Università degli Studi di Pisa
Facoltà di Ingegneria

Centro Interdipartimentale di Ricerca
"Enrico Piaggio"

TESI DI LAUREA IN INGEGNERIA INFORMATICA

Pianificazione e controllo in tempo reale per la presa di oggetti mobili mediante un dispositivo di lancio controllato e recupero

RELATORI: **Ch.mo Prof. Ing. A. Bicchi**
Ch.mo Prof. Ing. A. Balestrino
Ing. G. Tonietti

CANDIDATO: **Adriano Fagiolini**

Anno Accademico 2003/2004

Dedicato

A Mamma Amalia e Babbo Moreno
per l'inesauribile appoggio che mi danno
e la fiducia che ripongono in me.
Cercherò di ricambiare i vostri sacrifici.

A mio Fratello Davide
che nella sua realtà è la mia forza e
nella sua spontaneità un esempio di vita.

Ad Ilaria
per quel dolce e tenero amore quale è
e per il sostegno che mi ha dato
durante il lungo cammino di Ingegneria.

A Zio Franco e Zia Brunilde
per i loro consigli, la lealtà e
il bene che mi hanno dimostrato.

A tutti gli amici, colleghi di studio e non,
con cui ho condiviso le soddisfazioni
e le preoccupazioni di questi anni.

Indice

Prefazione

Questo documento presenta il lavoro svolto dall'autore come tesi di laurea presso il Centro Interdipartimentale di Ricerca Enrico Piaggio della Facoltà di Ingegneria di Pisa. L'obiettivo di tale tesi consiste nella pianificazione e nel controllo in tempo reale di un dispositivo di lancio e recupero, finalizzati alla presa di un oggetto in movimento. Per la rilevazione della posizione di tale oggetto si fa uso di un sistema di visione. Il lavoro è stato sviluppato sotto la supervisione del Dottor Hitoshi Arisumi e si configura come un'estensione dei suoi precedenti risultati sulla Casting Manipulation, ottenuti presso il National Institute of Advanced Industrial Science and Technology (AIST) di Tokyo.

In questo documento, si è scelto di adottare la traduzione Manipolazione del Lancio al posto di Casting Manipulation, al fine di rendere l'esposizione dell'argomento in lingua italiana più fluida.

Ogni capitolo si apre con un sommario nel quale viene brevemente spiegato l'argomento che sarà trattato nelle pagine successive. Il capitolo (1) introduce l'obiettivo di questo lavoro di tesi ed elenca le motivazioni che hanno spinto all'utilizzo del Casting Manipulator al posto di sistemi meccanici più "classici"; inoltre definisce la pianificazione ed il controllo del dispositivo scelto, finalizzati alla presa di un oggetto fermo. Nel capitolo (2) la trattazione

viene poi estesa all'ipotesi di un oggetto in movimento. Nei capitoli (3) e (4) si spiega rispettivamente il funzionamento del sistema di visione realizzato e la libreria software di simulazione e di controllo del robot. Proseguendo con il capitolo (5), troviamo una descrizione dettagliata del manipolatore costruito in laboratorio, allo scopo di verificare i risultati teorici presentati. Concludiamo con il capitolo (6) dove vengono inserite alcune considerazioni conclusive sul lavoro svolto correlate ad eventuali sviluppi futuri. Infine, nell'appendice (A), è riportata una guida di riferimento alle classi *C++* realizzate riguardanti l'acquisizione del flusso video, la rilevazione della posizione del bersaglio, la pianificazione e il controllo del robot e, per ultimo, la lettura e la scrittura diretta del hardware impiegato.

Preface

This document presents the work developed by the author as his final dissertation at the Interdepartmental Centre of Research Enrico Piaggio of the Faculty of Engineering of Pisa (Italy). The aim of the thesis is to realise a real-time planning and control of a particular manipulator in order to catch and retrieve a moving object, by using a vision system to detect the position of the object itself. The work has been done with the supervision of Doctor Hitoshi Arisumi and may be seen as an extension of his previous results on Casting Manipulation obtained at the National Institute of Advanced Industrial Science and Technology (AIST) of Tokyo (Japan).

Every chapter opens with a short summary where the topic to be analysed is briefly introduced. Chapter (1) presents the aim of the present work and outlines the reasons why it was decided to use a Casting Manipulator instead of a "classic" mechanical system; it also explains the planning and control of such a manipulator, referring to the case of a fixed object. In chapter (2) the description is then extended to the case of a moving object. Chapter (3) presents the operation of the vision system realised and then, chapter (4) describes the planning and control library developed. In chapter (5), it is possible to find a detailed description of the robot built in the laboratory in order to check the theoretical results. Chapter (6) includes some closing considerations about the work made and draws up a list of possible future

enhancements. Finally, in appendix (A), it is possible to find a reference guide of the *C++* classes implemented, regarding the video stream acquisition, the detection of the position of the target, the planning and control of the manipulator and the direct reading and writing to the hardware employed.

Capitolo 1

Presa e recupero di un oggetto fermo

Il primo passo, nella realizzazione di un sistema di presa e recupero di oggetti in movimento, consiste nella scelta del dispositivo meccanico da utilizzare. A tal scopo, come verrà chiarito di seguito, abbiamo deciso di impiegare un manipolatore del lancio. Il secondo passo consiste invece nella realizzazione della pianificazione e del controllo di tale meccanismo.

Nel primo paragrafo del presente capitolo indichiamo le motivazioni che hanno spinto all'uso di un "casting manipulator" al posto di una struttura meccanica "classica". Per semplificare poi la trattazione, il resto del capitolo è dedicato alla pianificazione e al controllo finalizzati alla presa e al recupero di un bersaglio fermo. Il secondo paragrafo descrive la tecnica di Manipolazione del Lancio nel caso più generale, indicandone le fasi di funzionamento dal punto di vista concettuale; il terzo ed ultimo paragrafo descrive poi suddetta tecnica più in dettaglio facendo riferimento al più semplice manipolatore della categoria.

1.1 Perché si usa un "casting manipulator"

Una delle specifiche più importanti di un manipolatore è l'ampiezza del suo *spazio di lavoro*. Quella di realizzare robot in grado di compiere operazioni di *pick and place* con il più ampio workspace è infatti un'esigenza molto sentita in ambiente industriale.

Una volta scelta la struttura di base di un manipolatore, è possibile allargarne lo spazio di lavoro in uno dei seguenti modi:

- ▷ aumentando la lunghezza dei bracci esistenti;
- ▷ aggiungendo ulteriori bracci alla catena cinematica;
- ▷ dotando il robot di una piattaforma mobile.

Sebbene le soluzioni proposte riescano ad ampliare lo spazio di lavoro di un robot, esse comportano i seguenti inconvenienti:

- ▷ l'aumento della lunghezza dei bracci appesantisce la struttura del robot ed, in generale, ne degrada le prestazioni;
- ▷ l'introduzione di un ulteriore braccio richiede l'uso di altri sensori e complica il controllo del robot;
- ▷ infine, l'utilizzo di una piattaforma mobile è una soluzione praticabile quando si conosce la natura del terreno su cui il manipolatore andrà a muoversi; in taluni casi, la presenza di una depressione o di un avvallamento impedisce al robot di proseguire.

In genere, al progettista è richiesto di realizzare una struttura semplice, dai costi contenuti, robusta, a basso consumo e di facile controllo. Deve pertanto trovare un compromesso tra esigenze contrastanti.

La tecnica della Manipolazione del Lancio è stata sviluppata proprio per superare i limiti delle strutture robotiche "classiche", ad esempio, in operazioni di soccorso e di recupero in ambienti disastrati. Essa si ispira all'arte della pesca: per raggiungere una determinata posizione, il pescatore fa inizialmente oscillare la canna per poi eseguire un lancio nella direzione desiderata; durante il volo dell'amo, egli ne corregge la traiettoria frenando il filo per piccoli intervalli di tempo. In questa maniera il pescatore riesce a raggiungere distanze notevoli, sebbene stia utilizzando un dispositivo molto semplice.

L'idea di base consiste, quindi, nell'utilizzo di componente cedevole e in grado di estendersi come il filo della canna da pesca, per realizzare l'ultimo braccio della catena cinematica del robot. In figura (1.1) è possibile vedere un manipolatore del lancio impiegato in operazioni di *pick and place*. In questa maniera, si riesce ad ottenere un manipolatore con spazio di lavoro ampio, ma di dimensioni e peso limitati.

1.2 La Manipolazione del Lancio

Nel caso più generale, un manipolatore del lancio è costituito da $n-1$ bracci rigidi, $L_1, \ldots, L_{n-1}$, attuati da altrettanti motori in grado di fornire le coppie generalizzate $\tau_1, \ldots, \tau_{n-1}$, e da un braccio cedevole ed estensibile L_n non attuato. Sull'estremità di quest'ultimo viene montato l'*end-effector* del manipolatore, nella fattispecie una pinza G opportunamente progettata in grado di afferrare l'oggetto bersaglio. Si tratta pertanto di una struttura meccanica *sotto-attuata*. In letteratura sono noti gli studi di A. De Luca [12], riguardanti la stabilizzazione di un manipolatore planare e sotto-attuato a giunti rotoidali, ed i risultati di Mark W. Spong [13], sui sistemi meccanici sotto-attuati in generale.

Il braccio L_n può essere costituito di materiale inestensibile o elastico e di

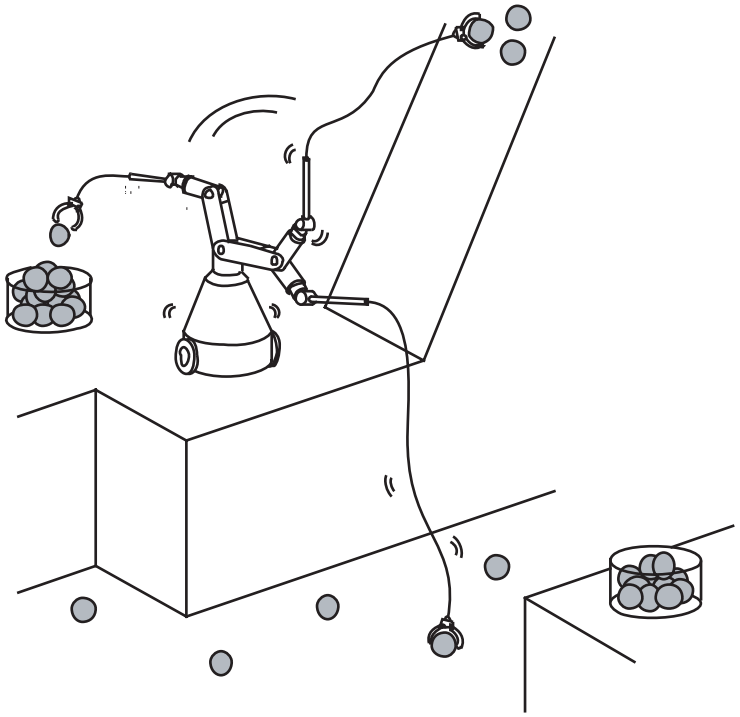

Figura 1.1: Un manipolatore del lancio utilizzato per operazioni di pick and place.

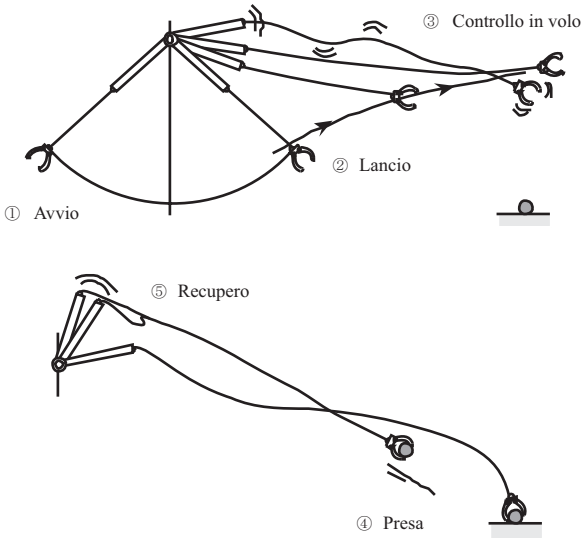

Figura 1.2: Le fasi di funzionamento di un manipolatore del lancio.

massa e inerzia trascurabili o non. In ogni caso, la presenza di un componente flessibile nella catena cinematica del manipolatore complica la pianificazione e il controllo di quest'ultimo rispetto a quelli di un manipolatore con sole parti rigide e richiede che il funzionamento avvenga sotto opportune condizioni. Come verrà chiarito a breve, si dovrà garantire che la parte flessibile rimanga sempre in tensione, almeno nella prima fase di funzionamento. Al contempo, però, è proprio la caratteristica di flessibilità del braccio L_n, unità alla sua estensibilità praticamente illimitata, che permette ad un manipolatore del lancio di superare in certe applicazioni i limiti delle strutture meccaniche esistenti e di avere uno spazio di lavoro più ampio.

1.2.1 Le fasi di funzionamento

Le fasi di funzionamento della tecnica di Manipolazione del Lancio, riportate anche in figura (1.2), possono essere schematizzate come segue. Durante la fase iniziale di *Avvio*, i primi $n-1$ giunti rigidi e attuati vengono controllati

in modo da imprimere alla pinza la velocità iniziale necessaria a raggiungere il bersaglio. In questa prima fase è importante prestare attenzione al comportamento del braccio flessibile L_n, che, per convenienza, deve rimanere sempre teso. Immagazzinata sufficiente energia, si passa alla fase di *Lancio*, durante la quale la pinza viene ancora tenuta in movimento fino ad un istante opportuno in cui viene rilasciata e lanciata nella direzione del bersaglio. Essa rimane comunque collegata alla catena cinematica, mediante il braccio flessibile L_n, che si estende durante il moto sotto l'azione dell'inerzia della pinza stessa. Senza un'ulteriore azione di controllo, la pinza si trova in volo libero e compie nello spazio un moto parabolico. Chiaramente, la massima distanza percorribile dalla pinza dipende dalla velocità iniziale. La fase che segue al lancio ha il compito di modificare la naturale traiettoria della pinza, mediante l'azione dei giunti attuati e di un dispositivo frenante, in modo che questa riesca ad afferrare il bersaglio, posizionato ad una distanza inferiore o uguale a quella massima. Questa terza fase è detta di *Controllo in volo*. La fase successiva, detta di *Presa*, può terminare con un successo o un fallimento. In ogni caso, si procede con il recupero della sola pinza oppure di quest'ultima insieme al bersaglio. Ciò avviene durante la fase di *Recupero* che prevede il riavvolgimento del braccio estensibile combinato ad un opportuno movimento dei bracci rigidi attuati.

Riassumendo, la Manipolazione del Lancio come tecnica di manipolazione robotica si compone delle seguenti fasi (figura 1.2):

- ▷ *Avvio*
- ▷ *Lancio*
- ▷ *Controllo in volo*

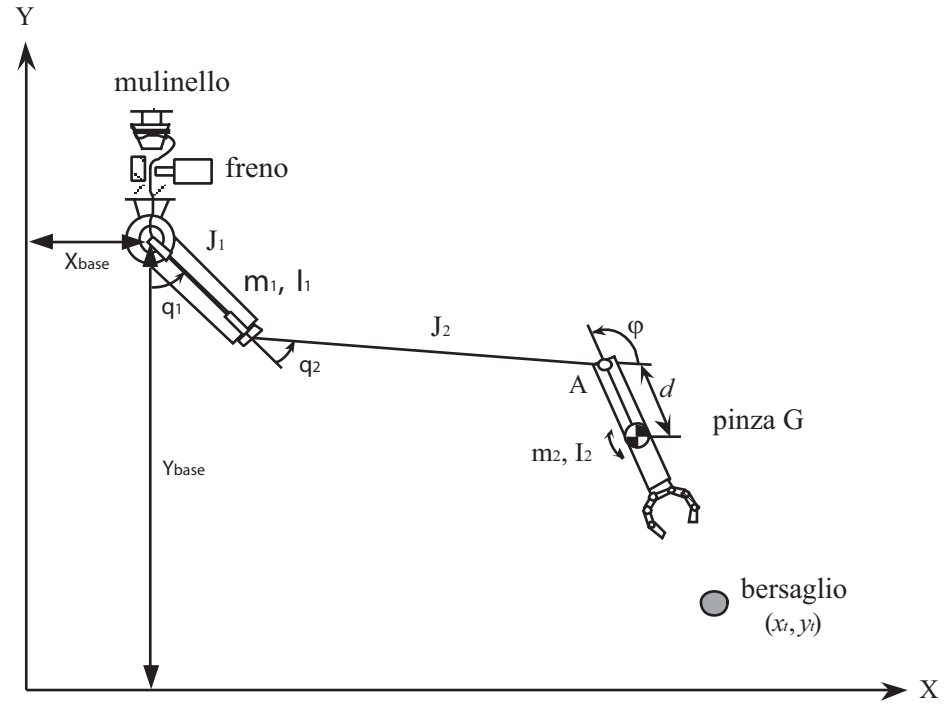

Figura 1.3: Il più semplice manipolatore del lancio costituito da un braccio rigido attuato ed un braccio flessibile ed estendibile di massa ed inerzia trascurabili. L'end-effector è costituito da una pinza utilizzare per afferrare il bersaglio.

▷ *Presa*

▷ *Recupero*

1.3 Un semplice dispositivo di lancio e recupero

Nel caso più semplice (figura 1.3), un manipolatore del lancio è costituito da un braccio rigido J_1 attuato da una coppia τ e da un braccio cedevole ed estensibile L_2 non attuato sulla cui estremità è montata una pinza G. I due giunti utilizzati sono entrambi rotoidali. Si tratta pertanto di un struttura meccanica planare, il cui piano di movimento coincide con il *piano di lancio*. Senza perdere di generalità, possiamo pensare di realizzare il braccio L_2 uti-

lizzando qualsivoglia materiale non elastico e di massa e inerzia trascurabili. In tal caso, la massa e l'inerzia di L_2 coincidono con quelle della pinza. Inoltre, la lunghezza di L_2 è pari a quella del filo sommata alla distanza d tra il punto in cui è montata la pinza e il suo centro di massa. L'estensione del braccio L_2 è inizialmente avvolta su di un mulinello montato alla base del manipolatore. Durante il lancio esso viene fatto scorrere lungo il braccio L_1. Inoltre, il manipolatore è dotato di un sistema frenante, usato per bloccare lo srotolamento del filo prima del lancio e come meccanismo di controllo durante il lancio stesso.

Prima di passare a descrivere in dettaglio le fasi di funzionamento, resta da osservare che la scelta del filo da pesca per la realizzazione di L_2 è valida solo in quelle applicazioni dove è richiesto uno sforzo mediamente basso. A dispetto di ciò, la sua massa ed inerzia trascurabili semplificano notevolmente il controllo. Più in generale, altre applicazioni potrebbero richiedere di realizzare L_2 con un materiale più resistente, ad esempio in acciaio. In tal caso, la dinamica di L_2 non può venir trascurata e richiede ad esempio un trattamento mediante modello agli elementi finiti. E' questo uno degli aspetti che potrebbero essere oggetto di studi futuri sulla Manipolazione del Lancio.

Infine, indichiamo con x_c, y_c e φ la posizione del centro di massa della pinza e il suo orientamento rispetto all'asse x e con $\mathbf{p}_t = (x_t, y_t)^T$ la posizione del bersaglio.

1.3.1 Avvio del manipolatore

Ogni operazione di presa e recupero comincia con una fase iniziale di avvio, durante la quale il giunto attuato J_1 è controllato in modo far acquistare all'end-effector l'energia necessaria per raggiungere e afferrare il bersaglio.

Dinamica del manipolatore

Prima di poter realizzare un qualunque controllo, è necessario disporre di un modello matematico del sistema meccanico in esame che ne descriva la dinamica diretta durante questa prima fase. Tuttavia, l'individuazione di tale modello è complicata dalla presenza della parte flessibile L_2. Pertanto è necessario specificare alcune condizioni di funzionamento del sistema stesso. Ipotizzando, ad esempio, di scegliere una *strategia di avviamento* tale da far rimanere la parte flessibile sempre in tensione, la dinamica del robot è quella di un manipolatore a due giunti rotoidali e bracci rigidi soggetto alla forza peso $\mathbf{g}$.

