di detection del sottosistema di visione, è quindi molto importante. Se nelle coordinate del marker sono presenti errori, questi si ripercuoteranno sulla posizione finale raggiunta dal manipolatore, e potrebbe causare un fallimento del task di presa.

In un secondo momento è stata introdotta la ripianificazione della traiettoria ad intervalli temporali costanti. Nelle sezioni successive verranno riportati i risultati ottenuti.

### **4.2.1 Una singola traiettoria**

In questa analisi è stato possibile raggiungere l'oggetto target attraverso la pianificazione di una singola traiettoria. Il file di via point è generato in tempo reale attraverso i dati reperiti dalla telecamera. Sono stati utilizzati il numero minimo di punti necessari per la pianificazione, pari a tre. Ad ogni esecuzione il primo via point è determinato dalla posizione e dall'orientamento all'istante iniziale del manipolatore, e il valore esatto viene calcolato prima dalla routines di pianificazione. Per gli altri punti è stata prevista la possibilità di specifica in modo assoluto oppure in modo incrementale rispetto al punto precedente. Nel primo caso il secondo via point è determinato come il punto medio tra la posizione iniziale e la posizione finale che si deve raggiungere ed il terzo punto rappresenta le coordinate del marker nello spazio operativo del robot. Nel caso siano definiti in modo incrementale allora i due via point rappresentano la variazione rispetto al punto precedente, ed essendo solo due coincidono con il valore della media tra i via point. Il tempo di

esecuzione che viene specificato in ogni set di via point è stato fissato a priori per riuscire ad eseguire lentamente la traiettoria, in modo da analizzare eventuali errori di spostamento.

Per analizzare le prestazioni dell'inseguimento della traiettoria calcolata sono state effettuate molte prove ed è stato valutata la funzione di errore relativa alla distanza euclidea rilevata dalla telecamera all'istante iniziale,  $d(t_i)$ , rispetto alla distanza euclidea percorsa dal braccio robotico durante la presa,  $d(t_f)$ . Le prove hanno portato al risultato esposto nella tabella 4.1, da cui si può evincere come il manipolatore riesca effettivamente ad eseguire la traiettoria. La media della distanza misurata all'istante iniziale infatti, non si discosta molto dalla distanza percorsa rilevata alla fine della presa. Questo risultato riesce a confermare l'effettiva riuscita dall'inseguimento 3D della traiettoria determinata attraverso i via point dati.

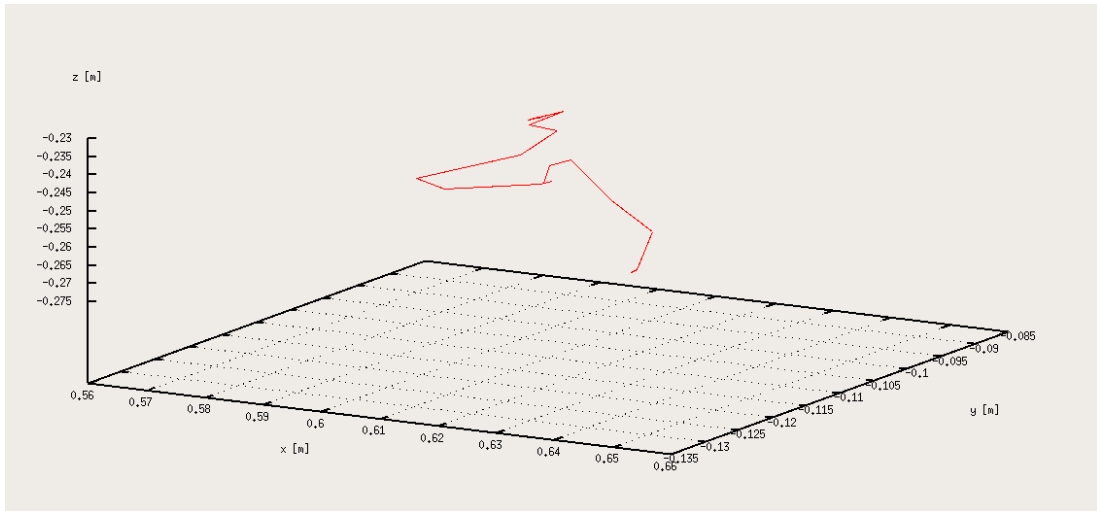
	$d(t_i)$	$d(t_f)$	errore
Media	0,252	0,258	-0,006
Deviazione std	0,03	0,04	0,03

**Tabella 4.1:** Distanza misurata  $d(t_i)$  all'istante iniziale vs distanza percorsa  $d(t_f)$  in metri

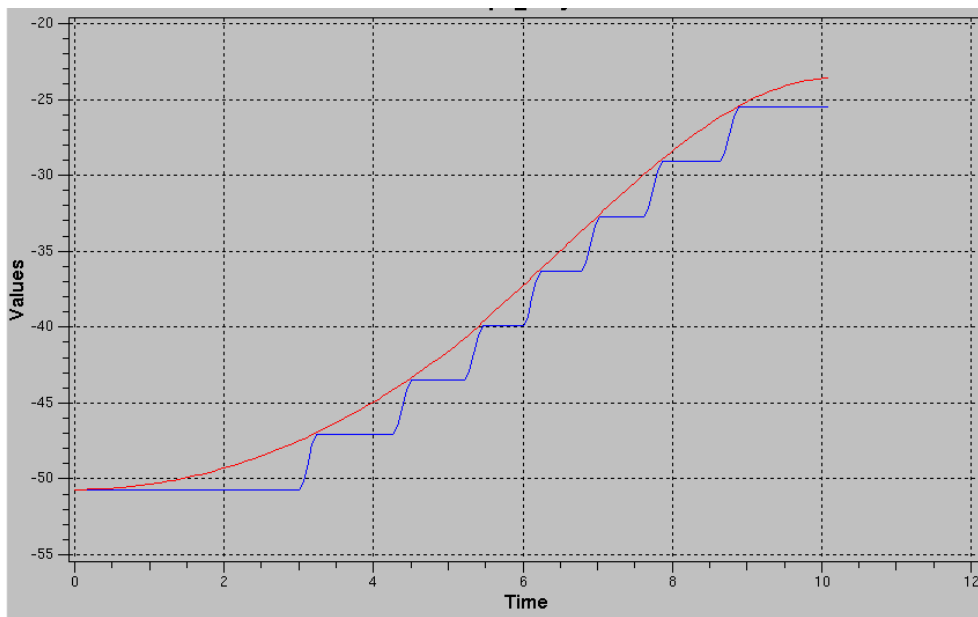
Nell'analisi della traiettoria sono state anche rilevate le posizioni raggiunte dal braccio robotico durante il moto, al fine di poter studiare la forma della percorso eseguito. Un'esempio è mostrato in figura 4.6, dove il braccio si è portato da una posizione iniziale da un'altezza di -27 cm rispetto all'asse z ad un'altezza di -23 cm.

Anche nello spazio dei giunti è stato possibile analizzare l'inseguimento della traiettoria teorica, infatti il grafico di figura 4.7, che rappresenta la traiettoria percorsa dall'ultimo giunto rilevata durante il moto del manipolatore, conferma la buona riuscita. L'andamento della velocità del sesto giunto durante la presa dell'oggetto viene mostrata in figura 4.8, mentre l'errore di inseguimento della traiettoria rilevato nello spazio dei giunti è rappresentato dal grafico di figura 4.9.

Nelle prove effettuate la presa non viene sempre realizzata, perché a volte il manipolatore non riesce a portarsi nella posizione ed orientazione corretta. La distanza dal marker in queste situazioni è tale da non permettergli la presa. Le analisi riguardanti l'inseguimento della traiettoria confermano che la causa del fallimento

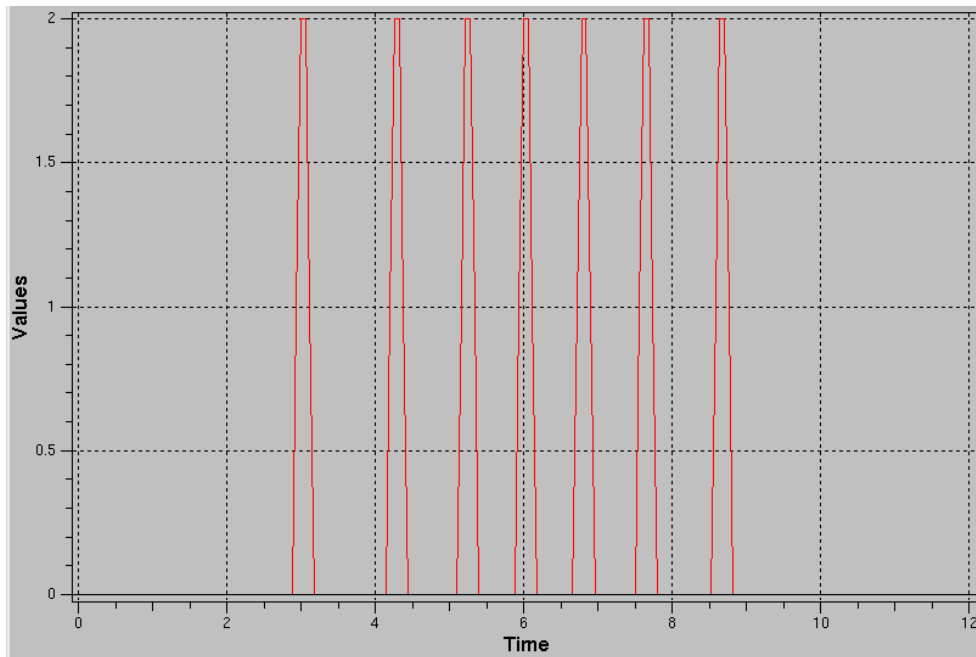


**Figura 4.6:** Percorso eseguito dal braccio robotico durante la presa dell'oggetto

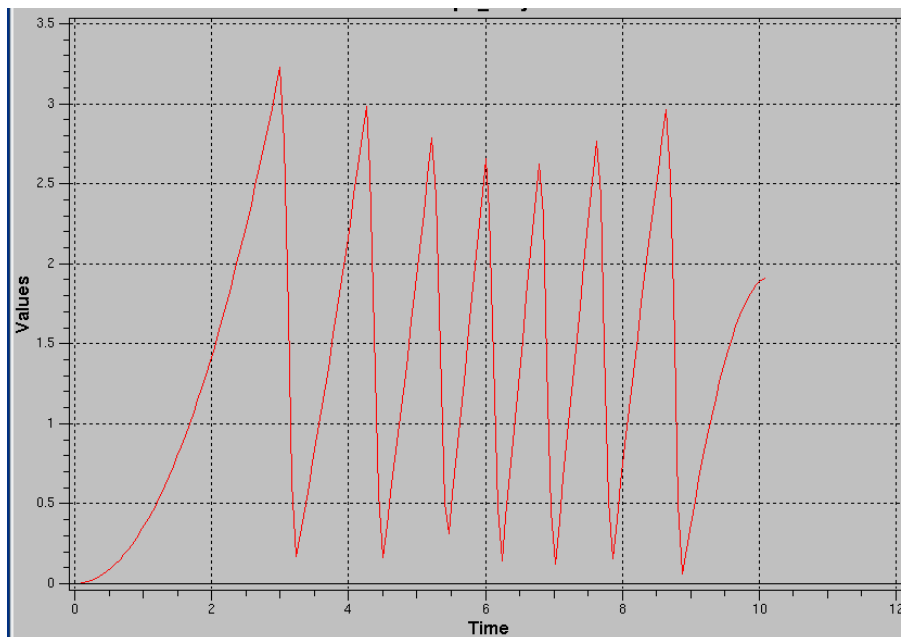


**Figura 4.7:** Traiettoria pianificata per il sesto giunto vs traiettoria eseguita

del task di asservimento visivo non è dovuto ad un errore sul percorso eseguito dal braccio robotico durante il moto, ma è originato quindi dalle imprecisioni presenti nella valutazione dei via point. È stata effettuata una verifica sui valori stimati



**Figura 4.8:** Velocità per il sesto giunto



**Figura 4.9:** Errore di inseguimento commesso dal sesto giunto

dai passaggi matematici, che determinano i via point, e risultano essere coerenti con le trasformazioni geometriche studiate. La causa dell'errata specifica dei via point, è quindi dovuta alle imprecisioni presenti nella stima del marker rispetto alla telecamera.