In generale la dinamica di un manipolatore a due giunti è esprimibile nella forma

$$\tau = \mathbf{B}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})\,, \tag{1.1}$$

in cui il vettore delle variabili di giunto è definito come $\mathbf{q} = (q_1, q_2)^T$ mentre le matrici di inerzia, delle forze di Coriolis e di gravità sono definite come

$$\begin{aligned}
&\mathbf{B}(\mathbf{q}) \in \mathbb{R}^{2\times 2}\,,\\
&\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{2\times 2}\,,\\
&\mathbf{G}(\mathbf{q}) \in \mathbb{R}^{2}\,.
\end{aligned}$$

In particolare, la matrice di inerzia $\mathbf{B}(\mathbf{q})$ è data da

$$\begin{aligned}
\mathbf{B}(\mathbf{q}) &= \sum_{i=1}^{2} \left(m_i\, \mathbf{J}_p^{(l_i)^T}\, \mathbf{J}_p^{(l_i)} + \mathbf{J}_o^{(l_i)^T}\, \mathbf{I}_i\, \mathbf{J}_o^{(l_i)} \right) = \\
&= m_1\, \mathbf{J}_p^{(l_1)^T}\, \mathbf{J}_p^{(l_1)} + m_2\, \mathbf{J}_p^{(l_2)^T}\, \mathbf{J}_p^{(l_2)} + \mathbf{J}_o^{(l_1)^T}\, \mathbf{I}_1\, \mathbf{J}_o^{(l_1)} + \mathbf{J}_o^{(l_2)^T}\, \mathbf{I}_2\, \mathbf{J}_o^{(l_2)}\,,
\end{aligned}$$

dove m_i è la massa del braccio i, I_i è la sua inerzia calcolata rispetto ad un asse passante per il centro di massa, $\mathbf{J}_p^{(l_i)}$ è il suo jacobiano di posizione e

$\mathbf{J}_o^{(l_i)}$ è il suo jacobiano di orientamento. Inoltre, gli elementi della matrice $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ possono essere ottenuti usando la tecnica dei simboli di Christoffel come indicato dall'espressione

$$c_{ij} = \sum_{k=1}^{2} c_{ijk}\, \dot{q}_k = c_{ij1}\, \dot{q}_1 + c_{ij2}\, \dot{q}_2 \,,$$

in cui ogni simbolo di Christoffel è ottenuto come

$$c_{ijk} = \frac{1}{2} \left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i} \right) .$$

Infine la matrice di gravità $\mathbf{G}(\mathbf{q}) = (g_1(\mathbf{q}), g_2(\mathbf{q}))^T$ si ottiene come

$$\mathbf{G}(\mathbf{q}) = -\sum_{j=1}^{2} m_{l_j}\, \mathbf{g}^T\, \mathbf{J}_{p_i}^{(l_j)}(\mathbf{q}) \,,$$

con $\mathbf{g} = (0, -g, 0)^T$.

Andiamo adesso a sviluppare i conti e ricavare la dinamica del manipolatore in esame, facendo uso delle seguenti abbreviazioni:

$$\begin{aligned} c_i &= \cos(q_i) \,, \\ s_i &= \sin(q_i) \,, \\ c_{ij} &= \cos(q_i + q_j) \\ s_{ij} &= \sin(q_i + q_j) \,. \end{aligned}$$

Con riferimento alla figura (1.4), si fissa un sistema di assi coordinati (x, y) giacenti sul piano di lancio e si indica con $\mathbf{p}_{base} = (x_{base}, y_{base})^T$ la posizione della base del manipolatore. Inoltre, chiamiamo m_1 e I_1 rispettivamente la massa e l'inerzia, calcolata rispetto ad un asse passante per il centro di massa, del braccio J_1, l_1 la sua lunghezza e a_1 la distanza tra il centro di massa e il giunto alla base della catena cinematica. In modo analogo si definiscono le

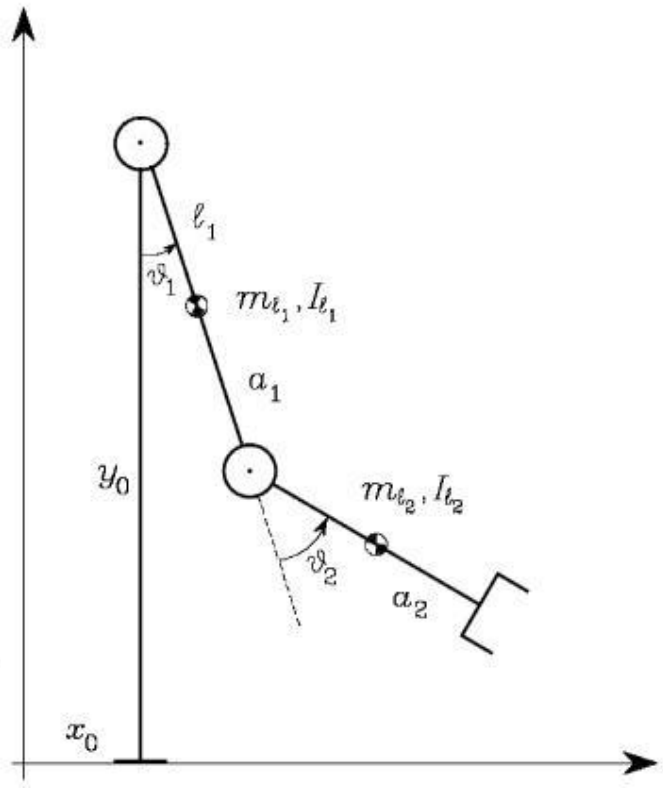

Figura 1.4: Modello di un manipolatore planare con due giunti rotoidali soggetto alla forza peso.

stesse grandezze relative al secondo braccio, ricordando che vale $l_2 = a_2$. La posizione dei centri di massa dei due bracci è data da

$$\mathbf{p}_{l_1} = \begin{pmatrix} l_1 \\ y_{base} - l_1\, c_1 \\ 0 \end{pmatrix},$$

$$\mathbf{p}_{l_2} = \begin{pmatrix} a_1\, s_1 + l_2\, s_{12} \\ y_{base} - a_1\, c_1 - l_2\, c_{12} \\ 0 \end{pmatrix}.$$

Derivando rispetto al tempo le posizioni dei centri di massa e mettendo in evidenza il vettore delle velocità delle variabili di giunto $\dot{\mathbf{q}} = (\dot{q}_1, \dot{q}_2)^T$, si ottengono le espressioni

$$\dot{\mathbf{p}}_{l_1} = \frac{d}{dt}\,\mathbf{p}_{l_1} = \begin{pmatrix} l_1\, c_1\, \dot{q}_1 \\ l_1\, s_1\, \dot{q}_1 \\ 0 \end{pmatrix} = \begin{pmatrix} l_1\, c_1 & 0 \\ l_1\, s_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix},$$

$$\dot{\mathbf{p}}_{l_2} = \frac{d}{dt}\,\mathbf{p}_{l_2} = \begin{pmatrix} a_1\, c_1 + l_2\, c_{12} & l_2\, c_{12} \\ a_1\, s_1 + l_2\, s_{12} & l_2\, s_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix},$$

che, confrontate con $\dot{\mathbf{p}}_{l_i} = \mathbf{J}_p^{(l_i)}\,\dot{\mathbf{q}}$, permettono di ricavare i due jacobiani di posizione:

$$\mathbf{J}_p^{(l_1)} = \begin{pmatrix} l_1\, c_1 & 0 \\ l_1\, s_1 & 0 \\ 0 & 0 \end{pmatrix},$$

$$\mathbf{J}_p^{(l_2)} = \begin{pmatrix} a_1\, c_1 + l_2\, c_{12} & l_2\, c_{12} \\ a_1\, s_1 + l_2\, s_{12} & l_2\, s_{12} \\ 0 & 0 \end{pmatrix}.$$

Ricordando che il filo da pesca ha massa trascurabile, abbiamo $a_2 = l_2$. Inoltre, la velocità di rotazione angolare dei due bracci è:

$$\omega_1 = \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix},$$

$$\omega_2 = \begin{pmatrix} 0 \\ 0 \\ \dot{q}_1 + \dot{q}_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix}.$$

Confrontando tali espressioni con $\omega_i = \mathbf{J}_o^{(l_i)}\,\dot{\mathbf{q}}$, si ricavano i due jacobiani di orientamento:

$$\mathbf{J}_o^{(l_1)} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} ,$$

$$\mathbf{J}_o^{(l_2)} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} .$$

Allora, gli elementi della matrice di inerzia

$$\mathbf{B}(\mathbf{q}) = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} ,$$

sono dati da

$$\begin{aligned} b_{11} &= m_1\, l_1^2 + m_2\, \left(a_1^2 + l_2^2 + 2\, a_1\, l_2\, c_2\right) + I_1 + I_2\,, \\ b_{12} &= I_2 + m_2\, \left(l_2^2 + a_1\, l_2\, c_2\right)\,, \\ b_{21} &= b_{12}\,, \\ b_{22} &= I_2 + m_2\, l_2^2\,. \end{aligned}$$

Usando, come detto sopra, la tecnica dei simboli di Christoffel, si ricava che gli elementi di $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ sono dati da

$$\begin{aligned} c_{11} &= -m_2\, a_1\, l_2\, s_2\, \dot{q}_2\,, \\ c_{12} &= -m_2\, a_1\, l_2\, s_2\, \left(\dot{q}_1 + \dot{q}_2\right)\,, \\ c_{21} &= m_2\, a_1\, l_2\, s_2\, \dot{q}_1\,, \\ c_{22} &= 0\,. \end{aligned}$$

Definendo la quantità $h = -\,m_2\, a_1\, a_2\, s_2$, è possibile riscrivere $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ come

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{pmatrix} h\,\dot{q}_2 & h\,\left(\dot{q}_1 + \dot{q}_2\right) \\ -h\,\dot{q}_1 & 0 \end{pmatrix} .$$

Infine la matrice di gravità è data da

$$\mathbf{G}(\mathbf{q}) = \begin{pmatrix} m_1\, g\, l_1\, s_1 + m_2\, g\, (a_1\, s_1 + a_2\, s_{12}) \\ m_2\, g\, a_2\, s_{12} \end{pmatrix} .$$

Riassumendo i parametri della dinamica del manipolatore durante la fase di avvio sono dati dalle espressioni:

$$\begin{aligned} b_{11} &= m_1\, l_1^2 + m_2\, \left(a_1^2 + l_2^2 + 2\, a_1\, l_2\, c_2\right) + I_1 + I_2\,, \\ b_{12} &= I_2 + m_2\, \left(l_2^2 + a_1\, l_2\, c_2\right)\,, \\ b_{21} &= b_{12}\,, \\ b_{22} &= I_2 + m_2\, l_2^2\,, \end{aligned}$$

$$\begin{aligned} c_{11} &= -m_2\, a_1\, l_2\, s_2\, \dot{q}_2\,, \\ c_{12} &= -m_2\, a_1\, l_2\, s_2\, \left(\dot{q}_1 + \dot{q}_2\right)\,, \\ c_{21} &= m_2\, a_1\, l_2\, s_2\, \dot{q}_1\,, \\ c_{22} &= 0\,, \end{aligned}$$

$$\begin{aligned} g_1 &= m_1\, g\, l_1\, s_1 + m_2\, g\, \left(a_1\, s_1 + l_2\, s_{12}\right)\,, \\ g_2 &= m_2\, g\, l_2\, s_{12}\,. \end{aligned}$$

Controllo a coppia calcolata

Come tecnica di controllo del manipolatore è stato scelto di utilizzare un controllo a coppia calcolata. Tale scelta richiede che i valori dei parametri geometrico-inerziali del modello matematico siano noti con sufficiente precisione. Indichiamo con $(\mathbf{q}_d(t), \dot{\mathbf{q}}_d(t), \ddot{\mathbf{q}}_d(t))$ la traiettoria desiderata da far

inseguire al sistema e supponiamo di poterne misurare, con un insieme opportuno di sensori, la posizione $\mathbf{q}$ e la velocità $\dot{\mathbf{q}}$.

Nota che sia la dinamica diretta

$$\tau = \mathbf{B}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})\,,$$

il controllo a coppia calcolata prevede di scegliere τ come

$$\tau = \mathbf{B}(\mathbf{q})\,\ddot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{B}(\mathbf{q})\,\mathbf{u}\,,$$

introducendo il nuovo ingresso $\mathbf{u} \in \mathbb{R}^2$. Premoltiplicando la dinamica per $\mathbf{B}(\mathbf{q})^{-1}$ e lasciando al primo membro solo il termine dell'accelerazione, otteniamo:

$$\ddot{\mathbf{q}} = \mathbf{B}(\mathbf{q})^{-1}\,\left(\tau - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})\right)\,.$$

Con la teorica scelta di τ, si ottiene $\ddot{\mathbf{q}} = \mathbf{u}$, ovvero il sistema meccanico visto dal nuovo ingresso $\mathbf{u}$ si comporta come un doppio integratore.

In realtà, trattandosi di un dispositivo sotto-attuato, non è possibile annullare contemporaneamente entrambe le dinamiche delle variabili di giunto. Infatti, con un solo ingresso τ possiamo soltanto fare in modo che una delle due dinamiche coincida con quella di un doppio integratore, mentre l'altra risulta dipendente dalla scelta fatta. Seguendo questa strada, esplicitiamo la dinamica di $\mathbf{q}$ come segue:

$$\begin{aligned}
\ddot{q}_1 &= \frac{b_{22}}{\Delta}\,\left(\tau - c_{11}\,\dot{q}_1 - c_{12}\,\dot{q}_2 - g_1\right) + \\
&\quad + \frac{b_{12}}{\Delta}\,\left(c_{21}\,\dot{q}_1 + c_{22}\,\dot{q}_2 + g_2\right)\,, \\
\ddot{q}_2 &= -\frac{b_{21}}{\Delta}\,\left(\tau - c_{11}\,\dot{q}_1 - c_{12}\,\dot{q}_2 - g_1\right) + \\
&\quad - \frac{b_{11}}{\Delta}\,\left(c_{21}\,\dot{q}_1 + c_{22}\,\dot{q}_2 + g_2\right)\,,
\end{aligned}$$

con $\Delta = b_{11}\, b_{22} - b_{12}^2$. Supponiamo di voler far convergere asintoticamente a zero l'errore sulla seconda variabile di giunto q_2. A tal scopo, scegliamo τ pari a

$$\begin{aligned}\tau &= c_{11}\,\dot{q}_1 + c_{12}\,\dot{q}_2 + g_1 + \\ &- \frac{\Delta}{b_{21}}\left(\frac{b_{11}}{\Delta}\,(c_{21}\,\dot{q}_1 + c_{22}\,\dot{q}_2 + g_2) + u\right) = \\ &= c_{11}\,\dot{q}_1 + c_{12}\,\dot{q}_2 + g_1 + \\ &- \frac{b_{11}}{b_{21}}\,(c_{21}\,\dot{q}_1 + c_{22}\,\dot{q}_2 + g_2) - \frac{\Delta}{b_{21}}\,u\end{aligned}$$

In tal caso la dinamica del manipolatore diventa

$$\begin{cases}\ddot{q}_1 = f_1\,(q_1, q_2, u)\ , \\ \ddot{q}_2 = u\,.\end{cases} \tag{1.2}$$

Per far convergere asintoticamente a zero, l'errore sulla seconda variabile di giunto q_2, è sufficiente scegliere il nuovo ingresso $u \in \mathbb{R}$ come

$$u = \ddot{q}_{2_d} + K_v\,(\dot{q}_{2_d} - \dot{q}_2) + K_p\,(q_{2_d} - q_2\,)\ ,$$

dove le costanti K_v e K_p possono essere fissate come

$$\begin{aligned}K_v &= 2\,\sigma\,, \\ K_p &= \sigma^2\,,\end{aligned}$$

avendo scelto $\sigma > 0$. Il che corrisponde ad aver piazzato due poli coincidenti in $-\sigma$ nella dinamica dell'errore. Infatti, definendo l'errore sulla seconda variabile di giunto come $e_2 = q_{2_d} - q_2$, la sua dinamica è

$$\begin{aligned}&\ddot{q}_{2_d} = \ddot{q}_{2_d} + K_v\,(\dot{q}_{2_d} - \dot{q}_2) + K_p\,(q_{2_d} - q_2\,)\ , \\ &\ddot{e}_2 + K_v\,\dot{e}_2 + K_p\,e_2 = 0\,.\end{aligned}$$

Pianificazione della fase di avvio

La pianificazione di un manipolatore del lancio è complicata anzitutto dalla presenza di un braccio flessibile, ma anche dal fatto che il secondo giunto non è attuato. Come detto in precedenza, dal punto di vista matematico, siamo di fronte ad un sistema dinamico sotto-attuato, con due variabili q_1 e q_2 da controllare, ma con un solo ingresso di controllo τ. Pertanto, non è possibile controllare una delle due variabili senza che l'altra sia influenzata dalla scelta della politica di controllo. Si tratta di una situazione che si presenta regolarmente in caso di manipolatori sotto-attuati. Decidiamo allora di non contrastare l'effetto della dinamica di una variabile sull'altra, ma di utilizzarlo a nostro vantaggio. Una volta fissata la traiettoria da far inseguire al giunto q_2, è determinato anche il controllo del primo giunto. Inoltre, sfruttando l'equazione della dinamica diretta (1.2) possiamo conoscere in ogni istante lo stato completo del manipolatore.

Innanzitutto, supponiamo che il robot sia stato portato in una configurazione con $q_2 = \dot{q}_2 = \ddot{q}_2 = 0$. In queste ipotesi, e in assenza di attrito sul primo giunto, la dinamica del manipolatore si semplifica e può essere riscritta come

$$\ddot{q}_1 + a \, \sin q_1 = 0 \, ,$$

dove il coefficiente a dipende solo dai parametri geometrici e inerziali ed è dato da

$$a = \frac{m_2 \, g \, l_2}{m_2 \, (l_2^2 + l_1 \, l_2) + I_2} \, .$$

Nel piano delle fasi $(q_1, \dot{q}_1)$, la dinamica semplificata del manipolatore è rappresenta da *pseudo-ellissi* concentriche. Con riferimento alla figura (1.5), è possibile distinguere due zone di funzionamento del manipolatore: la zona

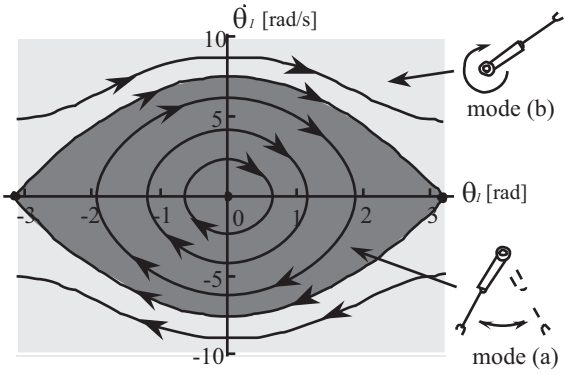

Figura 1.5: Diagramma di stato di un manipolatore del lancio in oscillazione con bracci in fase. Si riconoscono due modalità di funzionamento: rotazione in direzione costante ed oscillazione periodica.

di colore più chiaro coincide con un moto di rotazione in direzione costante, mentre la zona di colore più scuro corrisponde ad un'oscillazione periodica del sistema. Per motivi di più facile realizzabilità in laboratorio, decidiamo di sfruttare la modalità di *funzionamento in oscillazione*, concentrandoci soltanto sulla parte più interna del diagramma delle fasi.

Osservando tale grafico, ci si accorge immediatamente di un primo problema di avvio: la configurazione iniziale del manipolatore prevede $q_1 = \dot{q}_1 = 0$ e la corrispondente traiettoria coincide con l'origine. Pertanto, l'unica evoluzione possibile è quella che partendo dall'origine vi rimane indefinitamente.

Partendo invece da un generico punto del tipo $q_1 = q_{1_{avvio}}$ e $\dot{q}_1 = 0$, il robot continua a muoversi indefinitamente sulla pseudo-ellisse su cui si trova. Tuttavia, un secondo problema è rappresentato dagli attriti dell'aria e sul giunto J_1 che, col passare del tempo, smorzerebbero l'ampiezza di oscillazione del primo giunto fino a portarlo nell'origine.