#### **4.2.2 Aggiornamento della traiettoria in real-time**

Nell'ultima parte del lavoro di tesi è stato realizzato l'aggiornamento della traiettoria in real-time. Questa funzionalità consente di eseguire più traiettorie in sequenza temporale e quindi offre la possibilità di correggere l'andamento del percorso in tempo reale, che sarebbe utile anche nel caso si considerasse la presa di oggetti in movimento. Il numero di via point specificati per ogni traiettoria è tre, e i tempi di passaggio per i punti sono fissati a priori.

L'aggiornamento della traiettoria è determinato dalla frequenza di detection del marker nell'immagine, che viene stabilita dal modulo Camera del sistema di controllo, realizzato mediante Yara. Le prime prove sono state realizzate attraverso una ripianificazione ad intervalli di tempo elevati di circa 4 secondi, in modo da studiare l'effettiva riuscita di un aggiornamento del calcolo e dell'esecuzione. In seguito è stata provata una sincronizzazione del task di comando di esecuzione di una nuova traiettoria a frequenza più elevata di 0.5 Hz. Questa prova ha dimostrato che se viene interrotta l'esecuzione della traiettoria corrente, in attesa che ne sia disponibile un'altra, anche l'acquisizione dei dati dal manipolatore deve essere fermata, per non causare problemi nello scheduling dei processi.

# Conclusioni

In questa tesi è stato realizzato un sistema di asservimento visivo per robot manipolatori, finalizzato alla presa di oggetti. Il contributo principale di questo lavoro è consistito nella modifica dell'architettura di controllo del Manus, sviluppata sulla piattaforma YARA, in modo da permettere l'utilizzo di un sistema di visione, rappresentato dal modulo per la configurazione della telecamera e nel modulo per l'elaborazione dell'immagine per estrapolarne dei dati di posizionamento  $3D$ . Le prove eseguite sono state in grado di far emergere le potenzialità del sistema realizzato, evidenziando i vantaggi e gli svantaggi delle soluzioni tecniche adottate. Le funzionalità di visione artificiale utilizzate sono state sviluppate facendo uso della libreria ARToolkit, che ha consentito di ottenere la posizione  $3D$  di un marker noto, riducendo il carico computazionale in maniera significativa. L'accuratezza nella stima della posizione è condizionata dalle ridotte dimensioni del marker, e i test eseguiti hanno confermato delle imprecisioni dell'ordine del centimetro se aumenta la distanza dal marker. Questo ha compromesso in alcune sperimentazioni la riuscita del task di presa dell'oggetto.

Gli sviluppi futuri di questa tesi nel campo della visione artificiale potrebbero essere i seguenti:

- l'introduzione di telecamere stereoscopiche ed utilizzo della geometria epipolare per il visual servoing;
- l'implementazione di un algoritmo di riconoscimento degli oggetti più robusto alla variabilità dell'illuminazione della scena ed insensibile alle occlusioni;
- la considerazione di un possibile movimento dell'oggetto;

## *Conclusioni*

---

- l'integrazione con delle informazioni riguardanti features dell'immagine, tipiche di un visual servoing image based, per aumentare l'accuratezza del controllo di posizione.

Un'implementazione più accurata della ripianificazione della traiettoria in real-time, inoltre, potrebbe consentire di correggere eventuali errori introdotti dal sistema di visione e rendere più dinamico il controllo del robot, anche in caso di realizzazione di una presa di oggetti in movimento.

# Appendice A

## Cinematica del robot

### A.1 La convenzione di Denavit-Hartenberg

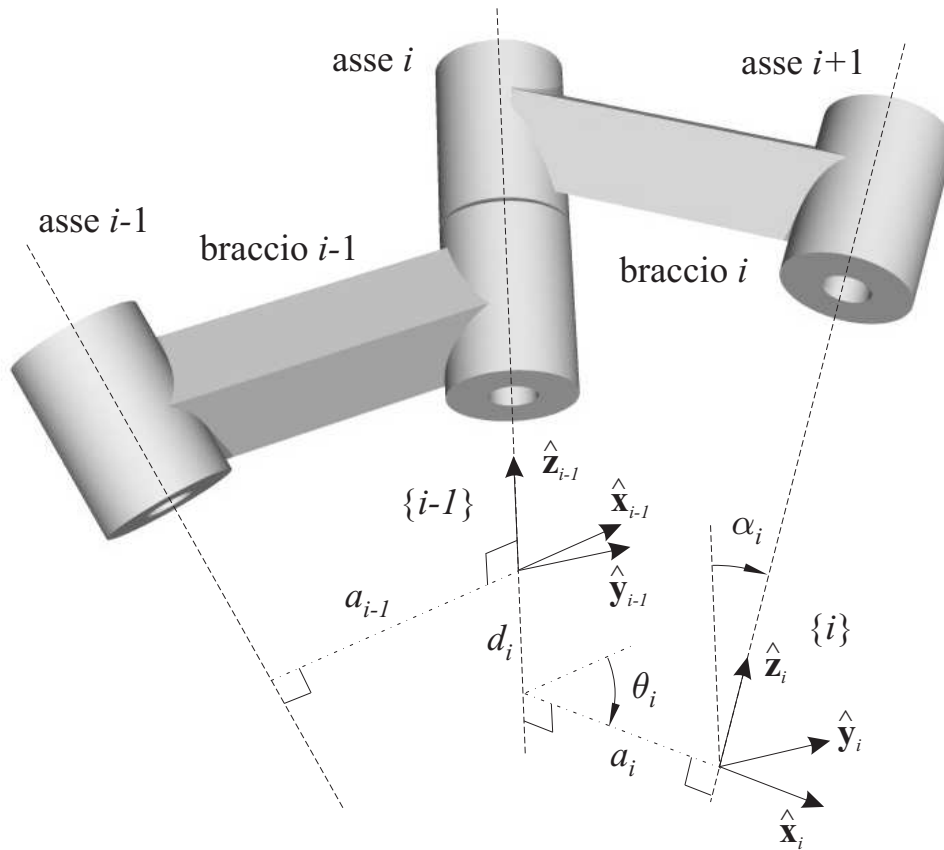
Su ciascun braccio del manipolatore verranno fissati dei sistemi di riferimento rispettando alcune regole che consentono di ricavare in modo semplice le matrici di trasformazione. La tecnica descritta è quella di Denavit e Hartenberg. Il procedimento prevede che su ciascun braccio venga fissata una sola terna utilizzata tanto per fornire indicazioni sulla geometria del braccio, che sull'accoppiamento tra quest'ultimo e il braccio successivo.

Esistono due versioni della convenzione di Denavit-Hartenberg in cui la differenza sostanziale tra la versione standard e quella modificata riguarda il posizionamento dell'origine delle terne di braccio. Mentre la versione modificata richiede di porre l'origine di ciascuna terna sul primo asse di giunto di ciascun braccio (quello più prossimo alla base del manipolatore), la versione standard richiede di porre l'origine sul secondo asse di giunto di ciascun braccio (quello più prossimo al polso del manipolatore).

#### **Versione standard della convenzione di Denavit-Hartenberg**

Con riferimento alla figura A.1, per fissare la terna relativa al giunto  $i$ -esimo si procede nel seguente modo:

1. si sceglie l'asse  $z_i$  giacente lungo l'asse del giunto  $i + 1$ ;



**Figura A.1:** Terne secondo la convenzione di Denavit-Hartenberg

2. si individua  $O_i$  all'intersezione dell'asse  $z_i$  con la normale comune agli assi  $z_{i-1}$  e  $z_i$  e con  $O_{i'}$  si indica l'intersezione con la normale comune con  $z_{i-1}$ ;
3. si sceglie l'asse  $x_i$  lungo la normale comune agli assi  $z_{i-1}$  e  $z_i$  con verso positivo dal giunto  $i$  al giunto  $i + 1$ ;
4. si sceglie l'asse  $y_i$  in modo da completare una terna levogira.

Nei seguenti casi la convenzione DH non fornisce una definizione univoca della terna:

- nella terna 0 solo la direzione dell'asse  $z_0$  risulta specificata,  $O_0$  e  $x_0$  possono essere scelti arbitrariamente;
- non essendovi un giunto  $n + 1$  nella terna  $n$  l'asse  $z_n$  non è univocamente definito, mentre l'asse  $x_n$  deve essere normale all'asse  $z_{n-1}$ ;
- quando due assi consecutivi sono paralleli, avendo la normale comune tra di loro non univocamente definita;
- quando due assi consecutivi si intersecano, in quanto il verso di  $x_i$  è arbitrario;
- quando il giunto  $i$  è prismatico, l'unica direzione determinata è quella dell'asse  $z_i$

La posizione e l'orientazione della terna  $i$  e  $i - 1$  risultano completamente definite dai seguenti parametri:

- $a_i$ : distanza tra  $O_i$  e  $O_{i'}$ ,
- $d_i$ : coordinata su  $z_{i-1}$  di  $O_{i'}$ ,
- $\alpha_i$ : angolo intorno all'asse  $x_i$  tra l'asse  $z_{i-1}$  e l'asse  $z_i$  valutato positivo in senso antiorario,
- $\theta_i$ : angolo intorno all'asse  $z_{i-1}$  tra l'asse  $x_{i-1}$  e l'asse  $x_i$  valutato positivo in senso antiorario.

$a_i$  e  $\alpha_i$  sono sempre costanti e dipendono solamente dalla geometria di connessione dei giunti consecutivi tramite il braccio. Tra  $d_i$  e  $\theta_i$  solo uno è variabile in base al tipo di giunto utilizzato nella connessione dei bracci  $i$  e  $i - 1$ : se il giunto è rotoideale la variabile è  $\theta_i$ , mentre se il giunto è prismatico la variabile è  $d_i$ .

A questo punto si può ricavare la trasformazione che lega la terna  $i$  con la terna  $i - 1$  secondo i seguenti passi:

- si parte da una terna coincidente con la terna  $i - 1$ ;
- si trasla la terna di  $d_i$  lungo l'asse  $z_{i-1}$  e la si ruota di  $\theta_i$  attorno allo stesso asse; questa operazione viene descritta dalla matrice:

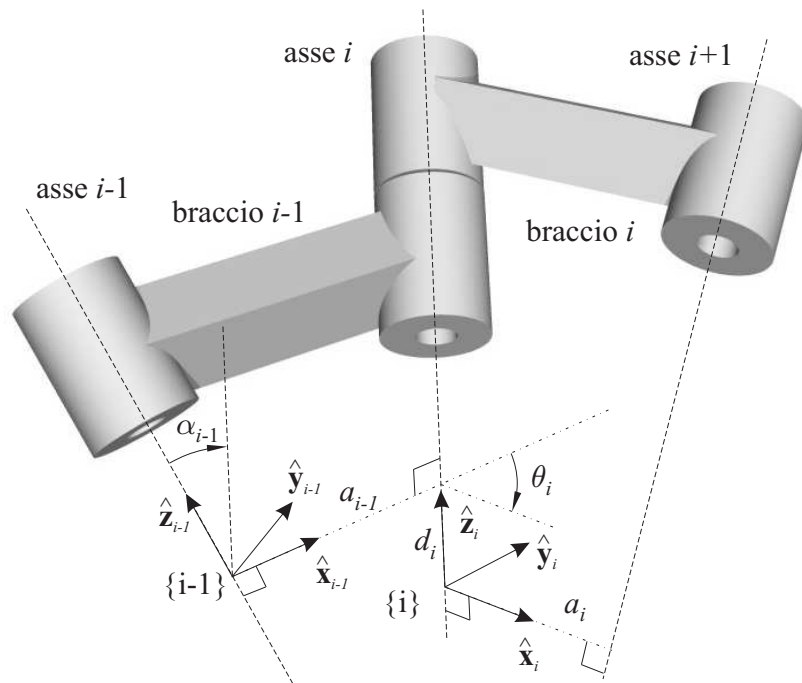
$$\mathbf{A}_{i'}^{i-1} = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- si trasla ora la matrice ottenuta nel passaggio precedente di  $a_i$  lungo l'asse  $x_{i'}$  e la si ruota di  $\alpha_i$  attorno allo stesso asse:

$$\mathbf{A}_i^{i'} = \begin{pmatrix} 1 & 0 & 0 & \alpha_i \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- la trasformazione di coordinate complessiva si ottiene moltiplicando le singole trasformazioni:

$$A_i^{i-1}(q_i) = A_i^{i-1} A_i^{i'} = \begin{pmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & \alpha_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & \alpha_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (\text{A.1})$$



**Figura A.2:** Terne secondo la convenzione di Denavit-Hartemberg modificata

### Versione modificata della convenzione di Denavit-Hartemberg

Si considerino coinvolte due generiche terne intermedie  $i$  e  $i-1$  e ci si riferisca alla figura A.2. L'algoritmo per l'assegnazione delle terne secondo la convenzione di Denavit-Hartemberg modificata, è il seguente:

1. Si identificano e si numerano gli assi di giunto(1,2,...,N). Si ponga un asse virtuale 0(asse di base) coincidente con l'asse 1. Per fissare le terne progressivamente dalla 1 alla N-1 si eseguano iterativamente i seguenti passi, ricordando che per fissare la terna  $i$ -ma al braccio  $i$ -mo si considerano sempre i due assi contigui  $i$  ed  $i + 1$ .
2. Si identifichi la perpendicolare comune tra l'asse  $i$  e l'asse  $i + 1$  (ovvero la distanza minima tra i due assi). Nel punto in cui tale perpendicolare incontra l'asse  $i$ -mo si assegni l'origine del riferimento del braccio  $i$ -mo. Se i due assi si intersecano in un punto, si assegni l'origine del riferimento del braccio  $i$ -mo in corrispondenza di tale intersezione. Se i due assi sono coincidenti, si assegni liberamente l'origine del riferimento lungo l'asse  $i$ -mo.

3. Si assegni il versore  $\hat{z}_i$  lungo l'asse  $i$ -mo. Il suo verso può essere fissato liberamente.
4. Si assegni il versore  $\hat{x}_i$  lungo la perpendicolare comune tra l'asse  $i$  e l'asse  $i+1$ , possibilmente con il verso diretto dall'asse  $i$ -mo all'asse  $(i+1)$ -mo. Nel caso in cui i due assi si intersecano in un punto, si assegni  $\hat{x}_i$  in modo che sia perpendicolare al piano contenente i due assi e se ne fissi liberamente il verso. Se i due assi sono coincidenti, si assegni liberamente la direzione e il verso di  $\hat{x}_i$ .
5. Si assegni il versore  $\hat{y}_i$ , in modo che la terna  $i$ -ma così ottenuta sia una terna levogira.
6. Se  $i < N$  allora si torni al passo 2.

La terna 0 e la terna N vengono fissate come segue:

- Se il primo giunto è di tipo rotoidale, si assegni il riferimento 0 in modo che la sua origine coincida con quella della terna 1 e il versore  $\hat{z}_0$  coincida con il versore  $\hat{z}_1$ . Si assegnino gli altri versori in modo che la terna ottenuta sia levogira. Se il primo giunto è di tipo prismatico, si orientino i tre versori della terna 0 esattamente come i tre versori della terna 1 (ovviamente  $\hat{z}_0$  e  $\hat{z}_1$  giaceranno sullo stesso asse).
- Se l'ultimo giunto è di tipo rotoidale, si assegni l'origine del riferimento N nel punto in cui la prosecuzione del versore  $\hat{x}_{N-1}$  interseca l'asse N e si fissi il versore  $\hat{z}_N$  lungo l'asse N stesso. Gli altri due versori possono essere scelti liberamente, prestando attenzione che la terna ottenuta sia levogira. Se l'ultimo giunto è prismatico si assegni liberamente l'origine del riferimento N lungo l'asse N e se ne fissi il versore  $\hat{z}_N$  lungo l'asse N stesso. Si scelga  $\hat{x}_N$  parallelo a  $\hat{x}_{N-1}$ , mentre  $\hat{y}_N$  venga scelto in modo che la terna ottenuta sia levogira.

## A.2 Il calcolo della cinematica diretta

Lo studio della determinazione della posa del dispositivo d'estremità nel riferimento di base essendo note le coordinate di giunto. La trasformazione di coordinate complessiva del sistema di riferimento del gripper rispetto a quello di base, rappresentata dalla matrice A.2, si ottiene moltiplicando tra loro tutte le matrici contenute in tabella (A.1).

$$\begin{aligned}
 A_1^0 &= \begin{pmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} & A_2^1 &= \begin{pmatrix} c_2 & -s_2 & 0 & L_2 c_2 \\ s_2 & c_2 & c_1 & L_2 s_2 \\ 0 & 0 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
 A_3^2 &= \begin{pmatrix} c_3 & 0 & s_3 & 0 \\ s_3 & 0 & -c_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} & A_4^3 &= \begin{pmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & c_1 & 0 \\ 0 & 0 & 0 & L_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
 A_5^4 &= \begin{pmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} & A_6^5 &= \begin{pmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 0 & L_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}
 \end{aligned}$$

**Tabella A.1:** Matrici di trasformazione dei giunti (Denavit-Hartenberg)

$$T_6^0 = A_1^0 A_2^1 A_3^2 A_4^3 A_5^4 A_6^5 = \left( \begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \quad (\text{A.2})$$

La matrice di trasformazione A.2 è scomponibile in una matrice di rotazione  $\mathbf{R}[3 \times 3]$  e un vettore  $\mathbf{x} = [xyz]^T$ , che sono rispettivamente l'orientamento e la posizione del tool frame rispetto al base frame. Gli elementi della matrice  $T_6^0$  sono i seguenti:

$$\begin{aligned}
 r_{11} &= ((c_1 c_{23} c_4 - s_1 s_4) c_5 - c_1 s_{23} s_5) c_6 - (c_1 c_{23} s_4 + s_1 c_4) s_6 \\
 r_{12} &= -((c_1 c_{23} c_4 - s_1 s_4) c_5 - c_1 s_{23} s_5) s_6 - (c_1 c_{23} s_4 + s_1 c_4) c_6 \\
 r_{13} &= (c_1 c_{23} c_4 - s_1 s_4) s_5 + c_1 s_{23} c_5 \\
 r_{21} &= ((s_1 c_2 c_3 c_4 + c_1 s_4) c_5 - s_1 s_{23} s_5) c_6 - (s_1 c_2 c_3 s_4 - c_1 c_4) s_6 \\
 r_{22} &= -((s_1 c_2 c_3 c_4 + c_1 s_4) c_5 - s_1 s_{23} s_5) s_6 - (s_1 c_2 c_3 s_4 - c_1 c_4) c_6 \\
 r_{23} &= (s_1 c_2 c_3 c_4 - c_1 s_4) s_5 + s_1 s_{23} c_5 \\
 r_{31} &= -(s_{23} c_4 c_5 + c_{23} s_5) c_6 + s_{23} s_4 s_6 \\
 r_{32} &= (s_{23} c_4 c_5 + c_{23} s_5) s_6 + s_{23} s_4 c_6 \\
 r_{33} &= c_{23} c_5 - s_{23} c_4 s_5 \\
 x &= ((c_1 c_{23} c_4 - s_1 s_4) s_5 + c_1 s_{23} c_5) L_4 + c_1 s_{23} L_3 + c_1 c_2 L_2 - s_1 L_1 \\
 y &= ((s_1 c_2 c_3 c_4 - c_1 s_4) s_5 + s_1 s_{23} c_5) L_4 + s_1 s_{23} L_3 + s_1 c_2 L_2 + c_1 L_1 \\
 z &= (c_{23} c_5 - s_{23} c_4 s_5) L_4 - c_2 c_3 L_3 - s_2 L_2
 \end{aligned}$$