I due problemi possono essere risolti individuando una strategia in grado

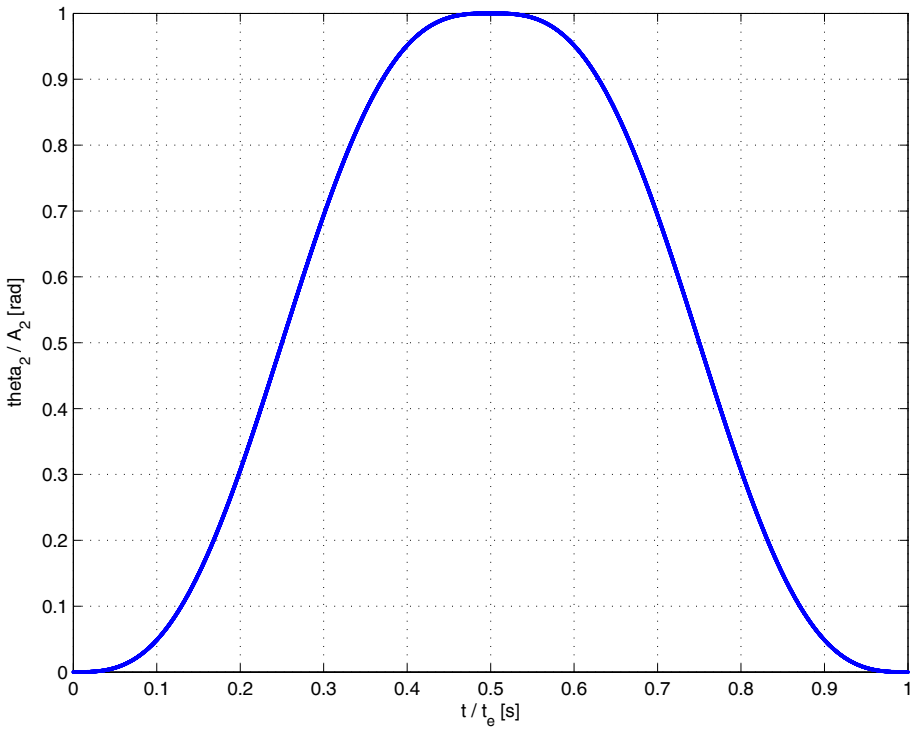

Figura 1.6: Traiettoria cicloidale da far inseguire al secondo giunto q_2, per eccitare o smorzare l'oscillazione del primo giunto q_1.

di eccitare o smorzare l'oscillazione del primo giunto, facendo saltare da una pseudo-ellissi all'altra. Il metodo proposto in [4] sfrutta pienamente l'effetto della dinamica di una variabile di giunto sull'altra, come abbiamo detto prima. In particolare il manipolatore viene controllato in modo da far inseguire alla seconda variabile di giunto una *traiettoria cicloidale* (figura 1.6) descritta da

$$\begin{cases} q_{2_d} = A_2 \left(\frac{2\,t}{t_e} - \frac{1}{2\pi} \sin\left(\frac{4\pi\,t}{t_e} \right) \right) & \text{per } t \in \left[0, \frac{t_e}{2}\right], \\ q_{2_d} = A_2 \left(-\frac{2\,t}{t_e} + \frac{1}{2\pi} \sin\left(\frac{4\pi\,t}{t_e} \right) \right) + 2\,A_2 & \text{per } t \in \left[\frac{t_e}{2}, t_e\right]. \end{cases} \tag{1.3}$$

Il periodo di oscillazione t_e viene scelto in modo che sia $t_e \in [0, T]$, dove T è il periodo di oscillazione di un pendolo

$$T = 2\,\pi \sqrt{\frac{l_1 + l_2}{g}}\,.$$

Supponiamo di voler stabilizzare il moto di oscillazione del primo giunto sulla pseudo-ellisse di ampiezza massima $q_{1_{max}}$. Risulta cruciale la scelta del coefficiente A_2 che rappresenta la massima ampiezza dell'oscillazione sul secondo giunto. Si può dimostrare [4] che per q_1 sull'orbita con ampiezza massima 80°, basta scegliere la sequenza $A_2 = 30, -45, 50, -75$. Una volta che l'ampiezza massima sul primo giunto è molto vicina a quella desiderata, il controllo commuta: da questo momento l'ampiezza A_2 è scelta in modo da annullare l'errore sul primo giunto. In particolare abbiamo

$$A_2 = \pm K_c \left(A_{1_{max_d}} - A_{1_{max}} \right) .$$

in cui $A_{1_{max_d}}$ rappresenta la massima ampiezza desiderata, mentre $A_{1_{max}}$ quella attualmente ottenuta. Infine si può ricavare che la scelta migliore per la costante moltiplicativa è $K_c = 1.14$.

In simulazione è possibile vedere l'andamento delle variabili di giunto e riconoscere la commutazione del controllo, come sarà mostrato nel capitolo (4).

Dinamica della pinza

Durante la fase iniziale di avvio/oscillazione, la posizione (x_c, y_c) del centro di massa della pinza e il suo orientamento φ rispetto all'asse x sono legati alle variabili di giunto q_1 e q_2 e alle sue derivate prime dalle relazioni:

$$\begin{cases} x_c = x_{base} + a_1\, s_1 + a_2\, s_{12} \,, \\ y_c = y_{base} - a_1\, c_1 - a_2\, c_{12} \,, \\ \varphi = q_1 + q_2 + \dfrac{\pi}{2} \,. \end{cases} \tag{1.4}$$

Inoltre le componenti della velocità di traslazione della pinza e la velocità di rotazione si ottengono derivando (1.4) rispetto al tempo:

$$\begin{cases} \dot{x}_c = a_1\, c_1\, \dot{q}_1 + a_2\, c_{12}\, (\dot{q}_1 + \dot{q}_2)\ , \\ \dot{y}_c = a_1\, s_1\, \dot{q}_1 + a_2\, s_{12}\, (\dot{q}_1 + \dot{q}_2)\ , \\ \dot{\varphi} = \dot{q}_1 + \dot{q}_2\ . \end{cases} \tag{1.5}$$

1.3.2 Decisione dell'istante di lancio

Una volta che l'end-effector ha un'energia sufficiente a raggiungere e afferrare il bersaglio, la fase di avvio si ritiene conclusa. Finita tale fase, il manipolatore viene ancora controllato in modo da non far perdere energia all'end-effector fino all'istante in cui non è verificata la *condizione di lancio*, istante nel quale esso viene lanciato. Da questo momento in poi, il braccio L_2 comincia ad estendersi trainato dall'end-effector. La scelta della condizione di lancio e, di conseguenza, dell'istante di lancio dipende da più fattori tra cui la distanza che l'end-effector deve percorre per raggiungere il bersaglio, la presenza di ostacoli nel cammino, ecc.

Risulta poi importante conoscere lo stato dell'end-effector al momento del lancio, poiché questo rappresenta lo stato iniziale $\mathbf{S}_0$ del volo e da questo dipende tutta la traiettoria e il controllo della fase successiva.

Nel nostro caso, la fase di avvio si conclude non appena l'oscillazione del braccio L_1 avviene da $-\theta_{1_{max}}$ a $\theta_{1_{max}}$ e quella del braccio L_2 è molto piccola. Da questo momento la pinza viene ancora fatta oscillare fino al momento in cui risulta

$$q_1 = q_{1_{lancio}}\ .$$

Più in generale, la condizione di lancio potrebbe essere $q_1 = f(\dots)$. Per ricavarsi lo stato della pinza al momento del lancio, si consideri innanzitutto

la configurazione in cui si viene a trovare il manipolatore. Si può dimostrare che essa vale:

$$\begin{cases} q_1 = q_{1_{lancio}}\,, \\ \dot{q}_1 = \sqrt{2\,a\;(cos(q_{1_{lancio}} - cos(q_{1_{max}}))}\,, \\ q_2 \simeq 0\,, \\ \dot{q}_2 \simeq 0\,, \end{cases}$$

dove $a = \frac{m_2\, g\, l_2}{m_2\left(l_2^2 + l_1\, l_2\right) + I_2}$. La posizione del centro di massa della pinza e il suo orientamento rispetto all'asse x sono dati da

$$\begin{cases} x_{c_0} \simeq x_{base} + (a_1 + a_2)\;\sin(q_{1_{lancio}})\,, \\ y_{c_0} \simeq y_{base} - (a_1 + a_2)\;\cos(q_{1_{lancio}})\,, \\ \varphi_0 \simeq q_{1_{lancio}} + \dfrac{\pi}{2}\,, \end{cases}$$

mentre le componenti della velocità di traslazione della pinza e la velocità di rotazione, ottenute per derivazione rispetto al tempo dalle precedenti espressioni, sono:

$$\begin{cases} \dot{x}_{c_0} \simeq a_1\, c_1\, \dot{q}_1 + a_2\, c_{12}\;(\dot{q}_1 + \dot{q}_2)\;, \\ \dot{y}_{c_0} \simeq a_1\, s_1\, \dot{q}_1 + a_2\, s_{12}\;(\dot{q}_1 + \dot{q}_2)\;, \\ \dot{\varphi}_0 \simeq \dot{q}_1\,. \end{cases}$$

Lo stato iniziale $\mathbf{S}_0$ della pinza al momento del lancio è dato quindi da

$$\mathbf{S}_0 = (x_{c_0}, y_{c_0}, \varphi_0, \dot{x_{c_0}}, \dot{y_{c_0}}, \dot{\varphi}_0)\,.$$

1.3.3 Controllo in volo

Ad eccezione di qualche caso fortuito, la traiettoria che l'end-effector percorre, partendo dallo stato iniziale $\mathbf{S}_0$ e soggetta alle forze fisiche dell'ambiente circostante, non è adatta a raggiungere ed afferrare il bersaglio. Pertanto, durante la fase di controllo in volo, il robot agisce sulla dinamica dell'end-effector allo scopo di correggerne la traiettoria, mediante l'uso del dispositivo frenante combinato al movimento dei bracci rigidi attutati.

Una volta definito il tipo di controllo in volo che si desidera utilizzare, il vero e proprio controllo è univocamente determinato dalla scelta di un insieme opportuno di n parametri, che inseriamo nel *vettore di controllo in volo* $\mathbf{b}^n$.

E' da osservare che, durante la fase di controllo in volo, l'end-effector si può portare in una configurazione dalla quale non è più possibile raggiungere e afferrare il bersaglio, poiché il tipo di controllo in volo scelto non lo permette. Tutto ciò è dovuto alla presenza del giunto flessibile J_n con il quale non è possibile spingere ma solo frenare l'end-effector. In tal caso la fase di controllo termina.

In generale, lo stato finale $\mathbf{S}_f$ nel quale si porta l'end-effector alla fine della fase di controllo in volo può essere visto come una funzione dello stato iniziale $\mathbf{S}_0$ e del vettore di controllo in volo $\mathbf{b}^n$, cioè risulta

$$\mathbf{S}_f = f(\mathbf{S}_0, \mathbf{b}^n)\,.$$

Nel caso specifico del manipolatore in esame, il robot dispone di un freno posto alla base della catena cinematica. Il controllo in volo $\mathbf{b}^n$ consiste in una serie n di frenate applicate agli istanti t_{b_i} e di durata Δ_{b_i}. Tale tecnica di controllo è detta *a frenata multipla* ed è, in base a quanto detto sopra, univocamente determinata dal vettore

$$\mathbf{b}^n = (t_{b_1},\, \Delta_{b_1},\, \ldots,\, t_{b_n},\, \Delta_{b_n})\ .$$

Ricerca del vettore di controllo in volo

Come abbiamo detto sopra, durante la fase di volo, il manipolatore agisce sull'end-effector, correggendone la traiettoria, in modo che questo posta raggiungere ed afferrare il bersaglio. Ciò equivale a chiedere che lo stato finale $\mathbf{S}_f = \left(x_{c_f}, y_{c_f}, \varphi_f, \dot{x}_{c_f}, \dot{y}_{c_f}, \dot{\varphi}_f\right)^T$ dell'end-effector coincida con uno *stato desiderato* $\mathbf{S}_d = (x_{c_d}, y_{c_d}, \varphi_d, \dot{x}_{c_d}, \dot{y}_{c_d}, \dot{\varphi}_d)^T$. Una parte del programma di controllo

deve quindi trovare un vettore di controllo $\mathbf{b}_*^n$ tale per cui risulta

$$\mathbf{S}_f\left(\mathbf{b}_*^n\right) = \mathbf{S}_d\,.$$

Questo è il compito dell'*algoritmo di pianificazione del controllo in volo.*

La ricerca di una soluzione analitica al problema non è un approccio facilmente praticabile, pertanto si è scelto di ricorrere ad una soluzione approssimata di tipo numerico. A tal scopo, il problema è stato riformulato in termini di ricerca del minimo di un funzionale obiettivo $\mathcal{D}$. Tale funzionale rappresenta la distanza opportunamente pesata tra la configurazione finale $\mathbf{S}_f$ ottenuta con i parametri di controllo $\mathbf{b}^n$ e quella desiderata $\mathbf{S}_f$. Essa vale

$$\begin{aligned}\mathcal{D} &= dist_{\mathbf{W}}\left(\mathbf{S}_f, \mathbf{S}_d\right) = \\ &= w_x\,\left(x_{c_f} - x_{c_d}\right)^2 + w_y\,\left(y_{c_f} - y_{c_d}\right)^2 + w_\varphi\,\left(\varphi_f - \varphi_d\right)^2 + \\ &+ w_{\dot{x}}\,\left(\dot{x}_{c_f} - \dot{x}_{c_d}\right)^2 + w_{\dot{y}}\,\left(\dot{y}_{c_f} - \dot{y}_{c_d}\right)^2 + w_{\dot{\varphi}}\,\left(\dot{\varphi}_f - \dot{\varphi}_d\right)^2\,,\end{aligned}$$

dove $\mathbf{W} = diag\left(w_x, w_y, w_\varphi, w_{\dot{x}}, w_{\dot{y}}, w_{\dot{\varphi}}\right)$ è la matrice dei pesi. Il vettore di controllo in volo $\mathbf{b}_*^n$ cercato è dunque la soluzione del problema

$$\mathbf{b}_*^n = \arg\min_{\mathbf{b}^n \in \mathbf{B}} \mathcal{D}\,,$$

dove abbiamo indicato con $\mathbf{B}$ il dominio di tutti i controlli ammissibili. Infine, fra i vari metodi di ricerca iterativa della soluzione, abbiamo adottato i due seguenti appartenenti alla classe del gradiente:

- ▷ DFS (Davidon - Fletcher - Powell),
- ▷ BFGS (Broyden - Fletcher - Goldfarb - Shanno).

Tale scelta è giustificata dalla rapidità di convergenza di entrambi gli algoritmi, il cui tempo di esecuzione, infatti, risulterà un parametro essenziale per

la ricerca di un vettore di controllo $\mathbf{b}_*^n$ finalizzato alla presa di un bersaglio in movimento. In realtà, nel caso di un bersaglio fermo, il controllo in volo ottimo $\mathbf{b}_*^n$ può essere calcolato off-line.

Dinamica della pinza in volo ed effetto delle frenate

L'evoluzione della pinza durante l'intero volo può essere descritta nei termini di un sistema ibrido: partendo dallo stato iniziale $\mathbf{S}_0$, la pinza si muove nello spazio inizialmente soggetta alla sola forza peso (*volo libero*); ad un certo istante, il giunto flessibile viene frenato e la dinamica della pinza cambia tenendo conto anche della forza applicata dal filo (*volo in frenata*); alla fine della frenata, la pinza torna a muoversi in volo libero fino al successivo istante di frenata.

Indichiamo con t_{b_i} il tempo trascorso dalla frenata con numero d'ordine $i-1$. Trascurando la forza che il filo applica sulla pinza, forza dovuta alla resistenza di srotolamento del filo e all'attrito che il filo incontra lungo il braccio J_1, e l'attrito dell'aria, la dinamica della pinza in volo libero è caratterizzata dalle equazioni

$$\begin{cases} x_c(t_{b_i}) = x_{c_{i-1}} + \dot{x}_{c_{i-1}}\, t_{b_i}\,, \\ y_c(t_{b_i}) = y_{c_{i-1}} + \dot{y}_{c_{i-1}}\, t_{b_i} - \dfrac{1}{2}\, g\, t_{b_i}^2\,, \\ \varphi(t_{b_i}) = \varphi_{i-1} + \dot{\varphi}_{i-1}\, t_{b_i}\,, \\ \\ \dot{x}_c(t_{b_i}) = \dot{x}_{c_{i-1}}\,, \\ \dot{y}_c(t_{b_i}) = \dot{y}_{c_{i-1}} - g\, t_{b_i}\,, \\ \dot{\varphi}(t_{b_i}) = \dot{\varphi}_{i-1}\,, \end{cases}$$

in cui $x_{c_{i-1}}$, $y_{c_{i-1}}$, $\varphi_{c_{i-1}}$, $\dot{x}_{c_{i-1}}$, $\dot{y}_{c_{i-1}}$ e $\dot{\varphi}_{c_{i-1}}$ sono lo stato della pinza immediatamente dopo la frenata $i-1$.

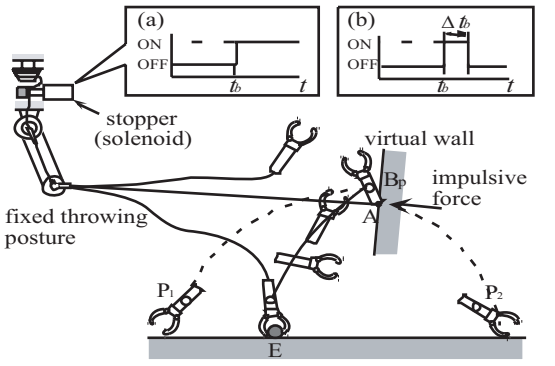

Figura 1.7: Urto della pinza contro il muro virtuale.

Si può poi dimostrare che l'effetto di una frenata di durata Δ_{b_i}, per $\Delta_{b_i} <$ 100 msec, sulla dinamica della pinza equivale all'urto di quest'ultima contro un *muro virtuale* (figura 1.7). A tal proposito, si introduce un *coefficiente di rimbalzo* dell'urto e_{b_i} che è in corrispondenza biunivoca con la durata dell'impulso di frenata Δ_{b_i}. In altre parole, una scelta del vettore di controllo in volo del tutto equivalente a quella fatta è

$$\mathbf{b}^n = (t_{b_1},\, e_{b_1},\, \ldots,\, t_{b_n},\, e_{b_n})\ .$$

Inoltre, si può dimostrare che ogni frenata non modifica le componenti di posizione e orientamento della pinza, ma solo le rispettive derivate temporali. Quindi l'effetto di una frenata può essere riassunto come segue:

$$\begin{cases} x_c(t_{b_i}) = x_{c_{i-1}} + \dot{x}_{c_{i-1}}\, t_{b_i}\ , \\ y_c(t_{b_i}) = y_{c_{i-1}} + \dot{y}_{c_{i-1}}\, t_{b_i} - \dfrac{1}{2}\, g\, t_{b_i}^2\ , \\ \varphi(t_{b_i}) = \varphi_{i-1} + \dot{\varphi}_{i-1}\, t_{b_i}\ , \\ \\ \dot{x}_c(t_{b_i}) = \dot{x}_{c_{i-1}}\ , \\ \dot{y}_c(t_{b_i}) = \dot{y}_{c_{i-1}} - g\, t_{b_i}\ , \\ \dot{\varphi}(t_{b_i}) = \dot{\varphi}_{i-1}\ , \end{cases}$$

In conclusione, lo stato $\mathbf{S}_f$ della pinza alla fine del volo è dato da

$$\begin{cases} x_{c_f} = f_x(\mathbf{S}_0, t_{b_1}, e_{b_1}, \ldots, t_{b_n}, e_{b_n}) \,, \\ y_{c_f} = f_y(\mathbf{S}_0, t_{b_1}, e_{b_1}, \ldots, t_{b_n}, e_{b_n}) \,, \\ \varphi_f = f_\varphi(\mathbf{S}_0, t_{b_1}, e_{b_1}, \ldots, t_{b_n}, e_{b_n}) \,, \\ \\ \dot{x}_{c_f} = f_{\dot{x}}(\mathbf{S}_0, t_{b_1}, e_{b_1}, \ldots, t_{b_n}, e_{b_n}) \,, \\ \dot{y}_{c_f} = f_{\dot{y}}(\mathbf{S}_0, t_{b_1}, e_{b_1}, \ldots, t_{b_n}, e_{b_n}) \,, \\ \dot{\varphi}_f = f_{\dot{\varphi}}(\mathbf{S}_0, t_{b_1}, e_{b_1}, \ldots, t_{b_n}, e_{b_n}) \,, \end{cases}$$

ovvero in forma più compatta

$$\mathbf{S}_f = f(\mathbf{S}_0, \mathbf{b}^n) \,.$$

Pianificazione e controllo dei giunti rigidi

Dal punto di vista del controllo, il manipolatore è adesso costituito da un unico braccio attuato da una coppia τ e soggetto alla forza peso $\mathbf{g}$. La sua dinamica è dunque

$$\ddot{q}_1 = \left(I_1 + m_1\, l_1^2\right)^{-1} \left(\tau - m_1\, g\, l_1\, s_1\right) \,.$$

Sempre seguendo la tecnica di controllo a coppia pre-calcolata, scegliamo il controllo τ come

$$\tau = m_1\, g\, l_1\, s_1 + \left(I_1 + m_1\, l_1^2\right)\, u \,,$$

dove u è il nuovo ingresso di controllo. Per far convergere asintoticamente a zero l'errore sulla traiettoria del primo giunto q_1, scegliamo u dato da

$$u = \ddot{q}_{1_d} + K_v\, \left(\dot{q}_{1_d} - \dot{q}_1\right) + K_p\, \left(q_{1_d} - q_1\,\right) \,.$$

Infine, dal punto di vista della pianificazione, per realizzare in modo migliore l'effetto di rimbalzo sul muro virtuale, la posizione del braccio rigido

L_1 agli istanti di frenata t_{b_i} viene scelta in modo che tale braccio sia parallelo al filo. Pertanto, la traiettoria desiderata sul primo giunto è

$$\begin{aligned} q_{1_d} &= \tan^{-1}\left(\frac{x_{c_i}}{y_{base} - y_{c_i}}\right), \\ \dot{q}_{1_d} &= 0\,, \\ \ddot{q}_{1_d} &= 0\,, \end{aligned}$$

dove x_{c_i} e y_{c_i} rappresentano le coordinate della posizione del centro di massa della pinza agli istanti di frenata t_{b_i}. In tal modo, l'impulso, che dalla pinza è trasmetto al filo, viene interamente assorbito dalla struttura che sorregge il manipolatore.