**Tabella A.2:** Soluzioni della cinematica diretta

### A.3 Il calcolo della cinematica inversa

Consiste nel determinare il movimento da assegnare ai motori per ottenere un prefissato movimento dell'organo terminale. Si consideri dato il vettore

$$\mathbf{x} = \begin{pmatrix} \mathbf{p} \\ \Phi \end{pmatrix}$$

in cui  $\mathbf{p}$  e  $\Phi$  indicano rispettivamente la posizione e l'orientamento espresso in notazione minima RPY<sup>1</sup>, si vuole determinare il vettore  $\theta$  che può essere diviso in due parti:

$$\theta = \begin{pmatrix} \tilde{\theta} \\ \hat{\theta} \end{pmatrix} = \left( \theta_1 \quad \theta_2 \quad \theta_3 \mid \theta_4 \quad \theta_5 \quad \theta_6 \right)^T$$

I valori delle variabili di giunto relative ai primi tre bracci del manipolatore  $\tilde{\theta}$  non dipendono dall'orientamento della terna utensile, ma dipendono esclusivamente dalla posizione dell'origine della terna di polso, ovvero da  $p_3^0$  descritta dall'espressione A.3. La cinematica inversa, relativamente a queste prime tre variabili di giunto, viene risolta attraverso un sistema di sole tre equazioni in tre incognite del tipo

---

<sup>1</sup>Notazione RPY (Roll, Pitch, Yaw) descrive l'orientamento di una terna usando solo tre parametri  $(\alpha, \beta, \gamma)$  rispetto ai nove delle matrici di rotazione.

rappresentato dall'equazione A.3.

$$\mathbf{p}_3^0 = \mathbf{p} - L_4 \mathbf{R}_6^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = \begin{pmatrix} x - L_4(c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma) \\ y - L_4(s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma) \\ z - L_4 c_\beta c_\gamma \end{pmatrix} \quad (\text{A.3})$$

$$\mathbf{p}_3^0(\tilde{\theta}) = \mathbf{p}_3^0 \quad (\text{A.4})$$

dove il vettore  $\mathbf{p}_3^0(\tilde{\theta})$  viene ricavato dalla matrice di trasformazione omogenea  $\mathbf{T}_3^0(\tilde{\theta})$  ottenuta al momento di valutare la cinematica diretta del manipolatore. Più nel dettaglio il sistema da risolvere sarà

$$\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = \begin{pmatrix} c_1 s_{23} L_3 + c_1 c_2 L_2 - s_1 L_1 \\ s_1 s_{23} L_3 + s_1 c_2 L_2 + c_1 L_1 \\ c_{23} L_3 - s_2 L_2 \end{pmatrix}$$

e la soluzione per le prime tre variabili di giunto è la seguente:

$$\tilde{\theta}_1 = 2 \arctan \left( \frac{-x_w \pm \sqrt{x_w^2 + y_w^2 - L_1^2}}{y_w + L_1} \right)$$

$$\tilde{\theta}_2 = \text{Atan2}(s_2, c_2)^2$$

$$\tilde{\theta}_3 = \pi - \arcsin \left( \frac{x_w^2 + y_w^2 + z_w^2 - L_1^2 - L_2^2 - L_3^2}{2L_2 L_3} \right)$$

Come si può notare  $\tilde{\theta}_1$  e  $\tilde{\theta}_3$  presentano due soluzioni che devono soddisfare rispettivamente le condizioni  $x_w^2 + y_w^2 \geq L_1^2$ , e  $L_1^2 + (L_2 - L_3)^2 \leq x_w^2 + y_w^2 + z_w^2 \leq L_1^2 + (L_2 + L_3)^2$ , mentre  $\tilde{\theta}_2$  ha una sola soluzione dipendente da  $\tilde{\theta}_1$  e  $\tilde{\theta}_3$ .

Esistono quattro soluzioni alla cinematica inversa per i primi tre giunti, vedi l'espressione A.5. Tra le possibili soluzioni deve essere scelta quella che minimizza il cammino del manipolatore e riesce quindi a minimizzare la norma della differenza della sua posizione con la posizione precedente, condizione rappresentata

---

<sup>2</sup>dove  $s_2 = (c_3 L_3 (c_1 x_w + s_1 y_w) - z_w (s_3 L_3 + L_2))$  e  $c_2 = z_w c_3 L_3 + (s_3 L_3 + L_2)(c_1 x_w + s_1 y_w)$ .

dall'equazioni A.6 e A.7.

$$\tilde{\theta}_i = \begin{pmatrix} \tilde{\theta}_{i,1} \\ \tilde{\theta}_{i,2} \\ \tilde{\theta}_{i,3} \end{pmatrix} \quad i = 1, 2, 3, 4. \quad (\text{A.5})$$

$$\tilde{\theta}_i(t) - \tilde{\theta}_i(t-1) = \begin{pmatrix} \tilde{\theta}_{i,1}(t) - \tilde{\theta}_{i,1}(t-1) \\ \tilde{\theta}_{i,2}(t) - \tilde{\theta}_{i,2}(t-1) \\ \tilde{\theta}_{i,3}(t) - \tilde{\theta}_{i,3}(t-1) \end{pmatrix} \quad i = 1, 2, 3, 4. \quad (\text{A.6})$$

$$\tilde{\theta} = \min \left\{ \left\| \tilde{\theta}_1(t) - \tilde{\theta}_1(t-1) \right\|, \dots, \left\| \tilde{\theta}_4(t) - \tilde{\theta}_4(t-1) \right\| \right\} \quad (\text{A.7})$$

Il valore delle prime tre variabili di giunto permette di conoscere la matrice di rotazione  $\mathbf{R}_3^0(\tilde{\theta})$  quindi data la matrice  $\mathbf{R}_6^0$  sarà possibile ricavare

$$\mathbf{R}_6^3(\tilde{\theta}) = \mathbf{R}_6^{3^{-1}}(\tilde{\theta})\mathbf{R}_6^0 = \mathbf{R}_6^{3^T}(\tilde{\theta})\mathbf{R}_6^0 = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad (\text{A.8})$$

dove i termini  $r_{ij}$  della matrice sono calcolati come indicato in tabella A.3.

$$\begin{aligned} r_{11} &= c_\beta c_{23} c_{(1-\alpha)} + s_\beta s_{23} \\ r_{12} &= s_\beta s_\gamma c_{23} c_{(1-\alpha)} + c_\gamma c_{23} s_{(1-\alpha)} - c_\beta s_\gamma s_{23} \\ r_{13} &= s_\beta c_\gamma c_{23} c_{(1-\alpha)} + s_\gamma c_{23} s_{(\alpha-1)} - c_\beta c_\gamma s_{23} \\ r_{21} &= c_\beta s_{(\alpha-1)} \\ r_{22} &= c_\gamma c_{(1-\alpha)} + s_\beta s_\gamma s_{(\alpha-1)} \\ r_{23} &= s_\beta c_\gamma s_{(1-\alpha)} - s_\gamma c_{(\alpha-1)} \\ r_{31} &= c_\beta s_{23} c_{(1-\alpha)} - s_\beta c_{23} \\ r_{32} &= s_\beta s_\gamma s_{23} c_{(1-\alpha)} + c_\gamma s_{23} s_{(1-\alpha)} + c_\beta s_\gamma c_{23} \\ r_{33} &= s_\beta c_\gamma s_{23} c_{(\alpha-1)} + s_\gamma s_{23} s_{(\alpha-1)} + c_\beta c_\gamma c_{23} \end{aligned}$$

**Tabella A.3:** Coefficienti della matrice di rotazione relativa ai primi tre giunti.

Il sistema indicato dall'equazione A.9 con matrice di rotazione A.10 relativa al polso sferico, permette di valutare le variabili relative agli ultimi tre giunti.

$$\mathbf{R}_6^3(\hat{\theta}) = \mathbf{R}_6^3(\tilde{\theta}) \quad (\text{A.9})$$

$$\mathbf{R}_6^3(\hat{\theta}) = \begin{pmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{pmatrix} \quad (\text{A.10})$$

Anche per le ultime tre variabili di giunto esistono due possibili configurazioni rappresentata dalle equazioni A.11, e per ognuna viene calcolata la soluzione riportata nella A.12, nel caso in cui  $\hat{\theta}_5 = 0$  o  $\hat{\theta}_5 = \pi$ .

$$\begin{aligned} \hat{\theta}_4 &= \text{Atan2}(r_{23}, r_{13}) & \hat{\theta}_4 &= \text{Atan2}(-r_{23}, -r_{13}) \\ \hat{\theta}_5 &= \text{Atan2}(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}) & \hat{\theta}_5 &= \text{Atan2}(-\sqrt{r_{13}^2 + r_{23}^2}, r_{33}) \\ \hat{\theta}_6 &= \text{Atan2}(r_{32}, -r_{31}) & \hat{\theta}_6 &= \text{Atan2}(-r_{32}, r_{31}) \end{aligned} \quad (\text{A.11})$$

$$\begin{aligned} \hat{\theta}_5 &= 0 & \hat{\theta}_5 &= \pi \\ \hat{\theta}_6(t) &= \theta_5(t-1) & \hat{\theta}_6(t) &= \theta_5(t-1) \\ \hat{\theta}_4 &= \text{Atan2}(r_{21}, r_{11}) - \hat{\theta}_6 & \hat{\theta}_4 &= \text{Atan2}(-r_{21}, -r_{11}) + \hat{\theta}_6 \end{aligned} \quad (\text{A.12})$$

In generale è possibile descrivere le due soluzioni per le ultime tre variabili di giunto attraverso la seguente espressione:

$$\hat{\theta}_i = \begin{pmatrix} \hat{\theta}_{i,1} \\ \hat{\theta}_{i,2} \\ \hat{\theta}_{i,3} \end{pmatrix} \quad i = 1, 2$$

Si considera d'interesse solo la soluzione per la quale risulta minimizzata la norma differenza della sua posizione con la posizione precedente, e quindi viene rispettata la condizione espressa dalle equazione A.13 e A.14.

$$\hat{\theta}_i(t) - \hat{\theta}_i(t-1) = \begin{pmatrix} \hat{\theta}_{i,1}(t) - \hat{\theta}_{i,1}(t-1) \\ \hat{\theta}_{i,2}(t) - \hat{\theta}_{i,2}(t-1) \\ \hat{\theta}_{i,3}(t) - \hat{\theta}_{i,3}(t-1) \end{pmatrix} \quad i = 1, 2. \quad (\text{A.13})$$

$$\hat{\theta} = \min \left\{ \left\| \hat{\theta}_1(t) - \hat{\theta}_1(t-1) \right\|, \left\| \tilde{\theta}_2(t) - \hat{\theta}_2(t-1) \right\| \right\} \quad (\text{A.14})$$

# Appendice B

## YARA

È un framework che consente lo sviluppo di applicazioni per la robotica secondo uno schema behaviour-based. Fornisce degli strumenti avanzati per lo scheduling real-time dei processi e un insieme di primitive per la comunicazione in grado di garantire la trasparenza rispetto ai meccanismi di implementazione utilizzati. È stato sviluppato presso il Dipartimento di Ingegneria dell'Informazione dell'università di Parma, con l'obiettivo di fornire agli sviluppatori gli strumenti per la realizzazione di applicazioni nell'ambito della robotica mobile[8].

I componenti del sistema robotico vengono definiti attraverso dei *moduli*, che possono rappresentare dei bahaviour specifici o un'interfaccia hardware in grado di comunicare ed interagire con gli altri moduli mediante lo scambio di dati e comandi. I moduli sono realizzati come classi C++ derivate dalla classe astratta *Module* e ad un modulo sono associate *activity* periodiche o sporadiche, che possono essere risvegliate da un evento esterno. I moduli sono identificati univocamente tramite il nome passato al costruttore, che consente al framework una loro indicizzazione e l'attivazione e disattivazione dei processi associati tramite i due metodi `turnOn()` e `turnOff()`.