1.3.4 Le condizioni di presa

Nel caso specifico del manipolatore in esame, lo stato desiderato $\mathbf{S}_d$ della pinza deve soddisfare le condizioni

$$\begin{aligned} x_{c_d} &= x_t\,, \\ y_{c_d} &= y_t\,, \\ \varphi_d &\in [\varphi_{min}, \varphi_{max}]\,, \end{aligned}$$

$$\begin{aligned} &\sqrt{\dot{x}_{c_d}^2 + \dot{y}_{c_d}^2} \in [v_{min}, v_{max}]\,, \\ &\dot{\varphi}_d = 0\,, \end{aligned}$$

con φ_{min}, φ_{max} e v_{min}, v_{max} opportunamente scelti.

La condizione sulla velocità minima di traslazione del centro di massa della pinza afferma la necessità che quest'ultima possieda un'energia cinetica residua tale da far scattare il meccanismo di chiusura della pinza. Al contrario, la condizione sulla velocità massima serve per impedire che la pinza rompa o rimbalzi sul bersaglio.

Infine, la condizione sulla velocità di rotazione $\dot{\varphi}_d$ è talvolta richiesta per impedire che il bersaglio venga rovesciato una volta afferrato.

1.3.5 Il recupero

L'ultima fase di funzionamento della Manipolazione del Lancio consiste nel recupero dell'end-effector e, in caso di successo nella presa, anche del bersaglio. Tale scopo in generale è ottenuto mediante un'azione combinata di riavvolgimento del braccio cedevole L_n e di controllo dei bracci attuati $L_1, \ldots, L_{n-1}$. La *strategia di recupero* deve tenere di conto della posizione dell'end-effector, della presenza di ostacoli lungo il cammino, della resistenza dell'oggetto preso, etc...

Nel caso del dispositivo in esame, si procede con il riavvolgimento del filo da pesca sul mulinello mediante un opportuno motore. Una volta recuperata la pinza e, in caso di presa, anche il bersaglio, il braccio L_1 viene riportato nella configurazione iniziale $q_1 = 0$, $\dot{q}_1 = 0$. Una volta che il secondo braccio, soggetto alla forza peso e a vari attriti, si ferma ($q_2 = 0$, $\dot{q}_2 = 0$), il manipolatore è pronto per eseguire un altro lancio.

Capitolo 2

Raggiungere un oggetto in movimento

Proseguendo nell'intento di definire una politica di controllo per la presa ed il recupero di un oggetto in movimento, cerchiamo ora di trovare un strategia che innanzitutto ci permetta di raggiungere un bersaglio mobile.

In questo capitolo vengono presentate due soluzioni alternative: la prima è una diretta generalizzazione della strategia di controllo discussa nel capitolo precedente, che consente di raggiungere il bersaglio attraverso n frenate; la seconda, invece, è una tecnica ottenuta in modo indipendente ed è in grado di portare l'end-effector del manipolatore sul bersaglio mediante una sola frenata.

2.1 Controllo in volo multi-frenata

Poniamoci come obiettivo quello di raggiungere un bersaglio mobile, la cui posizione istantanea (x_t, y_t) viene misurata dal sistema di visione che sarà descritto nel prossimo capitolo. Supponiamo di sostituire la pinza con una pallina che rappresenta un punto materiale.

Noto che sia lo stato iniziale di lancio della pallina e avendo fissato l'istante di frenata t_b, è possibile calcolare il valore del coefficiente di rimbalzo e,

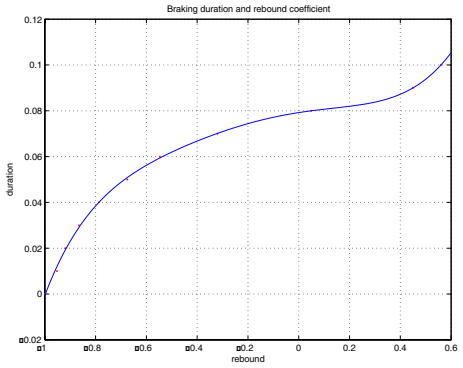

Figura 2.1: Relazione che lega il coefficiente di rimbalzo e con la durata ΔT_{b_i} della corrispondente frenata.

che teoricamente permette di raggiungere il bersaglio mediante una sola frenata [3]. In effetti, solo se il valore di e così ottenuto è in modulo minore o uguale all'unità, allora tale rimbalzo è fisicamente realizzabile ed è pertanto possibile raggiungere il bersaglio. In caso contrario, il lancio fallisce.

Volendo estendere questa tecnica al caso di un oggetto mobile, si procede come segue. Si fissano gli istanti di inizio t_{b_i} delle frenate e si assume, ad esempio, che siano distanti $T_{braking}$ secondi l'uno dall'altro. Resta da determinare a questo punto solo i coefficienti di rimbalzo e_{b_i} di ogni frenata. In figura (2.1) è riportata la caratteristica che lega il valore del coefficiente di rimbalzo alla durata della corrispondente frenata. Inoltre, in figura (2.2) viene schematizzato un esperimento in caduta libera usato per il calcolo di tale caratteristica. Si introduce un bersaglio virtuale la cui posizione (x_v, y_v) deve aggiornata opportunamente. Indichiamo con x_{v_i} la posizione del bersaglio virtuale all'istante della i-esima frenata. Il coefficiente di rimbalzo e_{b_i} viene calcolato in modo che la pallina termini la propria traiettoria esattamente sul bersaglio virtuale x_{v_i}.

Supponiamo di voler raggiungere un bersaglio a terra, $y_t = 0$. In tal

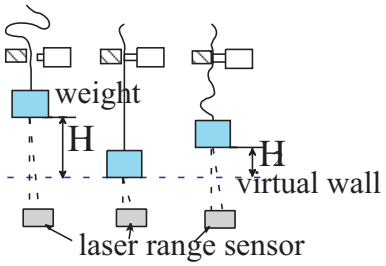

Figura 2.2: Un esperimento in caduta libera: è usato per calcolare la caratteristica $e - \Delta T_{b_i}$.

caso, la posizione del bersaglio virtuale è $(x_v, 0)$. L'aggiornamento di x_v deve essere fatta, in generale, considerando tutte le informazioni note a priori sulla possibile traiettoria seguita del bersaglio. Ad esempio, si può ipotizzare che il bersaglio si muova con velocità costante, con accelerazione costante oppure di moto casuale. In assenza di tali informazioni una possibile scelta è la seguente:

$$\begin{cases} x_{v_0} = \min\left(x_{land_{max}}, 2\,x_t\right) , \\ x_{v_i} = \dfrac{x_{land_i} + x_{t_i}}{2} . \end{cases}$$

Ad ogni passo, si procede a calcolare un coefficiente di rimbalzo. Se tale valore è un modulo minore o uguale all'unità, allora si calcola la corrispondente durata della frenata; in caso contrario, la frenata non avviene, ovvero si sceglie $e = -1$.

La figura (2.3) mostra la traiettoria ottenuta nel caso di un oggetto che si avvicina al manipolatore.

2.2 Controllo in volo a frenata singola

Consideriamo adesso una strategia di controllo in volo alternativa, che consente di raggiungere un bersaglio mobile mediante una sola frenata. Suppo-

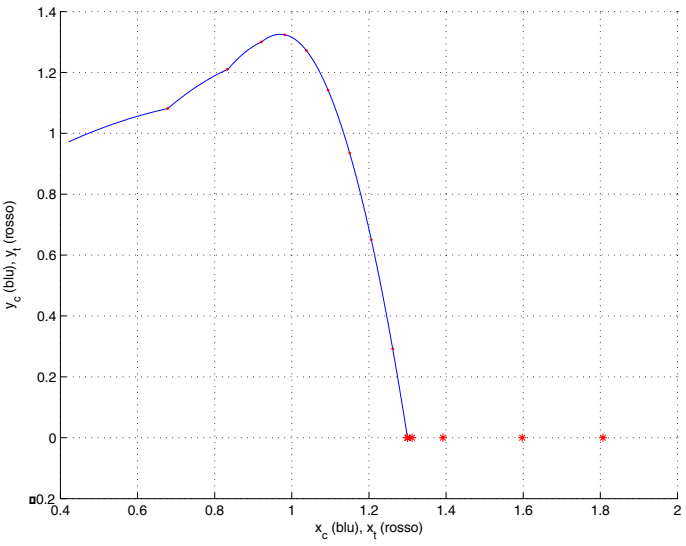

Figura 2.3: Il risultato di una politica di controllo in volo multi-frenata: la pallina viene controllata in modo da raggiungere il bersaglio mobile.

niamo ancora di aver sostituito la pinza con una pallina di massa m.

Ricordiamo innanzitutto che lo stato della pallina al momento del lancio è legato alla configurazione del robot in quell'istante dalle relazioni

$$\begin{cases} x_0 = x_{base} + a_1\, s_1 + a_2\, s_{12}\,, \\ y_0 = y_{base} - a_1\, c_1 - a_2\, c_{12}\,, \\ \dot{x}_0 = a_1\, c_1\, \dot{q}_1 + a_2\, c_{12}\, (\dot{q}_1 + \dot{q}_2)\,, \\ \dot{y}_0 = a_1\, s_1\, \dot{q}_1 + a_2\, s_{12}\, (\dot{q}_1 + \dot{q}_2)\,. \end{cases} \tag{2.1}$$

Partendo dallo stato iniziale $(x_0, y_0, \dot{x}_0, \dot{y}_0)$, la pallina percorre, sotto l'effetto della forza peso g, una traiettoria parabolica. Il suo stato al generico istante t

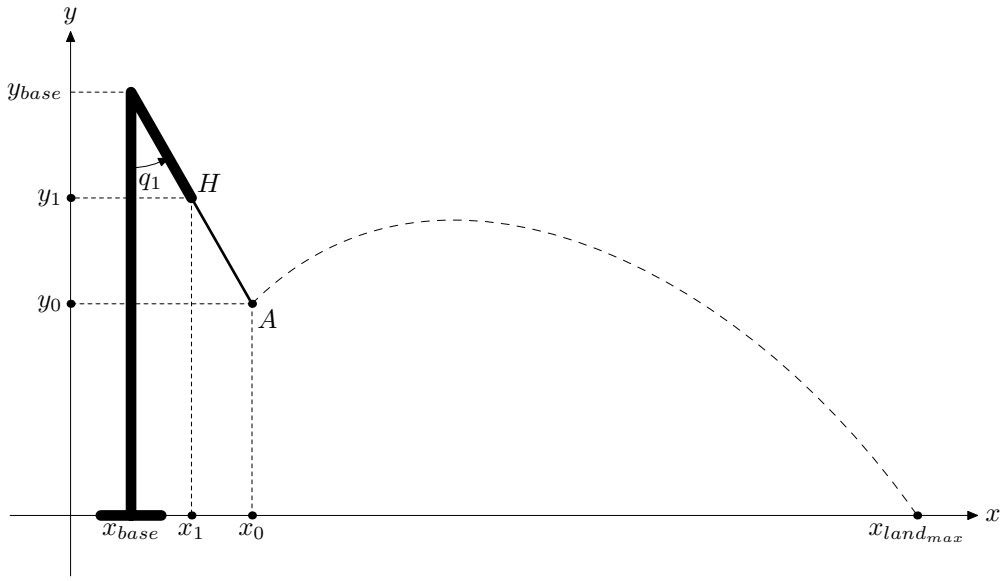

Figura 2.4: La traiettoria della pallina in assenza di controllo in volo.

è dato da:

$$\begin{cases} x(t) = x_0 + \dot{x}_0\, t\,, \\ y(t) = y_0 + \dot{y}_0\, t - \dfrac{1}{2}\, g\, t\,, \\ \dot{x}(t) = \dot{x}_0\,, \\ \dot{y}(t) = \dot{y}_0 - g\, t\,. \end{cases}$$

E' chiaro che, in assenza di controllo, la distanza massima raggiungibile (si veda figura 2.4). è data dall'espressione:

$$x_{land_{max}} = x_0 + \frac{\dot{x}_0}{2\, g}\left(\dot{y}_0 + \sqrt{\dot{y}_0^2 + 2\, g\, y_0}\right)\,.$$

Supponiamo che il primo giunto resti nella configurazione di lancio, cioè si abbia $q_1 = q_{1_{lancio}}$. Indichiamo con l_0 la distanza tra l'estremità del primo braccio (punto H) e la posizione istantanea $(x_t, 0)$ del bersaglio. Tale distanza è data da

$$l_0 = \sqrt{(x_1 - x_t)^2 + (y_1 - y_t)^2}\,.$$

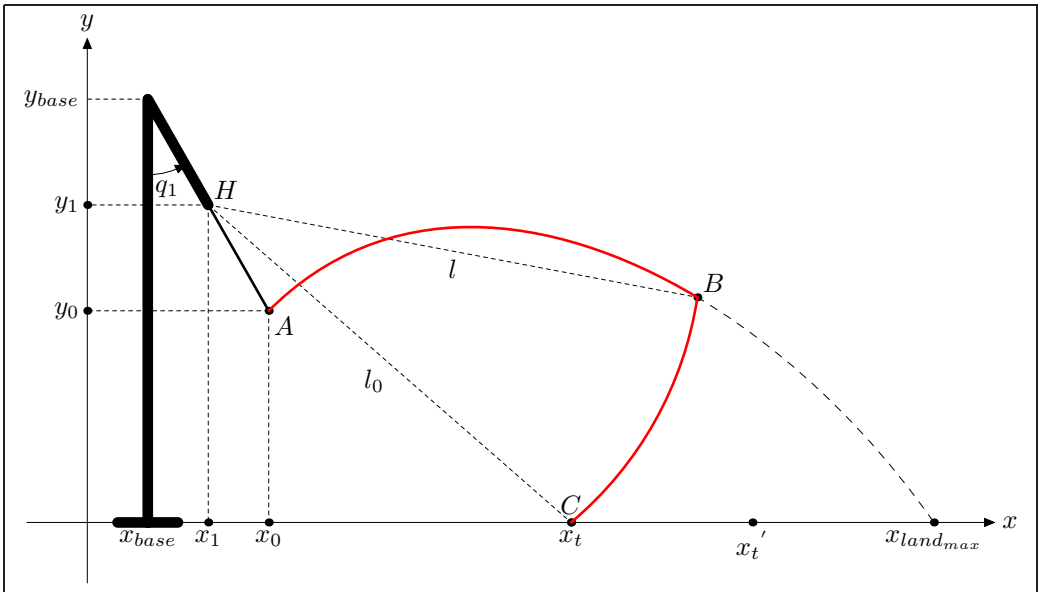

Figura 2.5: In rosso è riportata la traiettoria della pallina, ottenuta come risultato del controllo in volo a frenata singola, per un oggetto fermo.

Indichiamo invece, con $l(t)$ la lunghezza del filo, cioè la distanza tra l'estremità del primo braccio e la posizione istantanea della pallina. Tale distanza è data da

$$l(t) = \sqrt{(x(t) - x_1)^2 + (y(t) - y_1)^2}\,.$$

Quando le due distanze sono uguali ($l(t) = l_0$), si blocca lo scorrimento del filo mediante il freno. Durante la frenata, la pallina è vincolata a muoversi sull'arco di circonferenza BC che la conduce così direttamente sul bersaglio (figura 2.5).

Se il bersaglio si sposta nella posizione C', allontanandosi dal robot, l'inizio della frenata è ritardato fino all'istante in cui la pallina si trova nel punto B'. Il bersaglio viene comunque raggiunto a patto che non superi la distanza massima percorribile $x_{land_{max}}$ (figura 2.6).

Se, infine, il bersaglio si avvicina al robot (punto C''), l'istante di inizio

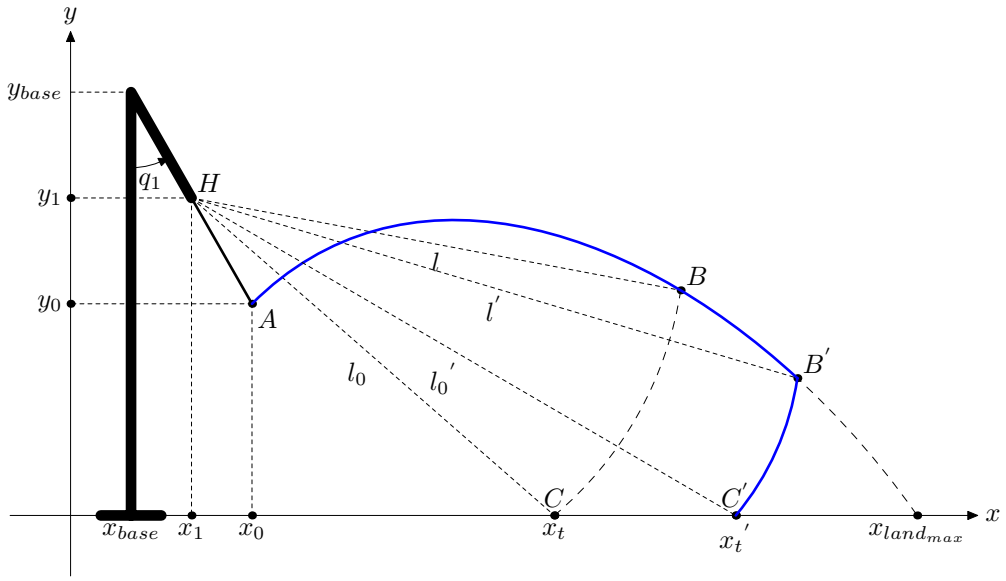

Figura 2.6: In blu è riportata la traiettoria della pallina, ottenuta come risultato del controllo in volo a frenata singola, per un oggetto fermo.

della frenata è anticipato al momento in cui la pallina passa da B'' (figura 2.6), riuscendo comunque a raggiungere l'oggetto.

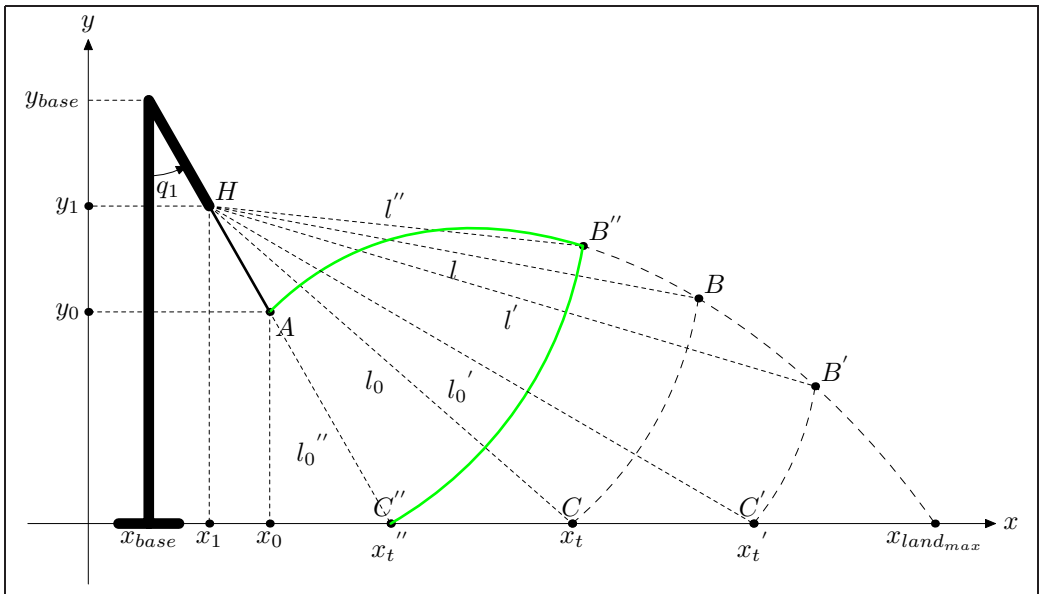

Figura 2.7: In verde è riportata la traiettoria della pallina, ottenuta come risultato del controllo in volo a frenata singola, per un oggetto fermo.

Capitolo 3

Il sistema di visione

Il sistema di presa e recupero di oggetti in movimento necessita di un meccanismo di visione in grado di rilevare alcune caratteristiche dell'ambiente circostante, tra cui la posizione istantanea del bersaglio e l'eventuale presenza di ostacoli. A tal fine, è necessario innanzitutto scegliere il tipo di dispositivo fisico da impiegare per la scansione dell'ambiente; in secondo luogo, occorre individuare un algoritmo in grado di estrarre le informazioni significative dai dati acquisiti; infine, per tenere di conto delle inevitabili deformazioni introdotte dal dispositivo, è richiesto di eseguire un'operazione di calibrazione.

Il presente capitolo è dedicato alla realizzazione di un sistema di visione che fa uso di una telecamera come dispositivo di scansione[1]*; nel primo paragrafo, discutiamo alcuni aspetti generali relativi all'impiego della visione come meccanismo di retro-azione in un sistema di controllo; il secondo paragrafo è invece indirizzato alla descrizione del processo di acquisizione del flusso video; nel terzo paragrafo si rivolge l'attenzione all'algoritmo di estrazione della posizione istantanea del bersaglio, algoritmo che fa uso delle librerie Open Computer Vision (OpenCV); infine, nel quarto ed ultimo paragrafo, si spiega la tecnica di calibrazione utilizzata.*

[1] L'algoritmo di estrazione della posizione del bersaglio e quello di calibrazione si basano sul lavoro di *Antonio Danesi*, *Daniele Fontanelli* e *Vincenzo Scordio*.

3.1 L'impiego della visione

Nel controllo di un manipolatore si possono impiegare dispositivi di visione di diverso tipo. Fra le varie possibilità troviamo l'uso di

- ▷ una telecamera;
- ▷ un laser;
- ▷ un sonar.

A prescindere della natura del dispositivo, esso viene utilizzato per scansionare l'ambiente circostante alla ricerca degli oggetti di interesse. Quest'ultimi devono essere ben riconoscibili; pertanto su ognuno di essi deve essere posizionato un marcatore identificabile dal meccanismo di visione scelto. I dati acquisiti vengono elaborati ed utilizzati, in generale, al fine di ricostruire una rappresentazione tridimensionale dello spazio circostante.

Nel caso più semplice, il sistema di visione è usato per rilevare la posizione istantanea del solo bersaglio e l'informazione ottenuta viene usata per controllare il robot in modo che questo possa raggiungere ed afferrare l'oggetto. In tale circostanza, l'uscita del sistema di visione rappresenta un segnale di riferimento in ingresso al controllo del manipolatore. In ragione di ciò, i requisiti temporali del sistema di visione dipendono dalla sola velocità dell'oggetto.

Più in generale, la visione può essere impiegata anche per determinare la posizione dell'end-effector del robot. In tal caso, l'informazione in uscita dal sistema di visione viene usata anche nell'anello più interno della retro-azione. Dal punto di vista realizzativo, ciò richiede che la velocità di acquisizione del

dispositivo di visione sia certamente più elevata di quella richiesta nel primo caso visto. In queste condizioni, i requisiti temporali possono essere eccessivi, impedendo l'uso della visione nel sistema di controllo.

Tuttavia, in taluni casi, si può ricorrere al seguente rimedio. Si arricchisce il sistema di controllo di un algoritmo in grado di fornire una stima della grandezza misurata attraverso il sistema di visione. Ogni qualvolta che il controllo non dispone dell'informazione esterna proveniente dalla sorgente di acquisizione, utilizza l'ultimo valore rilevato per determinare una stima del valore attuale. Nel momento in cui si rende disponibile un nuovo fotogramma, è possibile aggiornare la stima interna *fondendola* con l'informazione esterna. A tal scopo può essere usato ad esempio un *filtro di Kalman esteso*. In genere, questo accorgimento permette di ridurre i requisiti temporali richiesti.

Nel seguito facciamo implicitamente riferimento all'uso di una telecamera come dispositivo di acquisizione, anche se quanto detto ha una validità più generale. Inoltre, la telecamera viene impiegata per determinare soltanto la posizione del bersaglio mobile.

3.2 L'acquisizione del flusso video

L'acquisizione del flusso video in ambiente Ms Windows© fa uso delle librerie *Video for Windows*. Il codice sorgente per l'utilizzo di una telecamera è interamente contenuto nella classe *CCamera*, la cui interfaccia di programmazione è riportata nell'appendice (A), dove includiamo anche una breve descrizione delle funzioni membro più significative. Per l'applicazione in esame, è sufficiente istanziare una sola variabile di tipo *CCamera*, visto che si dispone di un'unica telecamera. Più in generale, è necessario definire un og-

getto *CCamera* per ogni sorgente di acquisizione video di cui l'applicazione fa uso.

Per inizializzare una telecamera si devono compiere i seguenti passi:

- ▷ creare una finestra di cattura, nella quale verranno visualizzati i fotogrammi acquisiti;
- ▷ richiedere al sistema operativo l'uso esclusivo di un dispositivo di cattura;
- ▷ collegare la finestra di cattura al dispositivo concesso dal sistema operativo;
- ▷ installare un gestore della telecamera, chiamato ad esempio *FrameCallBack()*;
- ▷ impostare i parametri di funzionamento della telecamera.

Una volta attivata, la telecamera fornisce una nuova immagine con una periodicità che dipende dalle caratteristiche hardware del dispositivo e da altre condizioni, tra cui la luminosità dell'ambiente circostante. Il sistema operativo provvede a mandare in esecuzione il gestore della telecamera *FrameCallBack()*, ogni volta che un nuovo fotogramma è disponibile. Tale gestore si occupa, tra le varie operazioni, di scaricare l'immagine del nuovo fotogramma in un buffer locale all'applicazione. Nel caso specifico, il buffer è rappresentato da una variabile di tipo *IplImage**.

3.3 L'estrazione della posizione istantanea

Esistono varie tecniche per l'analisi del flusso video, in grado di estrarre informazioni come la posizione di un certo oggetto, la sua velocità, la sua

area, ecc... Fra quelle usate per rilevare la posizione (u, v) di un oggetto all'interno dell'immagine acquisita, troviamo le tecniche note come

- ▷ feature tracking;
- ▷ pattern matching;
- ▷ estrazione dei contorni.

La prima tecnica, chiamata *feature tracking*, consente di ricavare la posizione di un oggetto attraverso un'analisi del flusso ottico; in altre parole, nota una stima iniziale della posizione, l'algoritmo cerca di aggiornare tale stima elaborando una sequenza di fotogrammi. Sebbene venga impiegata in molte applicazioni, tale tecnica non risulta molto robusta ed è facile che l'algoritmo perda il riferimento alla posizione e non riesca ad inseguire l'oggetto. In ragione di ciò, essa è spesso supportata da algoritmi di recupero.

Al contrario, la tecnica nota come *pattern matching* (riconoscimento di una maschera), estrae la posizione dell'oggetto dall'analisi di un solo fotogramma. Tale informazione è ottenuta scansionando il fotogramma corrente alla ricerca di una particolare sotto-immagine. In talune applicazioni, la scansione completa del fotogramma può richiedere un tempo eccessivo, per cui è necessario ricorrere al seguente stratagemma. Si scandiscono per intero i primi fotogrammi acquisiti dalla telecamera alla ricerca dell'oggetto di interesse; una volta individuata l'area in cui l'oggetto si trova, si scandiscono i fotogrammi successivi soltanto nella zona adiacente a tale area; qualora il riferimento alla posizione venga perso, si torna ad eseguire una scansione completa del fotogramma attuale. In questa maniera il tempo necessario all'estrazione è mediamente ridotto. La tecnica del pattern matching e le sue

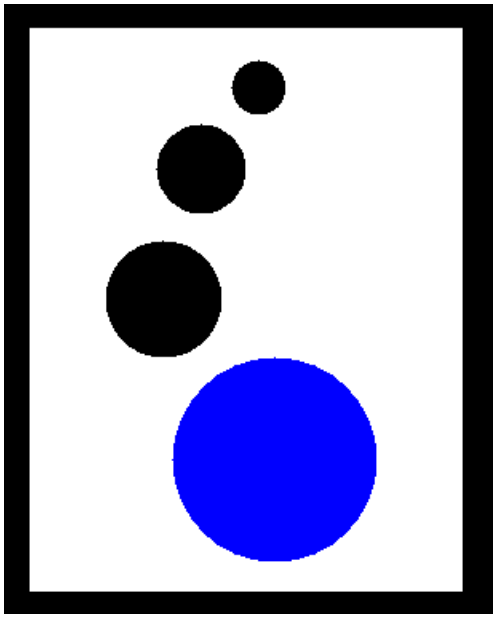

Figura 3.1: Il marcatore che viene montato sul bersaglio.

derivate vengono a volte impiegate proprio per la robustezza e la semplicità realizzativa.

Per ultima, troviamo la tecnica basata sull'*estrazione dei contorni*, tecnica che abbiamo scelto di usare per rilevare la posizione del bersaglio mobile. In generale, una volta noto il meccanismo di estrazione dei contorni di un'immagine, si può realizzare un algoritmo in grado di ricercare all suo interno delle maschere con opportune caratteristiche in termini di area, posizione all'interno del fotogramma, numero di contorni interni, ecc... Nel nostro caso abbiamo deciso di usare dei marcatori di colore chiaro al cui interno sono disegnati alcuni pallini. Il numero di pallini identifica univocamente il marcatore ed un pallino grande viene usato come *punto di riferimento* del marcatore stesso. Nelle figure (3.1 e 3.2) sono riportati due esempi di marcatori.

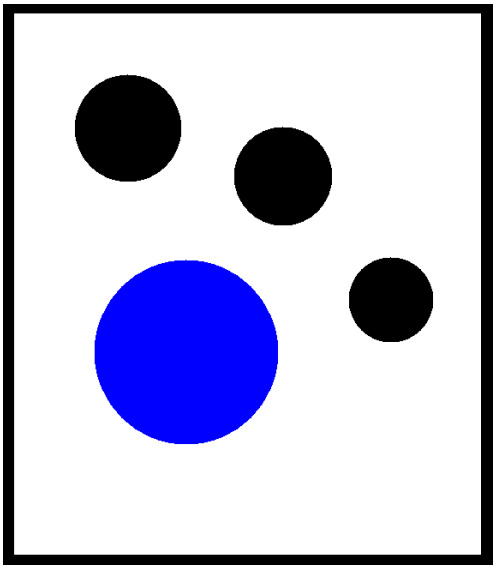

Figura 3.2: Un esempio di marcatore usato per la calibrazione.

La tecnica che descriviamo si appoggia alla libreria *Open Computer Vision* (*OpenCV*) per l'estrazione dei contorni e le altre operazioni di elaborazione dell'immagine. Si riesce ad individuare un contorno analizzando le variazioni di colore ed intensità all'interno del singolo fotogramma. Tale operazione è svolta dalla funzione di libreria *cvFindContours()*. Tale funzione prevede in ingresso l'immagine da analizzare e fornisce in uscita le informazioni relative ai contorni trovati, avendoli accuratamente organizzati in una struttura ad albero. Senza entrare nei dettagli, ogni nodo dell'albero rappresenta un contorno individuato; scorrendo l'albero con il puntatore *v_ next* si scende ad un livello inferiore e si trovano le informazioni relative ai contorni interni al contorno dato, mentre scorrendo l'albero con il puntatore *h_ next* si passa ad analizzare un nodo relativo ad un contorno disgiunto dai precedenti.

Il codice per la ricerca di un marcatore è riportato di seguito nella funzione *FindObject()*. Al suo interno si richiama la funzione *IdentifyAndLocaliseObject()* che conta il numero di pallini interni al marcatore e restituisce la

posizione del centro del punto di riferimento.

```
int CObjectLocalisation::FindObject(
    IplImage *img, CvPoint& ReferencePointPos)
{
    ...

    // Converts the image to black and white and
    // apply a threshold filter.
    cvCvtColor(img, img_bn, CV_RGB2GRAY);
    cvThreshold(
        img_bn, img_work, THRESHOLD,
        MAX_VALUE, CV_THRESH_BINARY);

    ...

    // Extracts the contours in the current image.
    cvFindContours(
        img_work, storage, &contour,
        sizeof(CvContour), CV_RETR_CCOMP);

    ...

    for(; contour != 0; contour = contour->h_next)
    {
        double area = fabs(cvContourArea(contour));

        if (area > AREA_MIN && area < AREA_MAX)
        {
            // Gets the information about the current contour.
            MarkerNumber = IdentifyAndLocaliseObject(
                CentrePosition, ReferencePointPos, contour, moments);

            // Checks if the recognised marker is valid.
            if (IsValid(MarkerNumber))
                break;
        }
    }

    ...

    return MarkerNumber;
}
```

```
44

46 int CObjectLocalisation::IdentifyAndLocaliseObject(
       CvPoint& CentrePosition, CvPoint& ReferencePointPos,
48     CvSeq *contour, CvMoments *moments)
   {
50     // Evaluates the position of the centre
       double m00, m10, m01;
52     cvMoments(contour, moments);
       ExtractMoments(moments, m00, m10, m01);
54     CentrePosition = cvPoint((int)(m10/m00),(int)(m01/m00));

56     int ball = 0;
       double area_max = 0, area_current;
58
       // Counts the number of balls and evaluate the position of
60     // the reference point, which is the centre of the biggest ball.
       for (
62         contour = contour->v_next;
           contour != 0;
64         contour = contour->h_next)
       {
66         area_current = cvContourArea(contour);

68         if (area_current > area_max)
           {
70             area_max = area_current;

72             // Gets the position of the reference point.
               cvMoments(contour, moments);
74             ExtractMoments(moments, m00, m10, m01);
               ReferencePointPos =
76                 cvPoint((int)(m10/m00), (int)(m01/m00));
           }
78
           ball++;
80     }

82     return ball;
   }
```

Il calcolo dell'area interna ad un dato contorno e il suo centro sono ottenuti sfruttando la nozione di *momento di un contorno* di ordine (p, q):

$$m_{pq} = \int_x \int_y x^p \, y^q \, dx \, dy \, .$$

L'area interna è infatti ottenuta come momento di ordine $(0, 0)$:

$$A = \int_x \int_y dx \, dy \, ,$$

mentre la coordinata x del centro è ottenuta come momento *normalizzato* di ordine $(1, 0)$:

$$x = \frac{1}{A} \int_x \int_y x \, dx \, dy$$

e la coordinata y come momento *normalizzato* di ordine $(0, 1)$:

$$x = \frac{1}{A} \int_x \int_y y \, dx \, dy.$$

Una volta calibrata, la telecamera è in grado di misurare la posizione (x, y) del bersaglio chiamando la funzione *FindObject()*. Per verificare la presenza del bersaglio nel campo visibile alla telecamera e, in tal caso, determinarne la posizione, si può chiamare la funzione *non bloccante ObjectDetected()*. Riportiamo di seguito il codice di tale procedura:

```
2 extern CCamera m_camera;
  extern double x, y;
4
  bool CObjectLocalisation::ObjectDetected()
6 {
      // Checks if a new frame is available
8     if (!m_camera.NewFrame)
          return false;
10
```

```
    m_camera.NewFrame = false;
12
    CvPoint TargetPos_OnImage;
14
    // Gets a pointer to the current frame
16  IplImage* img = m_camera.GetFrame().GetImage();

18  // Checks if the marker placed on the target is visible or not
    if (FindObject(img, TargetPos_OnImage) == MARKER_ON_TARGET)
20  {
        double u = TargetPos_OnImage.x;
22      double v = 480 - TargetPos_OnImage.y;

24      // Converts the coordinate of the image plane into
        // the correspondent coordinate of the movement plane.
26      ConvertToRealWorld(u, v, x, y);

28      return true;
    }
30
    return false;
32 }
```

3.4 La procedura di calibrazione

L'immagine fornita dal dispositivo di acquisizione è la proiezione su una certa superficie delle caratteristiche dello spazio circostante. Essa inevitabilmente contiene delle deformazioni, dovute al fatto che si cerca di rappresentare in uno spazio bidimensionale una realtà a tre dimensioni. A prescindere, dunque, dal tipo di dispositivo, si deve eseguire un'operazione di calibrazione per compensare tali effetti deformativi.

In quanto segue vogliamo trovare un modo per svolgere la calibrazione della telecamera, note che siano le posizioni di alcuni punti nel spazio reale e facendo uso dell'algoritmo di estrazione visto nel paragrafo precedente. A tal scopo, fissiamo innanzitutto due sistemi di riferimento: uno sul piano dell'immagine rilevata dalla telecamera e l'altro sul piano di movimento del

bersaglio; la posizione di un punto sul piano immagine è allora rappresentata in coordinate omogenee dal vettore

$$\mathbf{p}_{image} = (u, v, 1)^T \ ;$$

analogamente, la posizione di ogni punto nel piano di movimento è rappresentata dal vettore

$$\mathbf{p}_{real} = (x, y, 1)^T \ ;$$

In base a quanto detto sopra, i due sistemi di coordinate sono legati da una relazione del tipo

$$\mathbf{p}_{real} = \mathcal{H}\left(\mathbf{p}_{image}\right) \ , \tag{3.1}$$

dove la funzione vettoriale $\mathcal{H}$, detta *mappa di omografia*, è utilizzata per compensare gli effetti di deformazione introdotti dal dispositivo di visione. La sua complessità è generalmente molto alta e pertanto richiede, laddove è possibile, di effettuare il maggior numero di semplificazioni, compatibilmente con il grado di precisione richiesto. A tal proposito, nel caso di un moto planare e volendo trascurare gli effetti di curvatura, si può dimostrare che si ottengono buoni risultati di compensazione mediante una mappa di omografia $\mathcal{H}$ lineare, cioè del tipo:

$$\mathbf{p}_{real} = \mathcal{H}\,\mathbf{p}_{image} \ ,$$

dove $\mathcal{H} \in \mathbb{R}^{3\times 3}$ è una matrice di elementi

$$\mathcal{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix} \ .$$

Prima di proseguire, facciamo la seguente osservazione. Nell'ipotesi in cui la collocazione della telecamera è fissa, è sufficiente eseguire la calibrazione soltanto una volta, ad esempio all'inizio. Infatti, la mappa di omografia $\mathcal{H}$ ottenuta con tale calibrazione è sempre in grado di compensare le deformazioni introdotte. Al contrario, se la telecamera viene mossa, è opportuno aggiornare $\mathcal{H}$ con una certa periodicità. In tale circostanza, si deve fare in modo che la telecamera veda per la maggior parte del tempo un insieme di marcatori di posizione nota e possa ri-eseguire la calibrazione.

Detto ciò, torniamo alla procedura di calibrazione e cerchiamo di trovare una via per calcolare gli otto parametri della matrice $\mathcal{H}$, sfruttando l'algoritmo di estrazione della posizione, visto nel paragrafo precedente.