Le attività sono realizzate tramite istanze della classe template *Activity*. L'esecuzione delle attività è affidata a meccanismi interni del framework. Nel caso di funzionamento periodica viene mandato in esecuzione ad intervalli regolari il codice associato all'*Activity*, mentre nel caso di attivazione aperiodica l'istante di attivazione è il risultato dell'interazione tra il modulo e gli altri componenti del sistema.

All'atto della dichiarazione dell'attività viene passato al costruttore un parametro che rappresenta rispettivamente il periodo di attivazione espresso in secondi per le attività periodiche, ed il tempo minimo che deve intercorrere fra due attivazioni consecutive per le attività aperiodiche. Per entrambi i tipi di attivazione questo parametro ha anche il significato di deadline (l'istante temporale entro cui si deve concludere l'esecuzione dell'attività per poter essere considerata corretta). Per le attività periodiche, la loro attivazione e sospensione può essere invocata direttamente tramite i metodi `startPeriodicRun()` e `stopPeriodicRun()`; per quelle aperiodiche invece, ci si può limitare ad agganciarne l'attivazione ad uno specifico evento del sistema.

La cooperazione fra i componenti del sistema si fonda sul fatto che i moduli, trovandosi nello stesso spazio di indirizzamento, possono comunicare fra loro mediante accesso concorrente a strutture dati condivise. Per evitare allo sviluppatore le difficoltà legate all'uso esplicito della memoria condivisa e della sincronizzazione, il framework è dotato di una serie di primitive ad alto livello per la comunicazione fra i moduli che nascondono la complessità ed i dettagli utilizzati internamente per la loro realizzazione. Lo scambio reciproco di dati tra i moduli è consentito dal pattern *Information*. Per la ricezione o l'invio di un dato, il modulo deve dichiarare come propri *data members* uno o più oggetti di tipo *Receiver* e *Sender*. Essi sono classi template e permettono quindi di specificare il tipo di dato da scambiare, e ogni oggetto è identificato da un nome univoco in modo da poter essere identificato dal framework. Affinchè due moduli si possano scambiare un dato essi devono registrarsi rispettivamente come *Sender* e *Receiver* del dato in questione. Il pattern *Information* consente anche comunicazioni di tipo broadcast: se infatti un solo modulo può registrarsi come *Sender* di un dato, più moduli possono invece registrarsi come *Receiver* del dato, consentendo in questo modo l'invio contemporaneo di un dato a più moduli.

L'esecuzione dinamica del codice associato alle Activity è realizzato attraverso il framework TODS (Timed Object for Distributed Systems), che consente la realizzazione dello scheduling real time di task periodici ed aperiodici. TODS opera assegnando ad ogni task una thread di sistema, che viene schedulata dallo scheduler FIFO standard di Linux. Per l'assegnamento dinamico delle priorità a ciascuna thread in esecuzione viene applicata una politica di tipo EDF (Earliest Deadline First).

## YARA

---

La politica di scheduling EDF consiste nello schedulare ad ogni istante il task con la deadline più vicina, ed assegnare alle thread rimanenti una priorità decrescente con l'allontanarsi delle rispettive deadline.

# Bibliografia

- [1] Exact Dynamics. <http://www.exactdynamics.nl>. BV Bouriciusstraat 3, NL-6814 CS, Arnhem, The NETHERLANDS.
- [2] M.W. Corroppo. Implementazione di un sistema di controllo a giunti indipendenti per un manipolatore antropomorfo. Tesi di laurea specialistica in Ingegneria Informatica, Università degli Studi di Parma, 2006.
- [3] Philips Semiconductor. Sja1000: Stand-alone can controller, 2000.
- [4] Plx technology. <http://www.plxtech.com>.
- [5] Terratec cinergy. <http://www.terratec.it>.
- [6] Hirokazu Kato and Mark Billingham. Artookit-<http://www.hitl.washington.edu/artoolkit/>.
- [7] N. Otsu. A threshold selection method from gray-level histogram. IEEE Transactions on Systems, 1979.
- [8] F. Monica. Progettazione di un'architettura modulare , aperta ed in tempo reale per un robot mobile. Tesi di Laurea in Ingegneria Elettronica, Università degli Studi di Parma, 2003.
- [9] C. Guarino Lo Bianco. Cinematica dei manipolatori. Pitagora Editrice Bologna, 2003.
- [10] John J. Craig. Introduction to robotics mechanics and control. Addison-Wesley Publishing Company, 1986.
- [11] C. Bonivento, C.Melchiorri, and R. Zanasi. Sistemi di controllo digitale. Progetto Leonardo Bologna, 2004.
- [12] L. Sciavico and B.Siciliano. Modeling and control of robot manipulator. Pitagora Editrice Bologna, 2003.
- [13] G. Ferrari. Realizzazione di un'architettura software di governo per un robot manipolatore per compiti di assistenza. Tesi di laurea in Ingegneria Informatica, Università degli Studi di Parma, 2005.
- [14] L. Marchini. Sviluppo di un modulo software per la soluzione della cinematica di un manipolatore manus. Tesi di laurea in Ingegneria Informatica, Università degli Studi di Parma, 2005.
- [15] P. Ochi. Progettazione e realizzazione del sistema di controllo di un robot manipolatore per compiti di assistenza. Tesi di laurea in Ingegneria Elettronica, Università degli Studi di Parma, 2004.
- [16] Exact Dynamics. The arm users' manual, 2002.

- [17] Roboop. A robotic object oriented package in c++. <http://www.cours.polymtl.ca/roboop>.
- [18] Community Research and Development Information Service. <http://cordis.europa.eu>.
- [19] European Research Project MATS using Robotics for elderly and disabled people. <http://www.bcdi.be/mats/>.
- [20] Stime anticipatorie dei principali indicatori demografici del 2007. <http://demo.istat.it/altridati/indicatori/index.html>, Febbraio 2008.
- [21] Mark J. Kilgard. The opengl utility toolkit glut programming interface.
- [22] Mounir Mokhtari Mossaab Hariz, Stephane Renouard. Designing multimodal interaction for assistive robotic arm. IEEE International Conference on Rehabilitation Robotics, June 2007.

# Ringraziamenti

Vorrei ringraziare la mia famiglia, gli amici e tutte quelle persone che mi hanno permesso di vivere fino ad oggi credendo che

“...e se c’è un segreto  
è fare tutto come  
se vedessi solo il sole...”

da Qualcosa che non c’è di Elisa