Abbiamo già osservato che il generico punto $\mathbf{p}_{image}$ sul piano immagine è legato al corrispondente punto $\mathbf{p}_{real}$ sul piano di movimento dalla relazione (3.1). In forma estesa abbiamo

$$
\begin{aligned}
x &= h_{11}\, u + h_{12}\, (v_{max} - v) + h_{13}\,, \\
y &= h_{21}\, u + h_{22}\, (v_{max} - v) + h_{23}\,, \\
1 &= h_{31}\, u + h_{32}\, (v_{max} - v) + 1\,,
\end{aligned}
$$

dove v_{max} rappresenta la risoluzione lungo la coordinata y. Supponiamo di fissare quattro punti $\mathbf{p}_{m_1}$, $\mathbf{p}_{m_2}$, $\mathbf{p}_{m_3}$ e $\mathbf{p}_{m_4}$ e di conoscerne le posizioni sul piano di movimento (x_i, y_i) per $i = 1, 2, 3, 4$. Usando l'algoritmo di estrazione visto sopra si ricavano le corrispondenti posizioni (u_i, v_i) per $i = 1, 2, 3, 4$ sul

piano immagine. Possiamo adesso scrivere:

$$\begin{cases} x_1 = h_{11}\,u_1 + h_{12}\,(v_{max} - v_1) + h_{13}\,, \\ y_1 = h_{21}\,u_1 + h_{22}\,(v_{max} - v_1) + h_{23}\,, \\ 0 = h_{31}\,u_1 + h_{32}\,(v_{max} - v_1)\,, \\ x_2 = h_{11}\,u_2 + h_{12}\,(v_{max} - v_2) + h_{13}\,, \\ y_2 = h_{21}\,u_2 + h_{22}\,(v_{max} - v_2) + h_{23}\,, \\ 0 = h_{31}\,u_2 + h_{32}\,(v_{max} - v_2)\,, \\ x_3 = h_{11}\,u_3 + h_{12}\,(v_{max} - v_3) + h_{13}\,, \\ y_3 = h_{21}\,u_3 + h_{22}\,(v_{max} - v_3) + h_{23}\,, \\ 0 = h_{31}\,u_3 + h_{32}\,(v_{max} - v_3)\,, \\ x_4 = h_{11}\,u_4 + h_{12}\,(v_{max} - v_4) + h_{13}\,, \\ y_4 = h_{21}\,u_4 + h_{22}\,(v_{max} - v_4) + h_{23}\,, \\ 0 = h_{31}\,u_4 + h_{32}\,(v_{max} - v_4)\,. \end{cases}$$

Si tratta di un sistema di dodici equazioni nelle otto incognite cercate, $\mathbf{h} = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32})$, che possiamo riscrivere in forma matriciale come

$$\begin{pmatrix} x_1 \\ y_1 \\ 0 \\ x_2 \\ y_2 \\ 0 \\ x_3 \\ y_3 \\ 0 \\ x_4 \\ y_4 \\ 0 \end{pmatrix} = \begin{pmatrix} u_1 & v_{max} - v_1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1 & v_{max} - v_1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_1 & v_{max} - v_1 \\ u_2 & v_{max} - v_2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2 & v_{max} - v_2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_2 & v_{max} - v_2 \\ u_3 & v_{max} - v_3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_3 & v_{max} - v_3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_3 & v_{max} - v_3 \\ u_4 & v_{max} - v_4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_4 & v_{max} - v_4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_4 & v_{max} - v_4 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix},$$

ovvero in forma compatta

$$\mathbf{p} = \mathcal{M}\,\mathbf{h}\,,$$

dove $\mathbf{p} = (x_1, v_1, 0, x_2, v_2, 0, x_3, v_3, 0, x_4, v_4, 0)$ ed $\mathcal{M} \in \mathbb{R}^{12\times 8}$. Per ottenere i parametri cercati basterebbe invertire il sistema. Tuttavia, la matrice $\mathcal{M}$ non è quadrata e, di conseguenza, non possiede inversa. Ciò nonostante, possiamo risolvere il sistema usando la pseudo-inversione. Dunque, i parametri dell'omografia $\mathcal{H}$ sono dati da

$$\mathbf{h} = \mathcal{M}^{\dagger}\,\mathbf{p}\,, \tag{3.2}$$

dove $\mathcal{M}^{\dagger}$ rappresenta la pseudo-inversa di $\mathcal{M}$.

In conclusione, la calibrazione di una telecamera in posizione fissa si compone dei seguenti passi:

- ▷ sistemare la telecamera nella posizione desiderata;
- ▷ sistemare almeno quattro marcatori nel campo visibile dalla telecamera;
- ▷ misurare la posizione sul piano di movimento del punto di riferimento di ogni marcatore;
- ▷ estrarre la posizione sul piano immagine di tali punti mediante il sistema di visione;
- ▷ calcolare gli elementi della matrice di omografia $\mathcal{H}$ facendo uso della (3.2).

Nelle figure (3.3) e (3.3) sono riportate le immagini ottenute come risultato da due diverse calibrazioni. In particolare, nella (3.3) è possibile verificare

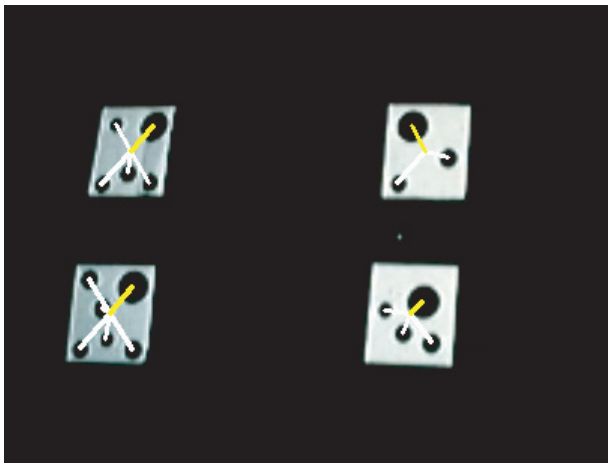

Figura 3.3: L'immagine vista durante la calibrazione, nel caso in cui i marcatori siano disposti su un piano ortogonale alla visuale della telecamera.

Figura 3.4: L'immagine vista durante la calibrazione, nel caso in cui i marcatori siano disposti su un piano *non* ortogonale alla visuale della telecamera.

che l'algoritmo usato è in grado di individuare i marcatori anche su un piano non ortogonale alla visuale della telecamera.

Il codice sorgente per l'estrazione della posizione del bersaglio e la procedura di calibrazione sono interamente contenuti nella classe *CObjectLocalisation*. Nell'appendice (A) riportiamo una descrizione delle funzioni membro più significative di tale classe.

Capitolo 4

La libreria di simulazione, controllo e diagnostica

Il software sviluppato, ad eccezione di quello relativo al sistema di visione, è stato raccolto in una sola libreria che abbiamo chiamato Casting Manipulation Library*. Tale libreria dichiara al suo interno unicamente la classe* CCastingManipulator *i cui metodi possono essere usati per la simulazione, il controllo e la diagnostica di un generico manipolatore del lancio. Sebbene l'implementazione dei metodi si riferisca al manipolatore costruito in laboratorio (capitolo 5), la sua interfaccia di programmazione risulta generale e valida anche per altri robot della stessa categoria.*

4.1 Le funzioni di simulazione

Tra i vari i metodi della classe *CCastingManipulator* troviamo innanzitutto quelli relativi alla simulazione del robot. In generale, ogni volta che si desidera simulare un sistema a tempo continuo su di un calcolatore, è necessario trovare un'approssimazione a tempo discreto della dinamica di tale sistema, ottenuta con tempo di campionamento T_{sample}.

Consideriamo ad esempio il metodo *OneLinkDirectDynamics(τ)* che viene

usato per aggiornare lo stato del manipolatore durante la fase di controllo in volo, in cui la dinamica del robot è descritta dall'equazione

$$\ddot{q}_1 = \left(I_1 + m_1\, l_1^2\right)^{-1} \left(\tau - m_1\, g\, l_1\, s_1\right) . \tag{4.1}$$

Più in generale, per un equazione differenziale del tipo $\ddot{q}_1 = f_1(q_1, \dot{q}_1)$, si può costruire la forma di stato

$$\begin{cases} \dot{x}_1 = x_2 \,, \\ \dot{x}_1 = f_1(x_1, x_2) \,, \end{cases} \tag{4.2}$$

dove abbiamo definito $x_1 = q_1$ e $x_2 = \dot{q}_1$. Usando l'approssimazione di Eulero in avanti per le derivate temporali

$$\frac{dx}{dt} \simeq \frac{x(t + T_{sample}) - x(t)}{T_{sample}} \,,$$

nella (4.2) otteniamo

$$\begin{cases} x_1(t + T_{sample}) = x_1(t) + T_{sample}\, x_2(t) \,, \\ x_2(t + T_{sample}) = x_2(t) + T_{sample}\, f(x_1(t), x_2(t) \,. \end{cases}$$

Passando alla notazione dei sistemi a tempo discreto, riscriviamo le ultime due equazioni come

$$\begin{cases} x_{1_{k+1}} = x_{1_k} + T_{sample}\, x_{2_k} \,, \\ x_{2_{k+1}} = x_{2_k} + T_{sample}\, f(x_{1_k}, x_{2_k}) \,. \end{cases}$$

Nel caso specifico abbiamo

$$\begin{cases} q_{1_{k+1}} = q_{1_k} + T_{sample}\, \dot{q}_{1_k} \,, \\ \dot{q}_{1_{k+1}} = \dot{q}_{1_k} + T_{sample}\, \dfrac{\tau - m_1\, g\, l_1\, \sin(q_{1_k})}{I_1 + m_1\, l_1^2} \,. \end{cases} \tag{4.3}$$

Le (4.3) sono proprio le espressioni usate all'interno del metodo *OneLinkDirectDynamics()* per far avanzare lo stato del sistema meccanico in esame fino al successivo istante di campionamento.

Per quanto concerne la fase di avvio, si procede in modo analogo. La dinamica del robot è quella di un manipolatore planare a due bracci con un solo ingresso di controllo τ applicato al primo giunto. Facendo riferimento alla notazione usata nel capitolo (1), la dinamica è data da

$$\begin{cases} \ddot{q}_1 = \dfrac{b_{22}}{b_{11} * b_{22} - b_{12} * b_{21}} (\tau - c_{11}\,\dot{q}_1 - c_{12}\,\dot{q}_2 - g1) + \\ \quad + \dfrac{b_{12}}{b_{11} * b_{22} - b_{12} * b_{21}} (c_{21}\,\dot{q}_1 + c_{22}\,\dot{q}_2 + g2) = f_1\,, \\ \ddot{q}_2 = \dfrac{-b_{21}}{b_{11} * b_{22} - b_{12} * b_{21}} (\tau - c_{11}\,\dot{q}_1 - c_{12}\,\dot{q}_2 - g1) + \\ \quad + \dfrac{b_{11}}{b_{11} * b_{22} - b_{12} * b_{21}} (c_{21}\,\dot{q}_1 + c_{22}\,\dot{q}_2 + g2) = f_2\,. \end{cases}$$

Usando ancora una volta l'approssimazione di Eulero in avanti per le derivate temporali e passando alla più comoda notazione dei sistemi a tempo discreto, otteniamo

$$\begin{cases} q_{1_{k+1}} = q_{1_k} + T_{sample}\,\dot{q}_{1_k}\,, \\ q_{2_{k+1}} = q_{2_k} + T_{sample}\,\dot{q}_{2_k}\,, \\ \dot{q}_{1_{k+1}} = \dot{q}_{1_k} + T_{sample}\,f_1(q_{1_k}, q_{2_k}, \dot{q}_{1_k}, \dot{q}_{2_k})\,, \\ \dot{q}_{2_{k+1}} = \dot{q}_{2_k} + T_{sample}\,f_2(q_{1_k}, q_{2_k}, \dot{q}_{1_k}, \dot{q}_{2_k})\,. \end{cases} \tag{4.4}$$

Le (4.4) vengono usate nel metodo *TwoLinkDirectDynamics(τ)* per far avanzare lo stato del manipolatore fino al successivo istante di campionamento durante la fase di avvio.

Avendo a disposizione tali discretizzazioni, siamo ora in grado di scrivere il ciclo principale di una simulazione. Ad esempio per la simulazione della fase di avvio scriviamo:

```
2 double theTime = 0;

4 InitRobot();
```

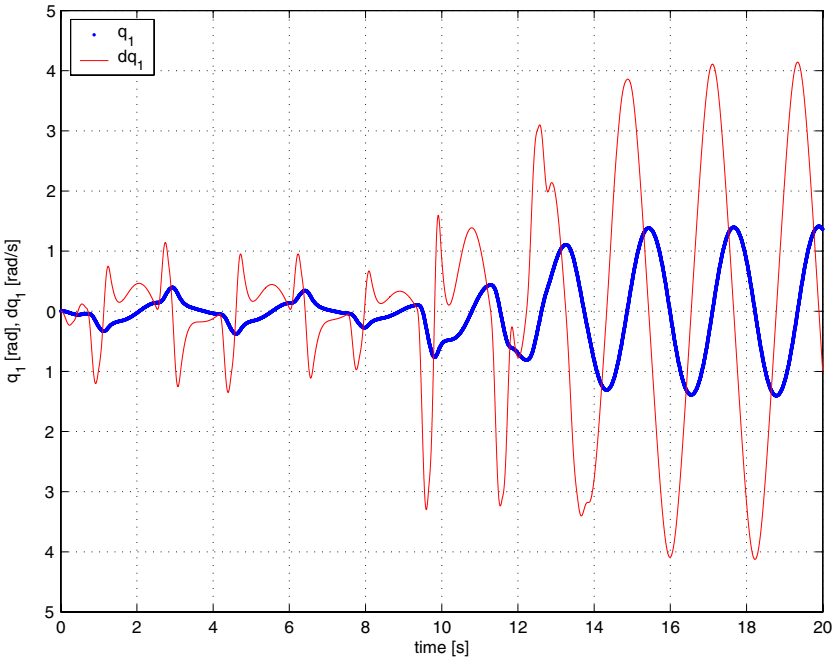

Figura 4.1: Andamento di q_1 e $\dot{q}_1$ durante la fase di avvio ottenuto in simulazione.

```
while (!EndOfSwing())
{
    SetDesiredTrajOnSwinging();
    double tau = TwoLinkComputedTorque();
    TwoLinkDirectDynamics(tau);
    SaveState();

    theTime += SAMPLE_TIME;
}
```

Il metodo *SaveState()* salva su disco lo stato attuale del manipolatore, comprendente lo stato delle variabili di giunto e quello dell'end-effector. Chiamando tale funzione alla fine di ogni ciclo, è possibile tracciare off-line l'evoluzione nel tempo di ognuna delle grandezze salvate. Come strumento per la visualizzazione su grafico, abbiamo scelto Matlab©. Nelle figure (4.1 e 4.2) sono riportati gli andamenti rispettivamente di q_1, $\dot{q}_1$, q_2, $\dot{q}_2$ relativi alla fase di avvio ottenuti mediante simulazione.

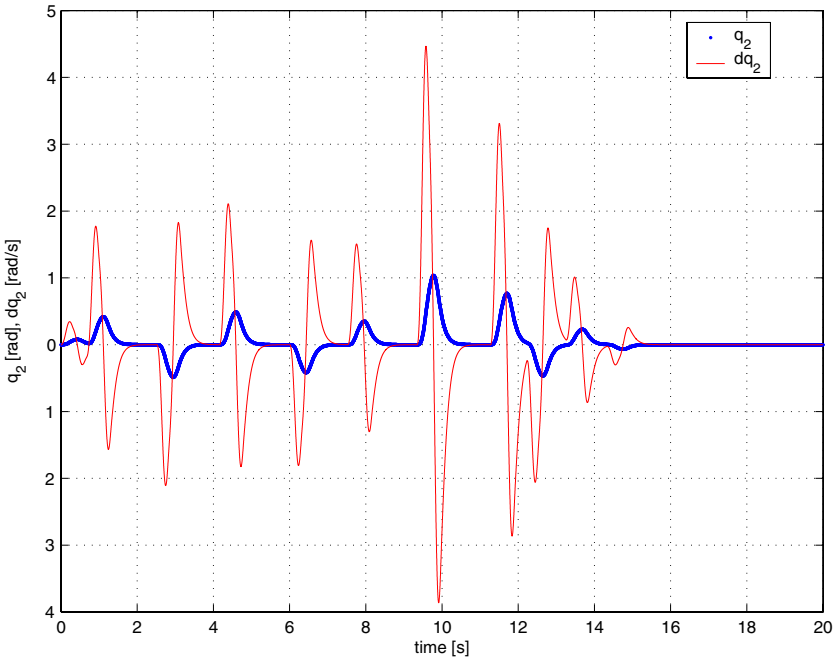

Figura 4.2: Andamento di q_2 e $\dot{q}_2$ durante la fase di avvio ottenuto in simulazione.

Per una descrizione delle altre funzioni citate si rimanda all'appendice (A) nella sezione relativa alla classe *CCastingManipulator*.

4.2 Le funzioni di controllo

Un altro gruppo di funzioni che appartengono all'interfaccia della classe *CCastingManipulator* riguarda la pianificazione ed il controllo in tempo reale del manipolatore. Nel realizzare tali metodi si deve prestare attenzione al rispetto dei vincoli temporali richiesti dal sistema da controllare; in altre parole il controllo del manipolatore deve essere aggiornato con un certa frequenza.

Tra le cause che rallentano l'esecuzione di un programma troviamo le operazioni di scrittura su disco, che devono pertanto essere eliminate, a meno che non siano strettamente necessarie. Qualora si presenti la necessità di salvare un dato, occorre tenere presente la possibilità di memorizzare tem-

poraneamente in memoria tale dato per poi salvarlo fisicamente solo alla fine dell'esperimento.

4.2.1 La gestione del tempo reale

Di seguito è riportato per intero il codice di gestione del tempo reale:

```
// Real-time management variables
unsigned int base_low, base_high;
unsigned int now_low, now_high;

// Read initial counter value
void CastingManipulator::InitRealTime()
{
    _asm
    {
        rdtsc
        mov base_low, eax
        mov base_high, edx
    }
}

// Read current counter value and evaluate elapsed time
inline void CastingManipulator::ReadRealTime()
{
    _asm
    {
        rdtsc
        mov now_low, eax
        mov now_high, edx
    }

    if (now_low < base_low)
        realTime = (((double) (now_high - base_high) - 1)*pow(2,32) +
                    (now_low - base_low)) / CPU_FREQUENCY;
    else
        realTime = (((double) (now_high - base_high))*pow(2,32) +
                    (now_low - base_low)) / CPU_FREQUENCY;
}

// Wait until next synchronisation time
void CastingManipulator::Synchronise()
```

```
{
38      theTime += SAMPLE_TIME;

40      ReadRealTime();

42      while (realTime < theTime)
            ReadRealTime();
44  }
```

L'istruzione assembler *rdtsc* legge il valore di un registro a 64 bit che viene inizializzato a zero alla creazione del processo di controllo ed incrementato all'arrivo di ogni clock. Tale registro contiene quindi ad ogni istante il numero di clock ricevuti dal momento di avvio del processo. Dividendo tale valore per la frequenza f_p di funzionamento del processore, si risale ai secondi trascorsi dall'istante di creazione del processo con precisione pari a $\frac{1}{f_p}$.

La funzione *InitRealTime()* inizializza la lettura del tempo reale, memorizzando il valore iniziale del contatore. La funzione *ReadRealTime()* legge il valore attuale del contatore, gli sottrae il valore iniziale e lo divide per la frequenza di funzionamento del processore. In tale maniera, si misurano effettivamente i secondi trascorsi dal momento in cui era stata chiamata la *InitRealTime()*. Infine la funzione *Synchronise()* imposta il successivo istante di sincronizzazione ed entra in un ciclo di attesa attiva fino all'arrivo di quell'istante. Una descrizione delle funzioni di gestione del tempo reale è riportata anche in appendice (A).

Tale approccio alla gestione del tempo reale sarebbe in realtà corretto solo in un ambiente mono-programmato. Infatti, il valore letto dall'istruzione *rdtsc* si riferisce a quello contenuto nel processore virtuale del processo di controllo. In altre parole, il tempo reale misurato è quello trascorso durante l'esecuzione del particolare processo che chiama la *rdtsc*. In un ambiente multi-programmato, in cui ai processi viene assegnato l'uso dell'unico pro-

cessore fisico secondo una politica concorrenziale, non viene considerato il tempo in cui il processore risulta assegnato ad altri processi.

Tuttavia, nel nostro caso, non si ha la necessità di sincronizzarsi perfettamente con un altro processo. Pertanto, il rallentamento del processo di controllo entro certi limiti non è percepibile. Durante la fase di avvio, tale rallentamento si presenta come un rallentamento dell'oscillazione. La fase più critica è quella del controllo in volo. Ciò nonostante, come chiariremo nel capitolo (5), l'attuale gestione del tempo reale non ha provocato problemi o malfunzionamenti del controllo.

Più correttamente si potrebbe pensare all'uso di sistemi operativi con esplicita gestione del tempo reale, tra i quali citiamo *VxWorks*, *RT Linux*, *RTAI* e *LinuxOs*.

4.2.2 L'indipendenza dal hardware utilizzato

Abbiamo cercato di svincolare la libreria di pianificazione e controllo dal tipo di hardware impiegato per la realizzazione del robot. A tal scopo, i metodi di controllo della classe *CCastingManipulator* non chiamano direttamente le funzioni di lettura e scrittura delle schede di acquisizione. Al contrario, si appoggiano ad uno strato intermedio di funzioni che abbiamo chiamato *Implementation Specific Library*.

Ad esempio l'aggiornamento della coppia τ viene fatto come segue:

```
// Codice dell'applicazione di controllo
void main()
{
    ...
    isl_WriteControls(tau);
    ...
}
```

```
// Codice dello strato intermedio
void isl_WriteControls()
{
    ...
    AO_VWrite(
        PCI6024E_DEVICE,
        DDMOTOR_CHANNEL,
        6.2066 * tau + 0.1035);
    ...
}
```

in cui *AO_ WRITE* è la funzione di scrittura relativa alla particolare scheda di acquisizione utilizzata. In questa maniera, qualora si renda necessario sostituire tale scheda con un'altra, non occorre modificare il codice della libreria di controllo, ma solo la procedura *WriteControls()*.

Nella appendice (A) è riportato l'elenco completo delle funzioni dello strato intermedio, tra cui troviamo anche l'inizializzazione delle schede e la gestione dei messaggi di errore.

4.3 Le funzioni diagnostiche

Per ultimo, tra i metodi della classe *CCastingManipulation*, troviamo quelli per la diagnostica del manipolatore costruito. Anche in questo caso i risultati delle misurazioni fatte sono salvati su file solo alla fine dell'esperimento e vengono visualizzati con Matlab©.

Capitolo 5

La realizzazione di un sistema di lancio e recupero

Per verificare i risultati teorici esposti nei precedenti capitoli, riguardanti la pianificazione ed il controllo in tempo reale finalizzati alla presa e al recupero di un oggetto in movimento, è stato costruito in laboratorio il manipolatore del lancio dalla struttura più semplice costituito da un solo braccio rigido e da una parte flessibile. Ciò rappresenta l'oggetto del presente capitolo.

Nel primo paragrafo viene descritto il setup hardware del robot, con particolare attenzione ai componenti fisici impiegati per la sua realizzazione; nel secondo paragrafo, si passa a parlare degli aspetti software ed infine, nel terzo ed ultimo paragrafo, sono riportati i risultati di alcune prove sperimentali di funzionamento.

5.1 Setup hardware

5.1.1 Struttura meccanica del manipolatore

Il manipolatore del lancio costruito in laboratorio è costituito da un solo braccio rigido J_1 attuato da una coppia τ e da un braccio flessibile ed estendibile J_2 non attuato e realizzato con filo da pesca (5.1). Entrambi i giunti

Figura 5.1: Il manipolatore del lancio costruito in laboratorio per la verifica dei risultati sperimentali.

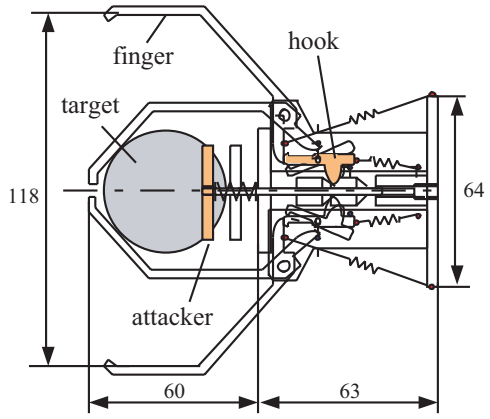

Figura 5.2: Progetto meccanico e realizzazione della pinza usata per afferrare il bersaglio.

sono rotoidali. Sull'estremità del secondo braccio è montato l'end-effector del robot. Per gli esperimenti di presa e recupero di un oggetto fermo è stata progettata e realizzata una pinza (figura 5.2), il cui meccanismo di chiusura viene azionato dalla forza di impatto con l'oggetto preso. Invece, per gli esperimenti di un bersaglio in movimento è stata utilizzata una *pallina* in alluminio.

Con riferimento alla simbologia usata nel capitolo (2), si riportano nella seguente tabella i valori delle grandezze geometriche ed inerziali del manipolatore realizzato:

Grandezza	Valore
x_{base}	0.000 m
y_{base}	1.695 m
m_1	1.1105 kg
I_1	0.0216 $kg\,m^2$
a_1	0.342 m
a_2	0.495 m
m_2	0.084 kg
I_2	1.3440E-005 $kg\,m^2$

5.1.2 L'attuatore

La coppia τ di attuazione del braccio rigido J_1 viene fornita da un *motore direct-driver*, montato sull'asse di rotazione del giunto corrispondente su cui è installato anche un encoder per la lettura della variabile di stato q_1.

Una volta fissata la posizione del peso, si fa variare la tensione di controllo v del motore fino al valore per cui l'asta, soggetta al momento della forza peso e alla coppia del motore τ, non risulta in equilibrio. Variando la distanza del peso e, quindi, il momento della forza peso da bilanciare è stato possibile calcolare la caratteristica tensione-coppia desiderata. I valori sperimentali ottenuti sono poi stati interpolati in Matlab©, approssimandoli con un polinomio del primo grado, cioè del tipo $v = a_1\,\tau + a_2$, in cui a_1 rappresenta la pendenza della retta e a_2 rappresenta uno spiazzamento iniziale. Il polinomio cercato è

$$v = 6.2066\,\tau + 0.1035\,. \tag{5.1}$$

In figura (5.3) sono riportati i valori sperimentali in rosso e la funzione approssimante in blu.

Sperimentalmente è stato possibile verificare che l'operazione di calibrazione ha migliorato il controllo del manipolatore.

5.1.3 Il meccanismo frenante

Come spiegato nei capitoli precedenti, il controllo in volo della traiettoria dell'end-effector è ottenuto mediante un'azione combinata di frenata e di movimento del braccio rigido J_1. Il meccanismo frenante impiegato a tal fine è riportato in figura (5.4). Su di esso, compare una fessura attraverso la quale viene fatto scorre il filo costituente il braccio flessibile J_2. Inoltre, il freno si compone di un cilindro metallico inserito all'interno di un solenoide. Un'estremità del cilindro è legata ad una molla di richiamo che, in assenza di tensione di controllo, impedisce al freno di bloccare lo scorrimento del filo; al contrario, quando il solenoide viene controllato con una tensione positiva esso attrae il cilindro in modo che l'altra estremità di esso prema sul filo e lo

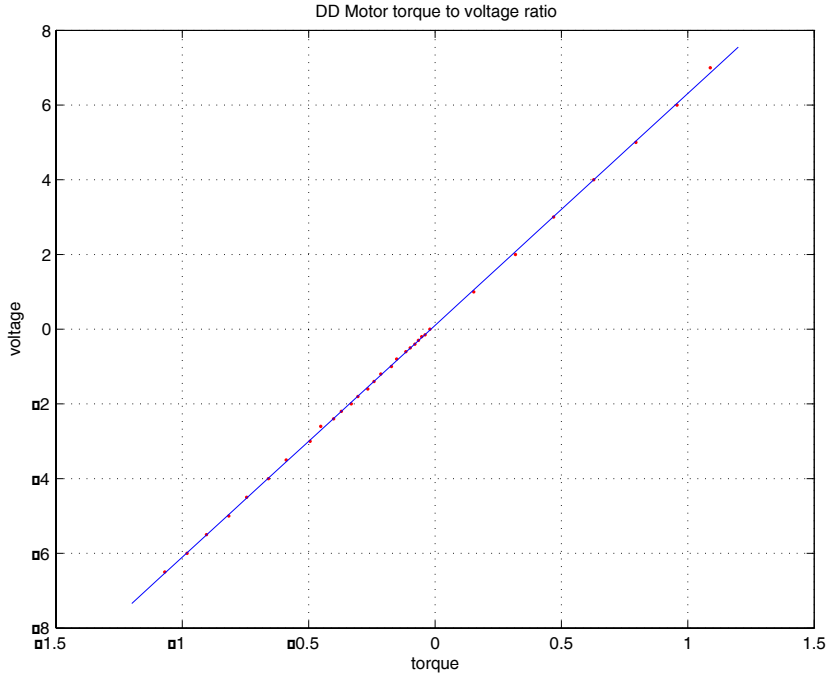

Figura 5.3: Caratteristica tensione di controllo del motore - coppia ottenuta in uscita per il motore direct-driver impiegato per l'attuazione del primo giunto.

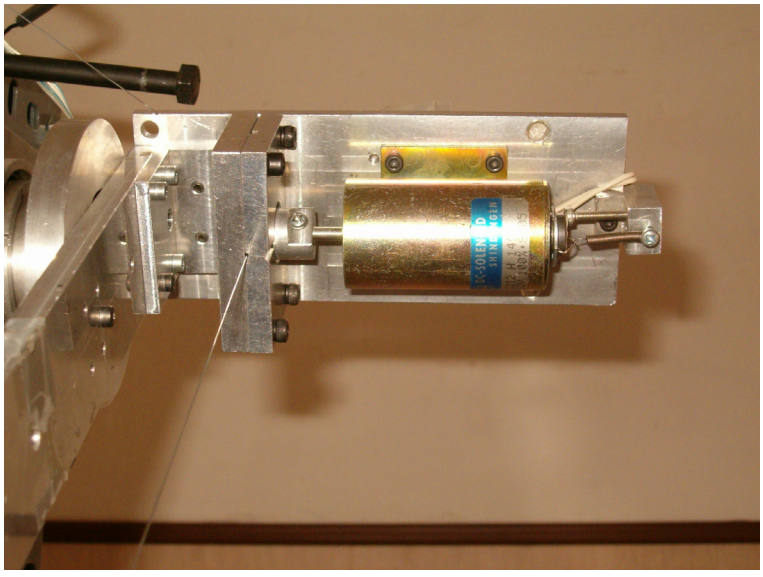

Figura 5.4: Freno a solenoide utilizzato per il controllo in volo della traiettoria dell'end-effector.

blocchi.

A dispetto della semplicità realizzativa e di controllo, il dispositivo frenante impiegato soffre di un ritardo non trascurabile per il controllo della dinamica dell'end-effector. Più in dettaglio, si è sperimentalmente visto che il tempo che intercorre tra l'invio dei comandi di frenata e di rilascio da parte del programma di controllo e l'istante di apertura o chiusura effettive del freno è dell'ordine di qualche decina di millisecondo, cioè risulta paragonabile alla durata di una singola frenata. E' stato pertanto necessario stimare tali ritardi e tenerne di conto nel controllo. I valori stimati sono:

$$t_{braking_{on}} = 0.024\, ms\,,$$
$$t_{braking_{off}} = 0.016\, ms\,.$$

5.1.4 I sensori e le schede di acquisizione

Per la lettura della variabile di giunto q_1, quella cioè relativa al braccio attuato, viene usato un encoder ad 81000 passi per rivoluzione (figura 5.5). Si tratta di un encoder di alta precisione per la cui lettura occorre utilizzare una scheda di acquisizione sufficientemente veloce. E' stato scelto di fare uso di una *US Digital PCI-4ES* (figura 5.6); si tratta di una scheda di acquisizione digitale da montare su bus PCI capace di leggere fino a quattro encoder in contemporanea con una frequenza di campionamento di 4 MHz.

In esecuzione è stato verificato che tale velocità non perde gli impulsi inviati dall'encoder e rappresenta uno strumento valido per la lettura di q_1.

Infine, per la lettura della variabile di giunto q_2 è stato impiegato un potenziometro (figure 5.7 e 5.8), la cui tensione di uscita è proporzionale all'an-

Figura 5.5: Encoder ad 81000 passi per rivoluzione impiegato per la lettura della prima variabile di giunto, q_1.

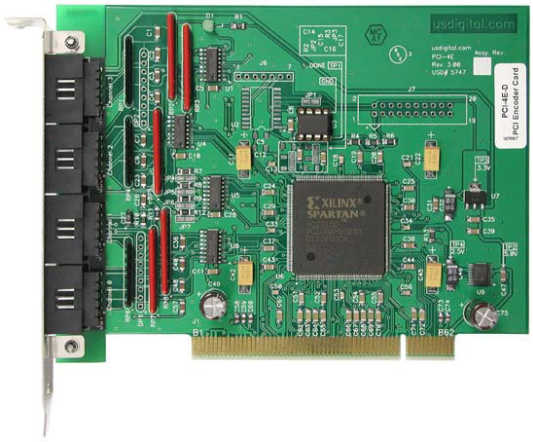

Figura 5.6: Scheda di acquisizione digitale *US Digital PCI-4ES*, utilizzata per la lettura dell'encoder.

Figura 5.7: Potenziometro utilizzato per la lettura della seconda variabile di giunto q_2.

golo misurato e viene acquisita mediante una *NI PCI-6024E* (figura 5.9); si tratta di una scheda di acquisizione analogica utilizzata anche per il controllo del dispositivo frenante e del motore di attuazione di q_1.

5.1.5 Ingresso e uscita del robot

Nella seguente tabella sono riportati i segnali di ingresso e quelli di uscita del manipolatore.

Segnale	Tipo
$v_{dd-motor}$	ingresso
v_{freno}	ingresso
$v_{potenziometro}$	uscita
$count_{encoder}$	uscita

I segnali di ingresso al manipolatore, nello specifico la tensione di controllo del motore direct-drive e del freno, sono forniti dalla scheda *NI PCI-6024E*. Tuttavia essi non vengono collegati direttamente ai dispositivi corrispondenti, ma attraversano uno *stadio di amplificazione* che provvede a fornire l'assorbimento di corrente richiesto.

5.1.6 La telecamera

Il sistema di visione fa uso di una telecamera Logitech *QuickCam© Orbit* con porta USB (figura 5.10). Si tratta di un dispositivo di acquisizione, che in condizioni medie di funzionamento, è caratterizzato da un frame-rate di 15 fotogrammi al secondo e una latenza di 66 millisecondi. Tali caratteristiche

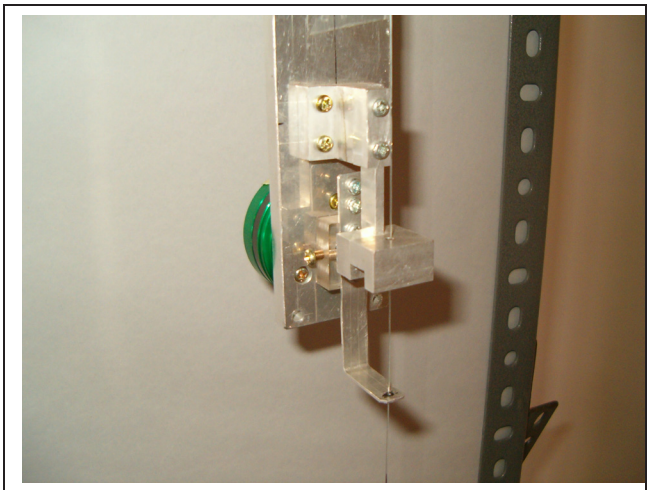

Figura 5.8: Il secondo giunto non attuato del manipolatore.

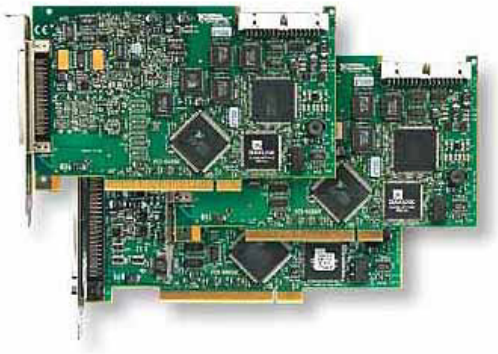

Figura 5.9: Scheda di acquisizione analogica *NI PCI-6024E* usata per la lettura del potenziometro, il controllo del motore e del freno.

Figura 5.10: La telecamera Logitech *QuickCam© Orbit*, usata per rilevare la posizione del bersaglio.

5.1.7 Il calcolatore

Il calcolatore utilizzato per la pianificazione ed il controllo in tempo reale del robot in esame monta un processore Pentium IV con frequenza di clock di 3.0 GHz. Le schede di acquisizione dati, USD-PCI-4ES e NI-PCI-6024E, sono state montate sul bus PCI di tale macchina, mentre la telecamera è stata collegata ad essa mediante una porta USB.

In fase sperimentale, con la macchina scelta è stato possibile elaborare i dati provenienti dalla telecamera e, parallelamente, controllare il robot.

5.1.8 Possibili aggiunte future

Nella realizzazione attuale il recupero dell'end-effector e, in caso di successo, anche del bersaglio viene svolto manualmente. Al fine di automatizzare tale operazione è richiesto di montare un altro motore in grado di riavvolgere il filo su di un mulinello anche esso tuttora mancante.

5.2 Setup software

Abbiamo già parlato della classe *CCamera* che contiene il codice per l'acquisizione del flusso video e della classe *CObjectLocalisation* che include invece il sorgente per la sua elaborazione e il calcolo della posizione del bersaglio. Inoltre, il codice di controllo del robot è contenuto nella *Casting Manipulation Library* che definisce la classe *CCastingManipulator*. Come già accennato, tale classe è stata realizzata cercando di *svincolarsi* dallo hardware utilizzato e si basa sull'ipotesi di disponibilità di un insieme di funzioni che leggono e scrivono direttamente sulle schede utilizzate (appendice A).

Per completare, quindi, anche dal punto di vista software, la fase sperimentale sono state realizzate tali funzioni e inglobate in un libreria detta *Implementation Specific Library*. Dal punto di vista notazionale ogni funzione di tale libreria comincia con il prefisso *isl_* .

Fra tali funzioni troviamo quelle per la lettura dello stato del manipolatore $\mathbf{x} = (q_1, q_2, \dot{q}_1, \dot{q}_2)$. Il valore della prima variabile di giunto q_1 si ottiene dalla lettura dell'encoder. A tal proposito, la scheda USD-PCI-4ES restituisce il valore di un contatore *count* a 24 bit che viene poi convertito nel corrispondente angolo mediante la relazione

$$\begin{cases} q_1 = \dfrac{count - 2^{24}}{81000}\, 2\,\pi & \text{se } count < 2^{23} - 1\,, \\ q_1 = \dfrac{count}{81000}\, 2\,\pi & \text{altrimenti}\,. \end{cases} \tag{5.2}$$

Il valore della posizione della seconda variabile di giunto q_2 è invece ottenuto leggendo la tensione v del potenziometro e convertendolo mediante la relazione

$$q_2 = 0.6077\, v - q_{2_{base}}\,, \tag{5.3}$$

dove $q_{2_{base}}$ è un valore sottratto in modo da annullare lo spiazzamento iniziale. Per quanto concerne poi le componenti della velocità, esse sono state ottenute mediante le approssimazioni

$$\begin{aligned} \dot{q}_1 &= \frac{q_1 - q_{1_{prec}}}{T_{sample}}\,, \\ \dot{q}_2 &= \frac{q_2 - q_{2_{prec}}}{T_{sample}} \end{aligned} \tag{5.4}$$

dove $q_{1_{prec}}$ e $q_{2_{prec}}$ rappresentano i valori precedenti e T_{sample} vale 0.0005 secondi. Infine, sperimentalmente si era osservata un'eccessiva rumorisità del potenziometro, con effetto negativo sulla derivata prima della variabile di giunto q_2. Per questo motivo, si è pensato di utilizzare un *filtro di Butterworth* software per reiettare parte del disturbo presente.

L'ultimo passo nella costruzione del sistema di lancio e recupero, è stato quello di realizzazione un'applicazione in grado di integrare le funzionalità sviluppate fino al momento. A tal proposito è stato creato il programma *Casting Manipulator Experiment* che fa uso delle seguenti classi o librerie:

- ▷ classi *Microsoft Foundation Classes* per l'interfaccia utente;
- ▷ classe *CCastingManipulator* per la pianificazione ed il controllo in tempo reale del robot;
- ▷ classe *CCamera* per l'acquisizione del flusso video;
- ▷ classe *CObjectLocalisation* per la rilevazione della posizione dell'end-effector;
- ▷ liberia *Implementation Specific Library* per la lettura e la scrittura dalle schede dal set di schede di acquisizione in uso.

I passi di esecuzione del suddetto programma sono i seguenti:

▷ Inizializzazione delle schede di acquisizione;

▷ Attivazione e calibrazione della telecamera;

▷ Avvio del thread relativo al controllo del manipolatore.

Riportiamo infine il codice del thread di controllo del manipolatore, in cui è possibile riconoscere l'interazione dei vari componenti software di cui abbiamo parlato:

```
extern CCamera m_camera;
extern CObjectLocalisation m_ObjectLocalisation;

UINT RobotControl(LPVOID pParam)
{
    // INITIALISATION

    CastingManipulator theRobot;
    theRobot.InitRobotInput();

    // Wait until the camera is ready and the target
    // is detected at least once.
    while (m_camera.GetFrame().GetImage() == 0);
    while (!m_ObjectLocalisation.ObjectDetected());

    AfxMessageBox("Initialise the robot and swing");

    theRobot.InitRobot();
    theRobot.InitRealTime();

    // STARTUP PHASE

    while (!theRobot.EndOfSwing())
    {
        theRobot.UpdateState();
        theRobot.SetDesiredTrajOnSwinging();
        theRobot.WriteControls();
        theRobot.BallOnSwinging();
        theRobot.Synchronise();
    }

```

```
    // THROWING PHASE
34
    // The manipulator is ready to throw, but waits until
36  // the users is ready too. So the end-effector is still
    // been swung.
38  AfxBeginThread(AskForReadyToThrow, NULL, THREAD_PRIORITY_TIME_CRITICAL);

40  while (!theRobot.EndOfSwing() || !ready)
    {
42      theRobot.UpdateState();
        theRobot.SetDesiredTrajOnSwinging();
44      theRobot.WriteControls();
        theRobot.BallOnSwinging();
46      theRobot.Synchronise();
    }
48
    theRobot.ThrowEndEffector();
50
    // MID-AIR CONTROL PHASE
52
    while (!theRobot.EndOfThrow())
54  {
        m_ObjectLocalisation.ObjectDetected();
56
        theRobot.UpdateState();
58      theRobot.SetDesiredTrajOnThrowing();
        theRobot.WriteControls();
60
        UpdateMidAirControl();
62      MidAirControl();

64      Synchronise();
    }
66


68  // REELING-UP PHASE
    theRobot.ReelUpEndEffector();
70
    // DISPOSAL OF ALL COMPONENTS
72  theRobot.ShutDownRobot();

74  AfxMessageBox("Robot is off");
```

```
76      return 0;
}
```

5.3 Verifiche di funzionamento

Durante la fase di realizzazione del setup sperimentale, sono state svolte alcune prove di funzionamento del sistema di lancio controllato e recupero. I risultati hanno mostrato che

- ▷ il sistema di visione calibrato ha un'alta ripetibilità e precisione;
- ▷ la posizione di atterraggio senza intervento di lancio varia attorno al valore medio di ± 2 cm.

Concludiamo riportando in figura (5.11), la traiettoria della pallina durante un esperimento di lancio controllato. Come è possibile vedere, la traiettoria della pallina è costituita da un arco di circonferenza, relativo alla fase di oscillazione, un arco di parabola, una frenata ed, infine, un altro arco di parabola che la porta a raggiunge il bersaglio fermo.

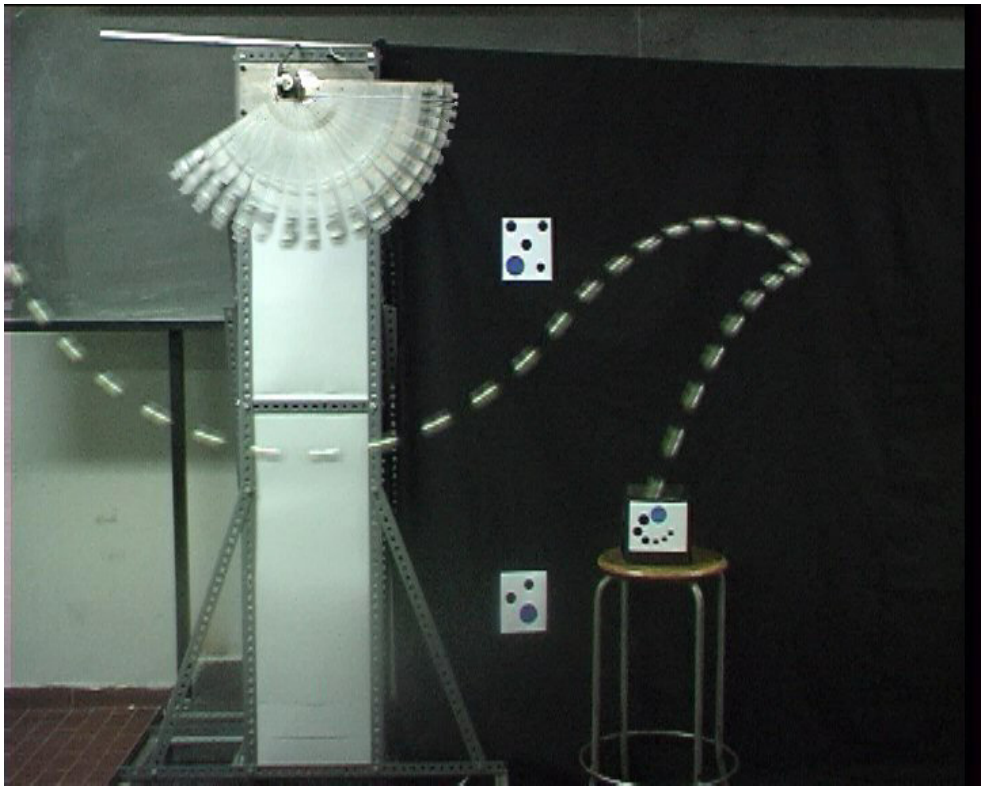

Figura 5.11: La traiettoria della pallina durante un lancio controllato. In essa è possibile riconoscere un arco di circonferenza, un arco di parabola, una frenata ed, infine, un altro arco di parabola che porta la pallina sul bersaglio.

Capitolo 6

Conclusioni

6.1 I risultati ottenuti

Durante questo lavoro di tesi sono stati realizzati:

- ▷ una libreria software per la pianificazione ed il controllo in tempo reale di un manipolatore del lancio;
- ▷ un sistema di visione che fa uso di una telecamera per rilevare la posizione istantanea del bersaglio;
- ▷ una strategia di controllo per la presa di un oggetto in movimento;
- ▷ un semplice manipolatore del lancio.

6.2 Gli sviluppi futuri

La pianificazione e il controllo del robot sono stati sviluppati nell'ipotesi in cui il braccio flessibile J_n non possieda massa ed inerzia proprie. Il modello che abbiamo così ottenuto è valido per il manipolatore del lancio che è stato costruito in laboratorio, in cui si è fatto uso del filo da pesca per realizzare il braccio J_2. Una prima estensione del lavoro svolto, riguarda l'utilizzo di un manipolatore del lancio nello spazio o su Marte, in cui l'assenza di gravità o

una gravità ridotta permettono di raggiungere distanze più lunghe di quelle ottenibili sulla Terra. Per la tipologia di utilizzo a cui si sta pensando, è necessario realizzare il braccio flessibile con un materiale più resistente, ma comunque flessibile. Si pensa ad esempio di impiegare una maglia in acciaio. In generale, *il braccio flessibile sara dotato di massa ed inerzia proprie.* In tal caso, nel controllo e nella pianificazione del robot, dobbiamo tenere di conto anche della dinamica del braccio J_n, per la quale si potrebbe adottare un modello agli elementi finiti.

Una seconda estensione riguarda poi il controllo in volo. Fino al momento, abbiamo supposto libero lo spazio che l'end-effector percorre per raggiungere ed afferrare il bersaglio. Più in generale, dovremmo considerare anche la *presenza di ostacoli.* Sarebbe comodo definire una politica di controllo in volo che tenga di conto esplicitamente di tali oggetti. Si potrebbe ad esempio pensare di scomporre la traiettoria dell'end-effector in una sequenza di sotto-traiettorie prive di ostacoli.

Resta inoltre da implementare interamente la *fase di recupero.* Durante tale fase si deve prestare attenzione ad evitare gli ostacoli, a non perdere o danneggiare il bersaglio e l'end-effector.

L'ultima estensione del lavoro svolto riguarda la realizzazione di un *controllo embedded* del manipolatore, le cui funzioni dovrebbero includere la capacità di comunicare senza fili con un calcolatore centrale. All'interno di tale obiettivo, si inserisce anche l'uso di *altri sistemi operativi* che abbiamo un'esplicita trattazione del tempo reale e lo studio di schede di acquisizione specifiche.

Ringraziamenti

Vorrei ringraziare innanzitutto il Prof. Antonio Bicchi per il sostegno e la comprensione, ancor prima dei suggerimenti, che mi ha dato. In secondo luogo, è giusto ringraziare l'Ing. Giovanni Tonietti, quale tutore e amico, che mi ha seguito sin dai tempi di Teoria dei Sistemi. Un altro ringraziamento va all'Ing. Hitoshi Arisumi per la supervisione nella realizzazione del manipolatore del lancio e le lunghe chiacchierate davanti ad un bicchiere di saké - «Ah lili!»

Vorrei poi ringraziare la Dott.essa Lucia Pallottino per i suggerimenti da vera matematica - almeno una che ci capisce in queste cose c'è - per la grande simpatia e la sopportazione anche nei periodi in cui le abbiamo invaso casa. Un altro ringraziamento va agli altri dottorandi del Centro Piaggio: primo fra tutti l'Ing. Nicola Sgambelluri - «works works!» - per l'aiuto nei tempi più bui della tesi e agli Ingg. Antonio Danesi, Vincenzo Scordio e Daniele Fontanelli per le delucidazioni sulla parte visiva. Poi vorrei ringraziare l'Ing. Davide Dente che mi ha risuscitato il computer da un falso collasso.

Voglio poi ringraziare il Maestro, Fabio Vivaldi, per aver fisicamente realizzato, anzi direi materializzato, le parti mancanti del manipolatore del lancio «Mah... 'sti cervelli!».

Non posso dimenticarmi a questo punto dei miei più cari colleghi di studio, nonché grandi amici, gli Ingg. Michele Bavaro e Riccardo Schiavi, con i quali ho percorso gli ultimi sforzi verso il raggiungimento di questa meta, assieme a tutti i dubbi e i ripensamenti delle scelte fatte. Voglio ringraziarli entrambi per il continuo aiuto e per l'amicizia, in ricordo delle giornate, serate e nottate passate insieme a far funzionare un *maledetto* algoritmo di identificazione. Spero veramente che l'amicizia del nostro *Trio* vada molto lontano.

Un grazie va anche ad Angela Cozzolino, a Giulia Gambetta e a Simona Bacchereti per avermi sopportato nel mio continuo parlare della tesi e avermi fatto coraggio nei momenti più tristi.

Infine, vorrei salutare tutti quelli che hanno vissuto con me il periodo del *Grande Trasloco*, passando dalla situazione di profughi rifugiati nell'aula A32 fatta un magazzino al comodissimo laboratorio che ci siamo conquistati.

阿度利吾乃　富和助理仁

Adriano Fagiolini

Appendice A

Guida di riferimento al software realizzato

Questa sezione costituisce una guida di riferimento alle classi e librerie che sono state sviluppate durante il lavoro di tesi. Per ognuna di esse, si trova un elenco dei metodi più significativi assieme ad una loro breve descrizione.

A.1 La classe *CCamera*

bool Initialize(...);

Chiede al sistema operativo l'uso esclusivo del dispositivo di acquisizione video e ne attiva il gestore.

void Uninitialize();

Disabililità il gestore del dispositivo di acquisizione e ne rilascia l'utilizzo.

void CalculateFrameRate();

Calcola il numero di fotogrammi al secondo che mediamente il dispositivo di acquisizione è in grado di fornire nelle impostazioni di funzionamento correnti.

LRESULT PASCAL FrameCallbackProc(...);

Rappresenta il gestore del dispositivo di acquisizione.

A.2 La classe *CObjectLocalisation*

int FindObject(IplImage img, CvPoint* p);*

Cerca all'interno dell'immagine *img* un marcatore noto; restituisce l'identificativo di tale marcatore e salva in p la posizione del centro del punto di riferimento.

int FindAllMarkers(IplImage img, CvPoint** marker_pos);*

Cerca all'interno dell'immagine *img* tutti i marcatori necessari per la procedura di calibrazione e salva in *marker_pos* la posizione dei centri dei rispettivi punti di riferimento.

bool ObjectDetected(IplImage img, CvPoint* p);*

Funzione *non bloccante* che cerca all'interno dell'immagine *img* il marcatore del bersaglio; in caso di successo restituisce *true* e salva in p la posizione del centro di riferimento; altrimenti restituisce *false*.

int ConvertToRealWorld(CvPoint p, double* x, double* y);*

Converte le coordinate p relative al piano immagine del centro di riferimento nelle corrispondenti coordinate (x, y) relative al piano di movimento, facendo uso la mappa di omografia $\mathcal{H}$.

void LoadHomography(const char filename);*

void SaveHomography(const char filename);*

Funzioni per il caricamento o il salvataggio della mappa di omografia $\mathcal{H}$ nel file $filename$.

void CalibrateCamera();

Esegue la calibrazione della telecamera.

A.3 La classe *CCastingManipulator*

A.3.1 Inizializzazione e fasi di funzionamento

void InitRobot();

Inizializza il manipolatore del lancio, azzerando la lettura dell'encoder e del potenziometro e la scrittura sugli ingressi di controllo del motore direct-drive e del meccanismo frenante.

void SwingMotion();

Realizza la fase di avvio del manipolatore; durante tale fase il robot viene controllato al fine di caricare l'end-effector con un energia sufficiente per raggiungere ed afferrare il bersaglio in movimento.

void ThrowEndEffector();

Rappresenta la fase di lancio del manipolatore; durante tale fase l'end-effector viene fatto oscillare in modo da non fargli perdere l'energia necessaria a raggiungere ed afferrare il bersaglio, fino al verificarsi della condizione di lancio; nell'istante in cui tale condizione si realizzata il filo che tiene bloccato l'end-effector viene rilasciato.

void MidAirControl();

Rappresenta la fase di controllo in volo; durante tale fase la traiettoria dell'end-effector viene modificata per permettere ad esso di raggiungere ed afferrare il bersaglio, attraverso un'azione combinata di movimento dei bracci rigidi attuati e di frenata della parte flessibile in estensione.

void ReelUpEndEffector();

Rappresenta la fase di recupero; durante tale fase l'end-effector e, in caso di successo nella presa, anche il bersaglio vengono recuperati attraverso un'azione combinata di riavvolgimento del braccio flessibile e movimento dei giunti attuati.

void ShutDownRobot();

Controlla il robot in modo da riportarlo nella configurazione di partenza e rilascia l'utilizzo dei dispositivi di lettura e scrittura.

A.3.2 Pianificazione della traiettoria

void SetDesiredTrajOnSwinging();

Imposta la traiettoria desiderata $\mathbf{q}_{2_d}$ del secondo giunto durante la fase di avvio.

void SetOneLinkDesiredState(...);

Imposta la traiettoria desiderata $\mathbf{q}_{1_d}$ del primo giunto dal momento del lancio in poi.

A.3.3 Controllo del manipolatore

double TwoLinkComputedTorque();

Fornisce la coppia τ richiesta al motore direct-drive secondo un controllo a coppia pre-calcolata durante la fase di avvio.

double OneLinkComputedTorque();

Fornisce la coppia τ richiesta al motore direct-drive secondo un controllo a coppia pre-calcolata dal momento del lancio in poi.

A.3.4 Gestione del tempo reale

void InitRealTime();

Inizializza la lettura del tempo reale.

void ReadRealTime();

Restituisce il numero di secondi trascorsi dall'istante di chiamata della funzione *InitRealTime()*.

void Synchronise();

Funzione *bloccante* che non restituisce il controllo al processo fino al successivo istante di sincronizzazione impostato da programma; nella pratica la sincronizzazione avviene su ogni decimo di millisecondo e viene utilizzata per rendere sincrona l'esecuzione del controllo del robot con il tempo reale.

A.3.5 Simulazione

void TwoLinkDirectDynamics(double tau);

Aggiorna lo stato del manipolatore, modellato come un robot planare a due bracci, in accordo alla sua dinamica e al valore dell'ingresso di controllo τ, usando come tempo di campionamento il valore impostato di T_{sample}.

void OneLinkDirectDynamics(double tau);

Aggiorna lo stato del manipolatore, modellato come un robot planare ad un braccio, in accordo alla sua dinamica e al valore dell'ingresso di controllo τ, usando come tempo di campionamento il valore impostato di T_{sample}.

A.4 Implementation Specific Library

void isl_ ClearBoards();

void isl_ InitBoards();

Chiamate in questa sequenza, le due funzioni chiedono al sistema operativo l'uso esclusivo delle schede di acquisizione *USD-PCI-4ES* e *NI-PCI-6024E* e provvedono alla loro inizializzazione.

void isl_ ShutDownBoards();

Rilascia l'utilizzo delle schede di acquisizione.

void isl_ ReadJointVariables(...);

void isl_ ReadFirstJointVariable(...);

La prima funzione legge lo stato completo $\mathbf{x} = (q_1, q_2, \dot{q}_1, \dot{q}_2)$ del manipolatore, mentre la seconda legge solo le grandezze relative alla prima variabile di giunto q_1. Laddove necessario si fa uso di (5.2), (5.3) e (5.4).

void isl_ WriteControls(double tau);

Cambia la tensione di controllo del motore direct-drive in accordo alla coppia τ richiesta, facendo uso della caratteristica coppia-tensione approssimata in (5.1).

void isl_ WriteDirectControls(double voltage);

Cambia la tensione di controllo del motore direct-drive in accordo al parametro fornito ingresso.

void isl_ SolenoidOn();

void isl_ SolenoidOff();

Funzioni di frenata e rilascio del filo da parte del meccanismo frenante.

Bibliografia

[1] Hitoshi Arisumi, Tetsuo Kotoku, Kiyoshi Komoriya, "A study of Casting Manipulation (Swing Motion Control and Planning of Throwing Motion)", *International Conference on Intelligent Robots and Systems*, 1997.

[2] Hitoshi Arisumi, Tetsuo Kotoku, Kiyoshi Komoriya, "Swing Motion Control of Casting Manipulation (Epxeriment of Swing Motion Control)", *International Conference on Robotics and Automation*, 1998.

[3] Hitoshi Arisumi, Tetsuo Kotoku, Kiyoshi Komoriya, "Study on Casting Manipulation (Experiment of Swing Control and Throwing)", *International Conference on Intelligent Robots and Systems*, 1998.

[4] Hitoshi Arisumi, Tetsuo Kotoku, Kiyoshi Komoriya, "Swing Motion Control of Casting Manipulation", *IEEE*, 1999.

[5] Hitoshi Arisumi, Kiyoshi Komoriya, "Posture Control of Casting Manipulation", *International Conference on Robotics and Automation*, 1999.

[6] Hitoshi Arisumi, Kiyoshi Komoriya, "Study on Casting Manipulation (Midair control of gripper by impulsive force)", *International Conference on Intelligent Robots and Systems*, 1999.

[7] Hitoshi Arisumi, Kazuhito Yokoi, Kiyoshi Komoriya, "Casting Manipulation (Braking Control for Catching Motion)", *International Conference on Robotics and Automation*, 2000.

[8] Hitoshi Arisumi, Kiyoshi Komoriya, "Catching Motion of Casting Manipulator", *International Conference on Intelligent Robots and Systems*, 2000.

[9] K. Furuta, M. Yamakita, S. Kobayashi, "Swing up control of inverted pendulum", *Proceedings of the IEEE International Conference on Industrial Electronics, Control and Instrumentation*, 1991.

[10] T. Sakaguchi, F. Miyazaki, "Dynamic Manipulation of Ball-in-cup Game", *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994.

[11] Y. Hashimoto, T. tsuchiya, I. Sugioka, T Matsuda, "Transversal Load-Swing suppression Control of Travelling Crane", *Transaction of RSJ*, 1993.

[12] A. De Luca, R. Mattone, G. Oriolo, "Stabilization of an Underactuated Planar 2R Manipulator", *International Journal of Robust and Nonlinear Control*.

[13] Mark W. Spong, "Underactuated Mechanical Systems", *Coordinated Science Laboratory (University of Illinois)*.

[14] A. Quarteroni, R. Sacco, F. Saleri, "Numerical Mathematics", Springer, 1991.

[15] W. H. Press et al., "Numerical Recipes in C++", Cambridge University Press, 1998.

[16] "Open Source Computer Vision Library - Reference Manual", Intel©, 2001.

[17] Robert Laganière, "Programming computer vision applications: A step-by-step guide to the use of the Intel OpenCV library", VIVA lab, University of Ottawa.

[18] Midori America Corporation, "Specification DWG for model CPP-45B Potentiometer", 2000.

[19] "Canon Laser Rotary Encoder", *http://www.industrialnewsroom.com/*.

[20] National Instruments Corporation, "NI-DAQ Function Reference Manual for PC Compatibles", 1998.

[21] National Instruments Corporation, "DAQ PCI-6023E/6024E/6025E User Manual", 1999.

[22] National Instruments Corporation, "NI-DAQ User Manual for PC Compatibles", 2000.

[23] US Digital, "PCI-4E Data Sheet", Gennaio 2003.

[24] US Digital, "PCI-4E Manual", Settembre 2003.

[25] "Logitech QuickCam© Orbit", *http://www.logitech.com/index.cfm/products/*.